Burroughs

B 1800 / B 1700 Systems

SYSTEM SOFTWARE

OPERATIONAL GUIDE

RELATIVE TO B 1700 SYSTEMS SOFTWARE RELEASE – MARK VI.1

PRICED ITEM

Printed in U.S.A.

August 1977

Burroughs

B 1800 / B 1700 Systems

SYSTEM SOFTWARE

OPERATIONAL GUIDE

RELATIVE TO B 1700 SYSTEMS SOFTWARE RELEASE – MARK VI.1

AA370509 AA401135 AA472198 AA551824 AA629484 AA728871

PRICED ITEM

Printed in U.S.A.

August 1977

1068731

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 Burroughs Corporation, P. O. Box 4040, El Monte, California 91734, Attn: Publications Department, TIO – West.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

INTRODUCTION

The productivity of a computer facility is largely dependent on an operator's experience and knowledge of the system. When the programs produced for the installation have been refined and are ready for use, the results obtained are largely due to the expertise of the operator. Therefore, some concept of the MCP and a knowledge of the peripherals used with the B 1800/B 1700 Systems are important in order to utilize the equipment effectively.

This manual is divided into sections to ease the operating personnel's task in referencing material to efficiently operate the B 1800/B 1700 system.

The purpose of the B 1800/B 1700 Systems System Software Operational Guide is to provide a general description of all Burroughs B 1800/B 1700 System Software without going into such detail as is required for a programming language or a reference manual. Formal documents pertaining to the system software described herein are referenced where applicable. Included in this manual are those operating instructions required to perform any major function of the described system software.

An explanation of the notational conventions used throughout this manual is as follows:

a. Key Words. All underlined upper case words are key words and are required when the functions of which they are a part are utilized.

EXECUTE

b. Optional Words. All upper case words not underlined are optional words, included for readability only, and may be included or excluded as desired.

FOR

c. Lower Case Words. All lower case words represent generic terms which must be supplied in the position described.

file-identifier

d. Braces. Words or phrases enclosed in braces ({ }) indicate a choice of entries of which one must be made.

{ EXECUTE / EX }

e. Brackets. Words or phrases enclosed in brackets ([ ]) represent optional portions of a statement which may be omitted.

[=]

f. Consecutive Periods (Ellipsis). The presence of ellipsis (...) within any format indicates that the control syntax immediately preceding the ellipsis notation may be successively repeated, depending upon the requirements of the operation.

```
[control-attributes] . . .
```

g. Question Mark. The appearance of a question mark (?) indicates that any invalid EBCDIC character or the question mark itself is acceptable. This convention is used primarily by the Master Control Program to indicate a control card instruction.

```
[?] LOAD
```

h. At Sign: Any data contained between "at signs" @ identifies that information to be hexadecimal information.

```
@0CF3@
```

i. Integer: The presence of the word integer within any format signifies that the data to be expressed may be decimal, octal, hexadecimal, or binary.

decimal – any valid decimal character or characters.

```
452
```

hexadecimal – any valid hexadecimal character or characters enclosed within @ signs.

```
@A22F@
```

octal – any valid octal character or characters enclosed within @ signs and preceded by the MODE indicator (3). The MODE indicator must be enclosed by parentheses.

```
@(3)036@
```

binary – any valid binary character or characters enclosed within @ signs and preceded by the MODE indicator (1). The MODE indicator must be enclosed by parentheses.

```
@(1) 1101001@
```

j. Master Control Program: The Master Control Program is abbreviated throughout this manual as MCP. Its functions are explained in a separate section of this manual.

SECTION 1
INTRODUCTION TO SYSTEM

SYSTEM INITIALIZATION

The MCP was designed as an integral part of the system and is intended to serve a wide range of installations and users. Therefore, provisions have been incorporated in the system to adapt the operation of the MCP to the particular requirements of a variety of installations. This has been accomplished by incorporating different environments within the MCP which may be specified at the time of system initialization. Some of the environment options can be changed or set after the system has been initialized by using a console printer input message.

In order to place the MCP in control of the system, the MCP must be loaded onto the system disk with the system's environment defined and the disk directory established. Then the SDL interpreter must be loaded to interpret the MCP S-language. When this procedure has been completed, the SDL interpreter starts interpreting and executing the instructions of the MCP.

Three separate procedures are performed during initialization thereby making the system operable: (1) Initializing Disks (System and Removable), (2) Performing a COLDSTART, and (3) Performing a CLEAR/START.

UNIT MNEMONICS

Mnemonic names are assigned to the peripherals attached to the system by the MCP. The mnemonics are:

CDx	Card Reader/Punch
CPx	Card Punch
CRx	Card Reader
CSx	Magnetic Tape Cassette
DCx	Disk Cartridge
DKx	Head-per-Track Disk
DPx	Disk Pack
FDx	Diskette ("Floppy Disk")
LPx	Line Printer
MTx	Magnetic Tape
PPx	Paper Tape Punch
PRx	Paper Tape Reader
SPO	Console Printer
SRx	MICR Reader-Sorter

NOTE

The "x" is replaced by a capital letter, A - Z, for multiple units of a specified type.

SYSTEM DESCRIPTION

The following functions are controlled by the MCP:

a. Loading

b. Interrupt handling

c. I/O control

d. Selection and initiation of programs

e. I/O error handling

f. System log maintenance

g. Storage allocation—memory and disk

h. Overlay functions—data and code

i. Multiprogramming

MCPII will service the following peripheral equipment:

a. Console Printer or Console Display (SPO)

b. 96-column Card Devices

c. 80-column Card Devices

d. Line Printers

e. Disk Cartridges

f. Disk Packs

g. Head-per-Track Disk

h. Diskette ("Floppy Disk")

i. Magnetic Tapes (7-track, 9-track, and phase-encoded)

j. Magnetic Tape Cassettes

k. Data Communications (Single Line and Multi-Line Controls)

l. MICR Reader-Sorters

m. Paper Tape Devices

HARDWARE REQUIREMENTS

The following list of equipment must be present for MCP operations. However, the listed equipment is not dedicated to the MCP and may be utilized by any user program.

Hardware Type	Usage
Console Printer or Console Display	Operator communication
Disk	Auxiliary storage
Card Reader	Control input

CENTRAL SERVICE MODULE

The Central Service Module (CSM or GISMO) is a microcoded routine which performs the following functions in an equivalent hard-wired machine:

a. Interrupt Detection and Handling.

b. Passes control to/from the MCP, usually on an interrupt.

c. Controls all I/O activity, such as:

   1. I/O Initialization

   2. Data Transfers

   3. I/O Termination

d. Manages Interpreter Activity.

INTERPRETERS

Interpreters are microcoded routines, or "firmware," that perform the operations specified by the programmer. Each language has its own interpreter.

SECTION 2
MASTER CONTROL PROGRAM

GENERAL

The Master Control Program (MCP) is a modular operating system which assumes complex and repetitive functions to make programming and operations efficient and productive. The MCP provides the coordination and processing control that is so important to system throughput by allowing maximum use of all system components. Operator intervention is greatly reduced through complete resource management. Since all program functions are performed under this centralized control, changes in scheduling, system configuration, and program size can be readily accommodated resulting in greater system throughput.

A detailed description of the MCP is presented in the B 1700 Master Control Program Reference Manual.

MCP DISK STRUCTURES

A significant aspect of the MCP design is the disk handling technique. Because this handling is the responsibility of the MCP, programs are less complicated and easier to write.

Areas handled by the MCP include:

a. Directory Maintenance

Users need only to specify LOAD, DUMP, ADD, UNLOAD, CHANGE, or REMOVE directives by file-name. All other actions pertaining to disk table maintenance are automatic.

b. Disk Allocation

Programs need only specify the amount of disk space they require. The MCP will handle the actual allocation of a physical area containing only the amount requested.

c. File Assignment

As for all files within the system, disk file assignment is made according to the programmatically specified file name and type.

d. Record Addressing

Programs need only specify the accessing method, and in the case of random files the specific record desired. The actual disk location is the sole responsibility of the MCP. This means the programmer need not be concerned with the physical locations of the files.

e. Paging

Paging is the technique by which the programmer may divide a disk data file into portions which may occupy non-contiguous areas of disk, rather than one huge area. Areas need not even be allocated until actually needed, thus decreasing the need for disk space until required by the size of the file.

DISK DIRECTORY

The Disk Directory is a disk-resident table that contains the name and type of file, together with a pointer to the disk file header or sub-directory for all files on which the MCP received a permanent disk directory entry request.

Disk-Pack-Identifier

The disk-pack-id is the name that is assigned to a user disk pack or cartridge at initialization time.

Example:

AAA/program-name/

AAA is the disk-pack-id

Main Directory File Name

If there were a set of programs that were all common to solving one problem, they could all have the same first "family" name. The disk pack-id is not used to access a system disk.

Example:

PAYROLL/program-name-1
PAYROLL/program-name-2
PAYROLL/program-name-3
PAYROLL/program-name-4

In this example, PAYROLL is the main directory file name or the family name, while program-name-1 through program-name-4 are the sub-directory file names.

Sub-Directory File Name

The main directory links to a sub-directory when the sub-directory file-id is used. This sub-directory will contain an address on disk of a File Header for each of the sub-directory file entries. The sub-directory is an extension of the main directory.

Main Directory Contents

The main directory entry contains:

1. Family Name
2. Address of the disk file header or sub-directory.
3. Type of File:

   1 = LOG
   2 = Directory (entry points to sub-directory)
   3 = Control Deck
   4 = Backup Print
   5 = Backup Punch
   6 = Dump File
   7 = Interpreter
   8 = Code File
   9 = Data File

Sub-Directory Contents

If the file has a family name and a secondary file name, the address in the main directory does not point to the disk file header but to a sub-directory. This sub-directory has the same format as the main directory, except that it uses only one segment of disk for each eleven file names. If there are more than eleven names in the sub-directory the MCP increases the size by one segment for each eleven additional names.

The sub-directory entry is identical to the main directory entry with the exception that the address always point to a Disk File Header.

Directory Reference

When a file is referenced on a removable disk, it must be preceded with the disk-pack-id. The removable disk directories and system disk directories are the same format.

MULTIPLE PACK FILES

Introduction

A multiple pack file is a file that can be contained on one or more removable disk packs (cartridges). The file attribute MULTI.PACK of the FILE statement may be used to declare a file to be a multiple pack file.

Restrictions

There are some restrictions imposed on multiple pack files that limit their selection for usage. They are as follows:

a. The maximum number of packs that can be assigned to a multiple pack file is 16, consisting of one base pack and 15 continuation packs.

b. There must be a minimum of two (2) disk drives present on the system (one system pack and one removable pack).

c. Only removable disk packs can be used for multiple pack files. A system pack cannot be used for a multiple pack file.

d. All packs containing a multiple pack file must have unique serial numbers. The disk pack-id is not the primary identifier for continuation packs.

NOTE:

It is suggested that all packs be initialized with unique serial numbers.

Base Packs

A multiple pack file can have only one base pack. The base pack must be on-line for any open or close operation performed on the file. It can be required at other times depending on the action requested by the program; if so, a message is printed on the console printer requesting the operator to mount the base pack if it is not on-line. A base pack can contain both single and multiple pack files; however, it cannot contain continuation files.

The header on the base pack retains all information concerning the file including the address of every area assigned to that file. For each area resident on a continuation pack, the base pack contains the serial number of the continuation pack. This allows the MCP using the base pack to control all processing, and thereby avoid updating each continuation pack as the file is processed.

Continuation Packs

When data overflows or "continues" to additional disk packs, the term Continuation Pack (CP) is used. There can be up to a maximum of 15 continuation packs for one multiple pack file, but a continuation

pack can be associated with only one base pack. A continuation pack can only contain continuation files, and all continuation files contained on a continuation pack must be assigned to the same base pack. A CP cannot contain single pack files, and cannot itself be a base pack.

When space is needed for a multiple pack file, the MCP searches for another continuation pack that is associated with the same base pack. If a continuation pack is not present on the system, the MCP then searches for a scratch pack (one that has just been initialized or purged and is of the same type, restricted or unrestricted, as the base pack). If such a pack is found, the scratch pack is automatically assigned to the multiple pack file.

General Information

DISK/COPY cannot be used to copy multiple pack files. Multiple pack files can be sorted (INPLACE sort only).

To obtain the maximum disk space available for a multiple pack file, assign 105 as the number of areas required and increase the BLOCKS.AREA.

Random files are allowed for multiple pack files.

A system pack cannot be used as a base or continuation pack.

The CHANGE (CH) message is not allowed on a multiple pack file; however, a REMOVE (RE) message is permissible.

An interrogation is performed by the MCP prior to opening a multiple pack file to predict whether a duplicate file situation exists. If so, the operator has the option of either removing the existing file at that time or waiting until file close time to remove the duplicate file, and using the OK console message.

A DS console message performs a normal close on a multiple pack file.

A base pack does not necessarily have to have any of its data residing on it. As soon as the file is opened and the tables built, the base pack can be removed or be off-line.

A scratch pack is one that has either been just initialized or purged. A pack which has had all files removed is not a scratch pack.

FILE SECURITY

General

In addition to the normal methods for referencing disk files, a mechanism is provided for securing files against accidental or deliberate misuse. For instance, a secured file cannot be listed, removed, or changed by an unauthorized user. A secured file is identified by the existence of parentheses surrounding its family-name. For example:

(PAYROLL)/<file-name>

A family-name enclosed in parentheses, such as (PAYROLL), is referred to as a usercode. Well-defined rules exist for creating and accessing files having usercodes. In addition, before files having usercodes can be created, a set of valid usercodes must be defined for the MCP. This is done with a program named SYSTEM/MAKEUSER which creates a file on system disk labeled (SYSTEM)/USERCODE, containing all valid usercode/password combinations.

The usercode provides a way of naming and grouping sets of disk files. The password authorizes entry into the system.

File security is independent of terminal users and data communications. Thus, the usercode/password combination can be specified through the card reader, SPO, or remote terminals. For simplicity, the following discussion on file security is only in the context of batch processing.

By executing SYSTEM/MAKEUSER, the file (SYSTEM)/USERCODE can be created or modified. Refer to the section on SYSTEM/MAKEUSER for detailed information on program operation. (SYSTEM)/USERCODE contains information on all valid usercode/password combinations, including default pack-id, priority, and security information. This file can be accessed only by SYSTEM/MAKEUSER and the MCP. Consequently, all access to (SYSTEM)/USERCODE may be prevented if SYSTEM/MAKEUSER is not on disk.

Once the (SYSTEM)/USERCODE file has been created, the next step in creating secured files is the execution of a program that creates files "under" a usercode. This is done by prefixing the EXECUTE command with a USER command containing a valid usercode/password. For example:

```
? USER PAYROLL/ACCT EXECUTE CHECK/WRITER
```

In this example, the MCP executes the program CHECK/WRITER and remembers the usercode/password pair (PAYROLL/ACCT) associated with it.

Operation of File Security

To fully understand the impact of file security on system operation, consider the following examples (in all cases, assume that the program CHECK/WRITER is executed under the usercode PAYROLL/ACCT):

a. The program CHECK/WRITER opens a new disk file labeled CHECKS, writes records into the file, then closes it with LOCK (RELEASE in COBOL) to enter it into the disk directory. When this happens, the MCP places into the directory the file-name (PAYROLL)/CHECKS. This file is secured and can only be accessed by programs run under the usercode PAYROLL. To perform library maintenance on this file, the input command must be preceded by the usercode and password. Thus, the following messages are valid:

   ```
   USER PAYROLL/ACCT REMOVE CHECKS
   USER PAYROLL/ACCT CHANGE CHECKS TO PAYCHECK
   USER PAYROLL/ACCT DUMP TO BACKUP =/=
   ```

   The first message removes the file (PAYROLL)/CHECKS, the second message changes the file (PAYROLL)/CHECKS to (PAYROLL)/PAYCHECK, and the third message dumps all files with the usercode of PAYROLL (i.e., (PAYROLL)/=) to a library tape.

   The following messages are invalid:

   ```
   REMOVE CHECKS
   REMOVE (PAYROLL)/CHECKS
   PD CHECKS
   ```

   The first message cannot find the file (PAYROLL)/CHECKS, and hence does not remove it (however, another file named CHECKS is removed if it exists on disk). The second message is rejected by the MCP because it attempts to remove a secured file. The third message acts in a manner similar to the first message, and cannot find the file (PAYROLL)/CHECKS.

   Control commands that are ZIPped by programs or entered through the card reader must also be prefixed by a USER command with the appropriate usercode/password pair if the commands access secured files:

b. If another program is executed without the usercode PAYROLL (either a different usercode or no usercode) and it attempts to access the file (PAYROLL)/CHECKS, the MCP disallows the request.

c. In some instances it may be desirable to create secured files that can be accessed by other usercodes. This can be done by designating the file as PUBLIC. In the previous examples, the file (PAYROLL)/CHECKS is made PRIVATE by default and no unauthorized user can access it. If (PAYROLL)/CHECKS is a PUBLIC file, then any program executed with a different usercode (or no usercode) can access this file by referencing the full file-name, (PAYROLL)/CHECKS.

When a program attempts to open this file as INPUT or INPUT/OUTPUT, the MCP checks to see if the file is PUBLIC and, if it is, allows access. The program can read the file, write into the file (if opened INPUT/OUTPUT), and then close it. (PAYROLL/CHECKS then contains all updates made to it by the program.

d. There are three ways of designating the security for a file, as follows:

1. By using the PROTECTION attribute of the FILE statement. For example:

   ? USER PAYROLL/ACCT EXECUTE CHECK/WRITER
   ? FILE CHECKS PROTECTION <security>;

   The <security> may be DEFAULT, PUBLIC, or PRIVATE. DEFAULT specifies that the security defined for the usercode in (SYSTEM)/USERCODE is to be used.

2. After a file has been created and locked into the disk directory, its security may be changed by modifying the disk file header. For example:

   USER PAYROLL/ACCT MH CHECKS PTN<security>

   The <security> may be PUBLIC or PRIVATE.

3. SYSTEM/MAKEUSER has an option whereby all files created by a specific usercode may be made PUBLIC.. When creating (SYSTEM)/USERCODE, the keyword PUBLIC can be associated with any usercode/password pair. This tells the MCP to make PUBLIC every new file created under that usercode/password. If PUBLIC is not specified, the default security is PRIVATE.

Currently, only SDL/UPL has source language constructs to allow designation of files as PUBLIC or PRIVATE. File declarations generated by other compilers specify DEFAULT security; that is, whatever security is specified for the usercode/password pair in the (SYSTEM)/USERCODE file.

e. If a file is designated as PUBLIC, then the creator of that file also has the option of controlling the type of I/O access that another user can perform (INPUT, OUTPUT, INPUT/OUTPUT). Thus, if (PAYROLL)/CHECKS is a PUBLIC read-only (INPUT) file, a program running under another usercode (or no usercode) can read the file but cannot write into it.

There are two ways of specifying the allowable I/O access, as follows:

1. By using the PROTECT.IO attribute of the FILE statement. For example:

   ? USER PAYROLL/ACCT EXECUTE CHECK/WRITER
   ? FILE CHECKS PROTECT.IO <access>;

   The <access> may be INPUT, OUTPUT, or I.O to specify read-only, write-only, or read-write, respectively.

2. After a file has been created and locked into the disk directory, its access may be changed by modifying the disk file header. For example:

   USER PAYROLL/ACCT MH CHECKS PIO <access>

The <access> may be INPUT, OUTPUT, or I.O to specify read-only, write-only, or read-write, respectively.

f. If an installation wishes to ignore file security, all programs may be run without usercodes and no noticeable impact. The file (SYSTEM)/USERCODE is not needed, and will never be accessed by the MCP. Two restrictions on file-names apply, as follows:

1. The family-name may not be enclosed in parentheses. A family-name surrounded by parentheses is assumed to be a usercode by the MCP.

2. The family-name may not begin with an asterisk ("*"). The use of the asterisk convention with file security is described below.

g. If a program is executed under a usercode and needs to access non-secured files (those without a usercode in the family-name), the following considerations apply:

1. Existing files on disk having two names can be accessed by a program running under a usercode by specifying the exact name. If, however, a program running under a usercode attempts to close a new file with LOCK that has two names, the MCP discards the file. This is because the family-name portion of the file-name is used by the MCP for storing the usercode. Thus, programs that create files having two names must be modified to use single file-names if they are to be run under a usercode.

   CHECK/WRITER may, therefore, open the output file NEW/INFO (which has two names), but the file may not be closed and entered into the directory. No naming conflicts exist if the file is never locked into the directory. In other words, CHECK/WRITER may open the output file NEW/INFO, write records to it, close it (a "temporary" close; "CLOSE" with no options), reopen it, and close it with PURGE. If, however, it attempts to close the file with LOCK (RELEASE in COBOL), the MCP displays a warning message and discards the file.

2. If CHECK/WRITER attempts to access an existing disk file labeled PAYABLES, the MCP first attempts to find a file labeled (PAYROLL)/PAYABLES. When it is unable to locate such a file on disk, the MCP automatically searches for a file labeled PAYABLES. Therefore, if CHECK/WRITER needs to access the file PAYABLES, it is possible only if there is no file on disk labeled (PAYROLL)/PAYABLES.

   To avoid this restriction, the asterisk ("*") naming convention is used.

   If CHECK/WRITER declares the file-name as *PAYABLES, the MCP does not modify the file-name in any way and searches only for the single-name file labeled PAYABLES. CHECK/WRITER may read and/or write this file and, when it is closed, the file-name remains unchanged.

   A program running under a usercode cannot create a file with a single file-name. The MCP always prefixes such output file-names with the usercode.

h. SYSTEM/MAKEUSER provides a mechanism to associate a default pack-id with any usercode/password pair, subject to the condition that all like usercodes in the (SYSTEM)/USERCODE file must have the same pack-id and security type (PUBLIC or PRIVATE), but different passwords.

If the pack-id specified in the program file declaration is blank, the file is automatically directed to the pack specified in the usercode entry in (SYSTEM)/USERCODE. If a pack-id is included in the file declaration, it overrides the default pack-id specified in (SYSTEM)/USERCODE. For example:

```
? USER PAYROLL/ACCT EXECUTE CHECK/WRITER
? FILE CHECKS PACK.ID PAYR;
```

In the above example, the MCP directs the file labeled CHECKS to the user pack labeled PAYR, creating a file labeled PAYR/(PAYROLL)/CHECKS, irrespective of the default pack-id contained in (SYSTEM)/USERCODE. If, however, a default pack-id of BACKUP is contained in (SYSTEM)/USERCODE and CHECK/WRITER is executed without the FILE statement (the pack-id in the file CHECKS is blank), the MCP directs the file to the default pack (labeled BACKUP) with the name BACKUP/(PAYROLL)/CHECKS.

If a default pack is specified in (SYSTEM)/USERCODE and it is necessary to create the file on systems disk, the asterisk naming convention may be used. For example:

```
? USER PAYROLL/ACCT EXECUTE CHECK/WRITER
? FILE CHECKS NAME *(PAYROLL)/CHECKS;
```

Whenever the MCP detects an asterisk in the first position of the declared family-name, it deletes the asterisk and uses the resulting file-identifier without further modification.

A program running under a usercode cannot create new files without a usercode by using the asterisk naming convention. It is, however, permissible to create a new file on systems disk with the asterisk naming convention used to override the default pack-id (for example, *(PAYROLL)/CHECKS).

i. There are programs such as Message Control Systems (MCS) and other utilities that need to be able to access and create files with any usercode. To permit this, a usercode/password pair can be defined in (SYSTEM)/USERCODE as PRIVILEGED.

In the following examples, assume that the usercode PAYROLL/ACCT is PRIVILEGED (with a default pack-id of PAYR), and that the usercode LEDGER/FILE is non-privileged (with a default pack-id of LEDG). If CHECK/WRITER attempts to create a new file, the MCP modifies the file-name as follows:

Declared file-name	Actual file-name used by MCP
CHECKS	PAYR/(PAYROLL)/CHECKS
*CHECKS	CHECKS
CHECKS/PAYCHECK	PAYR/CHECKS/PAYCHECK
*CHECKS/PAYCHECK	CHECKS/PAYCHECK
(LEDGER)/CHECKS	LEDG/(LEDGER)/CHECKS
*(LEDGER)/CHECKS	(LEDGER)/CHECKS
PAYR/*(LEDGER)/CHECKS	PAYR/(LEDGER)/CHECKS
PAYR/*CHECKS/	PAYR/CHECKS/
PAYR/CHECKS/PAYCHECK	PAYR/CHECKS/PAYCHECK

If CHECK/WRITER attempts to access an existing file, the MCP modifies the file-name as follows:

Declared file-name	Actual file-name used by MCP	
CHECKS	PAYR/(PAYROLL)/CHECKS	(note 1)
CHECKS/PAYCHECK	PAYR/CHECKS/PAYCHECK	(note 1)

All other cases of accessing existing files function as shown above for creating new files.

NOTE

1. If the file cannot be found, the MCP restores the originally declared file-name, then searches the system disk. If the file is then found, access is granted only if the file is PUBLIC and the PROTECT.IO <access> matches the open type.

j. Non-privileged usercodes are restricted in the file-names that can be accessed. For the following examples, assume that the usercode PAYROLL/ACCT is non-privileged (with a default pack-id of PAYR). If CHECK/WRITER attempts to create a new file, the MCP modifies the file-name as follows:

Declared file-name	Actual file-name used by MCP
CHECKS	PAYR/(PAYROLL)/CHECKS
*(PAYROLL)/CHECKS	(PAYROLL)/CHECKS
BACKUP/*(PAYROLL)/PAYCHECK	BACKUP/(PAYROLL)/PAYCHECK

If CHECK/WRITER attempts to access an existing file, the MCP modifies the file-name as follows:

Declared file-name	Actual file-name used by MCP	
CHECKS	PAYR/(PAYROLL)/CHECKS	(note 1)
*CHECKS	CHECKS	(note 2)
CHECKS/PAYCHECK	PAYR/CHECKS/PAYCHECK	(note 1)
*CHECKS/PAYCHECK	CHECKS/PAYCHECK	(note 2)
(PAYROLL)CHECKS	PAYR/(PAYROLL)/CHECKS	(note 1)
*(PAYROLL)/CHECKS	(PAYROLL)/CHECKS	
BACKUP/(PAYROLL)/CHECKS	BACKUP/(PAYROLL)/CHECKS	
BACKUP/*(PAYROLL)/CHECKS	BACKUP/(PAYROLL)/CHECKS	
(LEDGER)/CHECKS	LEDG/(LEDGER)/CHECKS	(note 2)
*(LEDGER)/CHECKS	(LEDGER)/CHECKS	(note 2)
BACKUP/*(LEDGER)/CHECKS	BACKUP/(LEDGER)/CHECKS	(note 2)

NOTE

1. If the file cannot be found, the MCP restores the originally declared file-name, then searches the system disk. If the file is then found, access is granted only if the file is PUBLIC and the PROTECT.IO <access> matches the open type.

2. Access is granted only if the file is PUBLIC and the PROTECT.IO <access> matches the open type.

k. If a program running under a usercode ZIPs a control statement, the MCP automatically prefixes the control string with the usercode/password of the zipping program. Thus,

ZIP "RE (LEDGER)/CHECKS"

is interpreted by the MCP as

USER PAYROLL/ACCT RE (LEDGER)/CHECKS

and is disallowed unless the usercode PAYROLL/ACCT is PRIVILEGED.

If the zipping program includes a usercode/password in the control string, this usercode overrides the one added by the MCP. Thus,

ZIP USER LEDGER/FILE RE CHECKS

removes the file, even if the usercode PAYROLL/ACCT is non-privileged. This allows one non-privileged program to remove the files of another, but only if the correct usercode/password combination is known to the zipping program.

l. The usercode/password information must be prefixed to a SPO input message if the command results in modification or removal of a secured disk file. The following input messages are affected by this rule:

CHANGE
COMPILE
EXECUTE
MH
MODIFY
QF
REMOVE

Example:

Invalid Message	Valid Message
EX (LEDGER)/XXX	USER (LEDGER)/FILE EX (LEDGER)/XXX (or) USER (LEDGER)/FILE EX XXX
CH (LEDGER)/A TO X	USER (LEDGER)/FILE CH A TO X
US (LEDGER)/FILE CH A TO *X	US (SITE)/X CH (LEDGER)/A TO *X (if SITE/X is PRIVILEGED)

The commands KA and KP can be applied to any file without usercode/password prefixed. For example:

KA (LEDGER)/A
KA (PAYROLL)/=
KP (LEDGER)/A

In addition, it is not necessary to prefix a PD message with a usercode/password combination, although the MCP response will appear somewhat differently. For example:

PD (LEDGER)/A
PD= "(LEDGER)/A"

is equivalent to

USER (LEDGER)/FILE PD A
PD= "A"

m. Backup files created by programs running under a usercode have a naming convention that is different from the non-secured backup file naming convention. For example, backup files created by CHECK/WRITER have the following names (assume a default pack-id of PAYR):

Printer backup: PAYR/(PAYROLL)/#<integer>
Punch backup: PAYR/(PAYROLL)/%<integer>

To print, punch, remove, or display secured backup files, the PB, RB, and BF commands must be preceded by the appropriate usercode and password. For example:

Message	Action
USER PAYROLL/ACCT PB 15	prints PAYR/(PAYROLL)/#15 (or) punches PAYR/(PAYROLL)/%15

Message	Action
USER PAYROLL/ACCT BF PRT/=	displays all printer backup files on pack PAYR with usercode PAYROLL
USER PAYROLL/ACCT RB =/=	removes all backup files on pack PAYR with usercode PAYROLL

NOTE

The percent sign ("%") is used by the MCP to terminate scanning of input messages. Thus, any attempt to reference individual secured punch backup files in input messages must have the file-id surrounded by quotes. For example:

KA (PAYROLL)/"%15"
US PAYROLL/ACCT PD "%15"

n. The usercode/password convention is applicable to LOAD, DUMP, ADD, and UNLOAD constructs. For example, the message

USER PAYROLL/ACCT DUMP TO SYSTEM =/=

dumps (PAYROLL)/= from the default pack (PAYR) to the library tape labeled SYSTEM. The message

USER PAYROLL/ACCT LOAD FROM SYSTEM =/=

loads (PAYROLL)/= from the library tape labeled SYSTEM to the default pack (PAYR). However, the message

USER PAYROLL/ACCT LOAD TO BACKUP FROM SYSTEM =/=

loads (PAYROLL)/= from the library tape labeled SYSTEM to the pack labeled BACKUP. If the message

LOAD FROM SYSTEM =/=

is entered for a library tape containing both secured and non-secured files, the non-secured files are loaded to the system disk and the secured files are loaded to the default packs as specified in the (SYSTEM)/USERCODE file.

o. Programs executed under a usercode and password are also subject to the default pack-id specified in the (SYSTEM)/USERCODE file. Assume that the default pack-id for the usercode PAYROLL/ACCT is PAYR and consider the following examples:

USER PAYROLL/ACCT EXECUTE CHECK/WRITER

The MCP first searches for the program file PAYR/CHECK/WRITER and, if found, executes it. If the program cannot be found on the default pack, the system disk is then searched. If program files of the same name exist on both the default pack and the system disk, the one on the system disk will never be located. To overcome this restriction, the asterisk naming convention is used:

USER PAYROLL/ACCT EXECUTE *CHECK/WRITER

The same problems occur when the program being executed is identified by a single file name. For example:

USER PAYROLL/ACCT EXECUTE PAYCHECKS

In this case, the MCP first attempts to locate the program file labeled PAYR/(PAYROLL)/ PAYCHECKS. If it cannot be found, the original name is restored, and the program is searched for on system disk. Again, the asterisk naming convention may be used to avoid the first search.

MCP OPTIONS

The MCP will perform certain functions based on the settings of various options. The system operator can use the SO input message to set an option, or the RO message to reset an option, except in the case of the LOG and SPOL options which are independently set with the SL message.

At COLDSTART all of the MCP options with the exception of DATE, TIME, DUMP, BOJ, and EOJ are reset and must be set if desired as part of the MCP's operations.

The DATE and TIME options are set automatically at COLD START time. The date and time must be entered after CLEAR/START before the MCP will allow programs to execute. However, these options may be reset, thereby making it unnecessary to enter the date and time after each CLEAR/START. After a CLEAR/START, the MCP options remain in the same state, set or reset, as they were before the CLEAR/ START was performed.

The following is a list of the available MCP options:

BOJ	DISP	LIB	PBT	SQRM
CHRG	DMS	LOG	RMOV	TERM
CLOS	DUMP	MEM	RMSG	TIME
DATE	EOJ	OPEN	SCHM	TRMD
DBUG	LAB	PBD	SPOL	ZIP

The MCP options are defined in the following paragraphs.

BOJ

The BOJ option specifies that a Beginning-of-Job message be displayed each time the MCP initiates an executable object program.

CHRG

The CHRG option requires that all program executions be accompanied by a charge number which will be entered in the log. Once set (by the SO input message), the CHRG option cannot be reset.

CLOS

The CLOS option specifies that a "file-id CLOSED . . ." message be displayed each time an object program closes a file.

DATE

The DATE option is set at COLDSTART and specifies that the "**DR PLEASE" message be displayed at CLEAR/START. When the "**DR PLEASE" message is displayed, the system operator must enter the date with the DR input message before program execution may begin.

DBUG

The DBUG option, when set, enables certain additional input messages and MCP functions intended for system software development and debugging. These debugging functions are not required for normal system operation.

DISP

The DISP option causes the MCP to include the code segment/displacement or page/segment/displacement information in the abnormal termination message for a program. When this option is reset, the terminal-reference information is not printed.

DMS

The DMS (Data Management System) option retains in memory the GISMO code segment containing the DMS search operator at CLEAR/START time. The DMS option must be set to cause the presence of the overlay and must be used for Data Management programs. A CLEAR/START is required by the MCP after the option is set or reset.

DUMP

The DUMP option must be set in order to dump system memory. If the DUMP option is reset, SYSTEM/DUMPFILE will be removed from disk and the space made available to the system. Any attempt to dump system memory (not DM or DP) will be ignored if the DUMP option is reset.

EOJ

The EOJ option specifies that an End-of-Job message be displayed each time an object program reaches normal End-of-Job.

LAB

The LAB option causes the MCP to display a tape label-name when a BOT (Beginning-of-Tape) is sensed. The character set for a Train Printer will also be displayed.

LIB

The LIB option causes the MCP to display library maintenance actions performed on disk files. The message displayed on the console printer can be one of the following:

file-dientifier	REMOVED
file-identifier	CHANGED TO file-identifier
file-identifier	LOADED
file-identifier	DUMPED

LOG

The LOG option will request the MCP to keep a log of all program executions on disk. See the LG, SL, and TL input messages for actions pertaining to the LOG. The LOG option cannot be referenced by the RO or SO input messages.

MEM

When reset, the MEM option will inhibit any messages from being displayed by the MCP regarding insufficient memory conditions.

OPEN

The OPEN option specifies that a "file-identifier OPENED . . ." message be displayed each time an object program opens a file.

PBD

The PBD option specifies that output files assigned to a printer or card punch will be diverted to a disk backup file if the required output device is not available when the object program tries to open that file. The file must be declared to allow backup output to disk.

PBT

The PBT option specifies that output files assigned to a printer or card punch will be diverted to a tape backup file if the required output device is not available when the object program tries to open that file. The file must be declared to allow backup output to tape.

NOTE

If both the PBD and the PBT options are set, backup will go to tape if a unit is available; if not, the backup will go to disk.

RMOV

The RMOV option if set will automatically remove the old file in "DUPLICATE FILE ON DISK" situations as though an RM message had been typed in by the system operator.

RMSG

The RMSG option, when set, causes MCP output messages directed to remote terminals to also be displayed on the host console printer or display. Remote messages are not displayed at the host console if the RMSG option is reset, except for error messages that may require the attention of the system operator.

SCHM

The SCHM option causes the MCP to display a message when a program is placed in the schedule. The message has the following format:

job-number program-name NEEDS integer KB, SCHED PR = schedule-priority,
IN FOR hh:mm:ss.t, number-of-levels DEEP IN ACTIVE SCHEDULE

SPOL

The SPOL option requests that the MCP maintain a log on disk of all input control messages, and input and output SPO messages. Refer to the LG, SL, and TL input messages for information pertaining to the SPOLOG. The SPOL option cannot be referenced by the RO or SO input messages.

SQRM

The SQRM (SMCP Queue and Remote Messages) option, when set, causes the code in the MICRO.MCP that handles queue and remote messages to be bypassed. This results in all such communicates being processed exclusively by the SMCP. When the SQRM option is reset, queue and remote messages are generally handled by the MICRO.MCP; however, the code from both MCPs may be resident in memory at certain times in order to handle exception conditions. In cases where system usage is extremely heavy and memory space is limited, an increase in throughput can be realized by setting the SQRM option.

TERM

The TERM option specifies that the MCP automatically discontinue processing (DS) of a program when an error condition is encountered. If an error condition occurs and it is necessary to obtain a memory dump of the program, the TERM option should be reset, or the TRMD option should be set.

TIME

The TIME option is set at COLDSTART and specifies that the "**TR PLEASE" message be displayed at CLEAR/START. When the "**TR PLEASE" message is displayed, the system operator must enter the time with the TR input message before program execution may begin.

TRMD

The TRMD option specifies that the MCP automatically dump memory and discontinue processing (DP) a program when an error condition is encountered. If both TERM and TRMD are set, the TRMD option takes precedence.

ZIP

The ZIP option when set will display on the console printer all programmatic ZIP statements made to the MCP.

MCP-OPERATOR INTERFACE

Control Instructions

The Master Control Program is directed to perform particular actions by the system operator through the use of control instructions.

Control instructions may be supplied to the Master Control Program by punched cards, the console printer, the console display, or programmatically through the usage of ZIP statements in an executing program.

There are seven major types of control instructions:

a. File Security Instructions

b. Library Maintenance Instructions

c. Program Control Instructions

d. Program Control Instruction Attributes

e. File Parameter Instructions

f. System Control Instructions

g. Job Spawning Instructions

The following rules apply to all control instructions supplied to the MCP:

a. If the special character percent (%) appears in a control instruction, all information following the % is ignored for control purposes. This allows comments to be present on control cards.

b. The appearance of the "less than" (<) sign in a control message causes the MCP to backspace its pointer one position for each < sign while scanning the control instruction. This allows correction of mistakes without requiring that the entire message be re-entered. Even though this technique is intended mainly for messages entered from the console printer, it works with control instructions entered on punched cards as well. This use of the "less than" sign does not work with control instructions entered through the console display or through ZIP statements. The "less than" sign may not be used for any other purpose.

c. Any program-name or file-identifier which contains the special characters listed below must be enclosed in quotes.

;	semicolon
,	comma
=	equal size
/	slash
	blank or space
"	quote mark
@	at sign
%	percent sign

Any special characters not contained in the above list do not require quote marks to enclose the identifier. The < sign may not appear in an identifier.

Examples:

"FILE%001"
"%3"/"%ABC="
"/XYZ"
SDL.INTRIN/#000000001

The slash in the second example above separates the family-name from the file-name and is not enclosed in quotes.

In the third example, the slash is part of the family-name and is, therefore, enclosed in quote marks.

In the last example the pound sign (#) is not listed as a special character, so the identifier need not be enclosed with quote marks.

d. All control instructions are described on the following pages under headings which might indicate that each of them must consist of a separate card. This is not necessarily so; if the text of one control instruction is delimited by a space then this is considered the "logical end" of that control instruction.

Sources of Control Instructions

PUNCHED CARDS

If punched cards are used to communicate a control instruction to the MCP, the following rules apply:

a. Column 1 of the first control card must contain an invalid character (80-column cards) or a question mark (96-column cards). An invalid character or question mark cannot appear in any other column. The next 71 columns of the card can contain control instructions in free-field format. Control information is limited to the first 72 columns of the control card.

b. Control instructions can be contained on more than one card; however, control words cannot be split and continued to another card. The invalid character in column one is optional on continuation cards.

CONSOLE PRINTER

Control instructions may be communicated to the MCP from the console printer by the following procedure:

a. Press the INPUT REQUEST button on the console printer.

b. Wait for the READY indicator to light.

c. Enter the control instruction from the console printer. If more than one line is required, proceed to step d; otherwise, proceed to step e.

d. If more than one line is required for the message:

   1. Press the LINE FEED button.

   2. Press the CARRIAGE RETURN button.

   3. Press the END OF MESSAGE button.

   4. Return to step b.

e. Press the END OF MESSAGE button.

If there are errors that cannot be corrected by using the "less than" sign, press the ERROR button on the console printer. The entire message (including all lines of multiple-line messages) are discarded by the MCP; an exclamation point (!) is typed to aid in identification, and the READY indicator relights. The message can then be re-entered.

Messages entered from the console printer are subject to the following rules:

a. The "less than" sign will not correct any message entered prior to the current line. Errors in previous lines of multiple-line input messages cannot be corrected by using the "less than" sign.

b. A blank is implied at the end of each line when multiple-line messages are entered; therefore, words may not be split between lines on the console printer.

CONSOLE DISPLAY

Control instructions can be communicated to the MCP from the console display by the following procedure:

a. Place the unit in the LOCAL mode.

b. Set the cursor to the HOME position.

c. If the input message does not fit within the space provided at the top of the screen, use the KB INP.LINES command to increase the reserved space as necessary. There are no "continuation" lines allowed on the console display as on the console printer.

d. Enter the control instruction. Errors can be corrected directly by use of the cursor-positioning keys; the "less than" sign is not valid for error-correction on the console display.

e. When the entire message has been entered, press the ETX key to write an ETX (⧗) on the screen, and the cursor is placed at the HOME position.

f. Press the XMT key to alert the MCP to the presence of the input message.

When the MCP has read the input message, the console printer is left in the receive (RCV) mode, anticipating an MCP response or other output message. If it is necessary to view the screen for any length of time, place the console display in LOCAL mode so that the MCP is not able to change the display. The operator should always leave the console display in RCV mode when not actually entering an input message.

Console Display Scrolling

Because it is often necessary to view messages which are no longer displayed on the screen, "scrolling" the display backwards and forwards is possible. This is accomplished by entering a single-digit integer, as follows:

Integer	Action
0 or ETX	Return display to most current screen.
1	Scroll backward one screen-full of messages from the one currently being displayed.
2	Scroll forward one screen-full of messages from the one currently being displayed. If entered while the display is on the most current screen, the display will be blank.
3	Scroll backward a few lines (one disk sector in the SPO QUEUE) from the screen currently being displayed. This is useful if a message is split between two screens.
4	Scroll forward a few lines (one disk sector in the SPO QUEUE) from the screen currently being displayed. If entered while the display is on the most current screen, part of the display will be blank.
5 - 9	Same as 0 or ETX.

While scrolling, any MCP output message causes the display to return to the most current screen. Entering 0, ETX, or any input message also causes the display to return to the most current screen.

ZIP

Control instructions can be communicated to the MCP by the use of a ZIP statement in an executing program. The ZIP statement in the program must reference a defined data area where the control statement is located. Refer to the appropriate language reference manual for specific syntax regarding the ZIP statement.

Generic Terms

A number of generic terms are used within this manual to describe the syntax of input and output messages. These terms are defined as follows:

a. identifier: A word consisting of from one to ten alphabetic, numeric, or special characters in any combination.

b. disk-pack-id (dp-id): An identifier which is the name of a disk pack or cartridge.

c. family–name: An identifier which is a file name, or the name given to identify a main file with sub-directory entries.

d. program–name: A file-identifier which is the name of a program.

e. compiler–name: A file-identifier which is the name of a compiler.

f. interpreter–name: A file-identifier which is the name of an interpreter.

g. unit–mnemonic: A name which consists of from one to six characters, used to identify a peripheral device.

unit–mnemonic	Device
CDx	Card Reader/Punch
CPx	Card Punch
CRx	Card Reader
CSx	Magnetic Tape Cassette
DCx	Disk Cartridge
DKx	Head-per-Track Disk
DPx	Disk Pack
FDx	Diskette ("Floppy Disk")
LPx	Line Printer
MTx	Magnetic Tape
PPx	Paper Tape Punch
PRx	Paper Tape Reader
SPO	Console Printer
SRx	MICR Reader-Sorter

The "x" notation represents an alpha character which distinguishes multiple units of the same type. For example two Line Printers would have mnemonic names of LPA and LPB.

h. system disk: A disk pack or cartridge that is initialized as a system type pack. A system pack is under the control of the MCP and one or more must be present on the system for the MCP to function. Head-per-track disk is always considered system disk.

i. removable disk: A disk pack or cartridge that can be removed from the system during operations. The MCP does not need removable disk packs in order to function.

j. file–identifier: All disk-file-identifiers used on the system must be unique, therefore, there can be no duplication of file names. Throughout this manual "file–identifier" will incorporate all the combinations allowed for a file–identifier, such as the following:

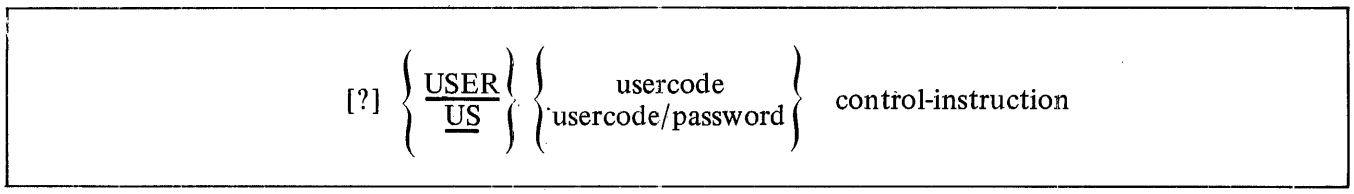
file–identifier
family–name/file–identifier
dp–id/family–name/file–identifier
dp–id/file–identifier/

FILE SECURITY INSTRUCTIONS

USER

The USER statement provides a means of invoking the MCP file security mechanism and its associated naming convention.

The format of the USER statement is:



The USER statement may be abbreviated as US.

The USER statement causes the MCP to verify the usercode and password against the directory of valid usercodes and passwords maintained in the (SYSTEM)/USERCODE file on system disk. The validated usercode is carried with the specified control instruction string, and is used to apply the MCP file security naming conventions to any subsequent file-identifier references.

Any programs executed under a usercode also carry the validated usercode along with them and thus use the MCP file security naming conventions for all files referenced during their execution.

Examples:

```
USER SITE/PRIV PD =/=
  PD= "FILE1"
  PD= "FILE2"
  PD= "FILE3"
PD (SITE)/=
 PD= "(SITE)/FILE1"
 PD= "(SITE)/FILE2"
 PD= "(SITE)/FILE3"

US STUDENT/JK EX TESTPROG PR 3
 (STUDENT) TESTPROG =5 BOJ. . . .

PD (FINANCE)/CHECKREG
 PD= "(FINANCE)/CHECKREG"
RE (FINANCE)/CHECKREG
 "(FINANCE)/CHECKREG" NOT REMOVED–SECURITY ERROR
US FINANCE/VP RE CHECKREG
 "CHECKREG" REMOVED
```

LIBRARY MAINTENANCE INSTRUCTIONS

CHANGE

The CHANGE statement changes the file-identifier of a disk file, causing the file to be referenced by the new file identifier.

The format of a CHANGE statement is:

[?] { CHANGE / CH } file-identifier-1 TO file-identifier-2 [, file-identifier-3 TO file-identifier-4]. . . ;

The control word CHANGE may be abbreviated as CH.

Any CHANGE statements affecting more than one file must have the file-identifiers separated by commas.

The CHANGE statement will cause the MCP to change the file-identifier of specified disk files from one name to another. If the file referenced in the CHANGE statement resides on a removable disk, the disk-pack-id must precede the file-identifier in order for the MCP to locate the proper file to change.

```
? CHANGE ALPHA/BETAONE/ TO ALPHA/BETATWO/
```

If the CHANGE statement is entered and the MCP cannot locate the file or if the file is in use, the following message is displayed on the console printer:

file-identifier NOT CHANGED . . . (reason). . .

The CHANGE statement is not allowed on a Multi-Pack File.

The CHANGE statement may consist of additional cards where two or more "changes" may be made. Termination will occur when a semicolon (;) is detected.

If the CHANGE statement references a PRIVATE file, it must be preceded by a USER statement with the proper usercode/password.

Examples:

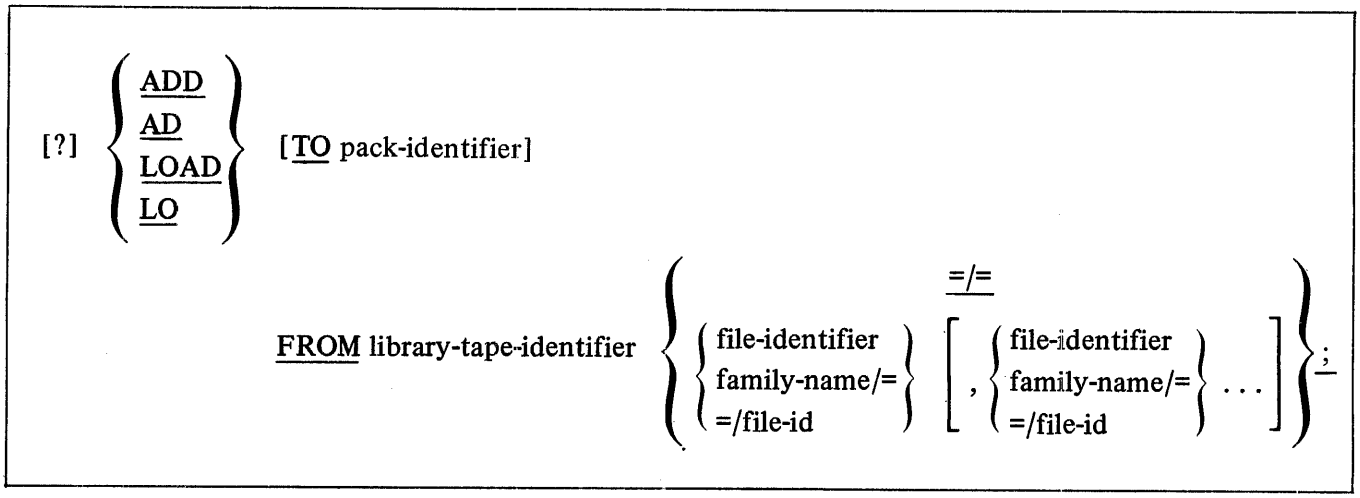
```
CHANGE A/B TO C/D
  "A/B" CHANGED TO "C/D"

? CH A/B C/D, X Y,
? ABC A/B;

CH (FINANCE)/CHECKREG TO (FINANCE)/CHECKS
  "(FINANCE)/CHECKREG" NOT CHANGED–SECURITY ERROR
USER FINANCE/VP CH CHECKREG TO CHECKS
  "CHECKREG" CHANGED TO "CHECKS"
```

ADD, LOAD

The ADD statement will cause a file or files on a LIBRARY tape to be placed on disk only if the file is not already on disk.

The LOAD statement will cause a file or files on a LIBRARY tape to be placed on disk. If the file is already on disk, the old file will be removed.

The format of the ADD and LOAD statement is:



The control words ADD and LOAD may be abbreviated as AD and LO, respectively.

The =/= option causes every file on the tape to be added or loaded.

The family-name/= option causes every file with the specified family-name to be added or loaded.

The =/file-id option causes every file with the specified file-id to be added or loaded.

The ADD or LOAD statements must be preceded by a USER statement with the required usercode/password when it is desired to load PRIVATE files from a library tape.

Examples:

```
?LOAD FROM SYSTEM COBOL,
?RPG, BASIC;

?ADD TO USERPACK FROM LIBTAPE
PAYROLL/=
ACCPAY/=,
MASTER/FILE,
=/REGISTER;
USER PAYROLL LO FROM SYSTEM=/=
```

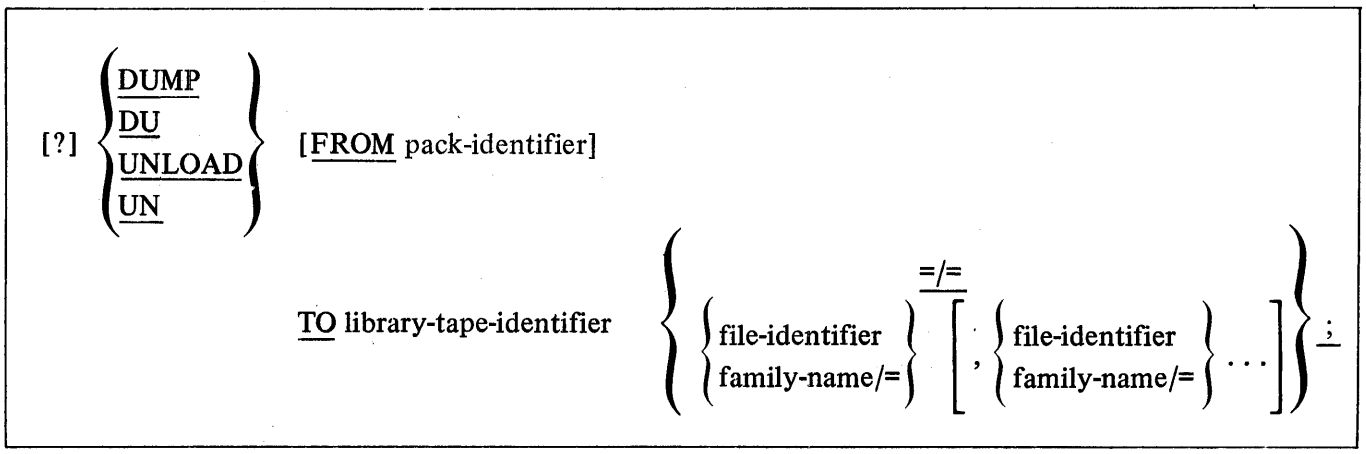
DUMP
UNLOAD

DUMP, UNLOAD

The DUMP statement causes one or more disk files to be placed on a LIBRARY tape. The file is not removed from disk by the dump.

The UNLOAD statement causes one or more disk files to be placed on a LIBRARY tape. The disk file is removed after the successful completion of the UNLOAD.

The format of the DUMP and UNLOAD statement is:



The control words DUMP and UNLOAD may be abbreviated as DU and UN, respectively.

The =/= option indicates that all files on the specified disk are to be dumped or unloaded.

A maximum of 2248 files can be handled with one DUMP or UNLOAD statement.

The DUMP or UNLOAD statements must be preceded by a USER statement with the required usercode/password when it is desired to dump PRIVATE files to a library tape.

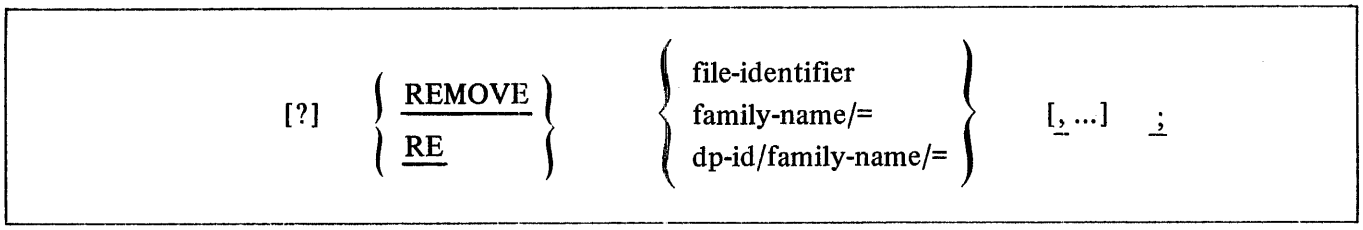
Examples:

```
?DUMP TO SYSTEM
X/Y,
Z/Q,
AAA,COBOL/=,
RPG/REORG,
SDL.INTRIN/=
;

?UNLOAD FROM USER TO BACKUP =/=;

USER FINANCE DUMP TO LIBRARY =/=
```

REMOVE

The REMOVE statement deletes specified files from the disk directory making the file space available to the MCP.

The format of the REMOVE statement is:



The control statement REMOVE may be abbreviated as RE.

The "/=" form will delete the main directory entry and in turn delete all the files in its sub-directory.

The REMOVE statement may delete any number of files. However, any statement affecting more than one file must have the file-identifiers separated by commas.

If the file-identifier referenced in the REMOVE statement resides on a removable disk pack, the disk-pack-id must precede the file-identifier in order for the MCP to locate the correct file. When the disk-pack-id is not included, the MCP assumes that the file resides on a system pack.

The REMOVE statement must be preceded by a USER statement with the required usercode/password when it is desired to remove PRIVATE files from disk.

Once a file has been removed, there is no means of recovering it.

The REMOVE statement may be continued to additional cards with the last "remove" terminated by a semicolon.

Example:

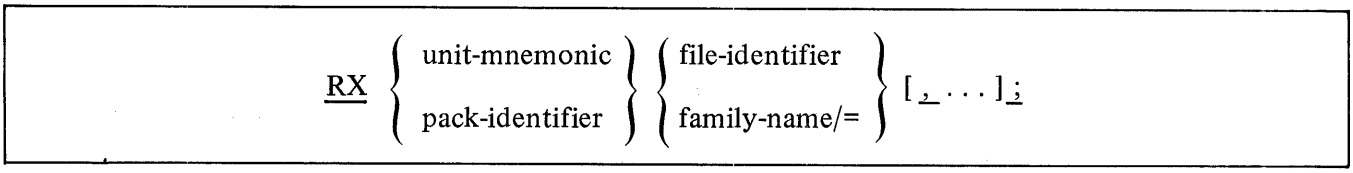
```
? REMOVE A/B
, X, Y,
Z;

USER SITE/X RE FILE1, FILE2
```

RX

RX (Exclusive Remove)

The RX statement deletes all files from the designated disk directory, exclusive of those files listed in the RX statement itself.

The format of the RX statement is:



All files on the disk designated by the unit-mnemonic (for system disks) or pack-identifier (for user packs) that are not specified in the RX message will be removed. Files that are in-use or are marked as "system" files (any file listed in the NAME TABLE) will not be removed, even if they are not included in the RX message file-name list.

Any disk I/O error encountered by the MCP during the remove operation causes the entire RX operation to abort at that point.

The file-identifiers in the list following the RX message must be separated by commas, and must be terminated by a semicolon (;) or a "? END" card (if entered from a card reader). The MCP does not begin the remove operation until the semicolon or "? END" card is detected.

NOTE

If an exception condition occurs on the card reader (for example, a card jam), it is possible to abort the RX operation without the necessity of a CLEAR/START. Simply clear the exception condition from the reader and enter a file-identifier card with invalid syntax (for example, "A/B/C/D").

Examples:

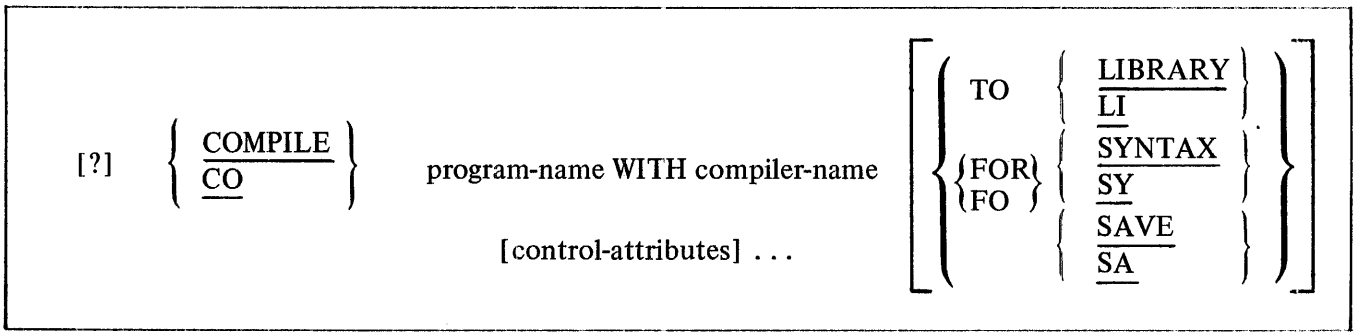
```
RX DPA SYSTEM/=, COBOL/=, RPG/=, PAYROLL/=;

?RX USERPACK FINANCE/=, PAYROLL/CHECKS,
  GENLEDGER/=,
  TESTPROG,
  MASTER/FILE, CUSTOMER/HISTORY
?END
```

PROGRAM CONTROL INSTRUCTIONS

COMPILE

The COMPILE statement designates the compiler to be used, and the type of compilation to be performed.

The format of the COMPILE statement is:



The COMPILE statement may be abbreviated as CO.

The compiler control statement must be the first statement in a set of control statements. The COMPILE statement has four options:

1. COMPILE
2. COMPILE TO LIBRARY
3. COMPILE SAVE
4. COMPILE FOR SYNTAX

The COMPILE is a "compile and go" operation. Providing the compilation is error-free, the MCP schedules the object program for execution. The program will not be entered into the disk directory, and must be recompiled to be used again. The "compile and go" is the default option of the COMPILE statement.

The COMPILE TO LIBRARY will leave the program object file on disk and will enter the program-name into the disk directory after an error-free compilation. The program is not scheduled for execution.

The COMPILE and SAVE combines the execute and library options. The MCP will enter the program-name into the disk directory and will leave the object program file on disk, as well as schedule the program for execution after an error-free compilation. The program remains in the disk directory.

The COMPILE FOR SYNTAX provides a diagnostic listing as the only output. This option does not enter the program-name into the disk directory or leave the program object file on disk. Some uses are as a debugging tool, first time compilation, or a new source listing.

DYNAMIC

The DYNAMIC statement will modify the working copy of a program that is already in the mix or scheduled for execution.

The format of the DYNAMIC statement is:

```
[?]  { DYNAMIC }  job-number  [control-attributes] . . .
     { DY      }
```

The DYNAMIC control word may be abbreviated as DY.

Any change that can be made by using the MODIFY statement is valid for the DYNAMIC statement; however, only the working copy of the program will be altered.

EXECUTE

The EXECUTE statement instructs the MCP to call a program from the library for subsequent execution.

The format of the EXECUTE statement is:

```
[?]  { EXECUTE }  program-name [control-attributes] . . .
     { EX      }
```

The EXECUTE control word may be abbreviated as EX.

The EXECUTE control statement must be the first statement in a set of control statements pertaining to the execution of a program.

If the program referenced in the EXECUTE statement resides on a removable disk cartridge or disk pack, the disk-pack-id must be part of the program-name in order for the MCP to locate the correct file.

Example:

```
? EXECUTE TEST
? DATA file-identifier
  (data cards)
? END
```

This example shows that a program named TEST is to be called out of the library on disk and executed. One of the files in the program TEST assigned as a card file is identified by the DATA control card. If the program does not require a card file, only the EXECUTE control statement is necessary and can be entered through the card reader with the "? EXECUTE TEST" or the console printer with the "EX TEST" command.

MODIFY

The MODIFY statement is used to permanently change attributes within a program.

The format of a MODIFY statement is:

```
[?]  { MODIFY }  program-name  [control-attributes] ...
     { MO     }
```

The MODIFY control statement may be abbreviated as MO.

The MODIFY statement has the same syntax as the EXECUTE statement, but does not execute the program.

Example:

```
? MODIFY  A/B PRIORITY 6
```

The above example will permanently change the priority of program A/B to six.

The MODIFY statement can be used to change the following attributes:

CHARGE
DYNAMIC.SPACES
FILE
FREEZE
INTERPRETER
INTRINSIC.NAME
INTRINSIC.DIRECTORY
MEMORY
OVERRIDE
PRIORITY
SCHEDULE.PRIORITY
SWITCH
UNFREEZE
UNOVERRIDE
VIRTUAL.DISK

PROGRAM CONTROL INSTRUCTION ATTRIBUTES

AFTER

The AFTER attribute is used to conditionally schedule a program after the termination of another program (by program-name).

The format of the AFTER statement is:

[?] { AFTER / AF } program-name

The AFTER control word may be abbreviated as AF.

Example:

EXECUTE ALPHA AFTER BETA or
EX ALPHA AF BETA

When BETA reaches EOJ, ALPHA will be placed in the ACTIVE SCHEDULE for execution as soon as memory resources are available.

If BETA was not either executing or scheduled when ALPHA was scheduled, ALPHA will remain in the WAITING SCHEDULE until BETA is executed and reaches EOJ, or until FS-ed by the system operator.

AFTER.NUMBER

The AFTER.NUMBER attribute is used to conditionally schedule a program after the termination of another program (by job-number) that is already in the mix or scheduled for execution.

The format of the AFTER.NUMBER statement is:

```
[?]  { AFTER.NUMBER }  job-number
     { AN           }
```

The AFTER.NUMBER control word may be abbreviated as AN.

Example:

EXECUTE ALPHA AFTER.NUMBER 7 or
EX ALPHA AN 7

NOTE

A job-number is assigned by the MCP to every job scheduled for execution on the system. Each job-number is unique and is incremented sequentially from the last COLDSTART.

THEN

THEN

The THEN attribute is used to conditionally schedule execution of a program in relation to another program.

The format of the THEN statement is:

```
[?]  { THEN }
     { TH   }
```

The THEN control word may be abbreviated as TH.

Example:

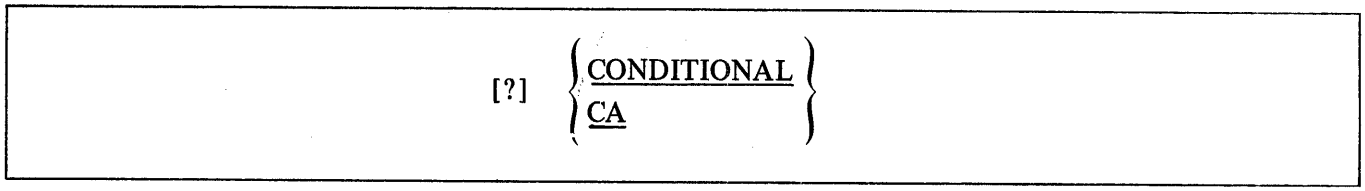
? EXECUTE ALPHA PRIORITY 14 MEMORY 20000 THEN COMPILE BETA COBOL SYNTAX

Program BETA will be executed (compiled) as soon as program ALPHA has terminated.

CONDITIONAL

The CONDITIONAL attribute is used in conjunction with the AFTER, AFTER.NUMBER, and THEN attributes and inhibits the program from being fired-up unless its predecessor successfully reaches EOJ. The CONDITIONAL attribute is a default statement.

The format of the CONDITIONAL statement is:



The CONDITIONAL control statement may be abbreviated as CA.

Examples:

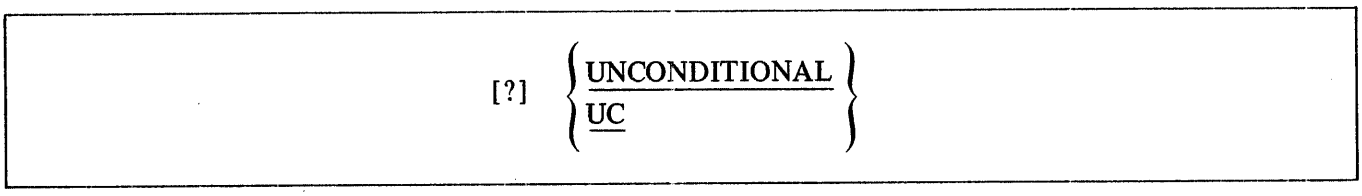
? EXECUTE A/B AFTER C/D CONDITIONAL

? EX A/B AF C/D CA

UNCONDITIONAL

The UNCONDITIONAL attribute is used in conjunction with the AFTER, AFTER.NUMBER, and THEN attributes and forces the program to be fired-up regardless of its predecessor's outcome.

The format of the UNCONDITIONAL statement is:



The UNCONDITIONAL control statement may be abbreviated as UC.

Examples:

? EXECUTE A/B AFTER C/D UNCONDITIONAL
? EX A/B AF C/D UC

? EXECUTE A/B THEN EXECUTE C/D UNCONDITIONAL
EX A/B TH EX C/D UC

CHARGE

The CHARGE attribute is used to insert a charge number into the log record for a program.

The format of a CHARGE statement is:

```
[?]  [OBJ]  { CHARGE }   [=]  integer
            { CG     }
```

The CHARGE control word may be abbreviated as CG.

The integer cannot exceed six digits. If less than six digits are used, leading zeros will be assumed. This number will be carried in the MCP log file for subsequent analysis.

If the MCP's CHRG option is set, the CHARGE statement must be used before a program will be scheduled.

DYNAMIC.SPACES

The DYNAMIC.SPACES statement allows the operator to specify the maximum number of overlays that will ever be present in a program's dynamic memory.

The format of the DYNAMIC.SPACES statement is:

```
[?]  [OBJ]  { DYNAMIC.SPACES }      [=]  integer
            { DS             }
```

The DYNAMIC.SPACES control word may be abbreviated as DS.

The purpose of DYNAMIC.SPACES is to allow dynamic memory space for the Memory Links that will be associated with the overlayable data within a program.

The MCP at run time will assign a value of 10 if the DYNAMIC.SPACES is zero. This attribute is normally used only when the exact memory requirement for a program's data overlays is specified.

For example:

```
? EXECUTE A/B MEMORY = 20000
```

The MCP will assign the program A/B 20000 bits of dynamic memory plus the following:

```
(2 * AVAIL.LINK) + (DYNAMIC.SPACES * IN.USE.LINK)
```

or

```
(2 * 175 BITS) + (DYNAMIC.SPACES * 163 BITS)
```

FILE

The FILE statement may be used to specify various attribute changes for both input and/or output files.

The format of the FILE statement is:

```
[?]  [OBJ]  { FILE }
            { FI   } internal-file-identifier file-attribute-1 [file-attribute-2] . . . ;
```

The control word FILE may be abbreviated as FI.

The FILE statement must have each element within the statement separated by at least one space, and must be terminated with a semicolon or END OF MESSAGE. If more than one card is required for a FILE statement, each of the continuation cards must have a question mark in column 1.

The FILE statement must immediately follow the COMPILE, EXECUTE, DYNAMIC, or MODIFY statement. The MCP modifies the information in a working copy of the program's FILE PARAMETER BLOCK (FPB).

The file-identifier used in the FILE statement must refer to the internal-file-name used in the program that opens the file. For example, if the external file-identifier is to be changed for this run only, the FILE statement would be as follows:

```
? EXECUTE program-name
? FILE internal-file-identifier NAME file-identifier;
```

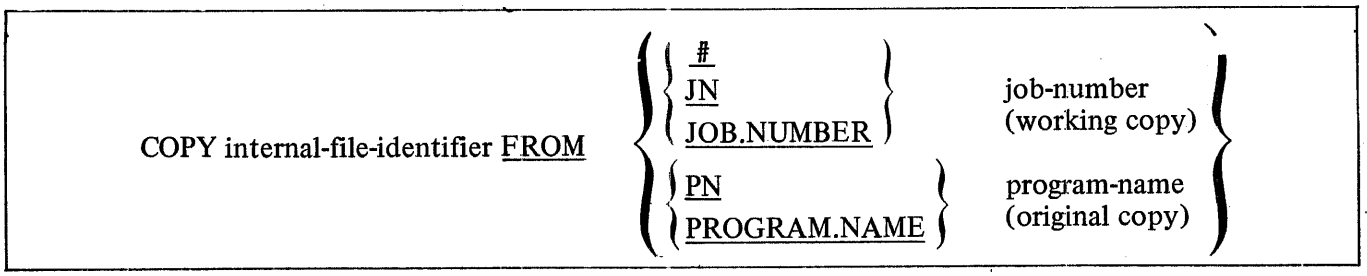
FILE ATTRIBUTES

Following is a list of the file-attributes that may be modified at execution time with the use of a FILE statement.

FILE ATTRIBUTE	FUNCTION
ALLOCATE.AT.OPEN	All of the areas requested by this file will be allocated at the time the file is opened.
AREAS[=] integer	The number of areas assigned to the file at compile time will be altered to the value of the integer. The integer must be between 1 and 105, inclusive.
ASCII	The recording mode of the file will be changed to ASCII.
BACKUP	The output of the file will be allowed to go to backup. This sets BACKUP.DISK and BACKUP.TAPE by implication.
BACKUP.DISK	If the PBD option is set, the output of the file will be allowed to go to disk backup.
BACKUP.TAPE	If the PBT option is set, the output of the file will be allowed to go to tape backup.

FILE ATTRIBUTE	FUNCTION
BCL	The recording mode of the file will be changed to BCL.
BINARY	The recording mode of the file will be changed to BINARY (80-column card and paper tape only).
BLOCKS.AREA[=] integer	Assign integer blocks (physical records) to each disk area.
BUFFERS[=] integer	The number of buffers assigned to the file will be altered to the value of the integer. The integer specified must not be zero. For QUEUE files, BUFFERS specifies the maximum number of messages allowed in memory at any one time. For REMOTE files, BUFFERS specifies the maximum number of messages in the input queue allowed in memory (the output queue attributes are declared in the network controller).
COPY	The entire File Parameter Block except the internal file identifier of one file will be copied to the receiving file's File Parameter Block. The internal file-identifier will not be changed.

SYNTAX



DEFAULT	Override the declared block and record sizes and use the block and record sizes specified in the disk file header or tape label instead. (Input disk and labeled B 7800/B 7700/B 6800/ B 6700/B 1800/B 1700 tape files only.)
DELAYED.RANDOM	The file-access method will be changed to DELAYED RANDOM.
DRIVE[=]integer	The file will be directed to the drive or EU specified by the integer. The drive must be a system disk. The integer must be a positive number from 0 to 15.
EBCDIC	The recording mode of the file will be changed to EBCDIC.
EOP	Indicates that the program requests end-of-page reporting from the MCP upon sensing a channel 12 punch on a line printer file.
EU[=]integer	Same as DRIVE.
EVEN	The file will be changed to even parity.

FILE ATTRIBUTE	FUNCTION
FILE.TYPE[=] { DATA / CODE / INTERPRETER / PSR.DECK / INTRINSIC }	An output disk file will be assigned the specified type when it is closed and has been entered in the disk directory.
FORMS	The program will be suspended and the MCP will display a message for the operator to load special forms in the device (printer or punch) before the file is opened.
HARDWARE	A printer or punch file will be allowed to go to the hardware device assigned.
HEADER	For REMOTE files only. Specifies that a message header is present on remote reads and writes.
INPUT.SELECTIVITY	For QUEUE files only. Specifies that the file is made up of multiple subqueues (a queue-file-family). The number of subqueues can be specified by the Q.FAMILY.SIZE attribute.
INVALID.CHARACTERS [=] integer	The integer may contain a value of 0, 1, 2, or 3, and determines the course of action for invalid characters output to a train printer.

0 = Report all lines that contain invalid characters. The following console message will be printed for each occurrence:

```
FILE file-name IS PRINTING
INVALID CHARACTERS ON LPx.
```

1 = Report all lines that contain invalid characters and stop the program at that point.

2 = Report once that the file is printing invalid characters. The following console message will be printed.

```
FILE file-name IS PRINTING
INVALID CHARACTERS ON LPx.
(one-time warning)
```

3 = Do not notify operator of invalid character output.

LABEL.TYPE[=] integer — The integer values and associated label types are as follows:

Integer Value	Label Type
0	ANSI
1	Unlabeled
2	Burroughs

LOCK — The file will be LOCKED, if still open, at program termination (DS or normal EOJ).

FILE ATTRIBUTE	FUNCTION
MAXIMUM.BLOCK.SIZE[=] integer	Fixed block size to be used for variable length records.
MULTI.PACK	The disk file is considered a multi-pack file.
NAME[=] file-identifier	The external file-identifier or disk pack-id will be changed to the value of file-identifier. If only the disk pack-id is to be changed the PACK.ID attribute may be used.
{NO / NOT} file-attribute	When this option is used it will negate the file-attribute following the word NO or NOT. For example, a file assigned to go strictly to backup could be changed to go to the printer by entering a NO BACKUP file statement. The following is a list of file-attributes that the NO or NOT statement can negate. ALLOCATE AT.OPEN BACKUP BACKUP.DISK BACKUP.TAPE DEFAULT EOP FORMS HARDWARE HEADER INPUT.SELECTIVITY LOCK MULTI.PACK OPTIONAL PSEUDO TRANSLATE USER.BACKUP.NAME VARIABLE WORK.FILE
NUMBER.STATIONS[=] integer	For REMOTE files only. Specifies the maximum number of stations that can be attached to the file.
ODD	The file will be changed to ODD parity.
OPTIONAL	Designates the file as optional.
PACK.ID[=] disk-pack-id	Alter the pack-id.
PROTECTION [=] {DEFAULT / PUBLIC / PRIVATE}	The security designation for a disk file will be changed to the type specified. A new disk file will be given the security specified by the FPB when it is closed and entered into the disk directory.
PROTECT.IO [=] {INPUT / OUTPUT / I.O}	The security I/O designation for a disk file will be changed to the type specified. A new disk file will be given the security I/O code specified by the FPB when it is closed and entered into the disk directory.

FILE ATTRIBUTE	FUNCTION
PSEUDO	Makes file a pseudo type.
Q.FAMILY.SIZE[=] integer	For QUEUE files only. Specifies the number of subqueues in a queue-file-family (multiple queue file).
Q.MAX.MESSAGES[=] integer	For QUEUE files, specifies the maximum number of messages allowed in the queue at any one time. For REMOTE files, specifies the maximum number of messages allowed in the input queue (the size of the output queue is declared in the network controller).
RANDOM	The file will be changed to a RANDOM access file.
RECORDS.BLOCK[=] integer	The number of logical records per block for a fixed record-length file.
RECORD.SIZE[=] integer	The number of bytes assigned for the logical record will be changed to the value of the integer.
REEL[=] integer	The value of the integer will determine the number of the first reel.
REPETITIONS [=] integer	Specifies (at file creation time) the number of copies of a backup file to be printed or punched by the SYSTEM/BACKUP program. The REPETITIONS value can be overridden by the COPIES option of the PB input message. The integer specified may not be greater than 63. If the REPETITIONS value for a printer or punch file is greater than one (1), the file will be directed to backup automatically by the MCP.
SERIAL	The file is to be processed sequentially.
SAVE[=] integer	A save factor representing the number of days a tape or disk file may be saved.
TRANSLATE	Specifies that the file is to be translated using the MCP "soft-translate" facility. Translation is not allowed on input/output files. The name of the translate file to be used must be specified by the TRANSLATE.NAME attribute.
TRANSLATE.NAME[=] identifier	Specifies the file-id portion of the file to be used, if "soft-translate" is requested. The family-name portion of the name is not specified with this attribute; the name is assumed to be "TRANSLATE." The translate file must have been created using SORT/COLLATE, and must reside on system disk.
UNIT.NAME[=] unit-mnemonic	The file will be directed to the device specified by unit-mnemonic.
USER.BACKUP.NAME	For disk printer and punch backup files. Specifies that the external-file-id be used (NAME attribute) rather than the default backup-file-id (BACKUP.PRT/integer or BACKUP.PCH/integer) when entering the file name into the directory. This attribute is ignored for backup files sent to tape.

FILE ATTRIBUTE	FUNCTION
VARIABLE	The file will be processed using variable length records.
WORK.FILE	Assigns this file as a work file used internally. A work file has the program's job number included as part of the family-name, in an attempt to make the file-identifier unique.

The following list of device attributes may be used to change the input or output device originally assigned to a file.

CASSETTE
CARD.PUNCH
CARD.READER
DATA.RECORDER.80
DISK
DISK.CARTRIDGE
DISK.FILE
DISK.PACK
PAPER.TAPE.PUNCH
PAPER.TAPE.READER
PRINTER
QUEUE
READER.PUNCH.PRINTER
READER.SORTER
READER.96
REMOTE
TAPE
TAPE.NRZ
TAPE.PE
TAPE.7
TAPE.9

FILE ATTRIBUTE ABBREVIATIONS

The following abbreviations may be used to identify the FILE statement attributes.

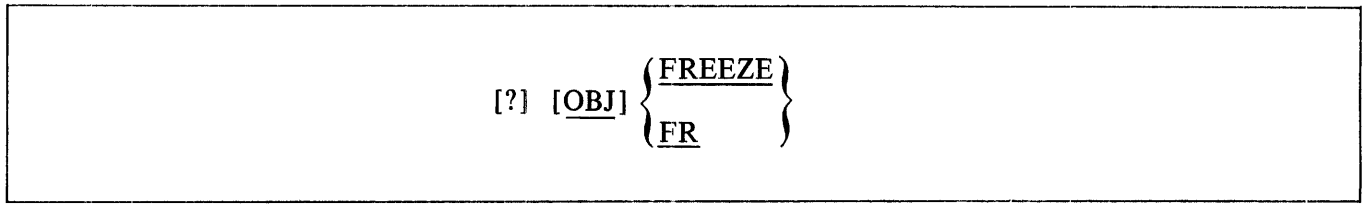
ADVERB	ADV
ALLOCATE.AT.OPEN	ALL
AREAS	ARE
ASCII	ASC
BACKUP	BAC
BACKUP.DISK	BDK
BACKUP.TAPE	BTP
BCL	BCL
BINARY	BIN
BLOCKS.AREA	B.A
BUFFERS	BUF
CARD.PUNCH	CPC
CARD.READER	CRD
CASSETTE	CAS
COPY	CPY
DATA.RECORDER.80	DRC
DELAYED.RANDOM	D.R
DEFAULT	DEF
DISK	DSK
DISK.CARTRIDGE	DCG
DISK.FILE	DFL
DISK.PACK	DPC
DRIVE	DRI
EBCDIC	EBC
EOP	EOP
EU	EU
EVEN	EVN
FILE.TYPE	FTP
FORMS	FMS

HARDWARE	HAR
HEADER	HDR
INPUT.SELECTIVITY	ISL
INTERPRETER	INT
INVALID.CHARACTERS	INV
LABEL.TYPE	LAB
LOCK	LOC
MAXIMUM.BLOCK.SIZE	MAX
MULTI.PACK	MUL
NAME	NAM
NO	NO
NOT	NOT
NUMBER.STATIONS	NST
ODD	ODD
OPTIONAL	OPT
PACK.ID	PID
PAPER.TAPE.PUNCH	PTP
PAPER.TAPE.READER	PTR
PRINTER	PRT
PROTECTION	PTN
PROTECT.IO	PIO
PSEUDO	PSE
QUEUE	QUE
Q.FAMILY.SIZE	QFS
Q.MAX.MESSAGES	QMX
RANDOM	RAN
READER.PUNCH.PRINTER	RPP
READER.SORTER	RSR
READER.96	R96
RECORD.SIZE	RSZ
RECORDS.BLOCK	R.B
REEL	REE
REMOTE	REM
REPETITIONS	REP
SAVE	SAV
SERIAL	SER
TAPE	TAP
TAPE.NRZ	TPN
TAPE.PE	TPE
TAPE.7	TP7
TAPE.9	TP9
TRANSLATE	TRN
TRANSLATE.NAME	TNM
UNIT.NAME	UNI
USER.BACKUP.NAME	U.N
VARIABLE	VAR
WORK.FILE	WFL

FREEZE

The FREEZE control attribute will prohibit rolling a program out to disk at any time during its execution, thereby remaining in the same memory location regardless of the situation until End-of-Job.

The format of the FREEZE statement is:



The FREEZE control word may be abbreviated as FR.

HOLD

The HOLD control attribute allows the system operator to place a program into the waiting schedule prohibiting its execution until it is forced (FS'ed) into the active schedule.

The format of the HOLD statement is:

```
[?]   { HOLD }
      { HO   }
```

The HOLD control word may be abbreviated as HO.

The HOLD attribute may not be used with the MODIFY or DYNAMIC control statements.

INTERPRETER

The INTERPRETER attribute allows selection of a different interpreter for use by a program.

The format of the INTERPRETER statement is:

[?] [OBJ] { INTERPRETER / IN / INTERP } [=] file-identifier

The INTERPRETER control word may be abbreviated as IN or INTERP.

Examples:

```
? EXECUTE ALPHA/BETA INTERPRETER COBOL/INTERP001
? EX X/Y IN CCC/SDL/INTERP3
```

INTRINSIC.NAME

The INTRINSIC.NAME attribute makes it possible to change the family-name of the intrinsic file requested by a program.

The format of the INTRINSIC.NAME statement is:

[?] [OBJ] { INTRINSIC.NAME / IT } [=] intrinsic-identifier

The INTRINSIC.NAME control word may be abbreviated as IT.

The file-id portion of the intrinsic file ("AGGREGATE") may not be changed.

Example:

```
? EXECUTE ALPHA/BETA INTRINSIC.NAME ZZZ.INTRIN
```

or

```
? EX ALPHA/BETA IT ZZZ.INTRIN
```

INTRINSIC.DIRECTORY

The INTRINSIC.DIRECTORY attribute makes it possible to reference intrinsic files from a selected removable disk pack.

The format of the INTRINSIC.DIRECTORY statement is:

[?] [OBJ] { INTRINSIC.DIRECTORY / ID } [=] disk-pack-id

The INTRINSIC.DIRECTORY control word may be abbreviated as ID.

Example:

```
? EX ALPHA/BETA INTRINSIC.DIRECTORY UTILPACKA
```

MEMORY

The MEMORY attribute makes it possible to override the dynamic memory size assigned by the compiler for a given program at execution time.

The format of a MEMORY statement is:

[?] [OBJ] { MEMORY / ME } [=] integer

The MEMORY control word may be abbreviated as ME.

The integer expresses the dynamic memory size in bits.

The program will be terminated if there is not enough dynamic memory assigned to execute.

When the MEMORY statement is used following a compile statement, the memory will be reserved for the compiler, not the program being compiled.

Examples:

```
? COMPILE    program-name    COBOL SYNTAX MEMORY = 50000

                    or

? COMPILE    program-name    COBOL SYNTAX

? MEMORY = 50000
```

Both of the above examples will assign 50,000 bits of dynamic memory for the compiler. The following example will assign 50,000 bits of dynamic memory for the execution of a program.

```
? EXECUTE    program-name    MEMORY = 50000
```

OVERRIDE

The OVERRIDE attribute makes it possible to bypass the compatibility check normally made between a program and its interpreter.

The format of the OVERRIDE statement is:

```
[?] [OBJ] { OVERRIDE }
          { OV       }
```

At BOJ time the MCP performs a compatibility check of a program and its interpreter unless OVERRIDE is specified. The compatibility check consists of the following:

a. Interpreter's HARDWARE.TYPE is U (Universal), or matches the type of the processor (S or M) on which it is running.

b. Interpreter's MCP.LEVEL matches the MCP's LEVEL.

c. Interpreter's GISMO.LEVEL matches GISMO's LEVEL.

d. Interpreter's COMPILER.LEVEL matches the program's COMPILER.LEVEL.

e. Interpreter's ARCHITECTURE (language) matches the program's INTERPRETER.FIRST.NAME.

f. Interpreter has at least every attribute required by the program.

Specification of OVERRIDE does not cause bypassing of the MCP's interpreter name generation process.

UNOVERRIDE resets OVERRIDE, thus causing the compatibility check to be performed.

Examples:

```
EX A/B OVERRIDE

MODIFY TEST OV

DY 275 OV
```

PRIORITY

The PRIORITY attribute specifies the operational priority assigned to a given program.

The format of a PRIORITY statement is:

[?] [OBJ] { PRIORITY / PR } [=] integer

The PRIORITY control word may be abbreviated as PR.

The system operator has the ability to assign program priorities to maximize output and scheduling. Priorities range from zero to fifteen (0-15), where zero is the lowest and fifteen is the highest.

When a PRIORITY of nine or greater is specified, the following action occurs in a multiprogramming mode:

a. If necessary, jobs which are running and which have a lower priority will be "rolled-out" from memory to disk to create space for the high-priority job. This action is called "crashout."

b. A high-priority job entered in the schedule will not automatically suspend any other high-priority job running in memory. However, the system operator may stop (ST) them.

c. Upon termination of the high-priority job, the suspended programs will be automatically reinstated to memory.

PROTECTED

The PROTECTED attribute allows a program to be "protected" from accidental tampering by certain input messages.

The format of the PROTECTED statement is:

```
[?] { PROTECTED }
    { PT        }
```

The PROTECTED control word may be abbreviated as PT.

The following input messages are rejected by the MCP if they reference a protected program:

CL
DP
DS
QC
ST
SW

However, the DS, DP, ST, and SW messages are allowed to reference a spawned job from the parent remote terminal.

If a protected program reaches an abnormal termination, the MCP unlocks it automatically so that it may be DS-ed.

- The LP input message may be used to add or remove protection for a program that has reached beginning-of-job (BOJ).

The PROTECTED attribute may not be specified in a MODIFY or DYNAMIC control instruction, nor may it be used with an OBJ attribute.

Examples:

```
? EX REMOTE/UPDATE PROTECTED

? COMPILE TEST/PROGRAM COBOL LIBRARY PT
```

SCHEDULE.PRIORITY

The SCHEDULE.PRIORITY attribute assigns priorities of programs in the schedule.

The format of the SCHEDULE.PRIORITY statement is:

[?] [OBJ] { SCHEDULE.PRIORITY / SC } [=] integer

The SCHEDULE.PRIORITY control word may be abbreviated as SC.

The priorities of the schedule are separate from the mix priorities in that SCHEDULE.PRIORITY will only alter or assign priorities pertaining to the schedule, not the mix.

The priority integer must be equal to or less than fourteen.

Jobs in the ACTIVE SCHEDULE having the same assigned priority are further discriminated by the actual time the jobs have been in the schedule.

Example:

```
? EXECUTE A/B SCHEDULE.PRIORITY = 12
```

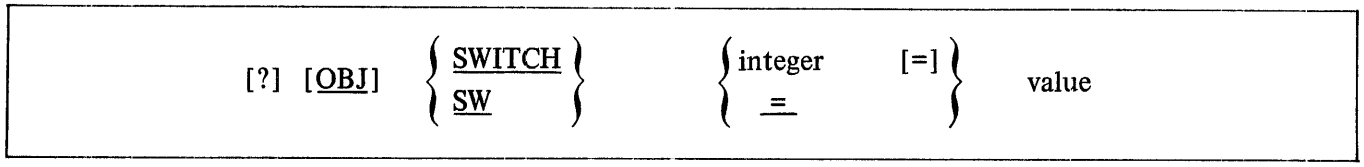
NOTE

Once the program has been placed in the schedule, the SP console message must be used to change the scheduled priority.

SWITCH

The SWITCH control attribute allows the system operator to set programmatic switches.

The format of the SWITCH statement is:



The SWITCH control word may be abbreviated as SW.

The integer must be a decimal digit from zero to nine (0-9) that references the switch to be set. To determine what switches are available, the specific language manual for the program for which the switches are being set must be referenced. If the "=" option is used, all ten switches are implied (40 bits of information).

The value is the value that the switch or switches are assigned.

Examples:

```
? SWITCH 0 = 5 SWITCH 1 = 3

? SW 0=5 SW 1=3

? SW = @0123456789@
```

To modify or query the switches after the program has gone to BOJ, use the SW and TS commands, respectively.

TIME

The TIME attribute allows specification of the maximum allowable processor time a job may accumulate.

The format of the TIME attribute is:

[?] [OBJ] { TIME / TI } [=] integer

The TIME attribute may be abbreviated as TI.

If the TIME attribute is specified for a program and that program exceeds the processor time limitation, it will be DS-ed with the following message:

EXCEEDED MAXIMUM RUN TIME ALLOWED

The integer is specified in minutes of processor time, not elapsed time.

Example:

? EX TEST/DEBUG TIME = 5

UNFREEZE

The UNFREEZE attribute allows the system operator to remove the FREEZE condition from a program, thus permitting the rolling-out to disk of a program that is in an interrupted state.

The format of the UNFREEZE statement is:

```
[?] [OBJ]  { UNFREEZE }
           { UF       }
```

The UNFREEZE control word may be abbreviated as UF.

UNOVERRIDE

The UNOVERRIDE attribute makes it possible to reset the compatibility check bypass caused by OVERRIDE.

The format of the UNOVERRIDE statement is:

```
[?] [OBJ] { UNOVERRIDE }
          { UV         }
```

For further information on the MCP interpreter compatibility check mechanism, refer to the OVERRIDE statement.

VIRTUAL.DISK

The VIRTUAL.DISK attribute gives the operator the ability to change the number of disk segments assigned by a compiler for saving data overlays during execution.

The format of the VIRTUAL.DISK statement is:

```
[?] [OBJ] {VIRTUAL.DISK} [=] integer
          {VI          }
```

The VIRTUAL.DISK control word may be abbreviated as VI.

If the integer is zero and the program requires disk space for data overlays, the MCP will assign a default size of 1000 segments.

Example:

```
? CO TEST UPL LI VI 5000
```

FILE PARAMETER INSTRUCTIONS

DATA

The DATA control instruction informs the MCP of the name of a punched card data file.

The format of the DATA control instruction is:

```
[?]  { DATA }   file-identifier
     { DA   }
```

The control word DATA may be abbreviated as DA.

The DATA control statement must be the last control instruction prior to the actual data.

When a DATA statement immediately follows a COMPILE or EXECUTE statement, the MCP saves the card reader for the job specified in the COMPILE or EXECUTE statement. This prevents any other job in the mix which is looking for a card file with the file-identifier specified on the DATA statement from being assigned the card file belonging to the job just executed. The MCP does not reserve the card reader in this manner if an END statement is detected before the DATA statement.

Examples:

```
? EX A/B CHARGE 123456
? DATA CARDIN
— data cards —
? END

? CO XYZ COBOL LIBRARY DATA CARDS
— data cards —
? END

? EXECUTE A/B ME 2000
? CG 2222
? FI CARDS NAME=TEMP DSK;
? END
? DATA CARDFILE
— data cards —
? END
```

END

END

The END statement indicates to the MCP that the card data input has reached the End-of-File (EOF).

The format of the END statement is:

[?] END [file-identifier]

The END control statement cannot be abbreviated.

When the END statement is used it must be the last card in that file. A program receives an EOF report from the MCP when it attempts to read the END control statement.

The END control card is not required at the end of a data deck if the program recognizes the last card in the file and closes that file without trying to read another record. However, if the program does try to read another record from that file and the card reader is empty, the MCP holds the card reader waiting for more data or a "? END" statement to be read.

A card file need not be terminated by an END card if it is immediately followed by another control card. The detection of this other control card by the MCP results in an EOF signal to the program, as though an END card had been read.

If a data card with an invalid punch in column 1 is read within a data deck, the MCP stops the card reader and notifies the operator that the card just read has an invalid punch in column (1). This allows the operator to correct the card and permit the program to continue reading cards.

The END statement is also used to separate a DATA statement from preceding EXECUTE or COMPILE statements when it is desired to disassociate the data file from the job specified in the EXECUTE or COMPILE statement. This is the only case where the END statement can appear on the same card as other control statements, and without a question mark/invalid character in column one. See the DATA statemer for further information on reserved card files.

Examples:

```
? EX A/B
? DATA CARDS
— data cards —
? END CARDS

? CO X/Z FORTRAN DA CARDS
— data cards —
? END

? EX TEST END
? DATA CARDFIL
— data cards —
? END CARDFIL
```

ENDCTL

The ENDCTL (END CONTROL) statement indicates to the MCP that the input file to SYSTEM/LDCONTRL has reached End-of-File (EOF).

The format of the ENDCTL statement is:

```
? ENDCTL
```

The ENDCTL statement cannot be abbreviated.

The ENDCTL statement must be the last card in a CONTROL DECK (the input file read by SYSTEM/LDCONTRL). The ENDCTL card causes the MCP to signal EOF to SYSTEM/LDCONTRL, which closes all files in use and goes to end-of-job.

Refer to the LD message for further information on pseudo-reader CONTROL DECKS.

STREAM

The STREAM control statement informs the MCP of the name of a punched card file that is to be treated as a STREAM DATA file.

The format of the STREAM control statement is:

```
[?] STREAM   file-identifier
```

The STREAM control statement cannot be abbreviated.

A stream of data consists of all data cards contained between the stream beginning and the stream end. The stream beginning is identified by the STREAM control instruction. The stream end is identified by the TERMINATE control instruction. The file-identifier on the TERMINATE card must be the same as on the corresponding STREAM card in order to end the data stream.

STREAM and TERMINATE are somewhat analogous to DATA CTLDCK and ENDCTL used in pseudo-reader loading, but are more generalized.

When reading a data stream, the EXCEPTION branch is taken any time a card with a question mark/invalid character in column one is read. The MCP replaces column one of that card image with binary zeros (@00@) prior to passing the card image to the reading program. The program receives the EOF branch only when the proper TERMINATE card has been read.

Example:

```
? STREAM CARDS
? COMPILE TEST WITH COBOL LIBRARY
? CHARGE 123456
? FILE CARDS NAME=SOURCE/TEST DISK;
? EX TEST CHARGE 123456
? DATA CARDIN
— data cards —
? END CARDIN
? TERMINATE CARDS
```

TERMINATE

The TERMINATE control instruction informs the MCP that the STREAM DATA input file has reached End-of-File (EOF).

The format of the TERMINATE control statement is:

```
? TERMINATE file-identifier
```

The TERMINATE control instruction cannot be abbreviated.

The file-identifier specified on the TERMINATE card must be the same as the one specified on the corresponding STREAM card in order to end the data stream.

Refer to the STREAM control instruction for further information regarding STREAM DATA.

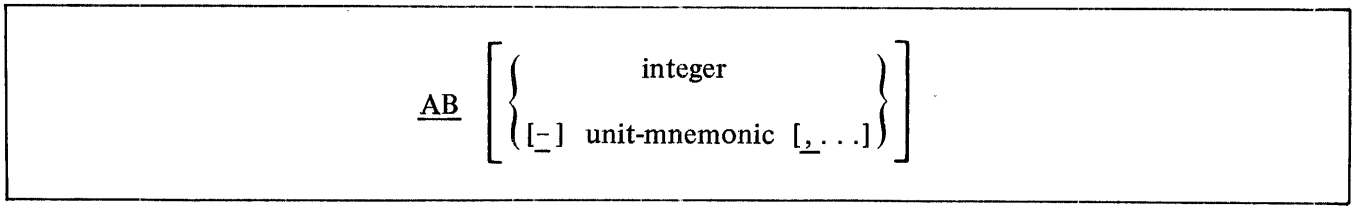
AB

SYSTEM CONTROL INSTRUCTIONS

AB INPUT MESSAGE (Auto Backup)

The AB input message allows the system operator to control the MCP AUTOPRINT mechanism, by reserving line printers and SYSTEM/BACKUP programs for automatic printer backup.

The format of the AB message is:



The AUTOPRINT mechanism retrieves printer backup files from disk and prints them automatically, directing the output to a designated line printer. Up to four (4) copies of SYSTEM/BACKUP may be specified for execution, and remain in the mix as long as there are backup files to be printed.

All printer backup files on the default backup disk (refer to the BD message) with a family-name of BACKUP.PRT are candidates for printing via AUTOPRINT. To explicitly avoid automatic printing, the file-attribute USER.BACKUP.NAME must be specified for the print file when it is opened and directed to backup, using a family-name other than BACKUP.PRT.

When the AUTOPRINT mechanism is invoked, a designated number of SYSTEM/BACKUP programs are executed. The names of candidate printer backup disk files are entered into a queue file named AUTOPRINT. Each SYSTEM/BACKUP program initiated by the AUTOPRINT mechanism reads entries from this queue and prints the file specified. Printing is done as though a simple PB message (with no options) had been entered. Multiple copies may be printed, however, if the REPETITIONS value in the backup file is greater than one (1). Refer to the FILE statement for a description of the REPETITIONS attribute. The backup files are always removed from disk after printing. There is no SAVE attribute allowed with AUTOPRINT.

The SYSTEM/BACKUP programs initiated by AUTOPRINT only use line printers reserved for AUTOPRINT use. Print files from other programs in the mix (including other SYSTEM/BACKUP programs not initiated by AUTOPRINT) may not open any line printer reserved for AUTOPRINT.

The AB message has three forms, as follows:

AB	Displays the current AUTOPRINT value, as well as the unit-mnemonics of all line printers reserved for AUTOPRINT.
AB integer	Sets the AUTOPRINT value (the maximum number of copies of SYSTEM/BACKUP allowed) to the specified integer. The integer specified may not be greater than four (4). If the integer is zero (0), the AUTOPRINT queue is purged by the MCP and no further backup file names are entered into it. This allows the currently-running SYSTEM/BACKUP programs to complete printing their respective backup files and go to an orderly end-of-job.

AB [-] unit-mnemonic — Reserves line printers for use by AUTOPRINT. If the unit-mnemonic is preceded by a minus sign ("-"), the AUTOPRINT reservation is removed from the specified printer.

NOTE

It is the responsibility of the system operator to match the number of line printers reserved for AUTOPRINT with the number of copies of SYSTEM/BACKUP specified by the AUTOPRINT value. It would not be desirable, for example, to specify two copies of SYSTEM/BACKUP while reserving a single line printer for AUTOPRINT. Likewise, nothing is gained from reserving more printers than the AUTOPRINT value.

The AUTOPRINT mechanism may be activated in one of the following ways:

a. AB integer. This is used for initial setup of the AUTOPRINT value and reserving of line printers, or to reactivate AUTOPRINT should all the copies of SYSTEM/BACKUP be explicitly DS-ed.

b. CLEAR/START. All the AUTOPRINT parameters are retained through CLEAR/START, and the mechanism will be automatically reactivated by the MCP.

c. CLOSE of a candidate backup disk file. If there are no copies of SYSTEM/BACKUP in use by AUTOPRINT when a backup disk file is closed and the AUTOPRINT value is non-zero, the MCP initiates the required number of copies of SYSTEM/BACKUP.

d. Default backup disk is readied. When the default backup disk (see the BD message) is made ready and the AUTOPRINT value is non-zero, the MCP activates the AUTOPRINT mechanism. Only backup files residing on the default backup disk will be printed by the AUTOPRINT mechanism, and the BD message will be rejected if the AUTOPRINT value is non-zero.

Examples:

```
AB
  AUTO BACKUP – AB = 0 NO PRINTERS RESERVED

AB 2
  NO BACKUP FILES ON DISK

AB LPA
  LPA RESERVED FOR AUTO BACKUP.

AB
  AUTO BACKUP – AB = 2 AND
  LPA RESERVED FOR AUTO BACKUP.

AB 1
  SYSTEM/BACKUP = 1 BOJ. . . .

AB LPB,-LPA
  LPB RESERVED FOR AUTO BACKUP.
  LPA NO LONGER RESERVED FOR AUTO BACKUP.

AB 0;AB-LPA,-LPB
  AB INTEGER = 0
  LPA NO LONGER RESERVED FOR AUTO BACKUP.
  LPB NO LONGER RESERVED FOR AUTO BACKUP.
```

AX

AX INPUT MESSAGE (Response to ACCEPT)

The AX message is a response to an ACCEPT message requested by an object program through the MCP.

The format of the AX message is:

```
mix-index  AX  . . . input message . . .
```

All responses are assumed to be alphanumeric format. The input message starts in the first position after the AX on the input line, and continues until the END OF MESSAGE button is pressed.

If the End-of-Message is depressed immediately after the AX, the MCP fills the area in the requesting program with blanks.

Example:

```
2  AX  CHECK  VOID  IF OVER 500 DOLLARS
```

Input messages shorter than the receiving field in the program are padded with trailing blanks. Longer messages are truncated on the right.

The AX message has an unsolicited console feature in that the operator may enter AX message responses for a given program prior to the actual ACCEPT. The AX messages must be entered in the order used, since they are queued on a first-in, first-out basis. Up to ten (10) AX messages may be queued for a program at any one time.

The queue is automatically cleared at program EOJ or an abort.

BB INPUT MESSAGE (Backup Blocks per Area)

The BB input message allows the operator to specify the default (minimum) number of blocks to assign each area of a printer or punch backup disk file.

The format of the BB message is:

BB [integer]

The value of the backup blocks per area is set to 200 by COLDSTART, and if the integer entered in the BB message is less than 5, a value of 200 is assigned by default.

If an integer is not entered with the BB message, the MCP displays the current setting of the backup blocks per area.

When an output printer or punch backup file is opened on disk, its attributes are set by the MCP as follows:

a. RECORD SIZE: Declared RECORD.SIZE plus one (1) byte.

b. RECORDS PER BLOCK: Optimized by the MCP for each particular RECORD.SIZE.

c. BLOCKS PER AREA: Declared BLOCKS.AREA or the BB value, whichever is larger.

d. AREAS: Declared AREAS or twenty-five (25), whichever is larger.

Thus, it is possible to declare (via the FILE statement) large file sizes (BLOCKS.AREA and/or AREAS) for specific backup files that are known to be larger than normal, while maintaining a system-wide default for all other backup files.

Examples:

```
BB

BB 350
```

BD

BD INPUT MESSAGE (Backup Designate)

The BD input message allows the operator to designate a specific disk pack or disk cartridge for backup files.

The format of the BD message is:

```
BD [pack-identifier]
```

When the BD message is entered without the pack-identifier the MCP will cause the current setting of the default backup disk to be displayed.

If " " (the quotes are required) is entered for the pack-identifier, the default backup disk is changed to the SYSTEM disk.

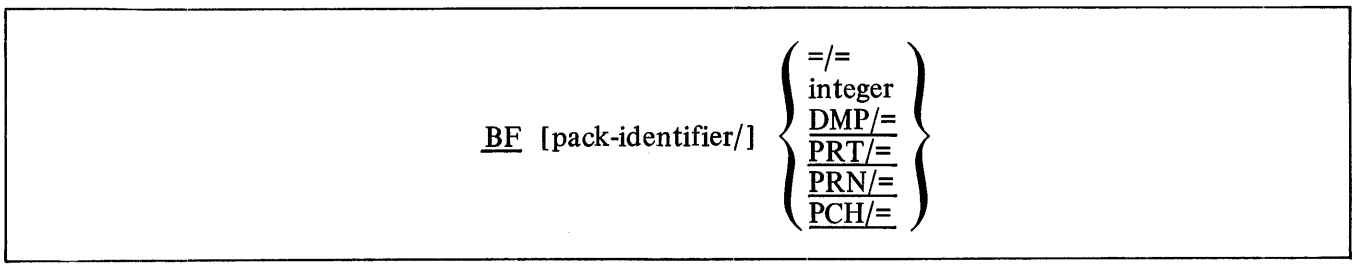
Examples:

```
BD

BD USERPACK

BD " "
```

BF INPUT MESSAGE (Display Backup and Dump Files)

The BF input message lists disk backup and memory dump file names on the console printer.

The format of the BF message is:



The BF message only finds those backup and dump files that have the MCP-generated name (BACKUP.PRT/integer, BACKUP.PCH/integer, or DUMPFILE/integer).

The PRT/= option lists all printer backup files on disk. The PCH/= option lists all punch backup files on disk. The DMP/= option lists all memory dump files on disk.

The =/= option lists all backup and memory dump files that are stored on disk.

PRN and PRT are both to be assumed to mean printer backup files. That is, PRN and PRT are equivalent.

The pack-identifier requests displaying the backup and dump files on the designated removable disk drive. If it is omitted, the MCP will display the backup files resident on system disk.

Examples:

```
BF =/=
BF PRT/=
BF 19
BF DMP/=
BF USERPACK/=/=
BF USER/PRT/=
BF USER/=/217
USER SITE/A BF PRT/=
```

CD

CD INPUT MESSAGE (List Card Decks in Pseudo Readers)

The CD input message allows the system operator to obtain a list of the pseudo card files and their file numbers that have been previously placed on disk by SYSTEM/LDCONTRL.

The CD message format is:

```
CD { integer [,integer] ... }
   { =/=                    }
```

The MCP displays the number of each pseudo deck specified and the first fifty (50) characters of the first card in the deck.

If a deck is in use, its name and the program using it are displayed.

Examples:

```
CD =/=
   DECK #1 = ?COMPILE TEST COBOL LIBRARY
   DECK #2 = ?EX PAYROLL/CHECKS PR 6 DATA CARDS

CD 5, 7
   DECK #5 = ?DATA CARDIN  % INPUT TO TEST/PROG
   DECK #7 IN USE BY DMPALL = 3
```

CL INPUT MESSAGE (Clear Unit)

The CL input message allows the operator to clear a unit on the system because of an apparent system software loop or hardware malfunction. Any program using the unit that has been cleared using the CL message will be discontinued (DS-ed).

The format of the CL message is:

CL unit-mnemonic

The CL message cannot be used with disk devices (DCx, DKx, DPx).

A CL message that references any unit which is not opened by a program is ignored. The only exception to this is the case where a card reader has been "reserved" for a program by job-number because the COMPILE or EXECUTE control statement immediately precedes the DATA statement without an intervening END statement. Without operator intervention (IL or UL message), such reserved card files can be opened only by the program for which they are reserved and, if unopened, remain reserved even after the program terminates or is RS-ed. The CL message permits the operator to remove the reserved condition from the reader and allow it to be assigned to any program that opens it with the correct file-identifier. To completely free the reader by removing the file-identifier, follow the CL message with an RY message.

Example:

```
CL LPA

CL CDB; RYCDB
```

CM

CM INPUT MESSAGE (Change System Software)

The CM input message allows the system operator to designate programs or usercode files as systems software with specific functions.

The format of the CM message is:

```
CM  system-software-mnemonic  { program-identifier }
                              { PURGE             }
```

The purpose of the CM message is to identify to the operating system the names of certain programs or usercode files to be used for specific functions (for example, as a new MCP or Network Controller). This is done by placing the file-identifier (and therefore the disk address) of the file into an entry in the NAME TABLE on disk. The NAME TABLE is the directory of all operating system software to be used by CLEAR/START and other system functions. Each entry in the NAME TABLE implies a specific function for the file entered into it.

The actual change resulting from a CM message is delayed until the following CLEAR/START. In the case of the Network Controller and Usercode files, however, the action taken is immediate.

For example, if a new MCP code file is placed into the "M" entry of the NAME TABLE, it will be loaded during the next CLEAR/START. If a new Network Controller is designated, it will be executed by the MCP whenever the currently-running Network Controller goes to EOJ, and a program attempts a REMOTE file open.

The PURGE option removes the file from the designated NAME TABLE entry.

Refer to the Clear/Start procedure for a list of the system software mnemonics that are used in the NAME TABLE.

Example:

```
CM MX MCP/XYZ

CM NC CANDE/HANDLER
```

CP INPUT MESSAGE (Compute)

The CP input message allows the operator to perform simple arithmetic functions on the console printer, as well as decimal/hexadecimal conversion.

The format of the CP message is:

CP integer-1 [operator integer-2] . . . (=;)

The valid operators recognized by the CP message are as follows:

+	addition
-	subtraction
*	multiplication
/	division
M	MOD (remainder divide)

An equal sign (=) or semicolon (;) terminates the expression and must be the last entry when entered from a card reader.

The CP message will evaluate an arithmetic expression strictly on a left-to-right basis. Therefore, quantities contained in parentheses or brackets are invalid. Spaces are not used as delimiters and are ignored. Operands and intermediate results are considered positive integers, and overflow beyond 16777215 will be truncated.

The response is displayed in both decimal and hexadecimal formats.

Example:

request: CP @ 3A@ * 4 + @F@

response: CP: @0000F7@=247

CP @F@

CP: @00000F@=15

CQ

CQ INPUT MESSAGE (Clear Queue)

The CQ input message causes all messages stored in the Console Printer QUEUE to be cleared.

The CQ message format is:

CQ

CU INPUT MESSAGE (Core Usage)

The CU message allows the system operator to interrogate the MCP regarding the amounts of SAVE and OVERLAYABLE memory in use by various programs in the mix, as well as by the MCP itself.

The format of the CU message is:

```
[mix-index] CU
```

If the mix-index is omitted, memory usage totals are displayed for all programs in the mix as well as for the MCP. The amount of AVAILABLE memory displayed is also included in the figure for OVERLAYABLE memory, since AVAILABLE is actually a subset of OVERLAYABLE.

Including the mix-index causes the MCP to display the memory usage statistics for only the program specified.

The output displayed by the MCP is also affected by the DBUG option; if set, additional information intended for systems software development and debugging is displayed. Most of this information is not useful for normal systems operation.

Examples:

```
CU
  CORE USAGE:  09:14:42.1
  SAVE=77153 BYTES
  OVERLAYABLE= 184990 BYTES
  AVAILABLE= 17224 BYTES
  CANDE/HANDLER =3 SAVE= 13751 BYTES, OVERLAYABLE= 10610 BYTES
  (USER) REMOTE/UPDATE =4 SAVE= 9219 BYTES, OVERLAYABLE= 2747 BYTES
  (SITE) CANDE =2 SAVE= 22069 BYTES, OVERLAYABLE= 26812 BYTES
  DMPALL =1 SAVE= 441 BYTES, OVERLAYABLE= 0 BYTES (ROLLED OUT)

2CU
  SYSTEM/LOAD.DUMP =2 SAVE= 9898 BYTES, OVERLAYABLE= 363 BYTES
```

DB

DB INPUT MESSAGE (Interrogate Data Base Status)

The DB input message is used to determine which DMSII data bases, if any, are currently in use on the system.

The format of the DB message is:

```
DB
```

Examples:

```
DB
  NO DATA BASES ACTIVE

DB
  THE FOLLOWING DATA BASES ARE ACTIVE:  STUDENTDB,PAYROLL
```

DF INPUT MESSAGE (Date of File)

The DF input message allows the operator to display on the console printer the compilation date and time for code and interpreter files, and the creation date for all other types of files.

The format of the DF message is:

```
DF { file-identifier  } [,...]
   { family-name/=    }
```

For DMSII data base files, the date displayed by the MCP also includes the version date and time, which is the last time the data base file was closed after an update. The version is stored in the data base dictionary for comparison the next time the file is opened.

Example:

```
DF COBOL, MCPII/=, DATA/FILE, DB/=
  "COBOL" COMPILED ON 04/20/77 AT 13:16:02.2
  "MCPII/ANALYZER" COMPILED ON 05/04/77 AT 14:15:04.4
  "MCPII/MICRO.MCP" COMPILED ON 04/22/77 AT 19:36:54.7
  "MCPII" COMPILED ON 04/25/77 AT 17:22:15.0
  "DATA/FILE" CREATED ON 05/03/76
  "DB/TRANS" CREATED ON 01/21/77 VERSION IS 04/11/77 18:10:14.0
  "DB/HIST" CREATED ON 01/21/77 VERSION IS 03/30/77 16:25:14.8
```

DM

DM INPUT MESSAGE (Dump Memory and Continue)

The DM input message allows the system operator to dump the contents of a program's memory space to disk for subsequent analysis by DUMP/ANALYZER.

The format of the DM message is:

mix-index DM

Processing automatically continues when the dump is finished.

The DM message will create a file called DUMPFILE/integer. The integer will be incremented by one each time a DM is performed in order to make each DUMPFILE unique.

The DUMPFILE may be printed by the DUMP/ANALYZER program. Refer to the "PM" message.

Example:

```
2 DM
```

DP INPUT MESSAGE (Dump Memory and Discontinue)

The DP input message allows the system operator to initiate a memory dump during a program's execution, and then abort that program.

The DP message format is:

mix-index DP

The input of the DP message signals the MCP to halt program execution, dump memory out to disk, and abort the program as though a DM message had been entered immediately followed by a DS message.

Example:

```
1 DP
```

{DR / DT} INPUT MESSAGE (Change MCP Date)

The DR, DT input message allows the system operator to change the current date maintained by the MCP.

The DR, DT message format is:

{DR / DT} mm/dd/yy

The MCP will accept only valid dates. The month entry must be between one and twelve, the day must be between one and thirty-one, and the year must be valid numeric digits.

DS INPUT MESSAGE (Discontinue Program)

The DS input message permits the system operator to discontinue the execution of a program.

The format of the DS message is:

mix-index DS

The DS message may be entered at any time after the BOJ and prior to EOJ.

The DS message signals the MCP to stop the program's execution and return the memory the program occupied to the system. Any files not previously entered into the disk directory are lost and the disk area occupied is returned to the disk available table. All other files are closed.

ED

ED INPUT MESSAGE (Eliminate Pseudo Deck)

The ED input message allows the system operator to eliminate a deck from a pseudo reader. This is equivalent to flushing the reader and then performing an RY message.

The format of the ED message is:

ED integer

The deck will be eliminated from the pseudo reader and from the disk directory by the ED message.

EM INPUT MESSAGE (ELOG Message)

The EM input message allows the operator to place a message into the ELOG.

The format of the EM message is:

EM input-message

The input-message starts in the first position after the EM on the input line and continues until the END OF MESSAGE is pressed.

ET

ET INPUT MESSAGE (ELOG Transfer)

The ET input message transfers the information in the file SYSTEM/ELOG to the file ELOG/*#integer*. The program SYSTEM/ELOGOUT is then executed label equating ELOG/*#integer* and prints the file.

The format of the ET message is:

ET

FM INPUT MESSAGE (Response to Special Forms)

The FM input message is a response to the "SPECIAL FORMS REQUIRED" message.

The format of the FM message is:

mix-index FM unit-mnemonic

The unit-mnemonic designates which unit is to be assigned to the file.

The message

program-name = mix-index SPECIAL FORMS REQUIRED FOR file-id

is displayed on the console printer requiring that a FM message be submitted by the system operator before the file can be opened.

The FM message overrides a SAVED condition on the specified device. This allows the system operator to mount the special forms before they are actually required and save (SV message) the device to prevent its use by other programs in the mix.

When a file with SPECIAL FORMS declared is closed, the device it is assigned to is saved automatically by the MCP, thus preventing other programs from accidentally using the special forms before they can be removed by the operator.

If the unit-mnemonic specifies a backup device (i.e., tape or disk), a backup file is created by the MCP. When printed or punched, the backup file will also request SPECIAL FORMS.

Example:

3 FM LPA

FN

FN INPUT MESSAGE (Display Internal File Name)

The FN input message allows the system operator to display the internal file names of an object program.

The format of the FN input message is:

```
FN    program-name    external-file-identifier
```

The MCP lists on the console printer all the internal-file-names of the object program which have the specified external-file-identifier in the following format:

FN = internal-file-identifier-1

FN = internal-file-identifier-2

FN = . . .

FR INPUT MESSAGE (Final Reel of Unlabeled Tape File)

The FR input message gives the operator the ability to notify the MCP that the last reel of an unlabeled tape file has completed processing, and there are no more input reels to be read.

The format of the FR message is:

mix-index FR

The FR message is a response to the message:

job-specifier FILE file-identifier (UNLABELED) REEL #integer NOT PRESENT

This message is the result of an unlabeled tape file reaching the End-of-Reel; the FR message notifies the program that the file has reached EOF.

The FR message is also allowed with labeled tape files in order to signal EOF without reading all of the reels of the file.

The FR message must be used with paper tape input files to signal EOF after all reels have been processed.

Examples:

```
 DMPALL =1 FILE INP.FILE (UNLABELED) REEL #5 NOT PRESENT
1FR

 TAPE/PRINT =3 FILE TAPEIN (LABELED "TAPFIL") REEL #2 NOT PRESENT
3FR
```

FS

FS INPUT MESSAGE (Force from Waiting Schedule)

The FS input message is used to force jobs from the WAITING SCHEDULE into the ACTIVE SCHEDULE.

The format for the FS message:

```
FS  { job-number }
    { =          }
```

The equal sign option will force all jobs into the ACTIVE SCHEDULE.

See the HS message for placing a job in the WAITING SCHEDULE.

NOTE

The WAITING SCHEDULE is a schedule of jobs that are waiting to be placed in the ACTIVE SCHEDULE. For example, an EXECUTE with the attribute THEN or AFTER.NUMBER would place the program in the WAITING SCHEDULE.

The ACTIVE SCHEDULE are those jobs that have satisfied all the requirements for execution and are only waiting for memory space to run.

In order for a program to be in the mix, it must have gone to BOJ.

GO INPUT MESSAGE (Resume Stopped Program)

The GO input message is used by the system operator to request resumption of a program that has been stopped (ST message).

The format for the GO message is:

```
mix-index GO
```

A program retains its assigned mix-index number when STOPped and rolled-out to disk. The MCP uses this mix-index number in the GO message to identify the program for resumption.

HS

HS INPUT MESSAGE (Hold in Waiting Schedule)

The HS input message will allow the system operator to place a HOLD on a specific job(s), thereby temporarily removing them from the ACTIVE SCHEDULE.

The format of the HS message is:

```
HS { job-number }
   {    =     }
```

The equal sign (=) option will place all jobs in the ACTIVE SCHEDULE into the WAITING SCHEDULE.

A job-number is assigned when a program is scheduled by the MCP.

A job that has been placed in the WAITING SCHEDULE by a HS message will remain in the WAITING SCHEDULE until FS-ed.

HW INPUT MESSAGE (Hold in Waiting Schedule until Job EOJ)

The HW input message allows the system operator to designate that certain jobs are to be placed in the WAITING SCHEDULE, awaiting the EOJ of another job (by job-number).

The format of the HW message is:

```
HW  { job-number-1 }  job-number-2
    {     =      }
```

The equal sign (=) option will place all jobs in the ACTIVE SCHEDULE into the WAITING SCHEDULE, and mark them as waiting for the completion of job-number-2.

A job that has been placed in the WAITING SCHEDULE by a HW message will remain in the WAITING SCHEDULE until job-number-2 reaches EOJ or until FS-ed by the operator.

IL

IL INPUT MESSAGE (Ignore Label)

The IL input message allows the system operator to ignore the label on the file mounted on the designated unit.

The format of the IL message is:

mix-index IL { unit-mnemonic / # integer }

The mix-index must be used to identify the program. In a multiprogramming environment there may be more than one "NO FILE" condition at a time.

The IL message may be used in response to the following messages:

NO FILE . . .

DUPLICATE INPUT FILE . . .

file-identifier NOT IN DISK DIRECTORY

It is assumed that the system operator knows that the file on the unit selected is the file needed regardless of the original file-identifier's location. If the unit-mnemonic specifies a disk drive, the directory on that drive will be searched for the required file-identifier. The # integer option is used to designate a pseudo-reader (by number) as the input device.

NOTE

A RESTRICTED disk cannot be assigned to a program with the IL message. The program must have the correct dp-id prior to the opening of the file.

An IL message referencing a card reader which has been reserved for a program by job-number overrides the reservation and assigns the reader regardless of the file-identifier requested.

KA INPUT MESSAGE (Analyze Disk Directory)

The KA input message allows the system operator to analyze the contents of a disk directory, including file area assignments.

The KA message has three formats:

Format 1:

```
KA  { disk-pack-id/=/= }
    {      =/=       }
```

Format 2:

```
KA  { disk-pack-id/DSKAVL/ }
    {       DSKAVL         }
```

Format 3:

```
KA  { [disk-pack-id/] family-name/= }
    {     file-identifier           }
```

Inclusion of the disk-pack-id causes the MCP to list the requested information for the specified user disk pack or disk cartridge; otherwise, system disk is assumed.

Format 1 results in a listing of available areas followed by a description of all files contained on the specified disk.

Format 2 lists only the available areas for the designated disk.

Format 3 lists only the description of the specified file or files.

If a line printer is not available, a KA listing (of a single file only) can be produced on the SPO by setting the DBUG option prior to entering the KA input message. Only the file-identifier option of format 3 may be used in this manner.

Examples:

KA =/=

KA USER/=/=

KA DSKAVL

KA USER/DSDAVL/

KA COBOL

KA RPG/=

KA USER/PAYROLL/=

KB

KB INPUT MESSAGE (Keyboard Options)

The KB message provides the system operator with control over certain characteristics of the console printer or console display.

The format of the KB message is:

KB option

Some options are valid for only one type of SPO, whereas others are valid for both types (but may function somewhat differently). Those options that are valid for both the console printer and console display are as follows:

TIME {ON / OFF}

When set ON, causes a "time stamp" to be included with each message displayed or printed. Set OFF by default. The time stamp is always included in the SPOLOG, regardless of the setting of this option.

LP {ON / OFF}

When set ON, causes all SPO messages (both input and output) to be printed on any line printer that is available. For console display only, the messages continue to appear on the screen. If the printer being used goes "not ready" or is required for program or MCP output (for example, a KA listing), the messages revert back to the SPO. Set OFF by default.

{WIDTH / W} integer

Controls the width of the output line (console printer) or each column displayed (console display). The default and maximum allowable value is 72 for the console printer and 80 for the console display. Values greater than the maximum are set to the maximum. Values less than 20 are set to 20.

integer

When entered on the console printer, integer causes the pointer to the SPO QUEUE on disk to be rolled back integer sectors. The messages are then displayed on the SPO again. If "KB LP ON" is entered first, the messages will be diverted to an available printer. The integer must be between 1 and 199, inclusive. A "CQ" message terminates the display.

When entered on the console display, integer causes a "scrolling" action similar to that produced by entering the integer by itself (see "Scrolling" on the console display); however, the output is directed to an available printer if the integer is 4 or less. If the integer is 5 or greater, it causes the pointer to the SPO QUEUE on disk to be rolled back by integer sectors and the messages displayed again as described for the console printer.

The options that are valid for the console display only are as follows:

COL integer

Controls the number of columns displayed on the screen. The integer may be either one (1) or two (2); values less than one are set to one, and values greater than two are set to two. The default setting is a two-column format.

{LENGTH / LTH / L} integer

Controls the number of output lines (each consisting of two columns) displayed on the screen. Set to 22 by default. The integer must be between 2 and (24 minus INP.LINES) inclusive. If less than 2, it will be set to 2.

{INP.LINES / INP.LTH / INP} integer

Controls the number of lines reserved at the top of the screen for entering input messages. Set to 2 by default. Changing INP.LINES automatically adjusts LENGTH as necessary to prevent loss of output lines at the bottom of the screen.

{SUPPRESS / SUP / UNSUPPRESS / UNS}

If SUPPRESS is specified, control cards entered from devices other than the SPO (ZIP, card reader, MCP internal) are not included on the screen. UNSUPPRESS is the default condition and causes such messages to be displayed.

{DIRECTION / DIRN / DIR} {0 / FIFO / 1 / LIFO}

Controls the order in which messages are displayed on the screen. FIFO (First In-First Out) and 0 specifies that the oldest message is displayed toward the top of the screen, with the most recent message at the bottom. LIFO (Last In-First Out) or 1 causes the messages to be displayed with the oldest at the bottom and the most recent toward the top of the screen. LIFO is the default setting.

Examples:

```
KB TIME ON
KB LP ON; KB 50
KB W 50
KB L 15
KB INP 3
KB SUP
KB DIR FIFO
```

KC
KP.

{KC / KP} INPUT MESSAGE (Print Disk Segments)

The KC or KP message provides a means for the system operator to print selected disk files or segments of a disk on the line printer.

The format of the KC or KP message is:

{KP / KC} { file-identifier / unit-mnemonic [integer-1] integer-2 } [integer-3]

The printout created by the KP message is in HEXADECIMAL format.

The printout created by the KC message is in CHARACTER format.

The file-identifier option will print a file by that name. Integer-1 is required with head-per-track disk only and designates the electronics unit.

Integer-2 is used to specify the disk address from which printing is to begin.

Integer-3 is used to specify the number of segments to print beginning either from the first segment of a file or the address specified by integer-2. If omitted, number of segments printed is one.

Examples:

KP A/B 10	Print 10 segments of file A/B
KP A/B	Print 1 segment of file A/B
KP CCC/X/	Print 1 segment of file A on pack CCC
KP DPA @ 5C @ 15.....	Print 15 segments from HEX LOC. 5C
KP DKA 1 200 10.....	Print 10 segments on EU 1 from DECIMAL LOC 200

LC INPUT MESSAGE (Load Cassette)

The LC message is used to load system programs (compilers, interpreters, object code, system software) from a cassette in the console cassette reader to disk with appropriate additions in the disk directory.

The format of the LC message is:

```
                                  { family-name/file-identifier }
LC [disk-pack-identifier]         { family-name/=               }
                                  { =/=                         }
```

The LC message cannot be used to load a freestanding program that does not execute under the control of the MCP.

The LC message calls the program SYSTEM/LOAD.CAS which loads the files.

LD

LD INPUT MESSAGE (Pseudo Load)

The LD input message is used by the system operator to initiate the building of pseudo card deck(s) on disk to be processed by pseudo readers.

The LD message format is:

LD

After receiving a LD message, the program, SYSTEM/LDCONTRL, looks for a "? DATA CTLDCK" control statement that initiates the read.

The card deck's "file-id" is assigned by a "? DATA file-id" control statement preceding the data deck to be read. Each data deck that is loaded will be numbered consecutively along with its file-id which is used in opening the pseudo card files.

Terminating the LD function requires a "? ENDCTL" control statement immediately following the last data deck that is to be read.

Example:

The following example demonstrates how two compile decks and one data deck can be loaded as pseudo card files to be used by pseudo readers.

```
                                  ? DATA CTLDCK
                                  ? COMPILE program-name COBOL SYNTAX
CONTROL                 DECK      ? DATA CARDS
DECK                    A
                                    data deck
                                  ? END
                                  ? COMPILE program-name FORTRAN
                                  ? DATA CARDS
                        DECK
                                    data deck
                        B
                                  ? END
                                  ? DATA file-id
                        DECK
                                    data deck
                        C
                                  ? END
                                  ? ENDCTL
```

{LG / LN} INPUT MESSAGE (Transfer and Print Log)

The LG, LN input message allows the system operator to transfer and print either the system log or the SPO log. The log files are numbered sequentially starting with LOG/#00000001 or SPOLOG/#00000001. The program SYSTEM/LOGOUT or SYSTEM/SPOLOGOUT is executed, performing the necessary file-equate to print the specified log. The program must be in the disk directory in order for the MCP to accept the message.

The format of the LG, LN message is:

{LG / LN} [SPO]

When the LG or LN message is entered without the SPO option, the system log file is transferred and printed. Inclusion of the SPO option causes the transfer and printing of the SPO log.

Examples:

```
LG
LN  SPO
```

LP

LP INPUT MESSAGE (Lock/Unlock Program)

The LP input message allows the system operator to protect a running program against accidental tampering by certain input messages.

The format of the LP message is:

```
mix-index LP [-]
```

The MCP will not allow the following input messages to reference a program that is protected:

CL
DP
DS
QC
ST
SW

A program may also be protected at execute time by using the PROTECTED control instruction attribute in the execute control string.

The "-" option removes the protection from a running program, thereby allowing use of the above input messages.

Examples:

1 LP

3 LP-

LT INPUT MESSAGE (Load Train Printer Translate Table)

The LT input message allows the system operator to change the translate tables for the 400 and 750 line-per-minute (LPM) Train Printers. Train printers have interchangeable print train modules which allow a variety of specialized character sets to be used. Each print module requires a unique translator to be loaded into the printer control.

The format of the LT message is:

LT unit-mnemonic [train-identification]

The train-identification may be either the train-id name or number. The printer referenced by the LT message must be ready and not in-use by any program.

The 1100 and 1500 LPM Train Printers (which require the B 1247-4 Printer Control) have an automatic train identification feature which allows the MCP to immediately recognize the print translator required, and to load it into the printer control when necessary.

The 400 and 750 LPM Train Printers (which can be connected to either the B 1247 or B 1247-4 Printer Controls) do not have this automatic feature, and require notification from the system operator (by use of the LT message) whenever the train module is changed.

The characteristics of the printer translate tables and the LT message depend on which printer control (B 1247 or B 1247-4) is being used. The control-id can be determined from a SYSTEM/ELOGOUT listing.

B 1247 Train Printer Control (Control-id = @10@)

If the B 1247 Printer Control (not allowed with the 1100 and 1500 LPM Train Printers) is used, the file SYSTEM/PRINTCHAIN is built automatically by the MCP on system disk. The MCP loads a translator from this file into the printer control based upon the setting of the TRAIN SELECTOR SWITCH on the printer, as shown in the following table:

Train Selector Switch Setting	Translator
1	64-character EBCDIC
2	48-character EBCDIC
3	16-character EBCDIC
4	96-character EBCDIC
5	48-character FORTRAN
6	48-character B300/B500
7	48-character RPG
8	Undefined

The MCP loads the translator specified by the switch setting the first time the printer goes ready following a CLEAR/START. If it is necessary to change the character set later, all that is required is to mount the new train module in the printer, select the proper switch setting, make the printer ready, and enter the LT message to alert the MCP to the presence of the new character set. The train-identification option is not used, because the MCP uses the setting of the TRAIN SELECTOR SWITCH to determine the proper translator.

Example:

```
LT LPA
  64-CHARACTER PRINT CHAIN SELECTED ON LPA
```

B 1247-4 Train Printer Control (Control-id = @3E@)

If the B 1247-4 Printer Control is used, the file SYSTEM/TRAINTABLE must be present on system disk. This file is built by the program SYSTEM/BUILDTRAIN, and should contain all print translators required by an installation. Refer to the section entitled SYSTEM/BUILDTRAIN for a list of the standard print translators and their train-identifications.

For 400/750 LPM Train Printers connected to the B 1247-4 control, the TRAIN SELECTOR SWITCH is ignored, and the MCP loads the translator specified by the LT message into the printer control.

The MCP displays the following message the first time a 400/750 LPM Train Printer connected to a B 1247-4 Printer Control goes ready after a CLEAR/START:

"LT" REQUIRED FOR unit-mnemonic

A translator is loaded by the MCP only when ordered to do so through use of the LT message. Once loaded, the MCP does not change the translator until another LT message is entered.

The first time that an 1100/1500 LPM Train Printer goes ready after a CLEAR/START, the MCP automatically loads the translator specified by the train module identification. To change a train module, it is only necessary to mount the new module in the printer, make the printer ready, and enter an RY message to alert the MCP to the change.

If the train-identification is omitted from the LT message, the MCP displays the train-id number of the train currently being used.

NOTE

The interrogation feature is only allowed with the B 1247-4 Printer Control.

Examples:

LT LPA EBCDIC96
EBCDIC96 = "016" LOADED ON LPA

LT LPB 255
EBCDIC3.64 = "255" LOADED ON LPB

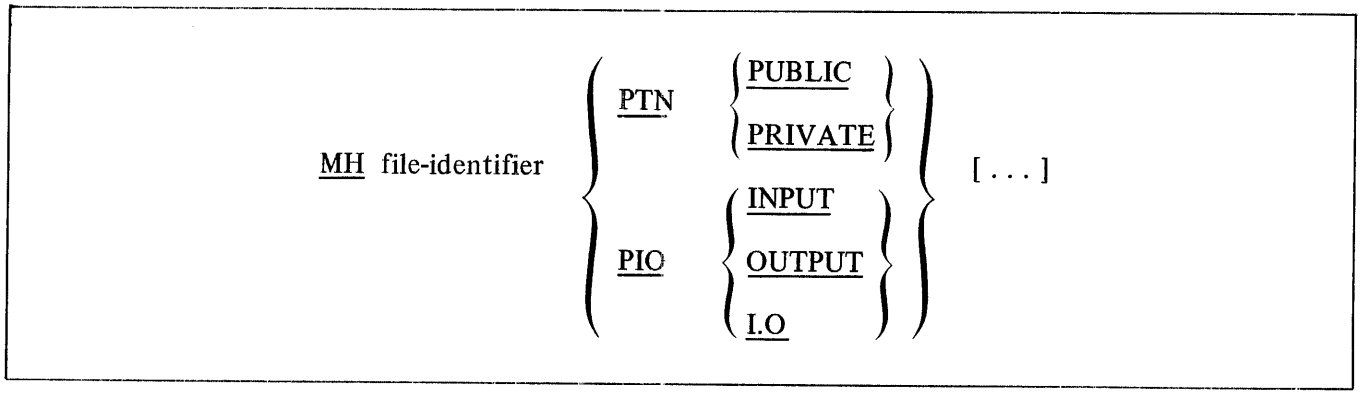
LT LPA
"005" WAS SELECTED ON LPA

MH

MH INPUT MESSAGE (Modify Header)

The MH input message allows the system operator to change the protection attributes of a disk file.

The format of the MH message is:



The PTN (PROTECTION) attribute specifies the access rights to the file. If PUBLIC is specified, no usercode/password is required to access the file. If PRIVATE is specified, the proper usercode/password for the file (or a PRIVILEGED usercode/password) is required for file access.

The PIO (PROTECT.IO) attribute allows limitation of the type of access to a PUBLIC file. A file may be limited to read-only (INPUT), write-only (OUTPUT), or read-write (I.O). If an object code file is marked INPUT, it is considered as executable but may not be opened as a data file by a program.

The PROTECTION and PROTECT.IO attributes of the FILE statement may be used to specify the file protection attributes at file-creation time. The MH message is used to set or modify the attributes of an existing disk file.

If a file is marked PRIVATE, the MH message must be preceded by a USER statement with the proper usercode/password.

Examples:

```
MH (PUBLIC/FILE PTN PRIVATE
  "(PUBLIC)/FILE" PROTECTION SET TO PRIVATE

USER (PRIVIL)/USER MH (PRIVATE)/FILE PIO INPUT PTN PUBLIC
  "(PRIVATE)/FILE" PROTECTION I.O SET TO INPUT
  "(PRIVATE)/FILE" PROTECTION SET TO PUBLIC

PD (USER)/TESTFILE
  PD= "(USER)/TESTFILE"
US USER/A MH TESTFILE PIN PUBLIC
  "TESTFILE" PROTECTION SET TO PUBLIC
CH (USER)/TESTFILE TESTFILE
  "(USER)/TESTFILE" CHANGED TO "TESTFILE"
MH TESTFILE PIO I.O
  "TESTFILE" PROTECTION I.O SET TO I.O
```

ML

ML INPUT MESSAGE (Mix Limit)

The ML input message allows the system operator to establish a maximum upper limit on the number of jobs allowed concurrently in the mix.

The format of the ML message is:

```
ML [integer]
```

The integer option sets the mix limit to the value specified, which must be between 1 and 63, inclusive. If the integer is omitted, the MCP displays the current setting of the mix limit on the SPO.

The value of the mix limit is used as the maximum number of jobs allowed in the mix concurrently; further attempts to execute programs after the mix limit has been reached remain in the ACTIVE SCHEDULE. The only jobs that are allowed to start after the mix limit has been reached are those having a SCHEDULE.PRIORITY of 15 (that is, SORT intrinsics).

Examples:

```
ML

ML 15
```

MP INPUT MESSAGE (List Multi-Pack File Tables)

The MP input message gives the operator the ability to interrogate the MCP's multi-pack file table which contains all multi-pack files that have been entered in the table since the last Clear/Start or RT message.

The format of the MP message is:

```
MP
```

MR

MR INPUT MESSAGE (Close Output File with Purge)

The MR input message gives the system operator the ability in a duplicate file situation to save the old file by purging the newly created file.

The format of the MR message is:

```
mix-index MR
```

MX INPUT MESSAGE (Display MIX)

The MX input message allows a system operator to request that the MCP display on the console printer all the programs currently in the MIX.

The format of the MX message is:

MX

The MX response lists the priority numbers, program-names and the MIX numbers of all programs currently running.

Example:

```
MX

    program-name = 1 #=127 PR=5

    program-name = 2 #=125 PR=4

END MX
```

NC INPUT MESSAGE (Set Memory Error Log Size)

The NC input message provides the system operator with the ability to interrogate or change the size of the table maintained in memory by the MCP and GISMO for logging memory errors. This table exists only on systems with error-correcting memories (B 1870/B 1860). On systems without error-correcting memories, the NC message has no effect.

The format of the NC message is:

```
NC [integer]
```

The integer specifies the number of entries to allocate to the table (each entry requires four bytes of memory). If the integer is greater than 255, the table size is set to 255 entries.

If the integer is zero, the MCP sets the table size to the default value, which is one entry for each 16 KB of main memory.

Any changes in the table size actually take effect when the next CLEAR/START is performed.

Entering the NC message without the integer causes the MCP to display the current size of the table, as well as the size it will assume at the next CLEAR/START.

Examples:

```
NC 255
  16 SPACES IN CORRECTABLE MEMORY ERROR TABLE
  WILL BE 255 SPACES AFTER NEXT CLEAR/START

NC0
  255 SPACES IN CORRECTABLE MEMORY ERROR TABLE
  WILL BE DEFAULT (1 PER 16K BYTES) SPACES AFTER NEXT CLEAR/START
```

OF

OF INPUT MESSAGE (Optional File Response)

The OF input message is used in response to the NO FILE message. It informs the MCP that the specified file is optional and can be bypassed.

The format of the OF message is:

```
mix-index OF
```

The OF message indicates that the file being requested is to be bypassed for this execution. Usage is restricted to input files that have been declared or label-equated (FILE control word) as OPTIONAL.

OK INPUT MESSAGE (Continue Processing)

The OK message is used by the system operator to direct the MCP to attempt to continue processing a program marked as WAITING.

The format of the OK message is:

```
mix-index OK
```

The OK message should only be given after the necessary action has been taken to correct the problem that caused the program to be placed in WAITING status.

Examples:

job-specifier DUPLICATE INPUT FILES . . .

job-specifier DUPLICATE FILE ON DISK . . .

job-specifier NO DISK . . .

job-specifier NO MEMORY . . .

job-specifier FILE file-identifier NOT PRESENT

If the corrective action is not taken before the OK message is entered, the original output message is repeated.

OL

OL INPUT MESSAGE (Display Peripheral Status)

The OL input message allows the system operator to interrogate the status of the system's peripheral units.

The format of the OL message is:

```
OL { PSR           }
   { unit-mnemonic }
   { unit-type-code}
```

The unit-mnemonic option displays the status of a specific unit.

The unit-type-code option displays the status of all peripherals of the same type.

Any invalid type unit used in the OL message will cause the MCP to display the following message.

NULL unit-type-code TABLE

The PSR option is used to interrogate the status of the pseudo readers on the system.

Examples:

CDA NOT READY

MTC UNLABELED

DPA LABELED "USER" #123456

MTA LABELED "MASTER" [123456]

OU INPUT MESSAGE (Specify Output Device)

The OU input message is a response to direct an output file to a specified output device.

The format of the OU message is:

mix-index OU unit-mnemonic

Example:

```
4 OU DPC
```

The OU is normally used in response to the "PUNCH RQD . . . or "PRINTER RQD . . ." message to direct the file to backup.

The OU message can also be used in a "SPECIAL FORMS REQUIRED" situation, and functions exactly the same as the FM message. In this case, the OU message overrides a saved condition on the specified device. This allows the system operator to mount the special forms before they are actually required and save (SV message) the device to prevent its use by other programs in the mix.

When a file with SPECIAL FORMS declared is closed, the device it is assigned to is saved automatically by the MCP, thus preventing other programs from accidentally using the special forms before they can be removed by the operator.

PB

PB INPUT MESSAGE (Print/Punch Backup)

The PB input message allows the system operator to initiate a copy of SYSTEM/BACKUP, that prints or punches a disk or tape backup file.

The PB message has two formats:

Format 1:

PB [unit-mnemonic] { =/= | PRT/= | PRN/= | PCH/= } [{ SAVE | LABEL | LABELS }]

Format 2:

PB [unit-mnemonic] { file-id | integer } [option-1 [option-2] . . .]

If specified, the unit-mnemonic must be a tape (MT) or disk (DC, DK, or DP) device, and indicates to the MCP the location of the requested backup file or files. If the unit-mnemonic is omitted, the default disk will be assumed (refer to the BD message).

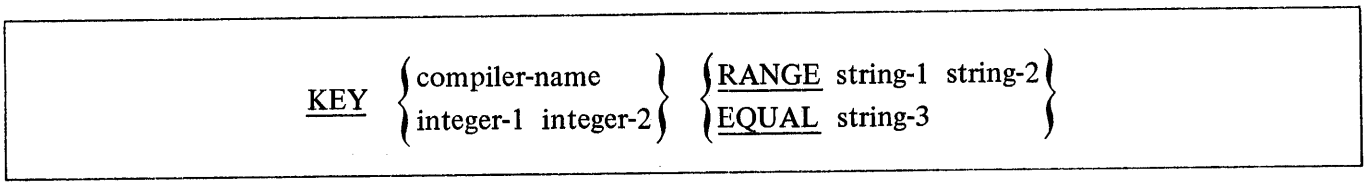
Format 1 is used to print or punch a number of backup files with one program. When the =/= option is used, all backup files existing on the designated disk or tape at the time the message is entered are printed or punched. If both printer and punch backup files exist on the disk, two copies of SYSTEM/BACKUP are executed; one copy handles printer files, and the second copy handles punch files. The PRT/= and PCH/= options cause the printing or punching of all printer of punch files, respectively, that exist on the designated unit. PRN/= is an acceptable equivalent of the PRT/= option. When this format is used, the only options that are allowed are SAVE, LABEL, and LABELS. Also note that backup files not having the MCP-generated names (BACKUP.PRT/integer and BACKUP.PCH/integer) are not found by this format.

Format 2 causes the printing or punching of one printer or punch backup file, as specified by the integer or file-id. The file-id form is used for backup files having user-assigned names (for example, those that do not have the MCP-generated names). When format 2 is used, options can be included to control the output and action taken by SYSTEM/BACKUP. A detailed description of these options follows:

Option	Function
COPIES [=] integer	Causes SYSTEM/BACKUP to produce integer copies of the specified backup file. One copy is the default if this option is not specified.
DOUBLE	Causes SYSTEM/BACKUP to double-space the entire printer listing, overriding any carriage control specified in the backup file.
KEY	Allows specification of a range of records to be printed or punched. A detailed description of the syntax is given below. All records in the file will be printed or punched if this option is omitted.

Option	Function
{LABEL / LABELS}	If this option is included, the option must be the only one specified. The option causes SYSTEM/BACKUP to list on the printer the name of each backup file specified in the PB message, followed by the file label that it would print or punch (even for unlabeled files). These file labels contain information such as program-id, file-id, date, time, and MCP level.
unit-mnemonic	If this option is included, it must specify a printer (LP) or card punch (CD or CP). It will cause SYSTEM/BACKUP to direct its output to the designated unit, if that unit is available.
RECORD integer-1 [integer-2]	Allows specification of a range of records to be printed or punched. Output will begin with the physical record specified by integer-1 (the first record in the backup file is record number 1) and continue until the physical record specified by integer-2. If integer-2 is omitted, end-of-file is assumed as the terminator. All records in the file are printed or punched if this option is omitted.
SAVE	Causes SYSTEM/BACKUP to leave the backup file on disk when the file is closed. The file will be removed from disk if this option is omitted. Tape backup files are always closed with release, so specification of SAVE is unnecessary.
SINGLE	Causes SYSTEM/BACKUP to single-space the entire printer listing, overriding any carriage control specified in the backup file.

The complete syntax for the KEY option follows:



Use of the KEY option allows specification of a range of records to be printed or punched according to information within the records themselves (e.g. a sequence number). The portion of each record to be compared may be specified, as well as the information that will start and stop the output.

Integer-1 specifies the column number of the subfield to be used for the compare argument, and integer-2 specifies the length. Integer-2 must be greater than 0 and less than 10.

The compiler-name option causes automatic generation of the proper column number and length pair that corresponds to the sequence number field of the output listing produced by the specified compiler. The permissible compiler-names that can be used are: BASIC, COBOL, DASDL, FORTRAN, MIL, NDL, RPG, SDL, and UPL.

The RANGE and EQUAL parameters specify the argument to which the subfield in each record is to be compared, and the action to be taken when a "true" comparison is detected. The strings can be either an integer or an alphanumeric literal enclosed within quote marks. When the comparison arguments are of

different lengths, an integer string is left-truncated or left zero-filled to the same length as the subfield; an alphanumeric literal is right-truncated or right space-filled to the same length as the subfield.

If EQUAL is specified, printing or punching will begin when an exact comparison is made between the subfield and string-3, and will continue until end-of-file is reached.

If RANGE is specified, printing or punching will begin when an exact comparison is made between the subfield and string-1. The printing and punching continues until an exact comparison is made between the subfield and string-2, or until end-of-file is reached, whichever occurs first.

If string-1 is equal to string-2, the entire backup file will be searched. Every record in which the designated subfield matches string-1 is printed or punched.

Since the specified comparisons require an exact match between the string and the subfield, no sequential ordering of the backup file is necessary.

NOTE

If both the RECORD option and the KEY option are specified in the same statement, the comparisons specified by the KEY option will be made only within the range of records specified by the RECORD option.

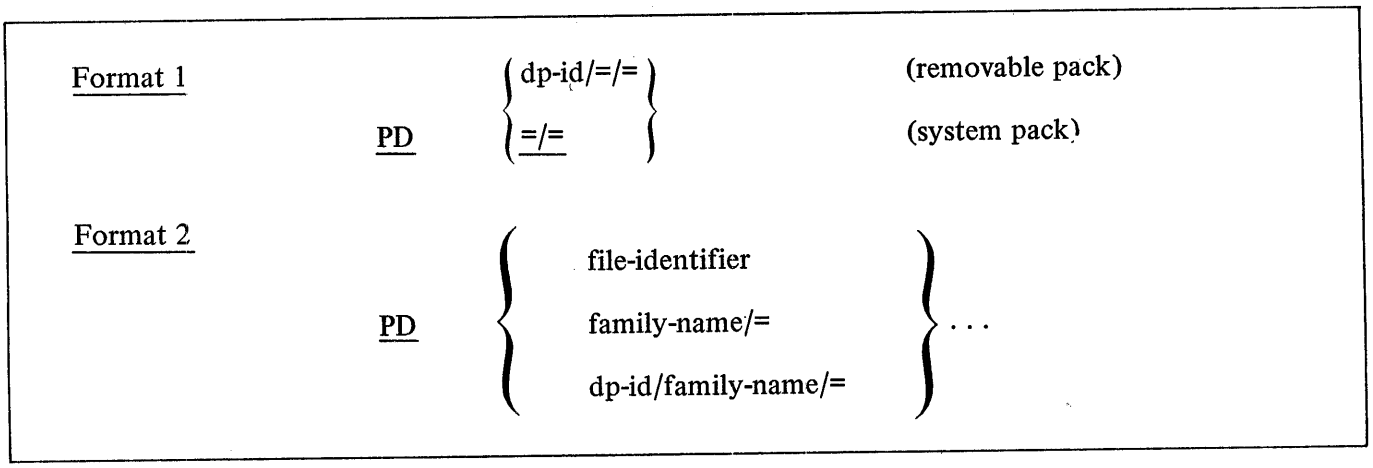
Examples:

```
PB 125
PB DISK/DOCUMENT
PB 17 LPA SAVE
PB =/= LABELS
PB DCC 4 RECORD 5
PB USERPACK/BACKUP/FILE
PB MTA =/=
PB MTC PRT/= LABELS
PB 3 KEY COBOL RANGE 123 567
PB 2 KEY 7, 6 EQUAL "ABC"
PB 53 RECORD 1 100 DOUBLE SAVE
USER SITE/A PB 315
USER PRIV PB=/= SAVE
```

PD INPUT MESSAGE (Print Directory)

The PD input message allows a system operator to request a list of all files on a disk directory or to interrogate a disk directory for a specific file(s).

The PD message has two formats:



The format 1 message will give a complete listing of all files in a disk directory.

The format 2 message will give a partial listing of the files in a disk directory.

The family-name/= format will list all files with the specified family-name.

If the file-identifier is not present in the disk directory the MCP will respond with the message:

file-identifier NOT IN DIRECTORY

Examples:

Does a file named COBOLZ reside on the system pack?

request: PD COBOLZ

response: PD = COBOLZ (affirmative response)

What files reside on the system pack?

request: PD =/=

response: PD = file-identifier-1

PD = file-identifier-2

PD = . . .

Does a family-name PAYROLL with a file-identifier QUARTERLY reside on a removable pack called MASTER?

request: PD MASTER/PAYROLL/QUARTERLY

response: PD = MASTER/PAYROLL/QUARTERLY

Do the files ALPHA, BETA, CHARLIE, reside on the system pack?

request: PD ALPHA, BETA, CHARLIE

response: PD = ALPHA

PD = BETA

CHARLIE NOT IN DIRECTORY

PG INPUT MESSAGE (Purge)

The PG message permits the system operator to purge a removable disk cartridge, disk pack, or magnetic tape.

The format of the PG message is:

PG unit-mnemonic [serial-number]

A disk cartridge/pack that is purged will be marked as UNRESTRICTED with its disk pack-id remaining unchanged.

The serial-number is required when purging a disk, and must be a six-digit number matching the serial number of the pack being purged.

Magnetic tape must have a write ring in place in order to be PURGED.

The serial-number is not used when purging a tape, and the serial number in the tape label will not be changed. To assign or change a tape serial number, use the SN message.

Examples:

PG DPA 000456

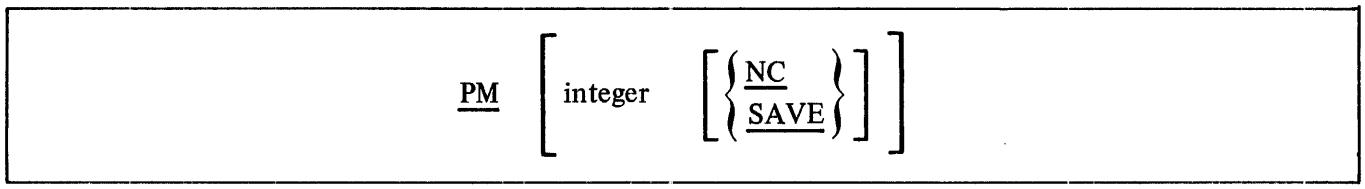
PG MTC, MTD

PM

PM INPUT MESSAGE (Print Memory Dump)

The PM input message allows a system operator to print the entire contents of memory or single program dump file.

The format of the PM message is:



A PM by itself causes the execution of the MCPII/ANALYZER program which will analyze and print the contents of SYSTEM/DUMPFILE. (System Memory)

The "integer" option causes the execution of the DUMP/ANALYZER program which analyzes and prints the contents of DUMPFILE/integer. (Program Memory)

The programs DUMP/ANALYZER, NDL/DUMP, and MCPII/ANALYZER must be located on systems disk to perform a PM message.

The SAVE option causes DUMP/ANALYZER to leave the specified DUMPFILE on disk at EOJ; without this option, the DUMPFILE will be removed from disk.

The NC option causes the execution of NDL/DUMP, which analyzes and prints the specified DUMPFILE (which must be of an NDL-generated network controller). NDL/DUMP does not remove the DUMPFILE from disk at EOJ; therefore, the SAVE option is not included if NC is specified.

Examples:

```
PM
PM 35
PM 201 SAVE
PM 10 NC
```

PO INPUT MESSAGE (Power Off)

The PO input message informs the MCP that a removable disk pack or cartridge is to be removed from the system.

The format of the PO message is:

PO unit-mnemonic

A system pack may not be powered off.

A PO message entered for a unit that is currently being used will cause the MCP to display the following message:

unit-mnemonic HAS integer USERS

A PO message entered for a unit that is not currently in use will cause the message:

unit mnemonic MAY NOW BE POWERED DOWN

to be displayed.

The PO message may be used on a multi-pack file base pack if there are no single-pack files in use at the time of the request.

PR

PR INPUT MESSAGE (Change Priority)

The PR input message allows the system operator to change to priority of a program that is currently in the MIX.

The format of the PR message is:

```
mix-index PR [=] integer
```

See the PRIORITY Control Instruction Attribute for a further explanation of priority.

PS INPUT MESSAGE (PROD Schedule)

The PS input message gives the system operator the ability to request that the MCP attempt to execute the "top" entry in the ACTIVE SCHEDULE.

The format of the PS message is:

PS

The normal function of the MCP checks the ACTIVE SCHEDULE at each EOJ or when a program is scheduled. The PS message will cause the MCP to check the ACTIVE SCHEDULE when the message is entered.

QC

QC INPUT MESSAGE (Quit Controller)

The QC input message allows the system operator to bring the NDL-generated Network Controller to End-of-Job.

The format of the QC message is:

QC

There can be only one NDL-generated Network Controller executing on the system. If any additional Network Controllers are attempted to be executed the following message will be output:

NETWORK CONTROLLER ALREADY RUNNING

After entering the QC message and all activity in the NDL system has stopped, the Network Controller issues STOP codes to all attached stations and then goes to End-of-Job.

If a station for any reason is unable to receive its output messages, the Network Controller waits indefinitely.

With a MCS in the system, the QC message is invalid and its function should be performed within the MCS.

QF INPUT MESSAGE (Query File)

The QF input message allows the system operator to interrogate a program on disk for the status of its control attributes.

The format of the QF message is:

QF program-identifier control-attribute-identifier [. . .]

Examples:

QF A/B CG

QF A/B FILE LINE BACKUP

QP

QP INPUT MESSAGE (Query Program)

The QP input message allows the system operator to interrogate a program while running on the system for the status of its control attributes.

The format of the QP message is:

```
QP job-number control-attribute-identifier [ . . . ]
```

Examples:

QP 14 PRIORITY

QP 15 CHARGE FREEZE

{RB / RF} INPUT MESSAGE (Remove Backup and Dump Files)

The RB or RF input message gives the system operator the ability to remove backup and memory dump files from disk.

The format of the RB, RF message is:

{RB / RF} [pack-identifier/] { =/= / integer / DMP/= / PRT/= / PRN/= / PCH/= }

The integer will remove the backup file specified by the integer.

The PRT/= and PCH/= options remove either all print backup files or all punch backup files, respectively. PRN is equivalent to PRT. Backup files that do not have the MCP-generated name (BACKUP.PRT/integer or BACKUP.PCH/integer) are not removed.

The DMP/= option removes all memory dump files (DUMPFILE/=) from the systems disk. The location of memory dump files is not influenced by the setting of the default backup disk designation.

The =/= option will remove from disk all memory dump files and all backup files having the MCP-generated backup file name.

The pack-identifier option specifies that the backup files to be removed are on the designated user disk.

Examples:

```
RB 37

RB =/=

RF PCH/=

RB USERPACK/=/=

RB USER/=/215

USER SITE/A RB PRT/=
```

RC

RC INPUT MESSAGE (Recover Database)

The RC input message is used in conjunction with DMSII audit and recovery to initiate recovery on a database that was in use when a system abort occurred.

The format of the RC message is:

```
RC database-name [ON pack-id] [file-identifier, . . .]
```

The program RECOVER/DATA.BASE must be present on system disk, and is executed by the MCP to begin recovery of the database.

The "ON pack-identifier" option is required if the data base dictionary resides on a user disk.

The list of file-identifiers allows a partial dump recovery to be performed on the specified DMSII files. This may be done if a subset of the data base, excluding the data base dictionary, has been lost or corrupted. Partial dump recovery requires that both a current copy of the dictionary and a backup copy corresponding to the files to be recovered be present on disk. The old dictionary must be labeled < database-name >/ OLD.DICT.

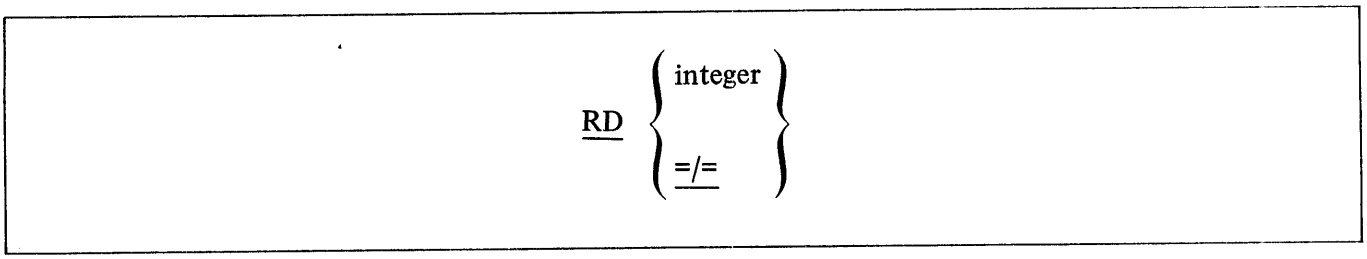
Examples:

```
RC STUDENTDB

RC PAYROLL ON USER1

RC UNIV FILE1, FILE2, FILE3
```

RD INPUT MESSAGE (Remove Pseudo Card Files)

The RD input message allows the system operator to remove pseudo card files from disk.

The format of the RD message is:



RL

RL INPUT MESSAGE (Relabel User Pack)

The RL input message gives the operator the ability to change the disk-pack-id and/or the type of user pack.

The format of the RL message is:

RL unit-mnemonic disk-pack-id { R / U }

RM INPUT MESSAGE (Remove Duplicate Disk File)

The RM input message allows the system operator to remove a disk file from the disk directory in response to a DUPLICATE FILE ON DISK message.

The format of the RM message is:

mix-index RM

The DUPLICATE FILE message is a result of a program trying to close a disk output file with the same name as a file already in the directory. This causes the program to go into a wait state. The RM message will remove the old file, close the new file, enter it in the directory, and continue processing.

Example:

```
1 RM
```

RN

RN INPUT MESSAGE (Assign Pseudo Readers)

The RN message is used by the system operator to assign a specific number of pseudo card readers.

The format of the RN message is:

RN integer

The RN message can be entered either before or after the creation of pseudo files.

It is the responsibility of the operator to determine the optimum number of pseudo readers in relation to the number of pseudo files to be processed.

By entering RN 0 (zero) all pseudo card readers will be closed as soon as they are finished processing the file that they are presently reading.

The pseudo card readers may also be closed by performing a Clear/Start.

RO INPUT MESSAGE (Reset Option)

The RO message allows the system operator to reset the options used to direct or control some of the MCP functions.

The format of the RO message is:

```
RO option-name-1 [,option-name-2] . . .
```

The MCP replies with a verification that the option has been reset after each RO input message.

Certain options cannot be reset by the RO message. If an attempt is made to reset any of these options with the RO message, the MCP displays the following error message:

option-name LOCKED

The TO message may be entered to determine which options are set at any given time. The option indicator equals one when set and zero when reset. A complete list of the MCP options and their status will be displayed.

Examples:

```
RO LIB

  LIB=0

RO DATE,TIME

  DATE=0
  TIME=0
```

RP

RP INPUT MESSAGE (Ready and Purge)

The RP message entered by the system operator will set a tape unit in READY status and PURGE the tape.

The format of the RP message is:

RP unit-mnemonic-1 [,unit-mnemonic-2] . . .

The RP message can be used for tape only.

Examples:

RP MTA

RP MTC, MTD

RS INPUT MESSAGE (Remove Jobs from Schedule)

The RS input message will allow the system operator to remove a job from the schedule prior to its being entered in the MIX for execution.

The format of the RS message is:

RS { job-number-1 [,job-number-2] ... }
   { =   }

The RS message can remove one or more jobs from the schedule.

The schedule number is the number assigned to the job by the MCP when it is entered into the schedule.

The job-number will be displayed by the MCP when the job is entered into the schedule if the SCHM option is set. The WS message will display the jobs in the schedule together with their job-numbers.

The "=" option will remove all jobs from the schedule.

If the requested program(s) are not in the schedule, the MCP will notify the operator that an invalid request has been entered.

Example:

```
RS   33 ,  34 ,  35 ,  36

#33 RS-ED
#34 RS-ED
#35 RS-ED

36 NULL SCHEDULE          (job 36 not in schedule)
```

RT

RT INPUT MESSAGE (Remove Multi-Pack File Table)

The RT input message allows the operator to remove an entry from the multi-pack file table.

The format of the RT message is:

RT file-identifier

Examples:

RT USER/A/B

RT BASEPACK/MASTER/

RY INPUT MESSAGE (Ready Peripheral)

The RY input message allows the system operator to ready a peripheral unit and make it available to the MCP.

The format of the RY message is:

```
RY unit-mnemonic-1 [,unit-mnemonic-2] . . .
```

Any number of units may be made ready with one RY message.

When a removable disk cartridge or disk pack is placed on a system, the MCP must be notified of its presence with the RY message.

If the designated unit is not in use and is in the remote status, the RY message causes all exception flags maintained by the MCP for the specified unit to be reset. After the unit has been made ready, the MCP attempts to read a file label (input devices only).

An RY message referencing a card reader causes the MCP to read the next card in the input hopper. If this card is a control card, the MCP does what is requested.

The only exception to this is the case where a card reader has been "reserved" for a program by job-number because the COMPILE or EXECUTE control statement immediately precedes the DATA statement without an intervening END statement. Without operator intervention (IL or UL message), such reserved card files can be opened only by the program for which they are reserved and, if unopened, remain reserved even after the program terminates or is RS-ed. The RY message has no effect on a card reader that is "reserved" in this manner. The CL message permits the operator to remove the reserved condition from the reader and allow it to be assigned to any job that opens it with the correct file-identifier. To completely free the reader by removing the file-identifier, follow the CL message with an RY message.

Examples:

```
RY DPB

RY MTC

RY LPA, LPB
```

SD

SD INPUT MESSAGE (Assign Additional System Drives)

The SD input message gives the system operator the ability to assign additional system drives for the MCP.

The format of the SD message is:

```
SD unit-mnemonic  serial-number
```

The SD message, after verification of the serial-number, will PURGE the pack, and add it to the system packs already on the system.

At COLDSTART, there is only one system drive, so additional drives may be added by the SD message. Once a system drive has been added to the system, it cannot be removed without performing a COLDSTART.

The following message is displayed when the new system drive is linked to the system.

unit-mnemonic IS NOW A SYSTEM PACK–CLEAR START REQUIRED

SE INPUT MESSAGE (Enable Console Switches)

The SE input message allows the system operator to enable sensing of the twenty-four Data Entry Switches on the B 1800/B 1700 console (while in RUN mode) by specified classes of programs and firmware.

The format of the SE message is:

```
SE integer
```

The value of the integer specifies the programs that are allowed to sense the Data Entry Switches, as follows:

Value of Integer	Program
0	Disable Switches
1	MCP
2	Normal-state Programs
4	Interpreters
8	GISMO

The integer may also be any sum of the above values, in which case multiple classes of programs are allowed to sense the Data Entry Switches.

Only SDL/UPL and MIL programs contain the source language constructs required to sense the Data Entry Switches. In both languages the CONSOLE.SWITCHES construct is used, and provides the 24-bit image of the Data Entry Switches.

Examples:

```
SE 2

SE 10
```

SL INPUT MESSAGE (Set LOG)

The SL input message gives the operator the ability to set the LOG or SPOL option, and allocate the area required.

The format of the SL message is:

```
SL [SPO] integer-1 [integer-2]
```

If the SPO option is included, the SL message affects the SPOLOG; otherwise, the system LOG is assumed.

The integer-1 entry is the size of each disk area to be assigned to the LOG or SPOLOG. It cannot be less than 100 or greater than 10,000 disk segments.

The integer-2 entry is optional and is the maximum number of areas desired. It must be between 2 and 105, inclusive. Default is 25.

The MCP responds with the following message when an SL message has been entered:

log-name NOW SET–CLEAR START REQUIRED

If there is insufficient disk space for the first area of the specified log, the following message is displayed:

NO SPACE FOR NEW log-name

If integer-1 is zero, the LOG or SPOL option is reset and the log is transferred (as though a TL message had been entered). A new LOG file is not created.

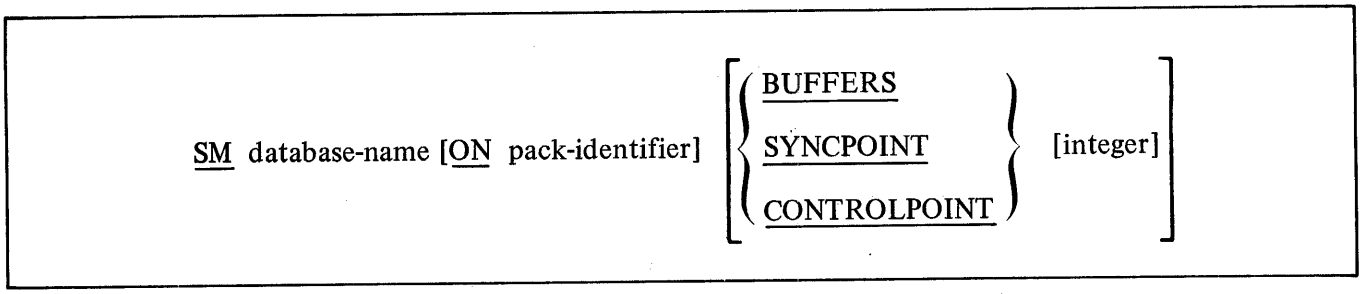
Examples:

```
SL 1000
SL 250 5
SL 0
SL SPO 2000
```

SM INPUT MESSAGE (Set Data Base Parameters)

The SM input message is used in conjunction with DMSII to set or interrogate certain parameters within a data base.

The format of the SM message is:



The "ON pack-identifier" clause is required if the data base dictionary resides on a user disk.

The integer given is the new value for the specified parameter; if omitted, the MCP displays the current setting of the parameter. The new value becomes effective the next time the data base is opened.

Examples:

```
SM STUDENTDB BUFFERS 25
  BUFFERS PARAMETER FOR DATA BASE STUDENTDB CHANGED FROM 5 TO  25

SM PAYROLL ON USER1 SYNCPOINT
  SYNCPOINT PARAMETER FOR DATA BASE PAYROLL = 5
```

SN

SN INPUT MESSAGE (Assign a Tape Serial Number)

The SN input message is used to initialize (and purge) a magnetic tape, assigning a volume serial number to the tape label.

The format of the SN message is:

SN unit-mnemonic serial-number [ASCII]

The SN message initializes a magnetic tape by putting a scratch ANSI label on the tape. Any tape that does not contain a valid ANSI label cannot be purged (PG message). This includes both unlabeled tapes and those that have the Burroughs standard label.

The serial-number is normally numeric, but any alphabetic or numeric string up to six digits in length is allowed. This serial number is placed in the label, and remains in all labels on the tape (even when purged). The serial number can be explicitly changed by another SN message.

ANSI tape labels may be maintained in either EBCDIC or 8-bit ASCII recording mode. The label recording mode is determined by the SN message, and remains unaltered (even by PG) until explicitly changed by another SN message. EBCDIC recording mode is the default, and is used unless the ASCII option is included in the SN message. Note that the recording mode of the tape label is completely independent of the mode in which the data are recorded on the tape.

Example:

```
SN MTA 123456

SN MTC BACKUP ASCII
```

SO INPUT MESSAGE (Set Option)

The SO input message allows the system operator to set the options used to direct or control some of the MCP functions.

The format of the SO message is:

```
SO option-name-1  [,option-name-2] . . .
```

The MCP replies with a verification that the option has been set after each SO input message.

Certain options cannot be set by the SO message. If an attempt is made to set any of these options with the SO message, the MCP displays the following error message:

option-name LOCKED

The TO input message may be entered to determine which options are set at any given time. The option indicator equals one when set and zero when reset. A complete list of the MCP options and their status will be displayed.

SP

SP INPUT MESSAGE (Change Schedule Priority)

The SP input message provides a means for the system operator to change the schedule priority of a program currently in the schedule.

The format of the SP message is:

SP job-number integer

The Schedule Priority is separate from the priority of the job when it is in the mix.

The job-number will identify the program in the schedule that is to be affected by the SP message.

The integer in the SP message specifies the new priority that will be assigned to the program. Priorities may range from zero through 14, where zero is the lowest priority and 14 is the highest priority.

To change the priority of a program in the schedule with a job-number of 33 to a priority of 7, the following SP message would be used.

```
SP  33  7
```

This program would be selected from the schedule ahead of the other programs with a lower priority.

The following message would be displayed in response to the above input message:

```
program-name  33  PR = 07
```

SQ INPUT MESSAGE (Squash Disk)

The SQ input message permits the system operator to initiate or terminate a "disk squash." When "squashing" a disk, the MCP attempts to move areas of data to numerically-lower disk addresses in order to alleviate disk checkerboarding.

The format of the SQ message is:

SQ unit-mnemonic [integer-1] [{ SIZE / STOP / TILL integer-2 }]

The unit-mnemonic specified must be a disk device (DC, DK, or DP). If head-per-track disk (DKx) is designated, integer-1 must be used to indicate the electronics unit (EU); integer-1 is not used with other types of disk.

A disk squash cannot be initiated if there are any jobs in the mix or in either schedule (WAITING or ACTIVE). Once the disk squash is initiated, only additional SQ input messages can be entered on the console printer. All card readers are inaccessible during the disk squash.

Both system and user disks can be squashed. With multiple system disks, only one drive or EU can be squashed at a time. The MCP automatically produces a KA listing of the specified disk both before and after the disk squash; therefore, a line printer must be available.

If the TILL option is specified, the MCP will terminate the disk squash as soon as integer-2 contiguous sectors have been made available. Otherwise, the squash will continue until normal termination or until explicitly stopped by the operator with the STOP option. The MCP displays the largest area currently available whenever interrogated by the SIZE option.

When the disk squash is terminated, the MCP displays the size of the largest available area as well as the total number of sectors available on the designated disk.

Examples:

```
SQ DPA

SQ DKB 1

SQ DCC TILL 5000

SQ DPB SIZE

SQ DCA STOP
```

ST

ST INPUT MESSAGE (Suspend Processing)

The ST input message provides a means for the system operator to temporarily suspend the processing of a program in the MIX.

The format of the ST messgae is:

mix-index ST

The mix-index identifies the program to be suspended.

The MCP will not suspend the program until all I/O operations in progress for that program have been completed.

When the MCP suspends a program, it is rolled-out to disk and the memory it was using is returned to the MCP for reallocation.

A suspended program will retain the mix-index and peripherals assigned to it; the MCP will use this to identify the program when referenced by another keyboard input message.

To restart a program after it has been suspended, either the OK or the GO message must be used. If for some reason all of the conditions necessary for the program to run are not met when the OK or GO message is issued, the MCP will not restart the program.

Example:

```
3) ST
```

SV INPUT MESSAGE (Save Peripheral Units)

The SV message allows the system operator to make a peripheral unit inaccessible to the MCP until a Clear Start operation occurs, or an RY input message is used to ready the unit.

The format of the SV message is:

```
SV unit-mnemonic [,unit-mnemonic] . . .
```

Any number of peripheral units may be saved with one SV input message.

When the SV message is entered and the unit is not in use, the specified unit is marked SAVED and "unit-mnemonic SAVED" is displayed by the MCP.

If the unit is in use, the MCP will respond with "unit-mnemonic TO BE SAVED" and will save the unit as soon as it is no longer being used.

Example:

```
SV LPA
```

SW

SW INPUT MESSAGE (Set Switch)

The SW input message allows the system operator to set programmatic switches.

The format of the SW message is:

mix-index SW { integer [=] / = } value

Programmatic switches may also be set at schedule time by using the SWITCH control statement attribute. Refer to Section 2, SWITCH control attribute.

The number must be a decimal digit from zero to nine (0-9) that references the switch to be set. To determine what switches are available, the specific language manual for the program for which the switches are being set must be referenced. If the "=" option is used, all ten switches are implied (40 bits of information).

The value is the value that the switch or switches will be assigned.

Examples:

```
5 SW1 = @F@

2 SW8 = 6

3 SW  = @0123456789@
```

TD INPUT MESSAGE (Time and Date)

The TD input message allows the system operator to request that the MCP type the current values of the time and date.

The format of the TD message is:

```
TD
```

The MCP displays the date and time in the following format:

DATE = mm/dd/yy TIME = hh:mm:ss.t

Where:

hh – hours
mm – minutes
ss – seconds
t – tenths of seconds

TI

TI INPUT MESSAGE (Time Interrogation)

The TI input message allows the system operator to interrogate the MCP as to the amount of processor time the program has used up to the time the interrogation was made.

The format of the TI message is:

mix-index TI

The mix-index identifies the program for which the interrogation was requested.

The time is given in hours, minutes, seconds, and tenths of seconds.

Example:

```
4TI
COBOL: A/B = 4    CPU TIME = 00:03:15.7
```

TL INPUT MESSAGE (Transfer LOG)

The TL input message allows the system operator to transfer the information in the SYSTEM/LOG or SYSTEM/SPOLOG file to LOG/#integer or SPOLOG/#integer. To transfer and print the log file, refer to the LG input message.

The format of the TL message is:

```
TL [SPO]
```

If the SPO option is included, SYSTEM/SPOLOG is transferred; otherwise, SYSTEM/LOG is transferred.

The TL message transfers the specified log file and creates a new (empty) log file. It does not cause the file to be printed.

TO

TO INPUT MESSAGE (Display Options)

The TO input message allows the system operator to interrogate the status of the MCP options.

The format of the TO message is:

```
TO [option-name-1 [,option-name-2] . . .]
```

The TO message entered by itself will display all of the options and their settings.

A value of zero (0) indicates a reset (off) condition; a value of one (1) indicates a set (on) condition.

Example:

```
TO LOG, TIME

  LOG = 1

  TIME = 0
```

or:

```
TO

  BOJ = 0 DATE = 1 . . . (lists all options)
```

TR INPUT MESSAGE (Time Change)

The TR message allows the system operator to change the current value of the time maintained by the MCP.

The format of the TR message is:

```
TR integer
```

The time specified by the integer is designated according to a 24-hour clock, and must be four digits in length.

This message is not accepted by the MCP if the value of the integer is greater than 2400 hours.

Example:

Set the time in the MCP to 7:19 P.M.

```
TR 1919
```

TS

TS INPUT MESSAGE (Test Switches)

The TS input message allows the system operator to test the programmatic switches set by the SW console message or the SWITCH control statement attribute.

The format of the TS message is:

mix-index TS

The output of the TS message is in hexadecimal format.

Example:

4 TS

PAYROLL/103 = 4 SWITCHES = @0123456789@

UL INPUT MESSAGE (Assign Unlabeled File)

The UL message allows the system operator to designate the unit on which a particular unlabeled input file is located in response to a "FILE NOT PRESENT" message from the MCP.

The format of the UL message is:

```
mix-index UL unit-mnemonic [integer]
```

The UL message is used only if the unit designated is to be acted on as an unlabeled file. The MCP assumes the file on the designated unit is the file requested by the program that caused the "FILE NOT PRESENT" message. The device specified in the UL message must be ready when the message is entered.

Unlabeled card reader files are not allowed by the MCP. Therefore, a UL message referencing a card reader functions exactly the same as an IL message.

The mix-index must be used to identify the program to which the file is to be assigned.

If integer is used, the MCP spaces forward "integer" blocks prior to reading the first data block into the object program. Tape marks are read as blocks. This is done at the time the file OPEN is performed.

Example:

A program with a mix-index of 1 calling for an unlabeled input tape file could be assigned a tape on a unit with the unit-mnemonic of MTA with the following UL message:

```
1 UL MTA
```

If the first three blocks on the tape are not desired, they can be skipped with the following UL message:

```
1 UL MTA 3
```

WD

WD INPUT MESSAGE (Display MCP Date)

The WD input message permits the system operator to request the current date used by the MCP.

The format of the WD message is:

```
WD
```

WM INPUT MESSAGE (Display Current System Software)

The WM input message allows the system operator to inquire which MCP and related System Software are currently being used since there can be more than one version residing on the system pack.

The format of the WM message is:

```
WM
```

The reply to the WM message is in the following format:

```
MCP = mcp-name INTERP = interpreter-name GISMO = gismo-name INIT = initializer-name
MICRO.MCP = micro-mcp-name NETWORK.CONTROLLER = network-controller-name
USERCODE.FILE = usercode-file-name
GISMO ALSO CONTAINS:
segment-name-1
segment-name-2

        .
        .
        .

segment-name-n
```

WS

WS INPUT MESSAGE (Display Schedule)

The WS input message allows the system operator to interrogate what program or programs are currently in the schedule and their status.

The format of the WS message is:

```
WS { job-number }
   {    =     }
```

The job-number is assigned by the MCP as the program is entered into the schedule.

The MCP response to the WS message gives the program-name, schedule number, memory required in KB's, program priority, and the length of time the program has been in the schedule.

Example:

```
WS 4

ALPHA = 4 NEEDS 8 KB PR = 4 IN FOR 00:08:37.4
```

WT INPUT MESSAGE (Display MCP Time)

The WT input message permits the system operator to request the current time used by the MCP. The reply is in the twenty-four hour clock format.

The format of the WT message is:

WT

WW

WW INPUT MESSAGE (List Contents of NAME TABLE)

The WW input message gives the operator the ability to list the different types of system software/ firmware in the NAME TABLE.

The format of the WW message is:

```
WW  { =/=                    }
    { system-software-mnemonic }
    { software-group/=       }
```

See the Clear/Start procedure for an explanation of the system software-mnemonics used in the NAME TABLE.

The =/= option displays the entire contents of the NAME TABLE on the SPO. If an individual system-software-mnemonic is specified, only the contents of the designated NAME TABLE entry are displayed. The software-group/= option permits portions of the NAME TABLE to be displayed, as follows:

N/= System Initializer Entries (N, NX)

G/= GISMO Entries (G, GT, GX)

I/= Interpreter Entries (I1, I2, I1T, I2T, IX)

M/= MCP Entries (M, MT, MX)

MM/= MICRO.MCP Entries (MM, MMX)

S/= Stand-alone Entries (SD, SDD, SDL, SIO, SL, SX)

C/= Network Controller Entries (C)

U/= Usercode-Password File Entries (US)

Examples:

```
WW GX
  GX = "GISMO/DEBUG"

WW MM/=
  MM = "MCPII/MICRO.MCP"
  MMX = "MICRO.MCP/DEBUG"
```

WY INPUT MESSAGE (Program Status Interrogation)

The WY message allows the system operator to check the current status of one program or all the programs in the MIX.

The format of the WY message is:

```
[mix-index] WY
```

The mix-index identifies the program in the MIX that is to be checked and its status displayed on the console printer. If the mix-index is omitted the MCP will display the status of every program in the MIX.

If the program is waiting for some type of operator action, the alternatives available to the operator will be identified.

Examples:

```
1 WY

    PAYROLL/PAY105=1 IL UL DS-NO FILE "PAYROLL/MASTER"

WY

    COBOL; LISTER=1 EXECUTING

    DMPALL=2 AX DS-WAITING FOR KEYBOARD INPUT

    USER/ACCPAY/=3 WAITING FOR I/O COMPLETE ON "CARDS" (CRA)
```

{XC / XD} INPUT MESSAGE

The XC and XD input messages allow the removal of contiguous disk segments from the MCP tables of available disk space temporarily (XC) or permanently (XD).

The format of XC, XD message is:

{XC / XD} unit-mnemonic [integer-1] interger-2 integer-3

The unit-mnemonic specified must be a disk device (DC, DK, or DP). If head-per-track disk (DKx) is specified, integer-1 must be used to indicate the electronic unit (EU). Integer-1 is not used when other types of disk are specified.

Integer-2 specifies the beginning segment address, and can be expressed in any format. If the operation is being performed on a disk cartridge (DCx), and the beginning segment address is not the address the first segment in a track, the MCP will automatically adjust it backward to the beginning of the track.

Integer-3 specifies the number of segments to be removed from use by the MCP. If the operation is being performed on a disk cartridge (DCx), the number of segments removed will always be a multiple of entire tracks. The MCP will make the necessary adjustments, both in starting address and number of segments.

The disk space to be removed must be available in order to be removed; therefore, if any portion of the space is occupied by files or headers, for example, the MCP will reject the request with the message:

REQUESTED SEGMENTS NOT REMOVED–NOT AVAILABLE

The requested disk space is permanently removed from use by the XD message. To return the removed segments a disk initialization (for disk packs or cartridges) or COLDSTART (for head-per-track disk) is required.

The XC message temporarily removes the disk space from use. The disk space is returned at the next CLEAR/START or, for user packs or cartridges, when the disk is powered down (PO message).

Examples:

```
XC DKA 0 200 1000

  DISK SPACE REMOVED FROM @EE00000C8@ THRU @EE00004AF@

XC DCC @46@ 30

  DISK SPACE REMOVED FROM @EA2000040@ THRU @EA200007F@
```

JOB SPAWNING INSTRUCTIONS

General

Job spawning can be generally defined as the ability to programmatically execute jobs and maintain an adequate level of control over these jobs during their execution. The mechanism to programmatically spawn and control jobs from an executing program requires the following specialized extensions to the MCP and control instruction handling:

a. Extensions to the ZIP function to allow propagation of control information when a spawned job spawns another job.

b. Control card commands to associate the necessary control information with spawned jobs.

c. MCP functions to recognize and apply the control information to such areas as job scheduling, message routing, file naming, and other functions.

The MCP job spawning mechanism is designed to allow a certain amount of flexibility in selecting the level of control over a spawned job. For example, it is possible to select a mode of operation where a controlling program is informed only of the birth and death of a spawned job, or to select a mode whereby the controlling job essentially becomes the SPO for the spawned job and receives all communication normally associated with the system SPO.

Jobs are spawned through use of extensions to the ZIP construct and are controlled through queue files referred to as the Control Queue and the ZIP Queue. The Control Queue is used by a controlling program to receive messages from the MCP regarding spawned jobs. The ZIP Queue is similar to the Control Queue, except that the ZIP Queue is used exclusively for job spawning. Refer to the ZQ control instruction for specific details on the differences between Control Queues and ZIP Queues. Messages in the Control Queue and ZIP Queue are variable in size. When opened, the attributes of the queue must be declared large enough to contain the messages anticipated from the MCP (both RECORD.SIZE and Q.MAX.MESSAGES).

Messages received in a Control Queue or ZIP Queue have a standard 50-byte header, as follows:

Field	Size	Description
Message Type	2 bytes	01 = DATA card label message 02 = Special schedule message 03 = SPO input message 04 = SPO output message 05 = Special BOJ message 06 = Special EOJ message 07 = Backup file ready message 08 = ZIP message from spawned job
Job Number	5 bytes	
Usercode	10 bytes	
Session Number	5 bytes	
Time Stamp	6 bytes	Time counter value (tenths of seconds)
Message Size	4 bytes	Message Text length (in bytes)
Usercode Index	4 bytes	Index into Usercode/Password File
Filler	14 bytes	Reserved for future expansion
Message Text	variable	Message Size contains length

The textual portion of the message varies according to the Message Type field, as follows:

Message Type	Message Text
01	The first ten bytes contain the actual label detected by the MCP on a DATA card (for example, "CARDS"). Note: This message type only appears in a ZIP Queue.
02	The first ten bytes contain the program pack-identifier; the next byte, if a "1", indicates that an FW command is required; and the next five bytes contain the job summary backup file number intended for HOST RJE).
03	Standard SPO input text (for example, "1AX . . .").
04	Standard output text which would normally appear on the system SPO.
05	The first two bytes contain the program mix-index.
06	The first two bytes contain the job termination type, as follows: 00 = Normal EOJ 01 = Aborted (DS or DP) 02 = Syntax Errors 03 = Not used 04 = RS-ed If the termination type is 02, the next four bytes contain the syntax error count.
07	Contains the 30-byte < file-identifier > of the backup file created: 10-byte < pack-identifier >, 10-byte < usercode >, and 10-byte < backup-file-number >. The first byte of the < backup-file-number > indicates the type of backup file ("#" = printer backup, "%" = punch backup).
08	The text portion of a ZIP from a spawned job which has its LS boolean set.

Job Spawning Control Instructions are valid only when "zipped" from a controlling program. They are not allowed as SPO input messages.

FW COMMAND (File Waiting)

The FW command is used by a controlling program to force a spawned program from the WAITING SCHEDULE into the ACTIVE SCHEDULE when its input data stream has been built.

The format of the FW command is:

SZ integer FW { job-number / = }

The FW command must be preceded by an SZ command specifying the appropriate session number.

A spawned job requiring a card file is placed into the WAITING SCHEDULE by the MCP to prevent its BOJ from occurring before the card input file has actually been created. An indication is returned to the controlling program in the Control Queue or ZIP Queue (Message Type 02) that the job is in the WAITING SCHEDULE and requires an FW command. It is the controlling program's responsibility to notify the MCP (using the FW command) when the card file required by the waiting program has been created.

The FW command is mainly intended for use by the HOST RJE system. When RJE/CONTROLLER receives an input stream, that stream is preceded by a control card section. RJE/CONTROLLER passes this control card section to the MCP with a ZQ message. If the control stream contains a command which causes a job to be scheduled and also contains a DATA command, the MCP places the job in the WAITING SCHEDULE in a special status. In addition, the MCP sends the special schedule message (Message Type 02) back to RJE/CONTROLLER indicating that an FW command is required when the card file has been constructed.

Examples:

```
SZ 1024 FW 735

SZ 15 FW =
```

LS

LS COMMAND (Log SPO)

The LS command causes a special bit (the LS boolean) to be set and carried with a control string and eventually a spawned job.

The format of the LS command is:

```
LS
```

The function of the LS boolean is to cause all control messages (both input and ouput) to be inserted in the Control Queue. A QUEUE command is required in the control string prior to the LS command.

The LS boolean also has the effect of bypassing the system SPO except for error messages requiring operator intervention, or if the RMSG option is set.

A spawned job which has its LS boolean set may not spawn another job with the LS command.

Example:

- The control string "QUEUE X/Y LS EX DMPALL" causes the following messages to be inserted in in the Control Queue:

    a. A special schedule message (Message Type 02).

    b. A message containing the text of the control string (Message Type 03).

    c. A special BOJ message (Message Type 05).

    d. The actual SPO BOJ message text (Message Type 04).

    e. The DMPALL DISPLAY message (Message Type 04).

    f. The DMPALL ACCEPT message (Message Type 04).

    g. Any input messages to DMPALL (Message Type 03), and any subsequent DISPLAY or ACCEPT messages (Message Type 04).

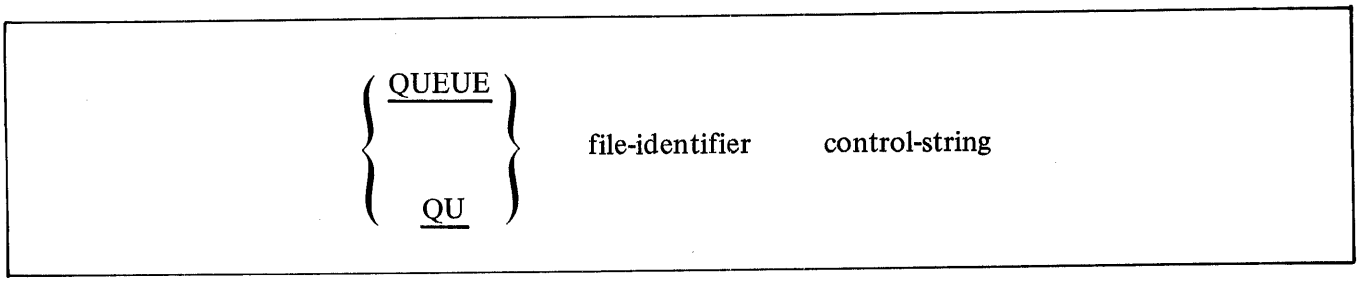
    h. A special EOJ message (Message Type 06).

    i. The actual SPO EOJ message text (Message Type 04).

QUEUE COMMAND (Designate Control Queue)

The QUEUE command is used to designate a previously-opened queue file as a Control Queue.

The format of the QUEUE command is:



The Control Queue provides a mechanism whereby the MCP can communicate to a controlling job on behalf of a spawned job, or in response to other control commands which the controlling job may wish to receive.

For example, the control string "QUEUE X/Y EX DMPALL" causes the MCP to insert in the Control Queue named X/Y the following messages:

a. A special schedule message (Message Type 02).

b. A special BOJ message (Message Type 05).

c. A special EOJ message (Message Type 06) when DMPALL terminates.

The QUEUE command may also be used to obtain MCP responses to normal SPO input messages. For example, the control string "QUEUE X/Y WY" causes the MCP to insert one or more messages in the Control Queue named X/Y with the standard SPO output text of the WY input message (Message Type 04).

SZ

SZ COMMAND (Session)

The SZ command causes a specified session number to be carried with a job or mix of jobs in order to associate independent jobs into logically related groups.

The format of the SZ command is:

```
SZ      integer
```

The SZ command is intended mainly for use by HOST RJE and CANDE. It need not be invoked in the course of simple job spawning.

The session number specified by the SZ command is carried with a ZIPped control string and is applied to a job or mix of jobs depending what other commands are in the control string.

If a job with a session number assigned to it spawns ZIPs another job, the session number is also carried to the ZIPped job.

For example, HOST RJE assigns a session number to a physical site at log-on time. All jobs submitted from that site are given identical session numbers, thus relating them back to the originating remote site. CANDE assigns session numbers in a similar manner, relating spawned jobs and MCP control strings back to the originating terminal.

Examples:

```
QUEUE CANDE/CONTROL USER STUDENT/JOB SZ 7 LS EX TESTPROG

SZ 1024 FW 159
```

ZQ COMMAND (Designate ZIP Queue)

The ZQ command is similar to the QUEUE command except that the specified queue is used exclusively for schedule messages and DATA card messages.

The format of the ZQ message is:

ZQ file-identifier

Where the Control Queue may contain many messages concerning spawned jobs, MCP responses to SPO commands, and so forth, the ZIP Queue only contains schedule records for jobs spawned by the controlling program, and the data card label message if a DATA control card is detected by the MCP in the control string.

The ZIP Queue effectively allows the controlling program to be immediately aware of the fact that a job has been scheduled for execution without the necessity of reading through the general Control Queue for pertinent messages.

In general, the Control Queue is designed for general communications, while the ZIP Queue is specifically used for job spawning control.

Example:

QUEUE A/B ZQ X/Y SZ 1024 LS CO TEST COBOL LI DATA CARDS

This control string causes the following two messages to be placed in the ZIP Queue named X/Y:

a. A DATA card label message (Message Type 01).

b. A special schedule message (Message Type 02). This message will also contain an indication from the MCP that an FW message is required.

MCP OUTPUT MESSAGES

General

The MCP communicates information to the system operator by the console printer. Messages can either be originated by the MCP for information and possible operator action, or they can originate from an executing program. In either case, the MCP has complete control over all messages.

All output console messages are indented one space by the MCP, in order for the operator to easily distinguish them from input messages. Messages originating from within a program (i.e., DISPLAY messages) are preceded by a percent sign (%) in order to more easily distinguish them from messages originated by the MCP concerning the program.

Syntax

The paragraphs below outline the syntax used in defining the MCP messages in this section.

Classification: MCP messages are listed in alphabetical order using the first word of the actual message as the key. The job-specifier portion and any "optional" type entries are not considered part of the key.

Job-Specifier: Job-Specifier is simply used to identify the job for which that message is intended. The format of the job-specifier is:

[[(usercode)] compiler-name:] [(usercode)] program-name = mix-index [SZ = session] . . .

The compiler-name portion is only printed when the executing program is a compilation.

Terminal-reference: The phrase "terminal-reference" following any message indicates that a termination message will be printed if the DISP option is set. Any time this message is printed, the program must be DS-ED or DP-ED, except when the TERM option is set causing the program to be terminated automatically.

The format of the terminal-reference message is:

: P = nn, S = nn, D = nn (@–@, @–@, @–@) DS or DP
or : S = nn, D = nn (@–@, @–@) DS or DP

NOTE

There are situations usually occurring when memory is approaching saturation that the program-identifiers will be omitted from console messages and referenced only by their mix-index number. This is done to conserve memory. For example, the following message:

3 NO MEMORY

might be used instead of

A/B/C = 3 NO MEMORY

MCP Messages

job-specifier–ABORTED

job-specifier–ACCEPT

job-specifier–ACCESS PPB TARGET OUT OF RANGE terminal-reference

ATTEMPTED TO WRITE OUT OF BOUNDS

unit-mnemonic ASSIGNED TO SYSTEM USE

unit-mnemonic AVAILABLE AS OUTPUT

BACKUP FILE nnnnnn NOT REMOVED–NOT ON DISK

BACKUP TAPE NOT FOUND–"RY" unit-mnemonic

BATCH COUNT COMMUNICATE ISSUED WHILE SORTER FLOWING terminal-reference

job-specifier–BEGINNING DATA OVERLAY ADDRESS = nnnn, WHILE BR = nnnn terminal-reference

job-specifier–{ BOJ. / EOJ. / DS-ED. } = job-number PR = nn [integer SYNTAX ERRORS] TIME = hh:mm:ss.

job-specifier–CANNOT ACCEPT "[IL'UL'OF'FR'FM'OU'OK'RM'MR]"MESSAGE

CANNOT ACCEPT DATA STATEMENT FROM THE SPO

unit-mnemonic CANNOT BE OPENED OUTPUT FOR file-identifier

CANNOT CHANGE PACK-ID OR FAMILY NAMES WITH EQUALS . . . id's . . .

CANNOT FIND UNIT REQUESTED FOR FN

CANNOT READ LABEL ON unit-mnemonic

CANNOT READ THE LABEL ON unit-mnemonic

CANNOT REMOVE PACK.ID OR FAMILY NAMES WITH = -S file-identifier

CANNOT SAVE THIS DEVICE unit-mnemonic

file-identifier CHANGED TO new-file-identifier

CHAR OR BIT STRING IS INCOMPLETE . . . . input message . . . .

CLEAR/STARTB1700 MCPII MARK nnn.nn mm/dd/yy hh:mm

***CLEAR/START REQUIRED

CLEAR/START REQUIRED–SYSTEM/PRINTCHAIN MISSING

COMPILE program-name CTRL RCD ERR: . . . .

job-specifier–CONTROL STACK OVERFLOW terminal-reference

job-specifier–"CONVERT" ERROR terminal-reference

COULD NOT CHANGE THE MCP

job-specifier–CPU TIME = hh:mm:ss.t

CURRENT MCP IS identifier USING interpreter-id

job-specifier–DATA OVERLAY RELATIVE DISK ADDRESS = nnnn, WHILE SIZE OF AREA = nnnn terminal-reference

DECK nnnn = 50 CHAR

DECK nnnn IN USE BY program-name

**DECK NUMBER nnnn NOT ON DISK

DEFAULT CHARGE NO. = nnnnnn

DISK ERROR ON OVLY { READ / WRITE } FROM { DISK ADDRESS@nnnn@ / MEMORY ADDRESS@nnnn@ }

job-specifier–DISK FILE DECLARED SIZE EXCEEDED ON file-identifier terminal-reference

job-specifier–unit-mnemonic DISK PARITY @nnnn@

job-specifier–nnnnDISK SEGMENTS REQUIRED FOR AREA OF file-identifier

job-specifier–DIVIDE BY ZERO terminal-reference

**DR PLEASE

job-specifier–DUPLICATE INPUT FILES file-identifier

END BF

END MX

END PD

job-specifier–ENDING DATA OVERLAY ADDRESS = nnnn, WHILE BR = nnnn
terminal-reference

"=" NOT PERMITTED IN FILE NAME FOLLOWING "FN"

unit-mnemonic ERROR/pack-id IS [RESTRICTED or INTERCHANGE] PACK

unit-mnemonic ERROR unit-id

job-specifier–EVALUATION OR PROGRAM PTR STACK OVERFLOW terminal-reference

EXECUTE program-name CTRL RCD ERR: . . .

job-specifier–EXPONENT OVERFLOW terminal-reference

job-specifier–EXPONENT UNDERFLOW terminal-reference

job-specifier–EXPRESSION OUT OF RANGE terminal-reference

job-specifier { INPUT / OUTPUT / INPUT/OUTPUT } FILE file-identifier CLOSED { RELEASE / PURGE / REMOVE / CRUNCH / NO REWIND / CODE / LOCK / CONDITIONAL / ROLLOUT / TERMINATE }

job-specifier–FILE internal-file-identifier LABELED . . . REEL nnnnnn NOT PRESENT

job-specifier–FILE internal-file-identifier NEEDS nnnn BITS TO OPEN, WHICH I COULDN'T FIND–"OK" WILL TRY AGAIN, ELSE "DS"

file name "file-identifier" REQUESTED BY "FN" NOT FOUND

FN = "internal-file-identifier"

FREE UP SOME DISK AND CLEAR/START

GOOD MORNING, TODAY IS name-of-day, hh:mm:ss.t {AM / PM} JLN DT = yy/ddd

unit-mnemonic HAS nnnn USERS

unit-mnemonic HAS BEEN PURGED

job-specifier–unit-mnemonic HOPPER EMPTY

INVALID BIT CHARACTER– . . .

INVALID BIT SPECIFIER– . . .

INVALID CHAR COL nn

INVALID CHARACTER . . .

INVALID CHANGE–PACK-IDS DO NOT AGREE . . . .

job-specifier–INVALID CASE terminal-reference

job-specifier–INVALID COMMUNICATE IN USE ROUTINE terminal-reference

unit-mnemonic INVALID CONTROL CARD

INVALID DECK NUMBER . . .

INVALID ED MESSAGE-DECK NUMBER

job-specifier–INVALID index terminal-reference

INVALID JOB NUMBER

INVALID MC-CHARGE OPTION ALREADY SET

INVALID MIX NUMBER

INVALID MNEMONIC . . .

job-specifier–INVALID LINK terminal-reference

job-specifier–INVALID OPERATOR terminal-reference

INVALID PACK.ID OR TAPE MNEMONIC FOR PB . . .

job-specifier–INVALID PARAM TO VALUE DESC terminal-reference

job-specifier–INVALID PARAMETER terminal-reference

INVALID PG

job-specifier–INVALID RETURN terminal-reference

INVALID SD–SERIAL NUMBER REQUIRED

INVALID SERIAL NUMBER

INVALID SL–LOG ALREADY SET

job-specifier–INVALID SUBSCRIPT terminal-reference

job-specifier–INVALID SUBSTRING terminal-reference

INVALID SYNTAX for {CHANGE / REMOVE} COMMA IS REQUIRED FOR MORE THAN ONE {CHANGE / REMOVE}

unit-mnemonic INVALID TYPE CODE . . .

INVALID unit-mnemonic

INVALID UNIT MNEMONIC FOR FN, MUST BEGIN WITH ALPHA

"IL" REQUIRES A PARAMETER

file-identifier IN USE

job-specifier–INSUFFICIENT MEMORY TO OPEN file-identifier

job-specifier IS EXECUTING

pack-id IS ALREADY A SYSTEM DRIVE

pack-id IS A NONREMOVABLE SYSTEM PACK OR IS ALREADY OFF LINE

pack-id IS AN INTERCHANGE PACK

unit-mnemonic IS NOT A USER PACK

pack-id IS NOT INITIALIZED

job-specifier IS NOT STOPPED

pack-id IS {RESTRICTED / INTERCHANGE} PACK

job-specifier IS SUSPENDED

job-specifier–INTEGER OVERFLOW terminal-info

INTRINSIC "intrinsic-name" REQUESTED BY program-name = job-number IS NOT IN DIRECTORY – FS or RS.

INV OPTION option-name

unit-mnemonic LABELED . . . . REEL nnnnnn

unit-mnemonic LABELED . . . . [S,R,U, or I] SERIAL.NO = nnnnnn

unit-mnemonic { LABELED . . . / UNLABELED } IN USE BY job-specifier . . .

file-identifier LOAD TERMINATED–DISK ESTIMATE ERROR

file-identifier LOADED

unit-mnemonic LOCK OUT

job-specifier LOCKED DISK FILE file-identifier

option-name LOCKED

unit-mnemonic LOCKED

LOG NOW SET–CLEAR/START REQUIRED

LOG OPTION NOT SET

LOG TRANSFER COMPLETE

pack-id MAY NOW BE POWERED DOWN

unit-mnemonic MEMORY ACCESS ERROR WAIT TILL UNIT IS RESET AND TRY AGAIN

job-specifier–unit-mnemonic MEMORY PARITY

MISSING PARENTHESIS . . .

unit-mnemonic MISSING PACK-ID

MCP RAN OUT OF WORK SPACE WHILE LOOKING FOR interpreter-id WANTED BY program-name = job-number

MODIFY program-name CTRL RCD ERR: . . .

NO SEGMENT DICTIONARY SPACE for program-name = job-number

job-specifier–NO SPACE AVAILABLE FOR [CODE or DATA] [PAGE nnnn] SEGMENT nnnn

NO SPACE AVAILABLE FOR interpreter-name SOUGHT BY program-name = job-number

NO SPACE FOR program-name = job-number

NO SPACE IN INTERPRETER DICTIONARY FOR interpreter-name SOUGHT BY program-name = job-number

**NO SYSTEM DISK FOR PSR DIRECTORY

**NO USER MEMORY FOR CD

file-identifier NOT A BACKUP FILE–REQUEST IGNORED

pack-id NOW A SYSTEM DRIVE–CLEAR/START REQUIRED

unit-mnemonic NOT AVAILABLE

NOT A DISK PACK–CANNOT RL

NOT A QUOTE-MARK . . .

file-identifier NOT CHANGED– { " < FILE-NAME > /=" NOT ALLOWED | BLACK OR ZERO FIRST NAME | file-identifier ALREADY ON DISK | NOT ON DISK | IN USE | RESTRICTED FILE }

NOT ENOUGH MEMORY FOR CM

job-specifier–NAME OR VALUE STACK OVERFLOW terminal-reference

job-specifier–NEEDS AN AX REPLY

program-name job-number NEEDS nnnnnnKB PR = nn hh:mm:ss.s

job-specifier–NO DISK AVAILABLE FOR DUMP

NO DISK SPACE TO BUILD LOG

job-specifier–NO MEMORY AVAILABLE FOR DUMP

NO MEMORY FOR KA

**NO MEMORY FOR PSEUDO READER

**NO MEMORY FOR PSR DATA DIRECTORY (PSR = Pseudo Reader)

NO OVLY DISK AVL FOR program-name = job-number AMT RQD: nnnn SEGMENTS–RS–ED

NO PRINTER AVAILABLE

NO PRINTER AVAILABLE FOR KP

NO PROGRAMS RUNNING

job-specifier–NO PROVISION FOR I/O ERROR ON file-identifier terminal-reference

job-specifier–NO PROVISION FOR END OF FILE ON file-identifier terminal-reference

NO PSEUDO DECKS ON DISKS

job-specifier–NO ROOM TO OPEN FILE file-identifier

file-identifier NOT IN DIRECTORY

file-identifier NOT IN DISK DIRECTORY

"=" NOT PERMITTED IN PROGRAM NAME FOLLOWING "FN"

file-identifier NOT LOADED–IN USE BY SYSTEM

file-identifier NOT { LOCKED / REMOVED } INVALID PACK-ID pack-id

file-identifier NOT ON DISK

pack-id NOT ON LINE

unit-mnemonic NOT READY

NULL SCHEDULE

NULL . . . TABLE

NUMBER OF PSEUDO READERS CHANGED TO nnnnnn

unit-mnemonic OFF LINE

OUT OF MEMORY SPACE

job-specifier–OUTPUT UNIT NOT AVAILABLE FOR BACKUP

job-specifier–unit-mnemonic { PARITY ERROR / ACCESS ERROR } – NO RECOVERY

PM CANNOT FIND DUMPFILE/integer FOR DUMP/ANALYZER

job-specifier–POCKET LIGHT COMMUNICATE REQUESTED WHILE SORTER FLOWING terminal-reference

job-specifier–PRIORITY CHANGED TO new-priority-number

job-specifier–unit-mnemónic PRINT CHECK

PRINTER NOT READY

job-specifier–PROGRAM ABORTED terminal-reference

job-specifier–PROGRAM IS NOT WAITING SPO INPUT–AX IGNORED

PSEUDO/nnnnnn NOT ON DISK

PSEUDO/nnnnnn NOT REMOVED–INUSE

job-specifier–unit-mnemonic PUNCH CHECK

unit-mnemonic = { PURGED LABEL | file-identifier [REEL nnnnnn] | UNLABELED }

unit-mnemonic READ CHECK

job-specifier–READ OUT OF BOUNDS terminal-reference

unit-mnemonic RELABELED pack-id { R | U }

job-specifier–REQUESTED A { CODE | DATA } SEGMENT OF LENGTH ZERO terminal-reference

job-specifier–REQUESTED A CORE SPACE NOT EQUAL TO THE SIZE I JUST COMPUTED AS HIS REQUIREMENT–RS–ED MY SIZE = nnnn HIS SIZE = nnnn

job-specifier–READ REQUESTED ON OUTPUT FILE file-identifier terminal-reference

program-name REQUESTED BY "FN" NOT IN DIRECTORY

program-name REQUESTED { READ | WRITE | SEEK } ON CLOSED FILE

{ RO | SO } REQUIRES THREE OR FOUR CHARACTERS

device-mnemonic REQUIRED FOR REEL nnnnnn file-identifier

message REQUIRES MIX NO.

job-number RS-ED

unit-mnemonic REWINDING

unit-mnemonic { SAVED | TO BE SAVED }

SD REQUIRES NULL MIX

SCHEDULED: program-name = Job-number PR = nn hh:mm:ss.s

job-specifier–SEEK REQUESTED ON SERIAL FILE file-identifier terminal-reference

nnnn SEGS REQ FOR SYSTEM DUMP FILE

SERIAL NUMBER REQUIRED

SPACE REQUIRED BEFORE " or @ . . .

job-specifier–STACK OVERFLOW terminal-reference

job-specifier–SUPERFLUOUS EXIT terminal-reference

SYSTEM/LOGOUT NOT IN DIRECTORY

job-specifier–TANK OVERFLOW terminal-reference

3 DISK SEGMENTS NEEDED FOR SYSTEM/PRINTCHAIN

THERE ARE NO ENTRIES IN LOG . . . NO TRANSFERS OCCURRED

THERE ARE NO RELEVANT BACKUP FILES–PB IGNORED

***THERE IS NO BACKUP PRINT OR PUNCH FILE WITH NUMBER nnnnnn [ON PACK-ID]

job-specifier–unit-mnemonic TIMEOUT @nnnnnn@

TOKEN TOO LONG–REQUEST IGNORED . . . .

job-specifier–TOO LONG IN USE ROUTINE

TOO MANY "=" IN NAME . . . TRY AGAIN

TOO MANY "/"-S IN NAME . . . TRY AGAIN

job-specifier–TRIED TO INITIALIZE A GLOBAL BLOCK LARGER THAN ENTIRE STATIC SPACE REQUESTED STATIC = nnnn GLOBAL = nnnn –RS-ED

job-specifier–TRIED TO { SEND TO / RECEIVE FROM } "program-name" WHICH IS NOT RUNNING

**TR PLEASE

job-specifier–UNDEFINED RUN TIME ERROR terminal-reference

job-specifier–UNEXPECTED POCKET SELECT terminal-reference

job-specifier–UNINITIALIZED DATA ITEM terminal-reference

unit-mnemonic UNIT PURGED

job-specifier–unit-mnemonic { NOT READY / JAM / MISSORT }

UNIT-MNEMONIC MUST START WITH ALPHA

unit-mnemonic UNLABELED

pack-id WRITE–LOCKOUT

job-specifier–WRITE REQUESTED ON INPUT FILE file-identifier terminal-reference

job-specifier ZIPPED AN INVALID CONTROL CARD

B 1800/B 1700 SYSTEMS SOFTWARE HALTS

When a software-controlled halt occurs on a B 1800/B 1700 system, various registers are used to display information about the halt. The most important of these is the L register, which provides the primary halt definition. Some halts display further descriptive information in other registers (usually X, Y, and T).

The L register is functionally divided into two portions:

(1) The left-most 16 bits (bits 0-15) describe the specific program or routine that halted, as follows:

@0000@	SDL Interpreter STATE light ON - MCP STATE light OFF - Normal-state SDL program
@0200@	MICRO.MCP
@0D00@	GISMO
@0F00@	Memory dump
@00F0@	CLEAR/START
@000F@	SYSTEM/INIT
@B000@	BASIC interpreter
@C000@	COBOL interpreter
@E000@	RPG interpreter
@F000@	FORTRAN interpreter

(2) The right-most 8 bits (bits 16-23) give the halt identification. This portion of the halt code is dependent upon the specific routine that halted bits 0-15.

SDL Interpreter Halts (L=@0000xx@)

@01@	Evaluation/Program Pointer stack overflow.
@02@	Control stack overflow.
@03@	Name/Value stack overflow.
@04@	REMAPS size error.
@05@	Invalid parameter passed to a PROCEDURE.
@06@	Invalid substring (SUBBIT or SUBSTR).
@07@	Invalid subscript.
@08@	Invalid value returned from a TYPED PROCEDURE.
@09@	Invalid CASE.
@0A@	Divide by zero.
@0B@	Invalid index.
@0D@	Invalid operator.
@0E@	Invalid parameter in VALUE.DESCRIPTOR.
@10@	Console halt (INTERRUPT switch).
@11@	HALT operator (T register contains further definition of the halt). Complete information on MCP halts is given MCP HALTS subsection.
@12@	Write out-of-bounds.
@13@	No memory for MCP TRACE.
@1E@	Invalid parameter in DYNAMIC declaration.
@1F@	Invalid TRANSLATE.
@35@	B 1870/B 1860 Cassette data error.
@37@	B 1870/B 1860 Read/Write outside administrative memory detected.

Basic Interpreter Halts (L=@B000xx@)

@10@	Console halt (INTERRUPT switch).

COBOL Interpreter Halts (L=@C000xx@)

@10@	Console halt (INTERRUPT switch).

RPG Interpreter Halts (L=@E000xx@)

@10@	Console halt (INTERRUPT switch).

FORTRAN Interpreter Halts (L=@F000xx@)

@10@	Console halt (INTERRUPT switch).

FIRMWARE Halts (L=@0F00xx@ or @00F0xx@ or @000Fxx@)

@01@	No device on designated port/channel.
@02@	Device on selected channel is not disk (T=device-id).
@03@	Disk is not idle (T=control status).
@04@	Time-out while waiting for SERVICE REQUEST (software timed).
@05@	Bad reference address (X=good, Y=bad).
@06@	Bad status count after SERVICE REQUEST (T=control status).
@07@	Bad result descriptor (T=Result descriptor).
@08@	Seek time-out (software timed).
@09@	Memory parity error on I/O descriptor.
@0A@	Memory parity error on I/O data.
@0B@	Time-out while waiting for OP.COMPLETE.
@0C@	Exception condition after 15 retries (T=Result descriptor).
@0D@	Exception condition on TEST OP.
@0E@	Designated port/channel is not disk.
@0F@	No disk found on system.
@10@	Designated port is invalid.
@11@	Designated channel is invalid.
@12@	Insufficient memory for this routine.
@13@	Memory parity error after GISMO overlay.
@14@	Memory parity error (no diagnosis of location).
@15@	NAME TABLE entry is empty (T=entry number).
@16@	SYSTEM/DUMPFILE port number is not 7.
@17@	SYSTEM/DUMPFILE disk address is zero. (DUMP option is probably reset.)
@18@	Disk address of SYSTEM/INIT IPB is zero.
@19@	MCP.TYPE field in HINTS is zero.
@1A@	Invalid STAND.ALONE program specified.
@1B@	STAND.ALONE SDL file not available.
@1C@	No SPO on system.
@1E@	SYSTEM/INIT checksum error.
@1F@	SYSTEM/MEMDUMP checksum error.
@20@	SYSTEM/MEMDUMP NAME TABLE entry is zero (memory dump is incomplete).
@80@	Requires too much memory to CLEAR/START.
@81@	Pseudo MAXS (LR bits 0-15) greater than real MAXS.
@82@	Insufficient memory to perform segment discard operation.
@83@	Optional GISMO segment required which does not exist in this version of GISMO. Push START to ignore requested segment and continue.
@84@	System-GISMO mismatch.
@85@	Old CLEAR/START or SYSTEM/MEMDUMP specified.

@86@ Software compatibility problem.
TE & TF contain values which identify the incompatible programs:

1=SYSTEM/INIT
2=GISMO
3=MICRO.MCP
4=MCP
5=SDL interpreter

@87@ Pseudo MAXM (LR bits 15-23) greater than real MAXM.

@88@ Attempt to CLEAR/START a B 1720 system with no control memory.

@8A@ Insufficient memory to CLEAR/START specified software.

@8B@ Pseudo MAXS too large (X=pseudo MAXS, Y=maximum value that can be used). Push START to use value in Y.

@8C@ GISMO checksum error.

@8D@ SDL interpreter checksum error.

GISMO Halts (L=@0D00xx@)

@10@ Console halt (INTERRUPT switch).

@20@ Reference address error (X=good, Y=bad, TA=channel, TE-TF=control status). Push START to assume good address and continue.

@21@ Irrecoverable reference address error (same register settings as halt @0D0020@).

@22@ Reference Address is zero (TB=channel, X=reference address).

@24@ Unknown device-id (TB=channel, X=reference address).

@25@ Invalid device-id for second OP.COMPLETE bit off (TB=channel, X=reference address).

@26@ Illegal request (STAND.ALONE GISMO).

@27@ Pause on uninitialized channel (TB=channel, X=reference address).

@28@ SOFT.ID seek complete problem (TB=channel, X=reference address).

@29@ Bad exchange entry in CHANNEL.TABLE (TE=port, TF=channel).

@30@ INTERRUPT.QUEUE overflow (X=number attempted, Y=maximum allowable).

@31@ MLC "hung" (TE=port, TF=channel, X=reference address).

@32@ Memory parity error detected by MLC. Push START to find parity error.

@33@ M.WAIT: MCP waiting only on time or does not have S.I.Q.EV set in WAIT.LIST.

@34@ Memory parity error (T=address of byte in error). If T=@FFFFFF@, then there was a read outside the physical bounds of memory or an intermittent parity error that could not be isolated.

@36@ COMMUNICATE.WITH.GISMO: Bad verb (T=FA register value when reading Communicate).

@37@ COMMUNICATE.WITH.GISMO: Bad adverb (X=Communicate function, T=FA value when reading Communicate).

@38@ COMMUNICATE.WITH.GISMO/GISMO.COMMUNICATE: Parameter list length error (T=FA register address).

@39@ GISMO.COMMUNICATE: Bad verb (X=verb).

@40@ COMMUNICATE.WITH.GISMO/GISMO.COMMUNICATE: Call by non-MCP (T=LIMIT.REGISTER).

@41@ USE.COMMUNICATE: Code not present.

@42@ Deleted function (T=LIMIT.REGISTER, X=SWAPPER value).

@43@ HI.PRI: MCP not in READY.O.

@44@	MCP destroyed RS field (TB=channel, X=reference address), or a reference address error halt because RFAC option was reset.
@45@	Attempt to initiate I/O with IO.COMPLETE or SOFT.IOC events set in I/O descriptor.
@46@	Disk exchange EU.BUSY from non-busy EU.
@47@	Hi-priority interrupt request from MLC on DISPATCH.READ (T=port/channel, X=reference address + 24).
@48@	SOFT.IO: Hi-priority request for a non-sorter device (T=channel, X=reference address, Y=RS field).
@50@	Attempt to reinstate non-present RUN.STRUCTURE.
@51@	Partial data transfer on disk pack (T=Result Descriptor, X=Reference Address).
@53@	B 1870/B 1860 Cassette data error at entry to GISMO (Y=STATE.FLAGS, X=LIMIT.REGISTER).
@54@	B 1870/B 1860 CPU multiple memory parity (T=Memory ELOG, Y=STATE.FLAGS, X=LIMIT.REGISTER).
@55@	B 1870/B 1860 Read/Write outside administrative memory detected (Y=STATE.FLAGS, X=LIMIT.REGISTER).
@56@	B 1870/B 1860 Read/Write outside administrative memory detected during memory scan to check location of parity error (usually indicates a processor failure).
@57@	B 1870/B 1860 Non-CPU parity error (T=Memory ELOG, X=LIMIT.REGISTER). If error was detected via a port interrupt rather than the memory ELOG, GISMO waits for an ELOG to be reported. However, it may never be reported, and T will not have ELOG.NON.CPU.MULTIPLE set on.
@58@	STAND.ALONE GISMO: B 1870/B 1860 correctable S-Memory error (T=Memory ELOG). Push START to continue processing. To suppress further reports for the failing location, change the T Register prior to pushing START.
@59@	Illegal I/O Operator with verify variant set (TB=Channel, X=Reference Address).
@60@	SERVICE.REQUEST from missing or deleted channel (TB=Channel).
@80@	IQ: Queue empty but an MCP count not equal to zero.
@81@	IQ: Queue empty but an IQ event is set.
@82@	IQ: Queue not empty but total count is zero.
@83@	IQ: Queue not empty but sub-queue count is zero.
@84@	IQ: Queue not empty but event is reset.
@85@	COMM: ITYPE=00, COMM=59.
@86@	Q.OUT: On READY.Q, not called by SCHEDULER.
@87@	Q.OUT: An MCP COMMUNICATE.Q not empty but event bit is reset.
@88@	HANG.RS: On time only, but clock is zero.
@89@	LRU: Old, presence bit reset.
@90@	ADJUST.INTERP: RS not MCP.
@91@	USE: See code.
@92@	USE: Not MCP which tried to transfer to USE.
@93@	USE: See code.
@94@	Bad event descriptor (T=RS).

MICRO.MCP Halts (L=@0200xx@)

@01@	INT: OP.COMPLETE OFF.
@02@	INT: IO.MCP.IO is invalid.

@03@	CD: Block and S.IOC.
@04@	CD: IO.MCP.IO not USER.IO.
@05@	CD: USE.IT, not IOC.E.
@06@	CD: Not IOC.E and IOC set. (These six MICRO.MCP halts all halt with T=reference address, Y=result descriptor, X=M.EVENTS CAT IO.MCP.IO)
@07@	HANDLE.INT: Bad result (T=result from GISMO).
@08@	DISPATCH: Bad result (T=result from GISMO).
@09@	MMCP.COMM: Bad RS.COMMUNICATE.MSG.PTR.
@10@	MMCP.COMM: COMM.VERB not 1, 2, 3, or 10.
@11@	Irrecoverable error (T=HEX.SEQ.NO).
@12@	MMCP CONDITIONAL.HALT (T=HEX.SEQ.NO).
@13@	DELAYED.RANDOM I/O: Result Descriptor indicates that the I/O has been initiated, but IO.MCP.IO value is incorrect.
@20@	FETCH.MSG returned null empty.
@21@	Incomplete prior dispatch.
@22@	LF not equal to 1 after dispatch.
@23@	Neither POCKET.COMPLETE nor DOUBLE.DOCUMENT set.
@24@	Neither POCKET.COMPLETE nor TOO.LATE.TO.POCKET set.
@75@	M.I.Q.EV set.
@76@	IQE.VALID OFF.
@77@	IQE.MMCP OFF.
@79@	COMMUNICATE.WITH.GISMO not zero-relative.
@80@	Explicit SIOC but not BEEN.THRU.ERROR.

MCP Halts (L=@000011@)

When the MCP executes an explicit HALT instruction, the L register is set to @000011@ amd a parameter is loaded into the T register to further define the nature of the halt. Usually, this parameter is the first six digits of a sequence number in the MCP itself; however, some halt conditions occur at more than one place in the MCP. These are given a common identifier, as follows:

T Register	Reason
@111111@	No memory.
@Cxxxxx@ @Dxxxxx@ @Exxxxx@	Disk I/O error (T contains the error Result Descriptor).
@000Cxx@	Systems software compatibility error. The low-order eight bits of the T register contains two 4-bit numbers identifying the two programs that are incompatible, as follows: 1 SYSTEM/INITIALIZER 2 GISMO 3 MICRO.MCP 4 SDL MCP 5 SDL Interpreter

All other halts point to a specific sequence number in the MCP. Unless otherwise noted, all halts are irrecoverable; a memory dump should be taken and submitted with supporting documentation and a B 1800/B 1700 Field Trouble Report.

T Register	Reason
@043275@	Control memory management problem at CLEAR/START.
@043315@	Integrity of system disk is bad.
@043550@	Attempt to initiate in-process I/O.
@043875@	DISPATCH to invalid port or channel.
@048366@	Disk I/O did not complete in 10 seconds. Push START to retry.
@048475@	MSG.TYPE=0 in S.MSG.Q.
@048479@	MSG.TYPE=2 in S.MSG.Q.
@061351@	Attempt to verify a usercode with no programs running.
@061353@	Attempt to verify a usercode with no programs running.
@061358@	Attempt to verify a usercode with no programs running.
@063960@	Could not find pseudo reader that was in use.
@071794@	Problem with skipping an empty area of a disk file.
@071794@	Call on GET.NEW.AREA with POSITION communicate is invalid.
@071795@	Could not update base pack header.
@071796@	Unrecognizable communicate called GET.NEW.AREA.
@098586@	Invalid parameters passed to PUT.QUEUE.MESSAGE.
@098946@	Attempt to check for invalid character in a non-existent Pseudo Reader.
@099672@	Invalid parameters passed to PUT.QUEUE.MESSAGE.
@099948@	Invalid parameters passed to PUT.QUEUE.MESSAGE.
@100668@	Incorrect interface between Q.DRIVER & Q.FILES.
@100725@	Incorrect interface between Q.DRIVER & Q.FILES.
@100946@	Micro-coded I/O not allowed.
@101555@	MCP is lost during punch check recovery.
@101558@	DISPATCH to invalid port or channel.
@101961@	Received an illegal interrupt from the MMCP.
@102015@	Memory parity on DISPATCH (Interrupt from channel 15).
@102691@	In IOCOMPLETE procedure and I/O in question is not complete.
@102759@	Begin address of I/O higher than MCP.LIMIT.
@102774@	DISPATCH to invalid port or channel.
@102928@	Invalid usercode.
@109725@	DISK.ADDR not part of Q.DISK.
@112228@	Self-checking in Q.DISK.DRIVER.
@112280@	I/O address must not be zero.
@112299@	I/O Descriptor not complete.
@112394@	No message found.
@112740@	No message found.
@113675@	Queue memory link data structure broken.
@113880@	Queue must be empty here.
@114057@	No disk passed to RETURN.DISK.
@114085@	Incorrect use of CLOSE.QUEUE parameters.
@115470@	Queue file FIB not in memory.
@123305@	Invalid parameters passed to OPEN.QUEUE.
@123905@	Invalid parameters passed to OPEN.QUEUE.
@125465@	Invalid parameters passed to CLOSE.QUEUE.
@130635@	Invalid parameters passed to PUT.QUEUE.MESSAGE.
@130653@	Invalid value returned from PUT.QUEUE.MESSAGE.

T Register	Reason
@135792@	Attempt to open queue for AUTOBACKUP failed.
@135793@	Attempt to open queue for AUTOBACKUP failed.
@135794@	Attempt to open queue for AUTOBACKUP failed.
@135795@	Attempt to open queue for AUTOBACKUP failed.
@135798@	Attempt to insert message in AUTOBACKUP queue failed.
@135808@	Attempt to receive AUTOBACKUP queue message failed.
@135812@	Attempt to receive AUTOBACKUP queue message failed.
@135813@	Attempt to receive AUTOBACKUP queue message failed.
@135814@	Attempt to receive AUTOBACKUP queue message failed.
@135817@	Attempt to close AUTOBACKUP queue failed.
@140330@	Invalid parameter passed to CLOSE.QUEUE.
@140535@	Scheduled a job that does not exist on disk.
@141572@	Invalid usercode.
@146050@	SY.TYPE=INDIRECT invalid in a code segment dictionary.
@154155@	CLUB.SOMEONE failed.
@171300@	CLUB.SOMEONE failed.
@171540@	Terminating a job that does not exist on disk.
@172125@	Terminating a job that does not exist on disk.
@172770@	RS.HIERARCHY not zero but cannot find calling program.
@173433@	No schedule entry for successor on unconditional execute.
@174166@	Q.KEY stored in QPTR is now bad.
@175606@	CLUB.SOMEONE failed.
@179595@	No disk space available for ROLLOUT.
@180515@	CLUB.SOMEONE failed.
@181463@	Cannot find a file already in use.
@189765@	Bad call on OPEN.QUEUE.
@189767@	Bad call on OPEN.QUEUE.
@190339@	Bad call on OPEN.QUEUE.
@190340@	Bad call on OPEN.QUEUE.
@190363@	Bad call on GET.QUEUE.MESSAGE.
@190364@	Bad call on GET.QUEUE.MESSAGE.
@190365@	Bad call on GET.QUEUE.MESSAGE.
@190366@	Bad call on GET.QUEUE.MESSAGE.
@196460@	Invalid MICR communicate.
@212466@	System disk with no available disk space.
@212467@	Cannot find an IOAT for this system disk.
@214830@	DISPATCH to invalid port or channel.
@215877@	A control did not respond to a TEST OP after a 10 second wait (at CLEAR/START). Push START once - T contains port and channel. Push START again - T contains I/O descriptor address.
@218068@	SPO did not respond to a TEST OP.
@220177@	CLEAR/START again (increased COLD.START.VARIABLES size).
@222458@	Disk is not a system disk.
@224730@	Failed to absolutize GISMO or SDL interpreter.
@233822@	I/O will not complete while attempting to ready a disk.
@237574@	SPO.QUEUE on disk has been destroyed.
@237867@	Invalid recursion in CONTROL.CARD.DRIVER.
@238173@	CRT SPO has read data beyond the buffer limits.
@238190@	Invalid recursion in CONTROL.CARD.DRIVER.
@243120@	Bad value passed to HEADER.UPDATE.

T Register	Reason
@244625@	Cannot find MPF.INFO.TABLE, already in use.
@246265@	Bad disk file header.
@247365@	Cannot find MPF.INFO.TABLE, already in use.
@247455@	More than 16 packs on-line in a MPF.
@260790@	Line printer or SPO not ready.
@263265@	No disk available for temporary available table.
@273025@	Bad entry passed to RETURN.DISK. Push START to resolve conflict and continue.
@273026@	Bad address passed to RETURN.DISK.
@273027@	Zero port and channel passed to RET.DISK.
@273062@	Bad disk address passed to RETURN.DISK.
@273639@	Bad disk address passed to RETDSKENMASSE.
@273643@	RETURN.DISK problem. Push START to continue.
@278718@	Tape I/O did not complete after a 10 second wait. Push START once - T contains I/O Descriptor address. Push START again - T shows whether OP was complete.
@292159@	Unexpected return from GET.QUEUE.MESSAGE.
@292161@	Unexpected return from GET.QUEUE.MESSAGE.
@292162@	Unexpected return from GET.QUEUE.MESSAGE.
@303283@	Unexpected return from PUT.QUEUE.MESSAGE.
@303305@	Unexpected return from PUT.QUEUE.MESSAGE.
@303664@	Attempt to receive AUTOBACKUP queue message failed.
@303665@	Attempt to receive AUTOBACKUP queue message failed.
@303667@	Attempt to close AUTOBACKUP queue failed.
@304042@	Incorrect call on OPEN.QUEUE.
@304043@	Incorrect call on OPEN.QUEUE.
@304059@	Incorrect call on PUT.QUEUE.MESSAGE.
@304060@	Incorrect call on PUT.QUEUE.MESSAGE'
@319771@	Incorrect call on PUT.QUEUE.MESSAGE.
@319772@	Incorrect call on PUT.QUEUE.MESSAGE.
@323147@	Invalid usercode.
@361155@	Bad call on CLOSE.QUEUE.
@421725@	Bad pseudo reader chain.
@429840@	No disk available for available table.
@433514@	Bad entry in SPC.TABLE during a SQUASH.
@433517@	No entry in AVL.TABLE segment.
@433521@	Bad virtual block number.
@433541@	RETURN.DISK failed.
@433561@	Problems in MERGE phase.
@433586@	Missing entry in SQ.FILE.
@433588@	Cannot RETURN.DISK while shrinking SQ.FILE.
@433593@	Cannot find an SQ.FILE area in SQ.DIRECTORY.
@433603@	Invalid buffer address.
@433604@	File header disk address missing.
@433606@	File AREA.ADDRESS missing.
@433608@	CHILD.DIRECTORY address missing.
@433610@	Bad data block being returned to disk.
@433611@	Bad data block being returned to disk.
@433630@	Cannot RETURN.DISK for SQ.FILE.SPACE.
@502864@	UNIT.FIB.ADDRESS does not match any FIB.
@507449@	Cannot find translate file that was already in use.

T Register	Reason
@511995@	Cannot find pack that is already in use.
@512613@	No space allocated for ANSI.BUFFER.SPACE.
@512832@	Illegal hardware type.
@512856@	Illegal hardware type.
@523979@	Lost a DISK.FILE.HEADER in MPF.OPEN.
@524004@	Cannot find pack that is already in use.
@546990@	QUEUE not a queue or not a disk queue.
@547008@	Cannot find disk.
@547356@	Invalid parameters passed to OPEN.QUEUE.
@547431@	Invalid parameters passed to CLOSE.QUEUE.
@547452@	Invalid parameters passed to CLOSE.QUEUE.
@547653@	Invalid parameters passed to PUT.QUEUE.MESSAGE.
@548373@	Bad call on OPEN.QUEUE.
@548430@	Cannot backtrack (CLOSE.QUEUE problem).
@548876@	Device is not a DATA.RECORDER.
@549104@	Attempted to open DATA.RECORDER other than input or output.
@549146@	Attempted to open DATA.RECORDER other than input or output.
@555820@	Bad disk directory.
@556520@	Irrecoverable disk I/O error on file header read.
@559005@	No disk available for directory expansion.
@559403@	Irrecoverable disk I/O error on file header read.
@567260@	Partial block on disk with no disk area allocated.
@567555@	Could not update base back header.
@571410@	Could not power-down continuation pack.
@571695@	Cannot find base pack.
@573187@	Lost a DISK.FILE.HEADER for a Pseudo Reader.
@573450@	MPF.TABLE has been destroyed.
@576782@	Not at EOF1 label on tape.
@578895@	Blown Q.KEY in QUEUE file structure.
@581873@	Invalid parameters passed to PUT.QUEUE.MESSAGE.
@582042@	Invalid parameters passed to CLOSE.QUEUE.
@582081@	Invalid parameters passed to CLOSE.QUEUE.
@583526@	Attempt to insert message in AUTOBACKUP queue failed.
@585525@	Q.KEY for SPO.QUEUE is blown.
@587862@	FIB indicates a partial block but BLOCK.COUNT=0.
@590517@	No label on multi-file tape.
@594766@	Tape I/O did not complete after a 10-second wait. Push START once - T contains I/O Descriptor address. Push START again - T shows whether OP was complete.
@602613@	Non-card device in CARD.STATUS.
@605362@	SCHED.OUT failed.
@608210@	Some I/O did not complete after a 10 second wait in the IO.ERROR routine. Push START once - T contains I/O Descriptor address. Push START again - T shows whether operation was complete.
@637530@	Bad Port and Channel in Descriptor.
@637550@	Bad Descriptor.
@642050@	Invalid AUDIT.EXCEPTION.STATUS value.
@644400@	Invalid AUDIT.EXCEPTION.STATUS value.
@645360@	Bad AUDIT.TYPE length.
@645410@	Bad STR.NUMBER length.
@645430@	Bad data descriptor.

T Register	Reason
@645450@	Bad data descriptor.
@645470@	Bad data descriptor.
@645510@	Bad DISK.L.ADDR length.
@645560@	Bad DISK.L.ADDR length.
@645600@	Bad DISK.L.ADDR length.
@658637@	Cannot find DFH area for data record of disk search.
@667120@	LOCK.DESCRIPTOR address is zero.
@667130@	CURRENT.LOCK set.
@669410@	Could not find header for index sequential.
@669940@	Exception while backing out DMS operation.
@672471@	First I/O did not complete when expected.
@674860@	CURRENT.LOCK set.
@681540@	LD.MIX.USER equal to zero.
@685300@	ML.SAVE not set for modify.
@704250@	Could not find key.
@706170@	CURRENT.LOCK reset.
@710360@	Invalid mix number in CURRENTS.
@723370@	No key match.
@723406@	Cannot find index sequential key to delete.
@723420@	Problem with WA.CT.ENTRY.
@728640@	Bad parameter (CASE 0).
@781710@	Ran out of disk during DMS.
@783525@	Missing LOG file.
@783540@	Missing LOG file.
@783542@	Missing LOG file.
@783544@	No disk available for new LOG area. LOG option is reset after CLEAR/START.
@789784@	Changed system disk without CLEAR/START.
@990451@	Invalid communicate (CODE overlay).
@990661@	Invalid communicate (INTERPRETER overlay).
@990662@	Invalid communicate (GISMO overlay).
@995253@	Logical I/O problems.
@995254@	Intervention not possible.

SECTION 3
SYSTEM SOFTWARE

DISK CARTRIDGE INITIALIZER

General

A disk cartridge must be initialized before it can be used on the system. The purpose of disk initialization is threefold. One, it assigns addresses to all segments on the disk. Two, it checks to see what segments, if any, are unusable (cannot be read from or written to). Any segment found to have errors will cause the entire track in which it resides to be removed from the MASTER AVAILABLE TABLE. If any flaws occur in track ZERO or ONE the entire pack is considered faulty and cannot be used on the system. Three, skeleton table entries, the disk directory, and available tables, for example, are built and the label is written in segment zero. Operation of the DISK CARTRIDGE INITIALIZER is interactive through the console printer.

Operating Instructions

The DISK CARTRIDGE INITIALIZER program does not operate under the control of the MCP and must be loaded and executed through the cassette reader on the control panel, as follows:

a. Place the DISK CARTRIDGE INITIALIZER cassette in the cassette reader in the control panel. The BOT light should be lit at this time.

b. Place the console printer on-line.

c. Set the system MODE switch to the TAPE position and press the CLEAR, then START buttons. This loads the bootstrap loader from the cassette tape and halts the processor. The L register must be equal to @AAAAAA@ at this time.

d. Set the system MODE switch to the RUN position and press START (DO NOT PRESS CLEAR). This will load and execute the initializer.

Upon execution of the DISK CARTRIDGE INITIALIZER, the following message is displayed on the console printer:

DISK CARTRIDGE INITIALIZER – MARK ⟨level-number⟩

The following succession of messages is then displayed, each requiring a response:

WHICH CARTRIDGE – DC ⟨X⟩ OR LEAVE BLANK TO TERMINATE

VERIFICATION ONLY? – ⟨YES OR NO⟩

ENTER 6 DIGIT SERIAL NUMBER

ENTER PACK.ID

ENTER CARTRIDGE TYPE – ⟨U, S, OR R⟩

ENTER JULIAN DATE – ⟨YYDDD⟩

ENTER OWNERS NAME

When initialization and/or verification is complete, the following messages are displayed:

ID = ⟨PACK.ID⟩ SER#= ⟨integer⟩ ⟨integer⟩ BAD SECTORS

INITIALIZATION COMPLETE DC ⟨X⟩

WHICH CARTRIDGE? – DC ⟨X⟩ OR LEAVE BLANK TO TERMINATE

At this time, an additional disk cartridge can be initialized or verified, or the program can be terminated with a null entry.

DISK PACK INITIALIZER

General

The DISK PACK INITIALIZER program satisfies B 1800/B 1700 disk pack initialization requirements. Input is accepted through either the console keyboard or a card reader.

Removable disk packs must be initialized before they can be used with B 1800/B 1700 system software. DISK PACK INITIALIZER is designed to initialize, verify, and label disk packs. It assigns addresses to the appropriate sectors, writes a pattern in the sector data area, and reads this pattern back to ensure that the sectors are not defective. If a sector is found bad, it is relocated into a spare sector. If more than five sectors on the same cylinder are bad, the sixth and successive sectors will be removed from the Master Available Table. Should a sector have to be removed from among the first 64 sectors of the disk, that disk cannot be used with B 1800/B 1700 systems software.

NOTE

Throughout this section, references are made to a disk pack capability termed "offset," which is a means of causing the disk unit to create a critical head alignment useful in testing marginal disk conditions. This capability is not available in Design Level I B 9499-7/B 9499-8 Disk Pack Drives or the Design Level I Disk Pack Electronics Controller (DPEC). Specifying offset on this level disk pack has no effect on initialization.

Operating Instructions

The DISK PACK INITIALIZER program does not operate under the control of the MCP and must therefore be loaded and executed through the cassette reader on the system console. Information is supplied to the program through either the console keyboard or a card reader. If a card reader is present at execution time, the following console printer message will be displayed:

IS CARD INPUT DESIRED? ⟨YES OR NO⟩

With an affirmative response, input will be expected from the card reader. A "NO" will cause input to be requested from the console keyboard.

CONSOLE KEYBOARD (SPO) OPERATION

When input is from the console printer or console display, DISK PACK INITIALIZER generates a series of messages to which a response is expected. These messages and their appropriate responses are listed below. An invalid response generates an error message. (See Error Messages section.)

ENTER UNIT ID ⟨DP?⟩

Enter unit mnemonic or blank to terminate.

VERIFICATION ONLY? ⟨YES OR NO⟩

A YES response will cause verification only, and break the message set at this point.

IS EXTENDED I/V REQUIRED? <YES OR NO>

This message is displayed only when initializing B 9499-7/B 9499-8 disk packs. A YES reply causes a 16-pass initialization/verification procedure to be performed. A NO reply causes a two-pass initialization/verification procedure to be performed, using an initialization pattern of @6363@. For B 9484-25/B 9484-55 disk packs, the 16-pass initialization/verification procedure is always used.

ENTER 6 DIGIT SERIAL NUMBER

Enter any decimal value except zero.

ENTER PACK ID

Enter up to ten characters with no embedded blanks.

ENTER PACK TYPE - <U, S, R, OR I>

U = unrestricted, S = system, R = restricted, I = interchange.

ENTER 5 DIGIT JULIAN DATE (YYDDD)

Enter the low-order two digits of year followed by a three digit numeric day of the year. For example: 76196

ENTER OWNER'S NAME

Enter up to 14 characters.

ENTER OPTIONS

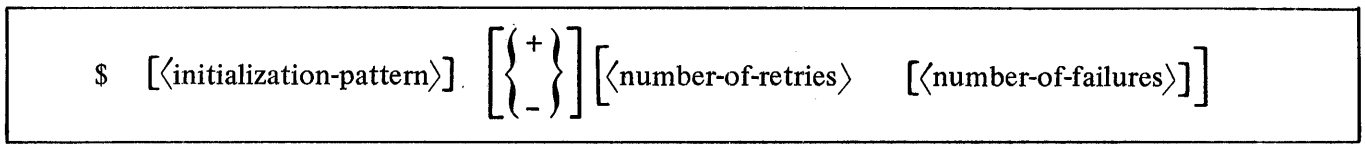
At this point, one or more occurrences of the "ENTER OPTIONS" message provides a means of entering a set of optional parameter statements which are described below. A blank response terminates the options requests.

MARGINAL SECTORS

Addresses of sectors that are known to require relocation or removal can be entered at this time. More than one sector address can appear in a message, but each must be a valid decimal address separated by spaces. All marginal sectors must be specified before the first dollar-sign option (described below).

DOLLAR-SIGN OPTIONS

Through the use of dollar-sign options, the user has the ability to verify with or without offset, compare during verification, change the number of retries on bad sectors, or modify the number of failures that may occur before a sector is considered bad. At least one dollar-sign statement must appear if extended initialization/verification is being requested, and marginal sector relocations are specified. The syntax of the dollar-sign option is as follows:



The initialization-pattern entry is a four-digit representation of a hexadecimal pattern (0000-FFFF). The first dollar-sign statement in a series of dollar-sign statements which omits this entry will cause a default pattern of @6363@ to be used for that pass. Any successive dollar-sign statements will be considered verify only, and only those entries described below will be valid.

The "+" or "–" entry is used to indicate that in (+) or out (–) offset is to be used during verification. Default is no offset.

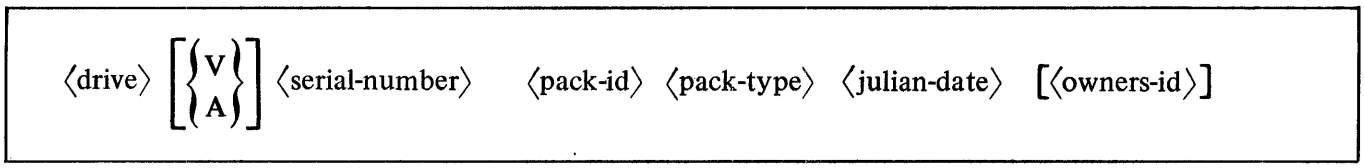
Number-of-retries specifies the number of times during verification that the same I/O operation will be attempted after an exception condition has occurred. Default is ten retries.

Number-of-failures specifies how many of the retries specified above are allowed to fail before the sector is considered bad and is relocated or removed. Default is one.

Note that if both default values for the previous entries are used while verifying a pack, all ten retries must be successful to allow the sector to remain.

CARD READER (80- OR 96-COLUMN) EXECUTION

All specifications are given in free-form format (up to 96 columns) and must be separated by one or more spaces. The specification card format is as follows:



The drive entry is the unit which is to be initialized, for example: DPA.

The V or A entry is optional. Specifying V indicates verification only, and the remainder of the specification card is ignored. Specifying an A indicates that an abbreviated initialization is desired. This will cause, as default, an initialization pattern of @6363@, followed by a single pass of verification without offset. If this entry is omitted, the default is four initialization passes, with each of the four passes followed by three verification passes, for a total of 16 passes.

The serial-number entry can be any valid decimal number except zero.

The pack-id entry is the name of the pack. There are to be no embedded blanks, and omitting this entry will cause an error for pack-type.

The pack-type entry indicates the type of access that is desired by the system software. Valid entry codes are:

U - unrestricted
S - system
R - restricted
I - interchange

The julian-date entry must have the format: YYDDD

The owner's-id entry is optional and can contain any 14-character string desired.

The following two optional parameter cards can also be used to provide a more flexible initialization and verification procedure:

a. Marginal-sector card.
b. Dollar-sign card.

These cards are in free-form format and contain the information described in two previous subsections entitled "Marginal Sectors" and "Dollar Sign Options."

SAMPLE EXECUTE STRINGS

a. Normal initialization/verification.

1. Console keyboard.

```
Msg:  ENTER UNIT ID ⟨DP(?)⟩
Rsp:  DPA
Msg:  VERIFICATION ONLY? ⟨YES OR NO⟩
Rsp:  NO
Msg:  IS EXTENDED I/V REQUIRED? ⟨YES OR NO⟩
Rsp:  YES
Msg:  ENTER 6 DIGIT SERIAL NUMBER
Rsp:  091543
Msg:  ENTER PACK ID
Rsp:  USERABC
Msg:  ENTER PACK TYPE - ⟨U, S, OR R⟩
Rsp:  U
Msg:  ENTER 5 DIGIT JULIAN DATE (YYDDD)
Rsp:  76150
Msg:  ENTER OWNER'S NAME
Rsp:  JOHN DOE
Msg:  ENTER OPTIONS
Rsp:
```

2. Card reader.

```
DPA 091543 USERABC U 76150 JOHN DOE
? END
```

Either of the above examples will cause the initialization and verification of the disk pack on DPA, using the following procedure:

Initialize with pattern @6363@
Verify three passes (No offset, offset in, and offset
Initialize with pattern @9C9C@
Verify three passes (No offset, offset in, and offset out)
Initialize with pattern @0000@
Verify three passes (No offset, offset in, and offset out)
Initialize with pattern @FFFF@
Verify three passes (No offset, offset in, and offset out)

The addresses of bad sectors are accumulated and relocated or removed during verification. The pack label contains the following:

Pack id = USERABC
Serial number = 091543
Type = unrestricted
Julian date = 76150
Owner's name = JOHN DOE

b. Verification only.

1. Console keyboard.

```
Msg:  ENTER UNIT ID ⟨DP(?)⟩
Rsp:  DPB
Msg:  VERIFICATION ONLY? ⟨YES OR NO⟩
Rsp:  YES
```

2. Card reader.

```
DPB V
? END
```

Either of the above examples will cause verification of the disk pack on DPB. Any bad sectors are reported.

c. Normal initialization/verification with specified marginal sector information.

1. Console keyboard.

```
Msg:  ENTER UNIT ID ⟨DP(?)⟩
Rsp:  DPA
Msg:  VERIFICATION ONLY? ⟨YES OR NO⟩
Rsp:  NO
Msg:  IS EXTENDED I/V REQUIRED ⟨YES OR NO⟩
Rsp:  YES
Msg:  ENTER 6 DIGIT SERIAL NUMBER
Rsp:  091543
Msg:  ENTER PACK ID
Rsp:  USERABC
Msg:  ENTER PACK TYPE - ⟨U, S, OR R⟩
Rsp:  U
```

```
Msg:  ENTER 5 DIGIT JULIAN DATE (YYDDD)
Rsp:  76150
Msg:  ENTER OWNER'S NAME
Rsp:  JOHN DOE
Msg:  ENTER OPTIONS
Rsp:  98355 18877
Msg:  ENTER OPTIONS
Rsp:  479665
Msg:  ENTER OPTIONS
Rsp:  $
Msg:  ENTER OPTIONS
Rsp:
```

2. Card reader.

```
DPA 091543 USERABC U 76150 JOHN DOE
98355 18877 479665
$
? END
```

These specifications will cause the same initialization/verification process as example 1, and will additionally relocate addresses 98355, 18877, and 479665 during the extra initialization pass.

d. Use of dollar-sign options.

1. Console keyboard.

```
Msg:  ENTER UNIT ID 〈DP(?)〉
Rsp:  DPD
Msg:  VERIFICATION ONLY? 〈YES OR NO〉
Rsp:  NO
Msg:  IS EXTENDED I/V REQUIRED? 〈YES OR NO〉
Rsp:  NO
Msg:  ENTER 6 DIGIT SERIAL NUMBER
Rsp:  673221
Msg:  ENTER PACK ID
Rsp:  USERB
Msg:  ENTER PACK TYPE - 〈U, S, OR R〉
Rsp:  U
Msg:  ENTER 5 DIGIT JULIAN DATE (YYDDD)
Rsp:  76118
Msg:  ENTER OWNER'S NAME
Rsp:  JACK SMITH
Msg:  ENTER OPTIONS
Rsp:  $ FFFF
Msg:  ENTER OPTIONS
Rsp:  $ +
Msg:  ENTER OPTIONS
Rsp:
```

2. Card reader.

```
DPD A 673221 USERB U 76118 JACK SMITH
$ FFFF
$ +
? END
```

Either of the above sets of specifications initializes DPD with an initialization pattern of @FFFF@, verifies one pass without offset, and one pass with offset in (+).

e. Combined marginal sector and dollar-sign options.

1. Console keyboard.

```
Msg:  ENTER UNIT ID ⟨DP(?)⟩
Rsp:  DPD
Msg:  VERIFICATION ONLY? ⟨YES OR NO⟩
Rsp:  NO
Msg:  IS EXTENDED I/V REQUIRED? ⟨YES OR NO⟩
Rsp:  NO
Msg:  ENTER 6 DIGIT SERIAL NUMBER
Rsp:  673221
Msg:  ENTER PACK ID
Rsp:  USERB
Msg:  ENTER PACK TYPE - ⟨U, S, OR R⟩
Rsp:  U
Msg:  ENTER 5 DIGIT JULIAN DATE (YYDDD)
Rsp:  76118
Msg:  ENTER OWNER'S NAME
Rsp:  JACK SMITH
Msg:  ENTER OPTIONS
Rsp:  98385
Msg:  ENTER OPTIONS
Rsp:  $ FFFF 8
Msg:  ENTER OPTIONS
Rsp:  $ 6363 + 40 5
Msg:  ENTER OPTIONS
Rsp:  $ 40 5
Msg:  ENTER OPTIONS
Rsp:  $ - 40 5
Msg:  ENTER OPTIONS
Rsp:
```

2. Card reader.

```
DPD A 673221 USERB U 76118 JACK SMITH
98385
$ FFFF 8
$ 6363 + 40 5
$ 40 5
$ - 40 5
? END
```

The above parameters will:

a. Relocate sector 98385.

b. Initialize with @FFFF@, verify without offset, and retry eight times on bad sectors.

c. Initialize with @6363@, verify with offset in (+), and retry forty times on bad sectors, relocating or removing the sector if it has failed at least five times.

d. Verify without offset, retry forty times on bad sectors, relocating or removing the sector if it has failed at least five times.

e. Verify with offset out (-), retrying and relocating as above.

ERROR MESSAGES

NO PACKS ON SYSTEM

THE FOLLOWING SECTORS ARE IN ERROR
THEY WILL BE RELOCATED

This message appears only if initialization has been requested.

DISK ERROR - RESULT STATUS IN "T"

WRITE LOCKOUT ⟨drive⟩

DISK NOT READY ⟨drive⟩

DISK NOT PRESENT ⟨drive⟩

Specified drive is not found.

PACK HAS EXCEEDED ERROR LIMITS

Master available table is filled.

PACK CANNOT BE USED WITH MCP

Either the master available table is filled, or 6 bad sectors were found within the first 64 sectors.

INVALID DRIVE ENTRY ⟨entry⟩

ERROR ON CYL 0 ⟨address⟩

INVALID SERIAL NUM ⟨serial number⟩

INVALID ENTRY BLANK ID

INVALID ENTRY ⟨entry⟩

INVALID SECTOR NUM ⟨sector number entry⟩

SECTOR REMOVED ⟨disk address⟩

PACK HAS EXCEEDED SPECS ERROR LIMIT ⟨unit-id⟩

The number of errors detected during verification exceed one of the following:

a. Two (2) errors per track,

b. Three (3) errors per cylinder, or

c. Fifty (50) errors per pack.

A pack that has exceeded these error limits cannot be used with the MCP.

SYSTEM/DISK.INIT

General

SYSTEM/DISK.INIT satisfies all B 1800/B 1700 disk pack and cartridge initialization requirements while running under MCP control. Input is accepted through either a card reader or the console keyboard.

Removable disk cartridges and packs must be initialized before they can be used with B 1800/B 1700 systems software. SYSTEM/DISK.INIT, which runs under MCP control, is designed to initialize, verify, and label both packs and cartridges. It assigns addresses to the appropriate sectors, writes a random pattern in the sector data area, and reads these patterns back to ensure that the sectors are not defective. If a sector on a disk cartridge is found bad, the entire track in which it resides is removed from the Master Available Table. When a sector on a disk pack is found bad, it is relocated into a spare sector. If more than five sectors on the same cylinder are bad, the sixth and successive sectors will be removed from the Master Available Table. Should a sector have to be removed from among the first 64 sectors of the disk, that disk cannot be used with B 1800/B 1700 systems software.

NOTE

Throughout this section, references are made to a disk pack capability termed "offset," which is a means of causing the disk unit to create a critical head alignment useful in testing marginal disk conditions. This capability is not available in Design Level I B 9499-7/B 9499-8 Disk Pack Drives or the Design Level I Disk Pack Electronics Controller (DPEC). Specifying offset on this level disk pack has no effect on initialization.

Operating Instructions

The SYSTEM/DISK.INIT program operates under the control of the MCP and can be executed through either the console keyboard or a card reader. A listing is produced containing the relocated or removed sectors. The listing will be sent to backup to free the printer for other jobs. Executing the program with SW 9 = 1 produces the report on the SPO instead of sending it to printer backup.

CONSOLE KEYBOARD (SPO) OPERATION

When executed from the console printer or console display, SYSTEM/DISK.INIT generates a series of messages to which an ACCEPT message response is expected. These messages and their appropriate responses are listed below. An invalid response generates an error message. (See Error Messages section.)

ENTER UNIT ID <DC? OR DP? >

Enter unit mnemonic or blank to terminate.

VERIFICATION ONLY? <YES OR NO>

A YES response will cause verification only, and break the message set at this point.

IS EXTENDED I/V REQUIRED? <YES OR NO>

This message is displayed only when initializing B 9499-7/B 9499-8 disk packs. A YES reply causes a 16-pass initialization/verification procedure to be performed. A NO reply causes a two-pass initialization/verification procedure to be performed, using an initialization pattern of @6363@. For B 9484-25/B 9484-55 disk packs, the 16-pass initialization/verification procedure is always used. For disk cartridges, the two-pass initialization/verification procedure is always used.

ENTER 6 DIGIT SERIAL NUMBER

Enter any decimal value except zero.

ENTER PACK ID

Enter up to ten characters with no embedded blanks.

ENTER PACK TYPE - <U, S, R, OR I >

U = unrestricted, S = system, R = restricted, I = interchange.

ENTER OWNER'S NAME

Enter up to 14 characters.

ENTER OPTIONS

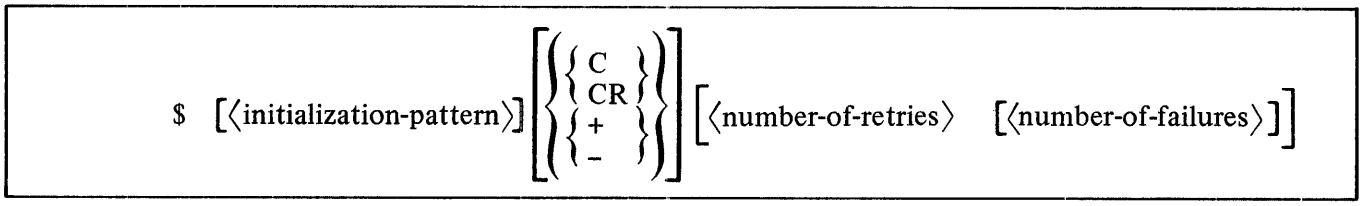
At this point one or more occurrences of the "ENTER OPTIONS" message will provide a means of entering a set of optional parameter statements which are described below. A blank response terminates the options requests.

MARGINAL SECTORS

Addresses of sectors that are known to require relocation or removal can be entered at this time. More than one sector address can appear in an ACCEPT, but each must be a valid decimal address separated by spaces. All marginal sectors must be specified before the first dollar-sign option, as described in the following subsection.

DOLLAR-SIGN OPTIONS

Through the use of dollar-sign options, the user has the ability to verify with or without offset, compare during verification, change the number of retries on bad sectors, or modify the number of failures that can occur before a sector is considered bad. At least one dollar-sign statement must appear if extended initialization/verification is being requested, and marginal sector relocations are specified. The syntax of the dollar-sign option is as follows:



The initialization-pattern entry is a four-digit representation of a hexadecimal pattern (0000-FFFF). The first dollar-sign statement, in a series of dollar-sign statements, which omits this entry will cause a default pattern of @6363@ to be used for that pass. Any successive dollar-sign statements are considered verify only, and only those entries described below are valid.

The "+" or "–" entry is used for disk packs to indicate that in (+) or out (–) offset is to be used during verification. Default is no offset.

The C or CR entry is used for disk cartridges to specify compare or compare-and-remove during verification. If this entry is omitted, normal verification is assumed.

Number-of-retries specifies the number of times during verification that the same I/O operation will be attempted after an exception condition has occurred. Default is ten retries.

Number-of-failures specifies how many of the retries specified above are allowed to fail before the sector is considered bad and is relocated or removed. Default is one for packs and eight for cartridges.

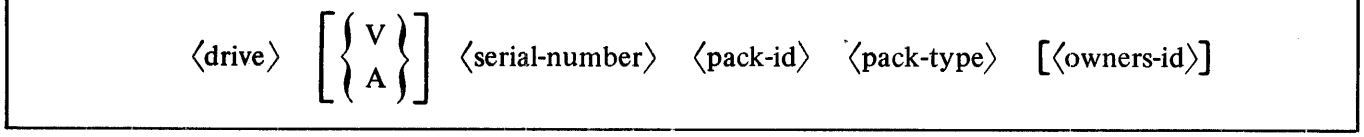
Note that if both default values for the previous entries are used while verifying a pack, all ten retries must be successful to allow the sector to remain.

CARD READER (80- or 96-COLUMN) EXECUTION

The SYSTEM/DISK.INIT execute control deck has the following format:

```
? EXECUTE SYSTEM/DISK.INIT FILE SPEC NAME ⟨spec-file-id⟩;
? DATA ⟨spec-file-id⟩
 ⟨specification cards⟩
? END
```

All specifications are in free-form format (up to 96 columns) and must be separated by one or more spaces. The specification card format is as follows:



The drive entry is the unit which is to be initialized, for example: DCA or DPB.

The V or A entry is optional. Specifying a V indicates verification only, and the remainder of the specification card is ignored. Specifying an A indicates that an abbreviated initialization is desired (disk pack only). This will cause an initialization pattern of @6363@ followed by a single pass of verification without offset.

If this entry is omitted, the default is four initialize passes and twelve verify passes for disk packs, or one initialize and one verify pass for cartridges.

The serial-number entry can be any valid decimal number except zero.

The pack-id entry is the name of the pack. There is to be no embedded blanks, and omitting this entry will cause an error for pack-type.

The pack-type entry indicates the type of access that is desired by the system software. Valid codes are:

U - unrestricted
S - system
R - restricted
I - interchange

The owners-id entry is optional and may contain any 14-character string desired.

Two additional optional parameter cards can be used to provide a more flexible initialization and verification procedure. These are:

a. Marginal-sector card
b. Dollar-sign card

These cards are in free-form format and contain the information described in the two previous sections entitled Marginal Sectors and Dollar-Sign Options.

SAMPLE EXECUTE STRINGS

a. Card Reader

1. Normal initialize/verify for disk pack.

```
DPA 091543 USERABC U JOHN DOE
```

This card will initialize and verify the disk pack on DPA with the following procedure:

Initialize with @6363@ pattern.
Verify three passes (no offset, offset in, and offset out).
Initialize with @9C9C@ pattern.
Verify three passes (no offset, offset in, and offset out).
Initialize with @0000@ pattern.
Verify three passes (no offset, offset in, and offset out).
Initialize with @FFFF@ pattern.
Verify three passes (no offset, offset in, and offset out).

The addresses of all bad sectors are accumulated and either relocated or removed in the final verification pass.

The pack label contains the following:

Pack-id = USERABC
Serial-number = 091543
Pack-type = Unrestricted
Julian-date = The System Date
Owner's id = JOHN DOE

2. Normal initialize/verify for disk cartridge.

```
DCA 084673 ABC U J. B. JONES
```

This card initializes a disk cartridge with a @6363@ pattern and verifies it on DCA with a label of:

Pack-id = ABC
Serial-number = 084673
Pack-type = Unrestricted
Owner's-id = J. B. JONES

3. Specification cards for disk cartridge.

```
DCD 123456 ABC S
7486
$ C
```

This set of cards initializes and verifies DCD as a system cartridge with a pack-id of ABC and a serial-number of 123456. The track containing sector 7486 is removed, and the initialize pattern is compared for content during verification.

4. Specification cards for disk pack.

```
DPC A 797601 XYZ U
98385 18877 19765
479665
$ FFFF
$ +
```

This set of cards initializes DPC with an initialization pattern of @FFFF@, verifies one pass without offset, and one pass with offset in (+). Addresses 98385, 18877, 29765, and 479665 are also relocated.

b. Console printer.

1. Normal initialize/verify operation.

```
Msg: SYSTEM/DISK.INIT = ⟨mix-index⟩ ENTER UNIT ID DC(?) OR DP(?)
Msg: SYSTEM/DISK.INIT = ⟨mix-index⟩ ACCEPT
Rsp: ⟨mix-index⟩ AX DPB
Msg: SYSTEM/DISK.INIT = ⟨mix-index⟩ VERIFICATION ONLY? YES OR NO
```

```
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ ACCEPT
Rsp:    ⟨mix-index⟩ AX NO
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ IS EXTENDED I/V REQUIRED?
        ⟨YES OR NO⟩
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ ACCEPT
Rsp:    ⟨mix-index⟩ AX YES
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ ENTER 6 DIGIT SERIAL NUMBER
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ ACCEPT
Rsp:    ⟨mix-index⟩ AX 179846
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ ENTER PACK ID
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ ACCEPT
Rsp:    ⟨mix-index⟩ AX ABC
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ ENTER PACK TYPE⟨U, S, R, OR I⟩
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ ACCEPT
Rsp:    ⟨mix-index⟩ AX U
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ ENTER OWNER'S ID
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ ACCEPT
Rsp:    ⟨mix-index⟩ AX S. B. GARVEY
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ ENTER OPTIONS
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ ACCEPT
Rsp:    ⟨mix-index⟩ AX
```

These responses initialize and verify DPB with the 16-pass initialize and verify procedure, and creates label information as follows:

```
Pack-id = ABC
Serial-number = 179846
Pack-type = Unrestricted
Julian-date = The System Date
Owner's-id = S. B. GARVEY
```

2. Normal verification only.

```
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ ENTER UNIT ID ⟨DC(?) OR DP(?)⟩
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ ACCEPT
Rsp:    ⟨mix-index⟩ AX DCC
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ VERIFICATION ONLY? ⟨YES OR NO⟩
Msg:  SYSTEM/DISK.INIT = ⟨mix-index⟩ ACCEPT
Rsp:    ⟨mix-index⟩ AX YES
```

These responses verify the cartridge on DPC and report any bad sectors encountered.

ERROR MESSAGES

INVALID ENTRY BLANK ID

INVALID ENTRY ⟨entry⟩

INVALID SERIAL NUMBER

INVALID SECTOR NUMBER

THE FOLLOWING SECTORS ARE IN ERROR AND WILL BE RELOCATED

THE FOLLOWING SECTORS ARE IN ERROR AND THEIR TRACKS WILL BE REMOVED

⟨unit-id⟩ IS INITIALIZED AS ⟨pack label information⟩

IS ⟨unit-id⟩ TO BE INITIALIZED?

This is a safety measure to ensure that this disk is the correct one. YES and NO are valid responses.

WOULD YOU LIKE TO RETRY? ⟨YES OR NO⟩

YES will retry the operation. NO will terminate.

DISK ERROR - RESULT = ⟨result-descriptor⟩

MUST RESTART TO CONTINUE

WRITE LOCK ⟨unit-id⟩

DISK NOT READY ⟨unit-id⟩

PACK CANNOT BE USED WITH MCP ⟨unit-id⟩

One of the first 64 sectors has been removed, or the Master Available Table is full.

PACK HAS EXCEEDED SPECS ERROR LIMIT ⟨unit-id⟩

The number of errors detected during verification exceed one of the following:

a. Two (2) errors per track,

b. Three (3) errors per cylinder, or

c. Fifty (50) errors per pack.

A pack that has exceeded these error limits cannot be used with the MCP.

ERROR CYL 0 ⟨unit-id⟩⟨disk-address⟩

SECTOR REMOVED ⟨disk-address⟩

COMPARISON ERROR ⟨disk-address⟩

The C or CR option is enabled, and a comparison error has been detected.

COLDSTART

General

The COLDSTART routine is used to load basic system software and firmware to disk. The routine is furnished on a cassette tape and is loaded via the control panel cassette reader.

The following actions are performed by COLDSTART:

a. Constructs and initializes the disk directory and available tables on the system disk.

b. Loads the MCP from magnetic tape to system disk.

c. Loads the SDL Interpreters from magnetic tape to the system disk.

d. Loads GISMO from magnetic tape to system disk.

e. Loads the System Initializer (SYSTEM/INIT) from magnetic tape to system disk.

f. Loads SYSTEM/LOAD.DUMP, FILE/LOADER, and SYSTEM/MEMDUMP from magnetic tape to system disk.

g. Loads the MICRO.MCP from magnetic tape to systems disk.

h. Makes appropriate entries in the NAME TABLE for all system software and firmware loaded.

i. Constructs the COLDSTART VARIABLES on system disk.

j. Displays a message on the console printer instructing the operator to perform a Clear/Start.

NOTE

When a COLDSTART is performed on a system disk that was previously in operation, all the files entered in the disk directory are lost and must be reconstructed. This is due to the disk directory being initialized and cleared by the COLDSTART.

Procedure

The COLDSTART procedure is as follows:

a. Mount a "system" pack or cartridge on drive 0 (if not a head-per-track system).

b. Mount the library tape (labeled "SYSTEM") on any available tape drive.

c. Set MODE switch to TAPE.

d. Place the COLDSTART cassette in the cassette reader. Cassette is automatically rewound.

e. Press CLEAR, then START.

f. Cassette reads a few feet and the system halts. The L register contains @AAAAAA@ at this time.

g. Set MODE switch to RUN, press START.

h. Cassette will continue to read. If the system HALTS with @000004@ in the L register, the cassette has a hash total error and must be reloaded. When the cassette has finished loading, the STATE light will come on, and COLDSTART will begin execution.

The library tape used for COLDSTART must be labeled "SYSTEM". It will be located automatically by the COLDSTART routine, and need not be mounted on any particular tape unit.

The files to be loaded by COLDSTART can be contained on the tape in any order, but must have the following file-identifiers:

MCPII	SYSTEM/INIT
SDL/INTERP1S	MCPII/MICRO.MCP
SDL/INTERP1M	FILE/LOADER
GISMO	SYSTEM/MEMDUMP
	SYSTEM/LOAD.DUMP

Files other than those shown above can also be contained on the same library tape, and can be loaded after completion of the COLDSTART (and subsequent CLEAR/START) by the following control message:

ADD FROM SYSTEM =/=

During the execution of COLDSTART, errors are identified by a system halt where the L register contains @000011@. The T register contains the specific error identification, as follows:

T Register	Description
@AAAAAA@	Normal End-of-Job. CLEAR/START required.
@0C0001@	Disk I/O Error. Press START once to display the Result Descriptor in the T register.
@0C0002@	Tape I/O Error. Press START once to display the Result Descriptor in the T register.
@0C0003@	Unexpected Data or Result Descriptor from tape.
@0C0004@	No tape control on system.
@0C0005@	No disk control on system.
@0C0006@	Disk not initialized in the proper format.
@0C0007@	Attempt to COLDSTART a pack or cartridge not initialized as "system" (S).
@0C0008@	Could not locate system tape. Make tape ready and press START.
@0C0009@	One or more files are missing from the system tape. By pressing START repeatedly until @0C0009@ is again displayed, a list of numbers corresponding to the missing files is displayed in the low-order (right-most) 4 bits of the T register, as follows:

File Number	File-identifier
1	MCPII
2	SDL/INTERP1S
3	SDL/INTERP1M
4	GISMO
5	SYSTEM/INIT
6	MCPII/MICRO.MCP
7	SYSTEM/LOAD.DUMP
8	FILE/LOADER
9	SYSTEM/MEMDUMP

T Register	Description
@0C000A@	Missing device on I/O DISPATCH operation.
@0C000B@	Insufficient disk for COLDSTART.
@0C000C@	Went past end-of-file on tape.
@0C000D@	Missing tape mark.

The system disk created by COLDSTART is a single system pack configuration, and does not contain a LOG. Once the system is running under MCP control, the number of system drives may be increased using the SD message, and the LOG option set with the SL message.

COLDSTART/DISK

General

COLDSTART/DISK is a stand-alone program designed to coldstart a disk using a second disk as the source of the required files. Its function is the same as COLDSTART, except that a disk is used as the input rather than a magnetic tape.

Only the first disk drive (DCA, DPA, or DKA) is allowed as the output system disk. The input disk may be either a user pack/cartridge or a single system disk. Multiple system packs/cartridges may not be used as input to COLDSTART/DISK.

The standard code file names (refer to COLDSTART) are used in the file name search. If these names cannot be located on the input disk, COLDSTART/DISK requests new names to be entered from the console keyboard (SPO).

An option is also provided to copy the remaining files from the input disk to the COLDSTARTed system disk after the standard files have been copied.

NOTE

Do not attempt to use the full copy option if there are more files on the input disk than can be contained on the output disk.

Procedure

COLDSTART/DISK does not operate under control of the MCP and must be loaded through the cassette reader on the system console. Refer to the COLDSTART procedure for cassette loading instructions.

All communication with COLDSTART/DISK is accomplished through the SPO. After the COLDSTART/DISK cassette has loaded successfully, the following messages appear:

```
COLDSTART/DISK MARK <mark-level>
ENTER OUTPUT DRIVE - <DCA, DPA OR DKA>
```

The correct response produces the following message:

```
ENTER INPUT DRIVE - <DC?, DP? OR DKA>
```

The correct response causes the following message to be displayed:

```
IS COMPLETE COPY DESIRED? <YES OR NO>
```

After the COLDSTART and optional COPY (if requested) are complete, the following message is displayed:

```
COLDSTART COMPLETE - CLEAR/START REQUIRED
```

Example:

```
COLDSTART/DISK MARK VI.1
ENTER OUTPUT DRIVE - <DCA, DPA OR DKA>
DKA
ENTER INPUT DRIVE - <DC?, DP? OR DKA>
DPB
IS COMPLETE COPY DESIRED? <YES OR NO>
YES
COLDSTART COMPLETE - CLEAR/START REQUIRED
```

These responses cause the head-per-track disk (DKA) to be COLDSTARTed using the disk pack on DPB as the input disk. The standard program names are used for the COLDSTART procedure, and the remaining files on DPB are copied to the new output disk.

ERROR MESSAGES

DISK ERROR RESULT IN 'T'
HIT START TO RETRY

The T REGISTER contains the error result descriptor. Push START to retry the I/O operation.

INVALID RESPONSE - TRY AGAIN

The operator attempted to respond with other than the requested information.

BAD FILE HEADER <file-name> ADDRESS <hex-address >

The input file has a bad header or the header is located on another disk. The file cannot be copied to the output disk.

INSUFFICIENT DISK SPACE FOR COLDSTART
INVALID COLDSTART
RESTART TO CONTINUE

The necessary coldstart files require more disk space than is available on the output disk. Obtain another disk and rerun COLDSTART/DISK.

NO DISK DEVICE ON SYSTEM
RESTART TO CONTINUE

There are no cartridges or packs on the system. COLDSTART/DISK cannot be used.

<family-name> INVALID SUB DIRECTORY

The sub-directory for the specified <family-name> is bad and cannot be used. COLDSTART/DISK cannot access any file with the specified <family-name>.

⟨file-name⟩ NOT FOUND
ENTER THE CORRECT ⟨file-name⟩ NAME

The specified program cannot be located on the input disk. Another name may be entered; however, it is the responsibility of the operator to insure that the file specified has the same function as the file requested by COLDSTART/DISK.

⟨file-name⟩ BAD AREA ADDRESS ⟨hex-address⟩

Retries could not correct bad reads on the specified file. The file is still copied to the output disk (if it is not a required system file), and should be verified manually.

⟨file-name⟩ IS A MULTI PACK FILE - CAN NOT COPY

Multi-pack files are not allowed on a system disk.

DISK NOT READY ⟨unit-mnemonic⟩

DISK NOT PRESENT ⟨unit-mnemonic⟩

WRITE LOCKOUT ⟨unit-mnemonic⟩

INSUFFICIENT DISK SPACE FOR ⟨file-name⟩
INVALID COLDSTART

During the complete copy, the space remaining on the output disk is not large enough to accommodate the specified file. Restart COLDSTART/DISK either without the complete copy option, or after unnecessary files have been removed from the input disk.

PACK LABEL BAD

The label on the input disk pack/cartridge is invalid.

CLEAR/START and MEMORY DUMP PROCEDURE

General

A CLEAR/START is used by the system operator to restore the system to an operable state. A CLEAR/START must be performed under any of the following conditions:

a. System Power-up.

b. An unscheduled halt.

c. An uninterruptible system software loop.

d. The system software/firmware is changed (via CM message).

A CLEAR/START performs the following functions:

a. Terminates all programs being executed.

b. Empties the schedule.

c. Writes correct parity and zeros throughout memory.

d. Loads the MCP, SDL Interpreter, System Initializer and GISMO specified by the NAME TABLE entries selected.

e. Returns control to the MCP.

If the processor is running at the time a CLEAR/START is to be performed, the INTERRUPT switch on the console should be used to bring the system to an orderly halt.

Clear/Start Procedure

a. Halt processor with the INTERRUPT switch.

b. Place CLEAR/START cassette in cassette reader.

c. Press CLEAR.

d. Set MODE switch to TAPE position.

e. Press START (When tape stops, check the L register for @AAAAAA@. At this point enter any temporary changes to be made in the T or X registers.)

f. Set MODE switch to RUN position.

g. Press START.

The same CLEAR/START program is usable on any B 1800/B 1700 system.

Name Table

The NAME TABLE is built during COLDSTART and resides on system disk. It identifies firmware and system software that can be used in the operational environment of the system.

The operator may select from NAME TABLE different environments for operation. However, not all systems will be able to use many of these programs since they are strictly for experimental system software development and system software debugging.

The main advantage of the NAME TABLE method of selecting an operating environment is the ability to at all times recover to the standard mode of operation.

A typical COLDSTART procedure will load and identify for the system the following:

a. A standard MCP (MCPII).

b. B 1830/B 1710 and B 1860/B 1720 SDL Interpreters.

c. A micro-coded I/O Driver (GISMO).

d. A System Initializer (SYSTEM/INIT)

e. SYSTEM/LOAD.DUMP

f. FILE/LOADER

g. SYSTEM/MEMDUMP

h. A micro-coded MCP (MCPII/MICRO.MCP)

This is enough system software and firmware to begin operations on whatever hardware is available. A system pack may be moved from one system to another and started by merely performing a CLEAR/START.

Operating Environments

The CM message is used to identify the function of various programs to the system for subsequent usage. See the CM input message for the syntax to be used.

The following list describes the function code or the system software mnemonic and its meaning.

NAME TABLE Entry Number	System Software Mnemonic (Function Code)	Meaning
0	N	Standard System Initializer (SYSTEM/INIT)
1	NX	Experimental System Initializer
2	G	Central Service Module (GISMO)
3	GT	Trace Central Service Module
4	GX	Experimental Central Service Module (GISMO/DEBUG)
5	I1	B 1830/B 1710 SDL Interpreter (SDL/INTERP1S)
6	I2	B 1860/B 1720 SDL Interpreter (SDL/INTERP1M)
7	I1T	B 1830/B 1710 Trace SDL Interpreter
8	I2T	B 1860/B 1720 Trace SDL Interpreter
9	IX	Experimental SDL Interpreter
10	M	Standard MCP (MCPII)

NAME TABLE Entry Number	System Software Mnemonic (Function Code)	Meaning
11	MT	Trace MCP
12	MX	Experimental MCP
13	MM	Standard MICRO.MCP (MCPII/MICRO.MCP)
14	MMX	Experimental MICRO.MCP (MICRO.MCP/DEBUG)
15	SD	System Memory Dump (SYSTEM/MEMDUMP)
16	SDD	Stand-alone DISK/DUMP
17	SDL	Stand-alone SDL Program
18	SID	Stand-alone I/O DEBUG
19	SL	Loader for Stand-alone SDL Program
20	SX	Experimental Stand-alone MIL Program
21	C	NDL Network Controller
22	US	Usercode-Password File ((SYSTEM)/USERCODE)

The purpose of the CM input message is to identify a file on System Disk to be used for a designated function.

Example:

```
CM   MX     MCP/XYZ
```

The above example makes the file MCP/XYZ the experimental MCP and will be the program executed when an experimental MCP is called for.

Selecting Environments

With the appropriate files loaded and CM-ed, there are two general environments which can be selected as a basis for operation:

a. Standard MCP (MCP II)

b. Trace MCP and System Software

A CLEAR/START is required to effect any change. The choices become the new basis for operation. They remain in effect until they are changed explicitly, but they can be switched on a temporary basis during the CLEAR/START procedure.

Temporary Environment Changes

Operations following a CLEAR/START can be tailored to the needs of system programmers by setting the following values in the T register during step e of the CLEAR/START procedure.

Bits on the control panel are numbered from LEFT to RIGHT.

Bits	Description
0	Dump Memory
1	Run a stand-alone program (see, below, bits 9-12)

Bits	Description
2	Not used
3	Switch TRACE vs. NON-TRACE
4	Run with experimental MCP (MX)
5	Run with experimental System Initializer (NX)
6	Run with experimental SDL Interpreter (IX)
7	Run with experimental GISMO (GX)
8	Run with experimental MICRO.MCP (MMX)
9-12	When bit 1 is set, the following programs will be run.

Value	Identification
0	SX
1	SDD
2	SIO
3	SDL, using SL to load with interpreter

13-23	Must be left zeros

Another option that can be made during CLEAR/START is the designation of the system disk. To override the usual CLEAR/START selection, load the following values in the X register during step e of the CLEAR/START procedure.

Bits	Value
16-19	Port
20-23	Channel

Memory Dump Procedure

The memory dump as well as other temporary changes may be accomplished during the CLEAR/START procedure. Between steps (e) and (f) in the CLEAR/START Procedure simply set the proper bits in the appropriate register and continue with the normal CLEAR/START procedure.

The memory dump requires that bit 0 of the T Register be turned on at this time.

DISK FILE COPY

General

The DISK/COPY program will copy one or more disk files from one disk to another or to another location on the same disk.

DISK/COPY accepts input specifications from either the card reader or console keyboard (SPO). If DISK/COPY is executed with switch 9 set to 1, input is expected as a series of ACCEPT (AX) messages; otherwise, a card file labeled CARDS is used. When input is specified by ACCEPT messages, entering a blank (null) ACCEPT message terminates the series and begins the specified file copy operations.

Any number of files may be copied during one execution of DISK/COPY.

DISK/COPY Operating Instructions

The following figure represents the DISK/COPY control deck.

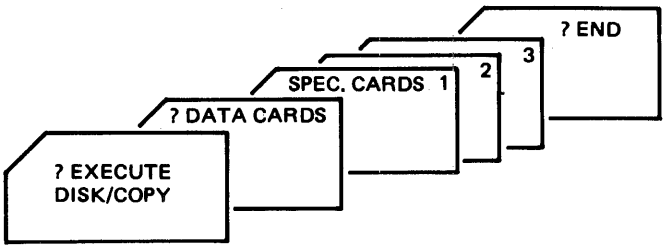


Figure 3-1. DISK/COPY Control Deck

COPY Specifications

Multiple specifications are allowed during one execution of DISK/COPY; however, each input card or ACCEPT message may specify only one copy operation.

Specifications are free-form. Each specification must contain two disk file-identifiers with the first file-identifier being the file to be copied, and the second file-identifier being the new copy of the file.

The format for the file identifiers is the same as used for MCP control cards. Specifications of "=/=", "⟨family-name⟩/=", "⟨pack-id⟩/=/=", and "⟨pack-id⟩/⟨family-name⟩/=" are also allowed.

When copying a single file and the file-identifier is to be retained when copying to another user disk, the new file-identifier may specify only the name of the pack-id followed by a slash.

Examples:

a. To copy file AAA on a systems disk to another location on the systems disk with the name BBB:

```
AAA     BBB
```

b. To copy a file AAA on a systems disk to another disk named NEWDISK and retain the file-identifier:

```
AAA     NEWDISK/AAA/
```

c. Since the file-identifier is not changed in example (b), the same result would be obtained by using the following specification card.

```
AAA     NEWDISK/
```

d. To copy all of the files with a<family-name>of AAA to a<family-name>of BBB:

```
AAA/=   BBB/=
```

e. To copy all of the files on PACK.1 to PACK.2:

```
PACK.1/=/=   PACK.2/=/=
```

f. To copy all files with a family-name of AAA on user pack PACK.1 to the system disk:

```
PACK.1/AAA/=     AAA/=
```

g. To copy all of the files on the system disk to the user disk labeled BACKUPPACK:

```
=/=          BACKUPPACK/=/=
```

DMPALL

General

The program DMPALL has two separate functions: (1) printing the contents of files, and (2) reproducing data from one hardware device to another. Execution may be from either the console printer or card reader.

Printing

Printing files consist of the following:

a. Data may be card, magnetic tape, paper tape, or disk.

b. Any file can be read up to 1000 bytes per logical record.

c. Contents can be printed in byte, digit, or combined form.

d. Printing may begin with a specified record number and terminate after a specified number of records are printed.

Reproducing

Reproducing files may be executed as follows:

a. A file may be reproduced from any card, magnetic tape, paper tape, or disk.

b. File-identifiers, record lengths, and blocking factors may be changed during the reproduction.

c. Reproducing may begin with a specified record number and terminate after a specified number of records.

Operating Instructions

CONSOLE PRINTER

DMPALL executed from the console printer responds with the following three messages:

```
DMPALL     =     mix-index   BOJ. . . .
% DMPALL   =     mix-index   ENTER SPECS.
DMPALL     =     mix-index   ACCEPT.
```

The operator replys to the ACCEPT message by entering an AX message containing the specifications needed to perform the DMPALL operation.

The directory of a LIBRARY tape created by the program SYSTEM/LOAD.DUMP can be either punched or printed using the following procedure:

```
mix-index AX PD [PUNCH] tape-identifier
```

When PUNCH is specified, the tape directory will be output to cards for use with the program SYSTEM/LOAD.DUMP. With PUNCH omitted, the default print option will list the directory.

CARDS

The DMPALL execute control deck has the following format:

? EXECUTE DMPALL FILE SPEC NAME specification-file-identifier;

? DATA specification-file-identifier

(specification cards)

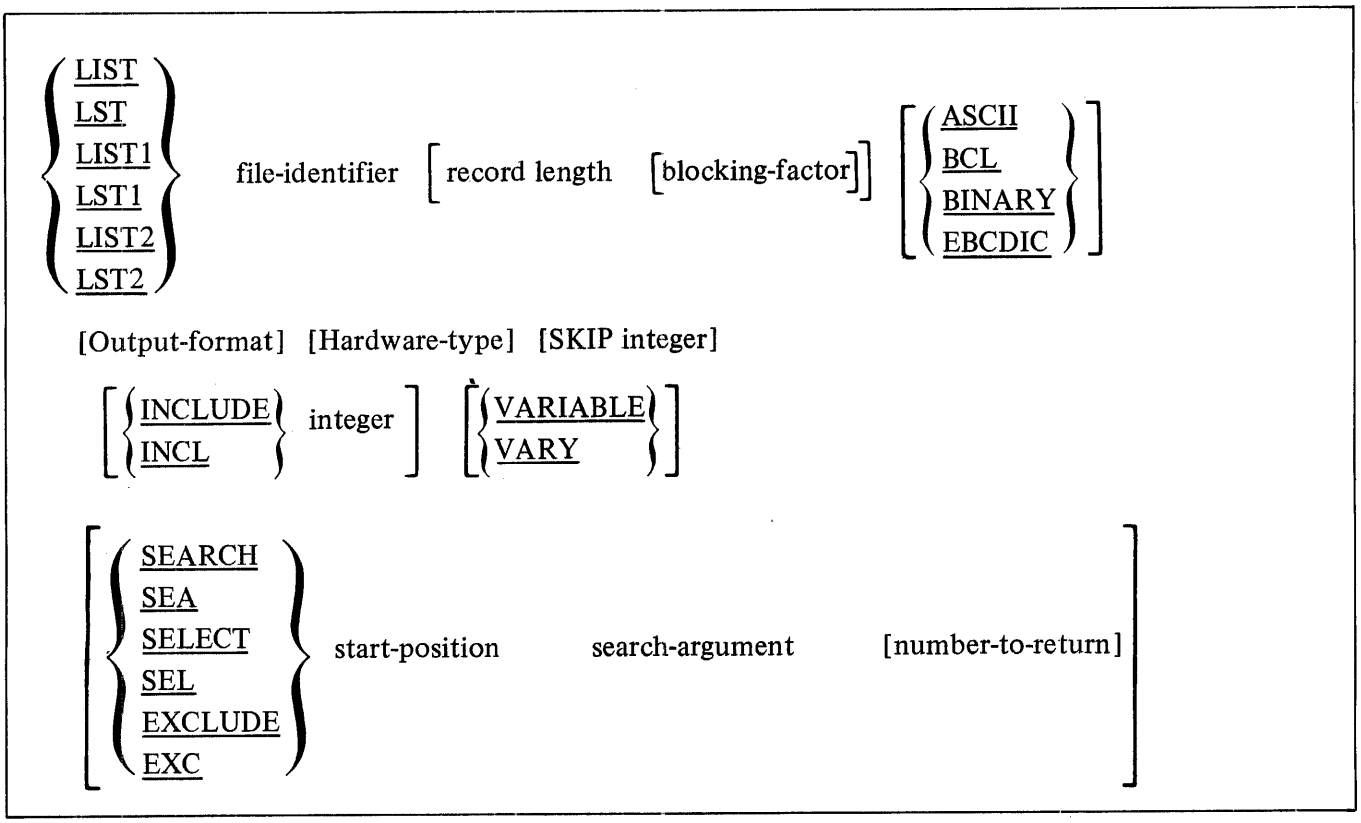
? END

A semicolon must terminate the specification string, after which comments may be entered. There may be more than one card in a specification card file.

All specification entries are free form in the first 72 columns of the card, and may be separated by either a space or a comma, or a combination thereof. The card file containing the specifications (one per card) is loaded to disk, and each specification is executed in turn from there.

Print Specifications

The specification string for printing a file is as follows:



If LIST2 or LST2 is specified, the printer listing will double-spaced; otherwise, the printer listing will be single-spaced.

The file-identifier entry must immediately follow the LIST entry, and is required for all files. The format of the file-identifier entry is the same as used MCP control instructions; therefore may consist of from one to three separate identifiers separated by slashes.· A file-identifier that is entirely numeric or which contains special characters must be surrounded by quotes.

The record-length in bytes must be the first numeric entry following the file-identifier. If omitted, a record-length of eighty is assumed. For disk files and labeled tapes the record-length used will be that of the file when created.

The blocking-factor must be the second numeric entry following the file-identifier. If omitted, a blocking factor of one is assumed. For disk files and labeled tapes when both the record length and blocking factor entry are omitted, the blocking factor with which the file was created will be used.

An optional access mode may be designated for the input file by including one of the key words ASCII, BCL, BINARY, or EBCDIC in the specification. If omitted, an access mode of EBCDIC is assumed by default. For labeled tape files, the access mode is that in which the tape was originally written.

The output-format entry may be specified as:

a. Alpha: A or ALFA.

b. Numeric: N, NUM, H, or HEX.

c. Alphanumeric: When entry is omitted.

The hardware-type entry may specify any one of the following:

	Hardware-type Entry	
Hardware Device	Long Form	Short Form
Card files	CARD	CRD
96-col. card files	CARD96	C96
Binary card files	BINARY	BIN
Magnetic tape files	TAPE	MTP
7-Track tape files	TAPE7	MT7
9-Track tape files	TAPE9	MT9
Cassette tape files	CASS	CAS
Paper tape files	PAPER	PPT
Disk files	DISK	DSK
Multi-pack disk files	MULTI	MPF

If the hardware-type entry is omitted, DISK is assumed by default. BINARY is applicable only to 80-column card files; if BINARY is specified for any input file, a card with END-OF-DECK punched in columns one through eleven (1-11) must be the last card in the input deck.

The SKIP integer entry may be entered to begin printing with a specified record as denoted by the integer.

The INCLUDE or INCL integer entry may be used to specify how many records should be included in the printout.

The VARIABLE or VARY entry may be used to specify tape or disk files having variable length records.

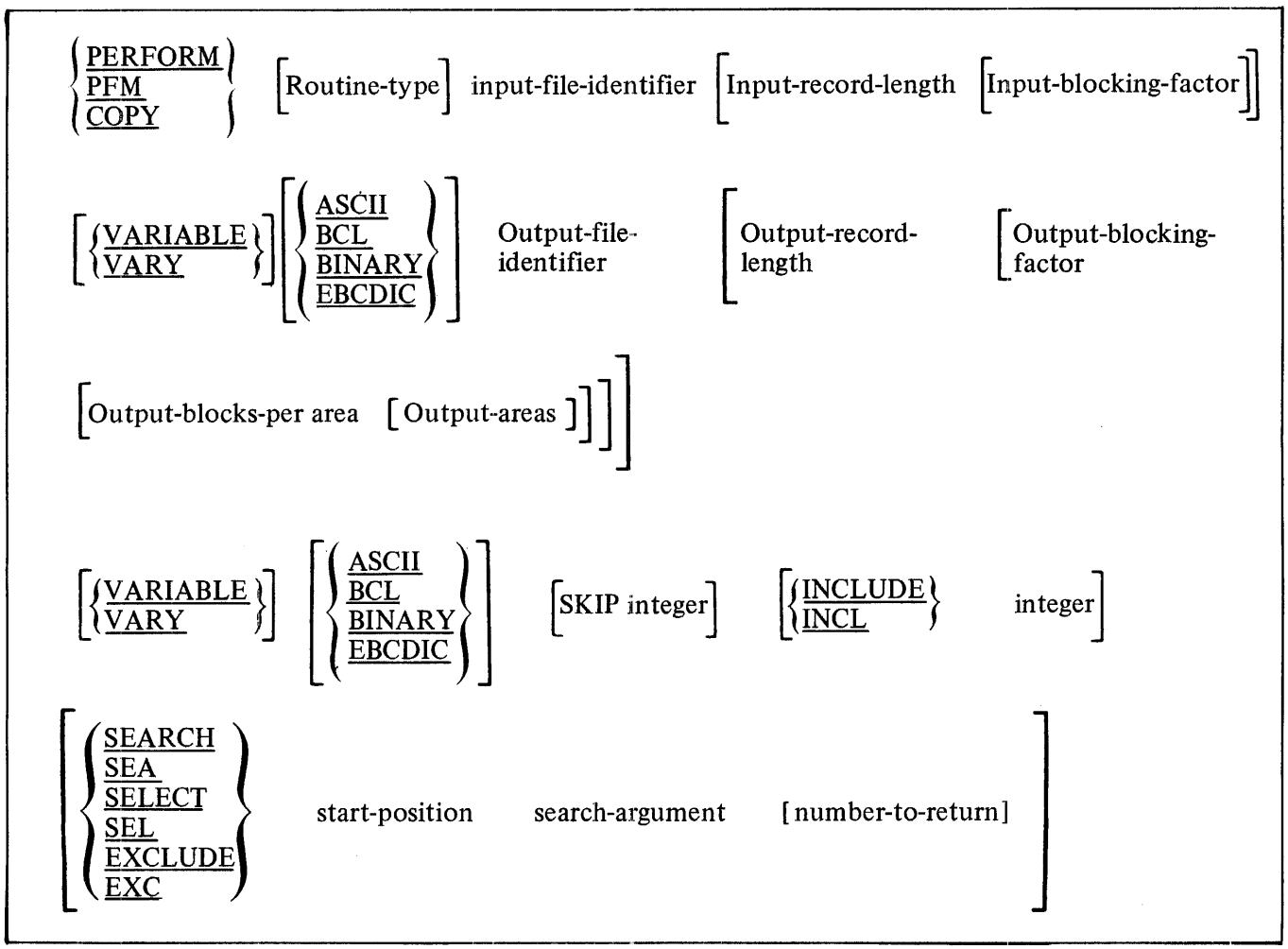
The SEARCH (or SEA) option may be used to specify that printing is to begin with the first record that contains the designated search-argument at the specified start-position (byte number) in the record. Note that the first byte in a record is considered relative position one. The search-argument must contain either EBCDIC or hexadecimal character values. An EBCDIC string is considered the default; if blanks or special characters are requested, quotes must delimit the string. If a hexadecimal argument is required, it must be delimited by "at" signs ("@").

The SELECT (or SEL) option lists only those records (within the range specified by the SKIP and INCLUDE options) which contain the designated search-argument at the specified start-position. The EXCLUDE (or EXC) option lists only those records (within the range specified by the SKIP and INCLUDE options) which do not contain the designated search-argument at the specified start-position. The number-to-return option is applicable only to the SELECT and EXCLUDE options, and specifies the number of records to list (those that satisfy the SELECT or EXCLUDE criteria) before terminating the command. Care should be taken when using both the INCLUDE and the number-to-return options in the same command, because the list is terminated when either of the two conditions is satisfied.

The printed output is headed with the file-identifier, record length, blocking factor, the current date, and the time. In addition a printout of a disk file will have the value of the End-of-File pointer in the heading. A running record count is printed in the left hand margin.

Reproducing Specifications

The reproduction string consists of the following specifications:



PERFORM, PFM, or COPY informs DMPALL that media conversion is desired.

The Routine-type entry may be either in the long-hand or short-hand form. The long-hand form utilizes two of the long form hardware-type entries, separated by spaces or the optional word "TO". The short-hand form utilizes two of the short form hardware-type entries, concatenated together (not separated by any spaces). If routine-type entry is omitted, DISK TO DISK (DSKDSK) is assumed by default.

Example:

To go from card to magnetic tape the short-hand form Routine-type would be CRDMTP. The long-hand form would be CARD TO TAPE with the TO being optional.

The format of input-file-identifier is the same as used in the LIST command.

Unlabeled tape files are specified by a file-identifier of "NONE" (quotes included).

The input-record-length must be the first numeric entry following the input-file-identifier in bytes. If omitted, a record length of eighty is assumed for all files except disk files which will use the record length of the file when created.

The input-blocking-factor must be the second numeric entry following the input-file-identifier. If omitted, a blocking factor of one is assumed. For a disk file where both the record length and blocking factor entries are omitted, the blocking-factor with which the file was created will be used.

The VARIABLE or VARY entry may be used after the input-file-identifier entries to indicate that the input file will have variable length records, but not variable length output.

An optional access mode for the input file can be specified in the same manner as used for the LIST command. To change the parity for an input tape file from ODD to EVEN, use the following FILE statement with the EXECUTE, MODIFY, or DYNAMIC control instruction:

```
FILE INP.FILE EVEN;
```

The format of the output-file-identifier is the same as for the input-file-identifier.

The first numeric entry following the output-file-identifier must be the output-record-length in bytes. If omitted, a record length of eighty is assumed unless the input file and the output file are both disk files. Then the default output-record-length will be assumed to be the same as the input-record-length.

The output-blocking-factor must be the second numeric entry following the output-file-identifier. If omitted, a blocking-factor of one is assumed unless the input file and the output file are both disk files and the output-record-length entry was omitted. Then the default output-blocking-factor will be assumed to be the same as the input-blocking-factor.

The number of blocks.per.area must be the third numeric entry following the output-file-identifier. This entry is only applicable to disk files. If omitted, 100 blocks.per.area is assumed unless both the input file and the output file are disk files and the record-length, blocking-factor entries were omitted for both the input file and the output file. Then the number of blocks.per.area for the input file will be used for the output file as well.

The Output-areas is the number of areas set for the output file. Default is 25.

The VARIABLE or VARY entry may be used after the output identifier to indicate variable length input records with variable length records being produced.

An optional access mode for the output file can be specified in the same manner as used for the LIST command. To change the parity for an output tape file from ODD to EVEN, use the following FILE statement with the EXECUTE, MODIFY, or DYNAMIC control instruction:

```
FILE OUTP.FILE EVEN;
```

The SKIP integer entry may be used to skip to a specified record prior to creating the output file.

The INCLUDE or INCL integer entry may be used to specify how many records should be included in the output file.

The SEARCH (or SEA) option may be used to specify that the copy is to begin with the first record that contains the designated search-argument at the specified start-position (byte number) in the record. Note that the first byte in a record is considered relative position one. The search-argument must contain either EBCDIC or hexadecimal character values. An EBCDIC string is considered the default; if blanks or special characters are requested, quotes must delimit the string. If a hexadecimal argument is required, it must be delimited by at-signs ("@").

The SELECT (or SEL) option copies only those records (within the range specified by the SKIP and INCLUDE options) which contain the designated search-argument at the specified start-position. The EXCLUDE (or EXC) option copies only those records (within the range specified by the SKIP and INCLUDE options) which do not contain the designated search-argument at the specified start-position. The number-to-return option is applicable only to the SELECT and EXCLUDE options, and specifies the number of records to copy (those that satisfy the SELECT or EXCLUDE criteria) before terminating the command. Care should be taken when using both the INCLUDE and the number-to-return options in the same command, because the copy is terminated when either of the two conditions is satisfied.

Examples:

a. Keyboard Console Input

EXECUTE DMPALL

DMPALL = mix-index BOJ.

DMPALL = mix-index ENTER SPECS.

DMPALL = mix-index ACCEPT.

A response of

LIST PACKA/PAYROLL/ A SKIP 50

causes a disk file located on the removable disk PACKA to be printed in alpha format beginning with the fiftieth record.

A response of

1AX COPY CRDDSK CARD SOURCE 80 2

causes a card file with the file-identifier of CARD to be written to a disk file, 80 character records, blocked 2, with a file-identifier of SOURCE.

A response of

1AX COPY PROGRAM/B CCC/PROGRAM/B

causes a disk file PROGRAM/B located on a system disk to be copied to the removable disk CCC with the file-identifier PROGRAM/B. The new copy on disk CCC will be an exact copy. Therefore, record length, blocking, number of areas, and area size will be the same as the original file.

b. Card Input

? EXECUTE DMPALL FILE SPEC NAME SPECCARDS will allow the operator to enter any number of specifications via a card reader. DMPALL will look for a card file with the file-identifier SPECCARDS. The specifications will be loaded to disk, and then executed one-at-a-time from there.

```
? EXECUTE DMPALL FILE SPEC NAME SPECCARDS;
? DATA SPECCARDS
  COPY CRDDSK XXX 80 1 DSKFIL 80 1
  LIST DSKFIL A
? DATA XXX
  (card data deck)
? END
```

The specifications will cause the card file XXX to be loaded to disk, then listed in alpha format.

DMPALL displays a message upon encountering any errors in an input file. DMPALL also interprets punched card output files (if punched on an 80- or 96-column data recorder). These capabilities may be changed, if desired, by setting switch nine (9) to one of the following values:

SW 9	Function
1	Ignore input errors
2	Suppress punched card interpretation
3	Both of the above

FILE/LOADER

General

The purpose of FILE/LOADER is to load card decks to disk punched by the program FILE/PUNCHER.

The FILE/LOADER card deck consists of the standard EXECUTE control card, a dollar card, an asterisk card, the data cards, and the END card.

Dollar Card

The dollar card is output by FILE/PUNCHER and identifies the file to be loaded. The dollar card can also be modified by the operator to change the name of the file-identifier.

The format of the FILE/LOADER dollar card is:

$ file-identifier

The "$" must be in column one and the file-identifier being free-form from column 2 through 80.

Dollar Dollar Card ($$)

Files produced by the MIL compiler (Micro Implementation Language) must be loaded using the $$ card to distinquish them from card files output by FILE/PUNCHER. The asterisk (*) card must not be used when using the $$ card.

Below is the card format produced by the MIL compiler which takes six cards to fill a disk segment.

Column	Description
1-6	Load address (Relative)
7	Blank
8-9	Number of bits on card
10	Blank
11-70	Data in hexadecimal format (30 Bytes)
71-72	Blank
73-80	Card sequence number

The format of the FILE/LOADER dollar dollar card is:

$$ file-identifier

Asterisk Card

The asterisk card is used to input the values for the file which is being loaded to disk. This card is produced by FILE/PUNCHER and should not be changed prior to input. When the asterisk card is missing, the card file is assumed to be a code file. The asterisk card must not be used when the first card of the file is a dollar dollar ($$) card.

The format of the FILE/LOADER asterisk card is:

Column	Description	
1	"*" Asterisk Sign	
3	File Type 1 LOG 3 Control Deck 4 Backup Punch 5 Backup Print 6 Dump 7 Interpreter 8 Code 9 Data	
5-10	EOF pointer	Right Justified, Leading Zeros Optional
12-17	Record Size in bits	
19-20	Records.per.Block	
22-24	Areas	
26-31	Segments.per.Area	

NOTES

(1) If a code file is being loaded, the asterisk card is optional and default values are assumed.

(2) If a code or interpreter file is designated on the asterisk card, only the EOF pointer is used. All other fields are ignored. If the EOF pointer field is blank, 100 segments for the interpreter or 500 segments for the code will be used as default values.

(3) All code and interpreter files will be closed with CRUNCH which frees the area not being used for the file.

Example:

```
? EXECUTE FILE/LOADER DATA CARDS
$ file-identifier
* . . . (Optional)
  data deck
[$ file-identifier ]
[*  ————————     ]
[  data deck       ]
? END
```

RESPONSE: File-identifier LOADED (Displayed after each load)

Error Messages

MISSING "$" IN COLUMN ONE

The first card of the input deck does not have "$" in column one.

MISSING file-identifier

The first card of the input deck has a "$" in column one, but is otherwise blank.

SEQUENCE ERROR FOLLOWING nnnnnnn–file-identifier NOT LOADED

The card following the card number specified is out of sequence.

RECORD.SIZE SPECIFIED nnnn–file-identifier NOT LOADED

AREAS SPECIFIED = 0 – file-identifier NOT LOADED

RECORDS.BLOCK SPECIFIED = 0 – file-identifier NOT LOADED

SEGMENTS.AREA SPECIFIED = 0 – file-identifier NOT LOADED

EOF.POINTER SPECIFIED = 0 – file-identifier NOT LOADED

INVALID FILE TYPE SPECIFIED–file-identifier NOT LOADED

BLOCK SIZE 56 – file-identifier NOT LOADED

EMPTY DECK–file-identifier NOT LOADED

There are no cards following the specification card(s).

"*" CARD INVALID–file-identifier NOT LOADED

An asterisk card following a dollar dollar card is invalid.

FILE/PUNCHER

General

The purpose of FILE/PUNCHER is to output disk files to cards in a hexadecimal format that is acceptable as input to FILE/LOADER. The dollar card and the asterisk card used by FILE/LOADER are also output when FILE/PUNCHER is executed.

The file-identifier is supplied to the program by an AX input message. For example:

EXECUTE FILE/PUNCHER

FILE/PUNCHER=mix-index ENTER FILE IDENTIFIER
FILE/PUNCHER=mix-index ACCEPT

mix-index AX file-identifier (free-form)

After punching the output file, the program will repeat the above messages and wait for another file-identifier to be entered. By responding with a blank file-identifier, the program will go to EOJ.

Below is the card format produced by FILE/PUNCHER which takes five cards to fill a disk segment.

Column	Description
1-72	Data in hexadecimal format (36 bytes)
73-80	Card sequence number

Error Messages

file-identifier NOT ON DISK

The file-identifier requested for output cannot be located by the MCP.

SORT

General

SORT is a system program that provides a means to invoke one of four sort intrinsics used for sorting or merging files of records. Specifications describe the input and output files, the keys by which the file(s) are to be sorted or merged, the sort intrinsic to be used, and the various sort options desired.

All SORT program reserved words and characters appear in uppercase type throughout the SORT subsection of this publication. A list of the SORT reserved words and characters appears at the end of this subsection.

Sort Intrinsics

A parameter table is generated by the SORT program and used by one of the four sort intrinsics:

a. SORT/QSORT

b. SORT/VSORT

c. SORT/MERGE

d. SORT/TAPESORT

The sort intrinsic does the actual sorting or merging of the file(s) in ascending or descending sequence, according to assigned keys, in an optionally user-specified, virtual collating sequence.

The SORT/QSORT intrinsic uses an inplace disk-sorting technique and can be specified when there is a minimum amount of disk space available. The SORT/QSORT intrinsic is invoked when the optional INPLACE specification statement is included in the SORT program specification card deck.

The SORT/VSORT intrinsic is a balanced-merge, vector sort with workfiles on disk, and is the default sort intrinsic.

The SORT/MERGE intrinsic merges from two to eight separate files according to common sets of ordered keys. This intrinsic is implicitly specified when a FILE statement containing more than one input-part is included in the SORT specification card deck.

The SORT/TAPESORT intrinsic is an unbalanced-merge, vector sort that uses from three to eight magnetic tape files as workfiles. This intrinsic can be specified by including the optional ⟨integer⟩ TAPESORT statement in the SORT program specification card deck.

SORT Execution Card Deck

Figure 3-2 contains an example of a SORT execution card deck.

If SORT is executed with switch 0 set to one (1), the card input file is not used and specifications are entered as a series of ACCEPT (AX) messages. A blank (null) ACCEPT message terminates the series and begins the sort procedure.

The generated sort program may either be executed immediately or compiled to library for subsequent execution. If SORT is called by an EXECUTE control instruction, the specifications are processed and the proper sort intrinsic is immediately invoked (if no syntax errors are detected). If, however, SORT is called by a COMPILE TO LIBRARY control instruction, an object program is compiled to the disk library for execution at a later time. For example, the following control string will compile an object sort program named SORTPROG to disk:

```
COMPILE SORTPROG WITH SORT TO LIBRARY
```

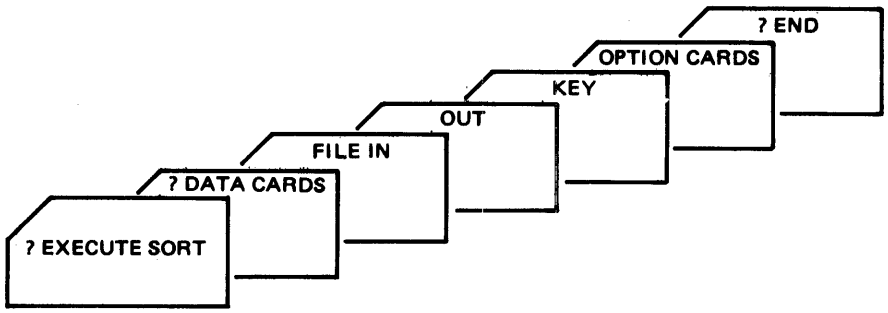


Figure 3-2. SORT Execution Card Deck

This compiled program may be executed at any time thereafter in the normal manner, for example:

EXECUTE SORTPROG

Three options are not allowed when a sort program is compiled to library. These are INCLUDE IN, DELETE IN, and DUPCHECK (with parameters). DUPCHECK specified without IN or OUT parameters is allowed, however.

NOTE

In order to compile a sort program to library, the data file SORT/SKELETON must be present on system disk. This file is used as input to the SORT program when a COMPILE is performed.

The SORT program specifications contain required and optional statements that are parameters to the sort. They are in free-form format, and are described in the following pages.

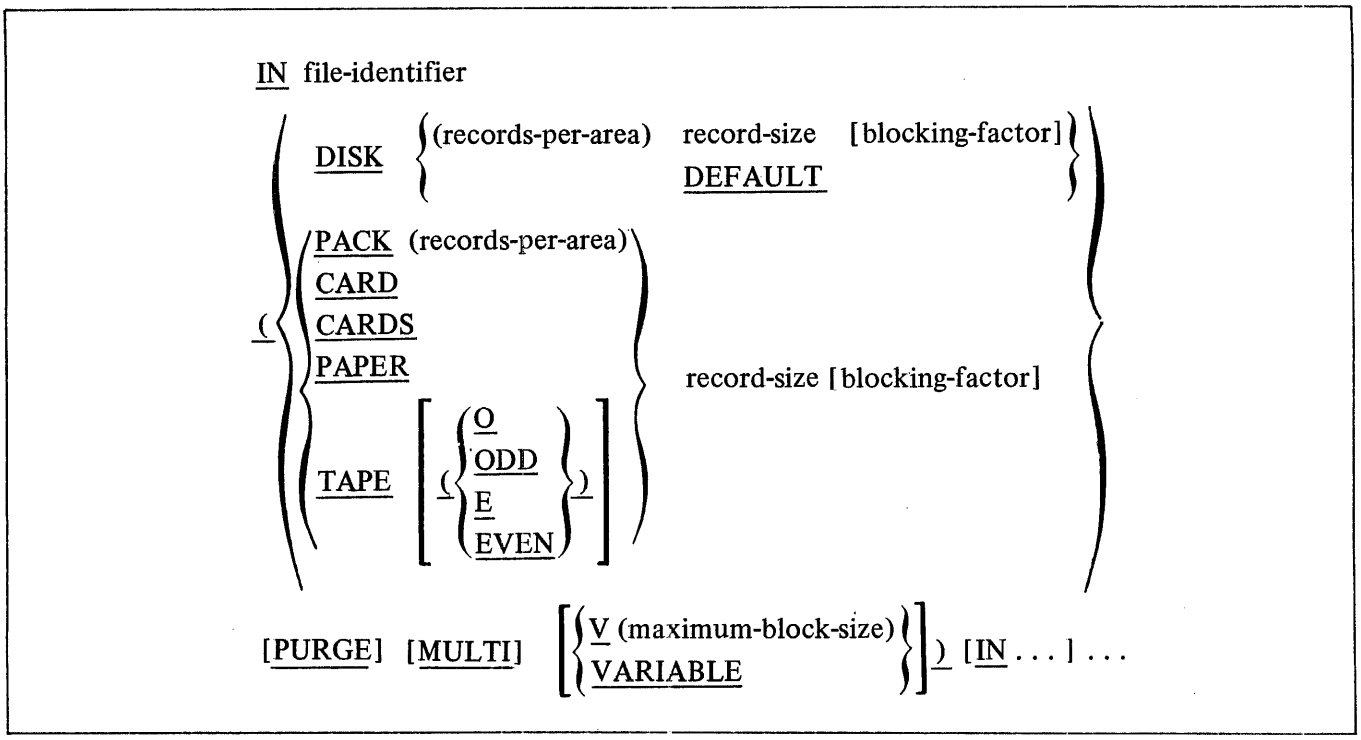
FILE Statement

The FILE statement is comprised of the reserved word FILE followed by at least one input-part (a description of the input file(s)) and only one (a description of the output file) output-part. The FILE statement is required in any set of SORT specifications, and has the following format:

FILE input-part [input-part] . . . output-part

INPUT-PART

The input-part of the FILE statement describes an input file to be merged or sorted, and has the following format:



FILE-IDENTIFIER

File-identifiers are the standard B 1800/B 1700 file format with the exception that any element (disk-pack-id, multi-file-id, or file-id) containing one or more non-alphanumeric characters, except the period, must be enclosed in quotation marks.

Example:

PACK1/WORK.FILE/"#00000002"

File-identifiers can contain any valid character except the question mark or quote mark.

DEVICE-ID

The device-id denotes the hardware type of the device associated with the input file. The allowable designations are as follows:

Device-Id	Hardware Type
DISK	Any disk
PACK	Disk pack only
CARD	Card reader
CARDS	Card reader
PAPER	Paper tape reader
TAPE	Any tape
TAPE (parity-specifier)	7-track tape

PARITY-SPECIFIER

Seven track tape is implicitly specified by the inclusion of a parity-specifier enclosed in parentheses following the tape device-id. ODD or O specifies odd parity, binary mode; EVEN or E specifies even parity with BCL translation.

RECORDS-PER-AREA

When the input file is located on disk, and the DEFAULT option is not used, the number of records-per-area must be specified and enclosed in parentheses.

RECORD-SIZE

The record-size is the actual size in bytes (characters) of the records to be sorted, and is a required entry except where the DEFAULT option is specified.

BLOCKING-FACTOR

The blocking-factor is optional and specifies the number of logical records in a block. When the blocking-factor is omitted, the default blocking-factor of one (1) applies.

DEFAULT

If the input file is on disk, the user may specify DEFAULT, which causes the SORT program to obtain the input file specifications (records-per-area, record-size, and blocking-factor) from the disk file header.

PURGE

If this option is specified, one of the following will occur when the sort intrinsic has finished reading the input file.

a. If the input file is a disk file, it will be removed from the disk directory.

b. If the input file is a magnetic tape file, the tape will be purged.

MULTI

The MULTI option allows a user to sort a multi-pack disk file. If MULTI is specified on the input-part of the FILE statement, it must also be specified in the output-part.

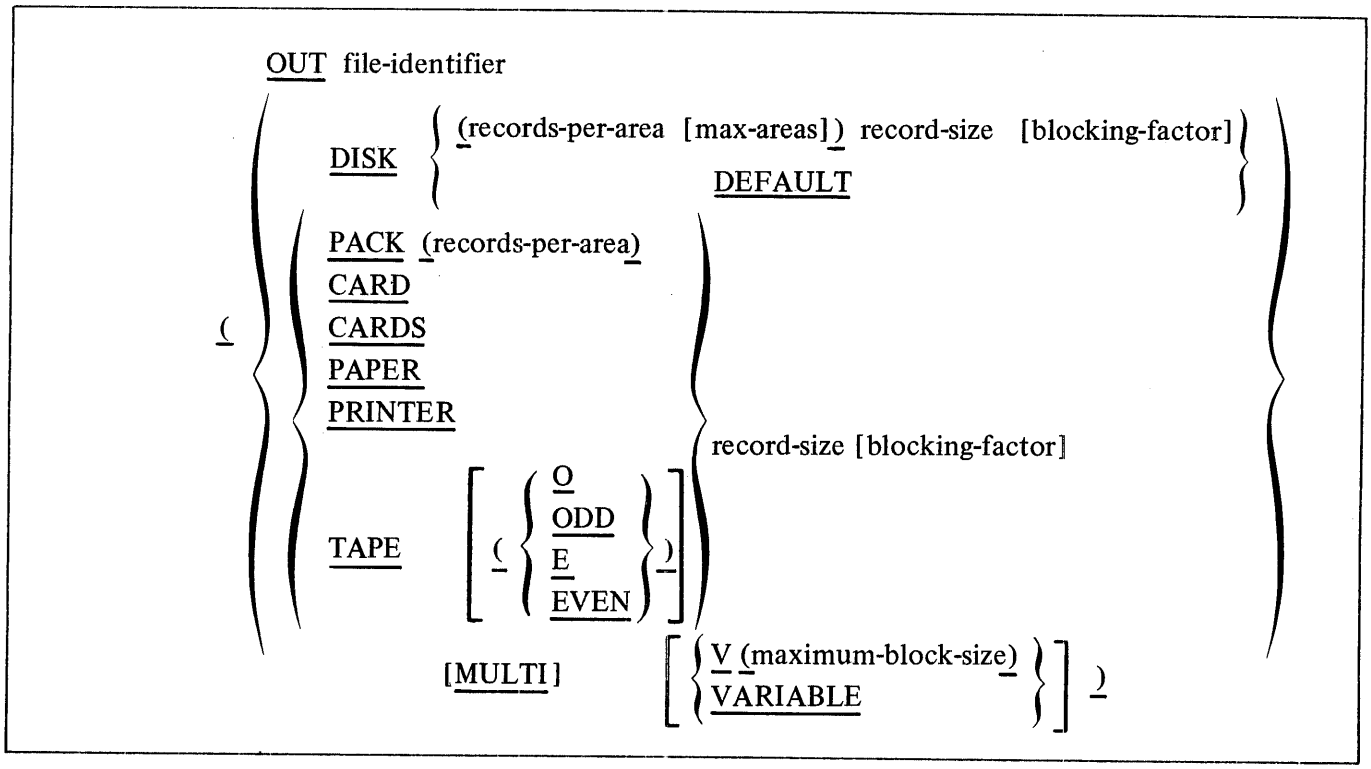
VARIABLE

The VARIABLE or V option allows the user to sort a file of variable length records. When V is specified, a maximum-block-size must be included. The default, when VARIABLE is specified, is equal to record-size. The workfile record size is fixed, and is equal to the size of the largest record specified in the input-part of the FILE statement.

Variable length records cannot be sorted with the INPLACE sorting technique.

OUTPUT-PART

The output-part of the FILE statement describes the sorted or merged file created by the sort intrinsic, and has the following format:



The elements of the output-part have the same relative function to the output file as the elements of the input-part have to the input file, with the exceptions described in the Device-Id and Default paragraphs below:

DEVICE-ID

The device-id denotes the hardware type of the device associated with the output file. The allowable designations are as follows:

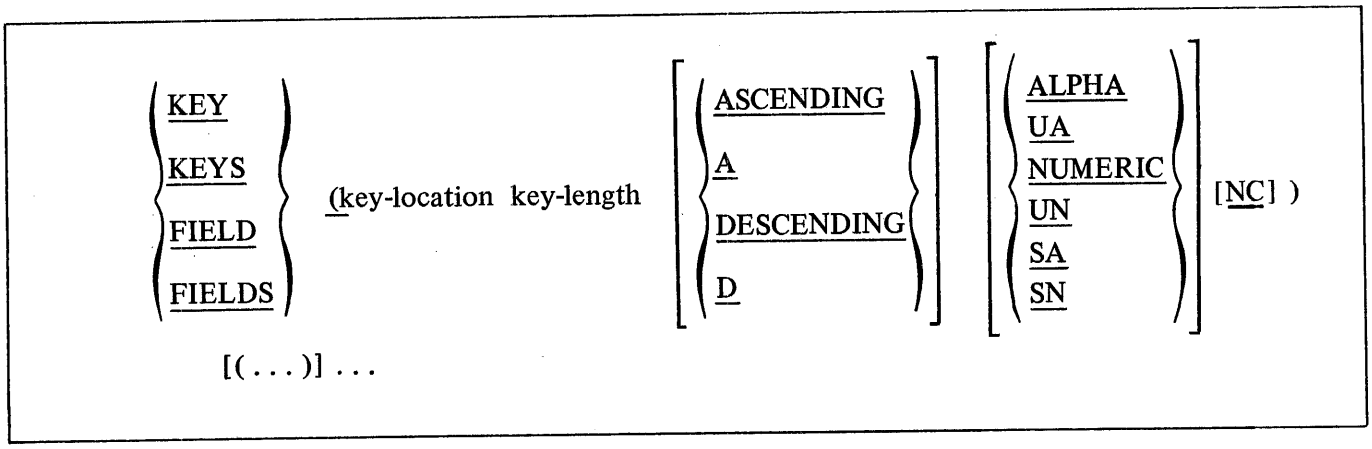
Device-Id	Hardware Type
DISK	Any disk
PACK	Disk pack only
CARD	Card punch
CARDS	Card punch
PAPER	Paper tape punch
PRINTER	Line printer
TAPE	9-track tape
TAPE (parity-specifier)	7-track tape

DEFAULT

The DEFAULT option can be specified for the output file only if DEFAULT is also specified for the input file. This option causes the records-per-area, record-size, and blocking-factor of the output file to be identical to those found in the input file disk header. DEFAULT cannot be specified when the SORT/MERGE intrinsic is invoked.

KEY Statement

The KEY statement defines the field(s) within a record by which the record is to be sorted or merged. The KEY statement is required in any set of SORT program specifications, and has the following format:



Multiple key descriptions are allows and must be enclosed in parentheses. The first key is the major key, and any additional keys are minor keys of successive, decreasing significance. Each minor key specified is subordinate to its predecessor.

The maximum number of keys permitted is 30 unsigned keys, or 15 signed keys, or any combination of signed and unsigned keys not exceeding 30, where each unsigned key counts as one and each signed key counts as two.

KEY-LOCATION

The key-location specifies the relative position of the most significant byte or digit (alphanumeric or numeric) of the field including the sign, if any, from the beginning of the record.

The first byte or digit of a record is considered as relative position one. The position is referred to in the number of units applicable to the data type for that key, thereby permitting different data types to appear within the same record.

Additional information and descriptions of key-location and position will be found in the following subsections titled ALPHA or UA and NUMERIC or UN.

KEY-LENGTH

The key-length specifies the number of significant bytes or digits in the key, including the sign where applicable. The length of a key cannot exceed 511 bytes or 1023 digits.

ASCENDING OR A

ASCENDING or A is the default condition, but can be specified if desired. The file is arranged with the record having the smallest major key appearing first in the output file, followed by records with successively equal or larger keys.

DESCENDING OR D

Specifying DESCENDING or D results in the file being arranged with the record having the largest major key appearing first in the output file, followed by records with successively equal or smaller keys.

ALPHA OR UA

ALPHA or UA (unsigned alphanumeric) indicates that the data is alphanumeric, and the key-location of the field is counted in 8-bit units from the beginning of the record. Alphanumeric is the default when no data type is specified.

NUMERIC OR UN

NUMERIC or UN (unsigned numeric) indicates that the data is 4-bit numeric, and the relative position of the field is counted in 4-bit units.

SA

SA (signed alphanumeric) indicates that the data is alphanumeric and that some, or all of the keys can contain a minus sign. For alphanumeric data, a minus sign is represented by a hexadecimal D in the most significant four bits of the first byte of the key.

SN

SN (signed numeric) indicates that the data is 4-bit numeric and that some or all of the keys can contain a minus sign. The minus sign is represented as a hexadecimal D in the first digit of the key.

NC

NC (no collate) specifies that the collate file is not to be used for the designated key (refer to the COLLATE option).

SORT Option Statements

The SORT program option statements have the following functions:

a. Specifying information about the input file, such as the number of records in the file or the bias (explained below) of the file.

b. Controlling the configuration or operation of the sort by using such options as MEMORY, INPLACE, TAGSORT, and RESTART.

c. Causing optional operations to be performed before, during, or after the sort execution. NOPRINT, SYNTAX, SEQUENCE, TIMING, ZIP, and COLLATE are examples of this type of function.

Each of the SORT program option statements is discussed in detail in the following paragraphs.

BIAS

The BIAS statement is used to indicate to the sort program the degree of ordering of the input file in relation to the specified key(s). The estimate is used to optimize sort execution.

The word BIAS must be followed by a number within the range of zero (0) to ninety-nine (99), where fifty (50) indicates completely random order and is the default if BIAS is not specified. Zero (0) indicates that the file is in reverse order from the sequence desired. A ninety-seven (97) indicates that the file is almost in the desired sequence.

Example:

BIAS 60%

The percent sign (%) is optional.

The bias of a file can be determined by specifying the SEQUENCE statement.

The BIAS option is not applicable for the INPLACE sort (SORT/QSORT).

⟨integer⟩ RECORDS

This option can be used to optimize sort operation by supplying an estimate of the total number of records in the input file(s). If this option is omitted, the default is 20,000 records.

Example:

12500 RECORDS

MEMORY

The MEMORY or ME option can be used to allocate more memory to the sort than the 8000 bytes assigned by default.

Increasing the memory available to the sort is the most significant means of increasing sort efficiency, up to an optimum sort memory size. The optimum sort memory size is dependent of many factors, but is usually 25 percent less than system memory size. The maximum sort memory size that can be specified is 125,000 bytes; 18,500 bytes for INPLACE sort.

Example:

MEMORY 24000

INPLACE

This option can be advantageous when only a minimum amount of disk space is available for sorting. The INPLACE sort requires work file space equal to the input file space, unless the input and output file identifiers are the same. In that case, no work file space is required, but the input file is overlaid by the output file during the sorting process.

TAGSORT

The TAGSORT option provides a means of sorting a file and creating an output file containing indices that point to the relative locations of records within the original file. The input file remains intact.

Input files can originate from any allowable hardware device type, but must reside on disk when using the file of indices to access the actual records.

The output file, referred to as the tagfile, must be defined as four characters per record, because it will consist of eight numeric-digit decimal numbers as indices that point to the records in the input file.

TAGSORT is allowable in SORT/VSORT and SORT/TAPESORT only.

Access to the input file is provided by using the tagfile as follows:

a. With COBOL the specified access method is RANDOM and the tagfile record is used as the ACTUAL KEY.

b. With RPG the specified access method is INDEXED and the relative record number is used as delivered in the tagfile to directly (DIRECT) access the original file.

TAGSEARCH

Specifying TAGSEARCH causes a TAGSORT to be performed, and a resultant sorted file of indices is used to build an output record file that is sorted. Beyond an undetermined number of records, TAGSEARCH can greatly increase overall sorting speed when compared with the other sorting methods available.

⟨integer⟩ TAPESORT

The SORT/TAPESORT intrinsic is invoked when ⟨integer⟩ TAPESORT is specified. The number of workfiles on magnetic tape can range from three to eight and is specified by ⟨integer⟩.

The workfiles are opened on whatever tape units are available. If tape output is requested, the first tape unit accessed as a workfile is of the same hardware type as specified for the output file. The sort algorithm causes this tape unit to be exhausted of data records on the merge pass that precedes the final merge pass, thereby making that tape unit available to receive the sorted output file.

All writing of workfile tapes is done in a forward direction, and all reading of workfile tapes is done in reverse direction; therefore, multiple tape reels are not allowed for any individual workfile. The user must use workfile tape reels of sufficient size to hold the original input file. For multiple reel input files, each tape reel can be sorted separately, and the resultant output files can then be merged using the SORT/MERGE intrinsic.

The following example specifies that five workfiles are to be used for TAPESORT:

```
5 TAPESORT
```

A feature of TAPESORT which aids in guaranteeing the integrity of the data is tapesort self-checking. As each block of data is written on a worktape, it is concatenated with the contents of an incremented counter. Thus, each block contains a number representing its relative position on the tape. Upon reading the worktape, these values are checked for incremental sequence.

RESTART

A sort which is abnormally terminated may be restarted at a point following the last successful pass in the following manner:

a. RESTART must be specified in any set of sort specifications if the ensuing sort is to be considered restartable.

b. After abnormal termination, the sort may be restarted by inserting a RESTART ⟨job-number⟩ statement into the sort specification deck and re-executing the sort. The ⟨job-number⟩ specified should be that which was assigned to the sort intrinsic at the time of the failure.

The RESTART feature is available for SORT/VSORT and SORT/TAPESORT.

Example:

```
RESTART 1259
```

NOPRINT

The NOPRINT option can be used to inhibit the printing of the SORT specifications on the line printer, and must be the first entry in the SORT specification deck.

The TIMING option is not affected by the use of the NOPRINT option.

SYNTAX

The SYNTAX option is used when the SORT specification cards are to be checked for errors only. The sort intrinsic is not executed, even when no errors are detected in the specification cards.

SEQUENCE

The SEQUENCE option checks the sequence of the input file according to the specified key(s). The number of records in the file and the bias are printed on the sort specification listing.

If the NOPRINT option is used, the number of records and the number of sequence errors are displayed on the console printer. The bias may be calculated as follows:

BIAS = 99 - ((Number of sequence errors x 100)/Number of records), rounded to the nearest integer.

The sort intrinsic is not invoked when SEQUENCE is specified.

TIMING

The TIMING option can be specified when the SORT/VSORT intrinsic is used, and provides the following information:

a. Distribute, Merge, and Final Merge Vector Size (VECTOR) in records.

b. Workfile blocking factor (BLOCK).

c. Workfile records-per-area (DISKRPA).

The above information is useful for debugging purposes.

The TIMING option is not affected by the use of the NOPRINT option.

ZIP

The ZIP statement contains the reserved word ZIP followed by a zip-string that is enclosed in parentheses. The zip-string is passed to the MCP upon successful completion of the sort. The zip-string cannot contain embedded right parentheses or question marks, and is limited in length to 100 characters.

Example:

```
ZIP (EX A/B AFTER SORT FILE DISK NAME C/D)
```

The SORT specifications can contain only one ZIP statement.

COLLATE

GENERAL

The COLLATE option invokes the optional, virtual collating-sequence capability according to a user-specified collate file. The collate file can be used to specify a new or unique collating sequence, or to retain the standard collating sequence with the exception of certain characters being interchanged or made equal in rank. This option permits the alteration of the sequence in which records are sorted or merged by any of the sort intrinsics. Alteration of the collating sequence may be desired for foreign alphabets or when converting from one processing system to another.

The format of the COLLATE statement consists of the reserved word COLLATE followed by a file-name enclosed in parentheses. The file-name is subject to the constraints described under the heading File-Identifier of the FILE statement subsection.

Example:

```
COLLATE (PACK2/TRANS.FILE/"#003")
```

The COLLATE option affects only those sort keys that are declared unsigned alpha (UA). All other sort keys are sorted in the hardware collating sequence of @00@ through @FF@, which is also the default collating sequence if no collate file is specified.

FUNCTIONAL DESCRIPTION

When the COLLATE option is specified, the MCP interface verifies that the collate file is on disk before processing. If it is not present, the user is directed to load the file by a console printer message. The MCP verifies the header information prior to opening the collate file to insure that the file consists of two 256-byte records in a single-area file; if not, a message is displayed and the SORT program is discontinued.

The sort or merge intrinsic brings the first record of the collate file into memory and, as the key(s) are extracted from each record, the keys which are declared unsigned alpha are processed through a translation operation before being passed to the sort or merge intrinsic comparison logic.

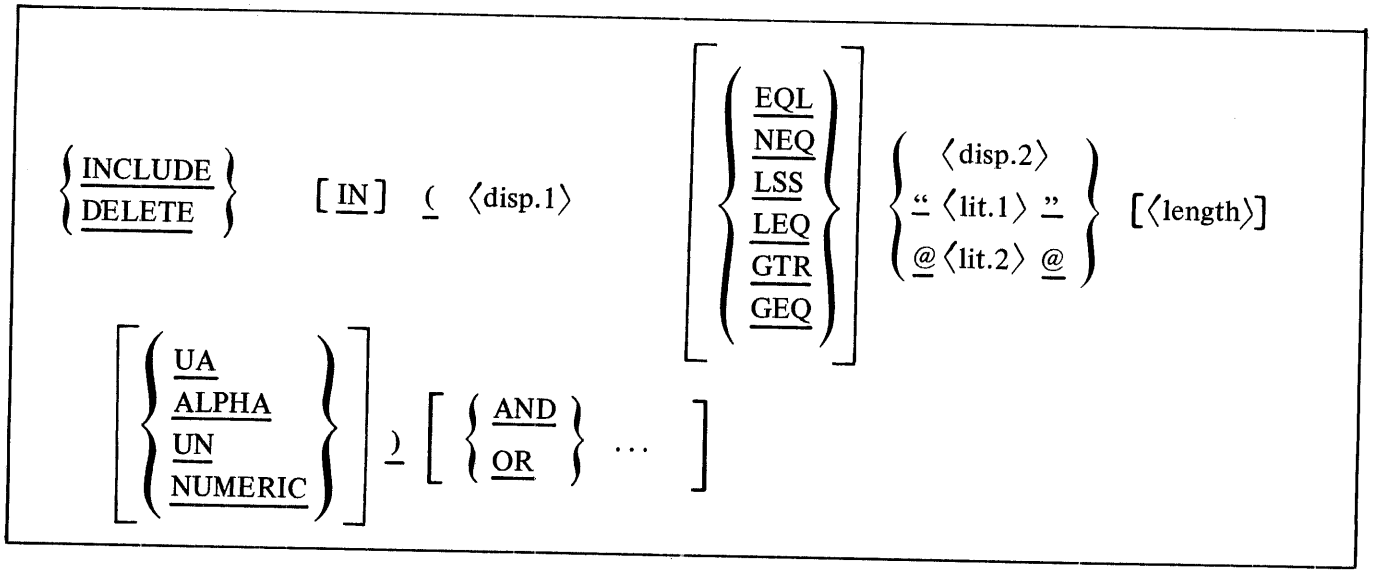
During the final merge phase of the sort intrinsics, the second record of the collate file is used to restore records to their original values.

For information pertaining to the creation of collate files, refer to the subsection titled COLLATE FILE GENERATOR.

INCLUDE/DELETE

The INCLUDE/DELETE option allows the inclusion or exclusion of certain records in a file, based upon relational evaluations of portions of the record with other portions of the same record or with specified literals. The function is available in SORT/VSORT, SORT/TAPESORT, and as a stand-alone function of SORT.

The syntax of the INCLUDE/DELETE statement is as follows:



The reserved words INCLUDE and DELETE can appear only once within a set of sort specifications, but can each be followed by a plurality of keys. The total number of keys may not exceed ten. Multiple keys within an INCLUDE or a DELETE are joined by the reserved words AND or OR. The resulting expressions

are evaluated element by element, left to right. (Note that if an OR is encountered, and all previous expression elements are satisfied, the evaluation is terminated as true. If an AND is encountered, and all previous expression elements are not satisfied, the evaluation is terminated as false. In all other circumstances, the entire expression is evaluated to a true or false conclusion.)

Specifying IN causes the INCLUDE/DELETE option to be performed within SORT, and a file is written based upon the output file specifications. No sort intrinsic is called. If COLLATE has been specified, the INCLUDE/DELETE option is performed while the file is in the re-collated state. If IN is not specified, the INCLUDE/DELETE option is performed within the appropriate sort intrinsic and, if COLLATE is specified, before the file is re-collated. (Note that the relationship of order between the COLLATE option and the INCLUDE/DELETE option is dependent upon whether it is performed in SORT or in one of the intrinsics.)

Disp.1 specifies the offset of the key (first operand) from the beginning of the record. The offset is in bytes or digits, depending upon the ALPHA or NUMERIC entry. The length of the field is as specified in the length field or a default of one if no length is specified.

The entry following the key displacement is the relational operator to be applied to the comparison of the specified operands. Valid entries are EQL (equal), NEQ (not equal), GTR (greater), GEQ (greater or equal), LSS (less), and LEQ (less or equal). Default, if no entry is specified, is equal.

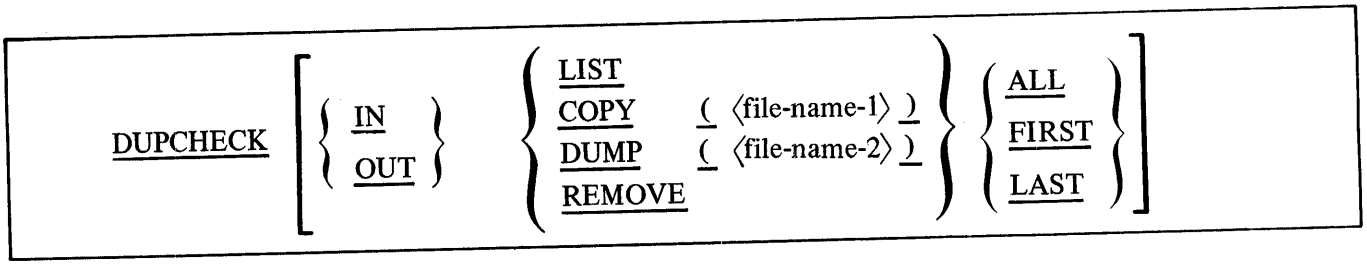
The second operand (disp.2) may be specified as a displacement into the record or as an alphanumeric or hexadecimal literal. As a displacement, the second operand is subject to the rules outlined for the first operand. Hexadecimal literals are delimited by at-signs (@) and may contain one to six hexadecimal digits. Character literals are delimited by quote marks, and may contain up to three characters. All characters are valid with the exception of the quote or the question mark. Literals are padded or truncated as necessary, based on the length attribute.

The length field specifies the length of the operands to be compared. The default, if no length is specified, is one. The maximum allowable length is three.

ALPHA and NUMERIC specify whether the operands are character or digit oriented. These entries can be abbreviated UA and UN respectively. Default is UA if nothing is specified.

DUPLICATE CHECKING

Duplicate checking is a means of describing an action to be performed upon the occurrence of records that are duplicated within the scope of the defined keys. DUPCHECK is available as a function of SORT, and may be used in conjunction with the sort intrinsics when specified for output. The syntax is as follows:



If DUPCHECK is specified without parameters, duplicate record checking is performed by the sort intrinsic during the output phase. If any duplicate records are detected, a disk file is built consisting of pointers to the records that are duplicates. The sort intrinsic also displays a warning message on the SPO, giving the name of the disk file containing the pointers. This file, labeled "SD.⟨integer⟩", consists of 8-byte records. Each record contains two 8-digit (4-byte) numbers. The first number is the relative position of the duplicate record in the output file, and the second number is the relative position of the record that it duplicates. Note that the first record in the output file is considered record number 1.

Specifying IN or OUT indicates at what point the duplicate checking is to be performed. If IN is specified, the duplicate checking is performed by SORT and no intrinsic is called, as the file is assumed to be ordered upon the specified key(s). An output file is produced, except in the case of a COPY or a LIST, where the input file is not altered. Specifying OUT will cause the output file to be checked in SORT after it has been sorted by the intrinsic.

Specifying LIST causes the duplicate records to be listed on the line printer. COPY creates a file with the name ⟨file-name-1⟩ containing a copy of the duplicate records. Neither the LIST nor the COPY options cause records to be removed from the file. REMOVE will cause the duplicate records to be removed and discarded, where DUMP builds a file of the removed records under the name ⟨file-name-2⟩ . The files created by the COPY and DUMP options have the same attributes as the input file.

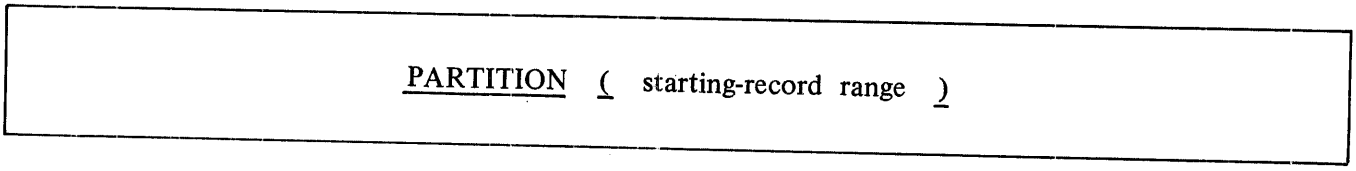
If ALL is specified, the set of duplicate records consists of all the subsets of records that are duplicates of each other. If FIRST is specified, the set of duplicate records consists of all records on all subsets except the first record in each subset. Likewise, LAST causes the set of duplicate records to consist of all records in all subsets except the last record in each subset.

If the COLLATE option is used in conjunction with DUPCHECK, the duplicate checking is performed while the file is in the re-collated state.

PARTITION

The PARTITION option is available in SORT/VSORT and SORT/TAPESORT, and allows any specified portion of the input file to be sorted. Partitioning occurs prior to any other file manipulations.

The syntax of this option is as follows:



The number of the first record in the file to be sorted is specified by starting-record (note that the first record in the file is considered number 1). The range portion indicates the number of records to be sorted.

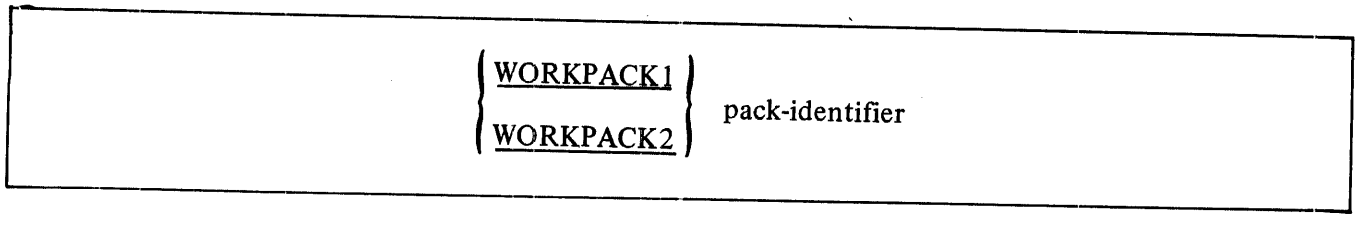
Only one PARTITION option may be included in a SORT specification deck.

PARITY DISCARD

The PARITY DISCARD option causes input file records to be discarded if an irrecoverable parity error is detected during the reading of the input file. If PARITY DISCARD is specified, the block of data in which the error occurs is discarded, and notification for each record lost, along with its relative record position is displayed on the console printer. If PARITY DISCARD is not specified, and a parity error occurs, the program is discontinued.

WORKPACKS

It is possible to designate specific user packs/cartridges for use by the two SORT/VSORT workfiles, W1 and W2. The syntax of the WORKPACK options is as follows:



Both WORKPACK options may be declared in one set of SORT specifications. The WORKPACK options are valid only when using the SORT/VSORT intrinsic.

Comments

Any commentary may follow a colon (:), which terminates SORT specification record scanning to the end of the next logical record. Otherwise, comments may be interspersed between SORT statements, but must not contain level 1 SORT reserved words.

SORT Reserved Words and Characters

A list of SORT reserved words and characters follows. The reserved words and characters in this list cannot be used in comments, unless the comment is preceded with a colon (:).

BIAS	KEY	SEQUENCE
	KEYS	SYNTAX
COLLATE		
	ME	TAGSEARCH
	MEMORY	TAGSORT
DELETE		TAPESORT
DUPCHECK		TIMING
	NOPRINT	
FIELD		
FIELDS		
FILE		
		WAIT
	PARITY	WORKPACK1
	PARTITION	WORKPACK2
	PASSPARAM	
		ZIP
INCLUDE	RECORDS	
INPLACE	RESTART	(

COLLATE FILE GENERATOR

General

The SORT/COLLATE program accepts input in the form of language statements described in this subsection, and generates a collate file on disk with a specified file-name. The collate file is then used by the sort intrinsic during the execution of a sort or merge that includes the COLLATE option. The collate file consists of two 256-byte records on disk. The first record of the file is used to collate during input to the distribute phase of the sort. The second record is used during output from the final merge pass.

Execution Deck

Figure 3-3 illustrates a SORT/COLLATE execution card deck consisting of specification cards and control cards.

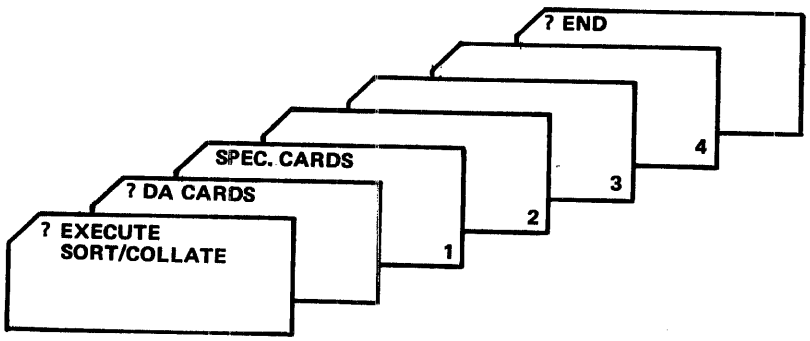


Figure 3-3. SORT/COLLATE Execution Card Deck

The specification cards contain statements that describe the collating sequence desired by the user, and are in free-form format in columns 1 through 71. The statements are checked for syntax errors, and if none are found a collate table file is produced. A detailed description of the specification statements and their functions follow.

Specification Statements

GENERAL

There are five specification statements used to specify a translate or collate file. The five statements are $ IDNT, $ table-name, $ NUMR, $ ALFA, and $ SEQN, and have the following format:

```
                                  $ table-name
                                  $ NUMR hex-1 hex-2 [hex-3 hex-4]
[$ IDNT file-name]  {             $ ALFA  ch-1 ch-2 [ ch-3 ch-4 ]   [{...}] ...
                                  $ SEQN  hex-5 [hex-6]
                                          hex-7 - hex-8 [hex-9 - hex-10]   [{...}] ... }
```

$ IDNT

The $ IDNT statement is optional and provides the user with a means to name the translate or collate file being created by the SORT/COLLATE program. The file-name must be in accordance with the standard MCP conventions for naming files. If the $ IDNT statement is not included, a default family-name of "TRANSLATE" (for translate files) or "COLLATE" (for collate files) is assigned to the file produced, and the file is placed on system disk.

$ table-name

The $ table-name statement is used to specify that a translate table is to be created by SORT/COLLATE for use by the MCP "soft-translate" facility (refer to the TRANSLATE and TRANSLATE.NAME attributes of the FILE statement). The table-name may be specified as one of the following:

table-name	Function
ASC8	8-bit ASCII
ASC7	7-bit ASCII
BCL	6-bit BCL (BCL tapes)
BCL8	8-bit BCL (BCL card files)
B500	6-bit B500 Interpreter disk files

All translation is to or from internal 8-bit EBCDIC. The translate table-name gives the external mode. Thus, "$ BCL8" generates a translate table that allows BCL card files to be read and translated to internal 8-bit EBCDIC, or BCL card files to be punched from internal 8-bit EBCDIC.

The default file-identifier for the translate file (if the $ IDNT statement is not included) is "TRANSLATE".

$ NUMR

The $ NUMR statement is used to specify a number of deviations from the standard collating sequence of @00@ through @FF@. This statement uses pairs of hexadecimal characters to indicate a replacement of standard collating position. In the syntax format shown above, the character represented by hex-1 would collate as hex-2, and hex-4 would replace hex-3 in the collating sequence. As many pairs of characters as necessary can be represented to achieve the desired collating sequence. The $ NUMR statement is particularly useful when dealing with characters that cannot be represented graphically.

Example:

```
$ NUMR 0040 0E3E
```

In the above example, @00@ would collate as @40@ and @0E@ would collate as @3E@.

The $ NUMR statement can be used alone or in conjunction with the $ ALFA statement.

$ ALFA

The $ ALFA statement is similar in function to the $ NUMR statement, with the exception that the representations are in the form of graphic characters. This method is more convenient for those characters that can be represented graphically.

Example:

```
$ ALFA AB ([
```

In the example, A will collate as B and left parenthesis will collate as left bracket.

When using either or both the $ NUMR and $ ALFA options, any characters not specifically mentioned retain their standard position in the collating sequence. For this reason, characters can either explicitly or implicitly be made to collate alike. When that is the case, the only way to restore the transmuted characters is to see that either TAGSORT or TAGSEARCH is specified in the sort specifications. Refer to the subsection titled SORT for additional information.

If a character appears more than once on the left side of a pair of characters in a $ NUMR or $ ALFA statement, the last appearance takes precedence in establishing its position in the collating sequence.

$ SEQN

The $ SEQN statement is a means of specifying a new collating sequence where a string of 2-digit hexadecimal character representations are used to describe each of the 256 character positions of the collate file. Each hexadecimal entry corresponds to a table position, beginning with @00@ and continuing through @FF@. Thus, the first character represented collates as @00@, the second as @01@, and so forth. If any portion of this hexadecimal string is in an ascending or descending contiguous form, the string can be represented as the first and last elements of the contiguous portion, separated by a hyphen. In this manner, the string of 2-digit hexadecimal characters 00 01 02 03 04 05 06 07 08 09 could be represented as 00-09, and the string 09 08 07 06 05 04 03 02 01 could be represented as 09-01.

The $ SEQN statement cannot be used in conjunction with either the $ NUMR or $ ALFA statements.

SYSTEM/MAKEUSER

General

SYSTEM/MAKEUSER is a utility program used to create, access, or modify (SYSTEM)/USERCODE, the file of allowable usercode and password combinations. (SYSTEM)/USERCODE maintains all information needed by the MCP to support the file security mechanism, including usercodes and their associated passwords, default pack-identifiers, charge numbers, maximum allowable run-time priorities, and default security codes. A maximum of 1024 usercode/password combinations is allowed in the (SYSTEM)/USERCODE file.

Operating Instructions

All communication and control of SYSTEM/MAKEUSER is performed through the console keyboard (SPO), except for certain "automatic" features described below. ACCEPT (AX) messages are used to provide commands and input data to the program. Success/failure messages are displayed on the SPO after each command has been processed. For example,

```
EX SYSTEM/MAKEUSER
  SYSTEM/MAKEUSER =1 BOJ. . . .
  SYSTEM/MAKEUSER =1 ACCEPT.
1AX ⟨command⟩ ⟨optional comment⟩
  % SYSTEM/MAKEUSER =1 ⟨success/failure message⟩
```

The ⟨optional comment⟩, if present, must begin with a percent sign ("%").

Several automatic features are included in the SYSTEM/MAKEUSER program, as follows:

a. If a card file labeled SYSTEM/USER.CODES (such as is produced by the PUNCH command) is present at BOJ or after execution of any command, SYSTEM/MAKEUSER automatically performs a CREATE command followed by a list of the newly-created (SYSTEM)/USERCODE file. SYSTEM/MAKEUSER then goes to EOJ. No operator action or intervention is required.

b. If SYSTEM/MAKEUSER is executed with SWITCH = L, a listing of the existing (SYSTEM)/USERCODE file is produced automatically. SYSTEM/MAKEUSER then goes to EOJ. No operator action or intervention is required. For example,

```
EX SYSTEM/MAKEUSER SW=L;
```

Input Record Format

Input records for the CREATE command (either explicit or automatic) are in a free-field format. Each file security option is preceded by a key word that identifies the option. Options may be entered in any order, but all options for any one usercode/password combination must be contained on one record. The usercode (US) and password (PW) options must be present on every record. Other options may be included as necessary; if omitted, the default values apply. The options and their default values are as follows:

Option	Description	Default Value
US=⟨usercode⟩	Usercode	No default
PW=⟨password⟩	Password	No default
PACK=⟨pack-id⟩	Default pack-id	System disk (" ")
CHG=⟨integer⟩	Default charge number	Zero (0)
PRI=⟨integer⟩	Maximum priority	Seven (7)
*PRIV	PRIVILEGED indicator	Not PRIVILEGED
PUBLIC	Default security	PRIVATE

The ⟨usercode⟩ may be up to eight (8) characters in length. A ⟨usercode⟩ of SYSTEM is not allowed. In addition, the ⟨usercode⟩ may not begin with an asterisk ("*"), may not begin or end with parentheses, and may not contain the "less than" sign ("<"). Other special characters are permitted in a ⟨usercode⟩, but may cause unpredictable results if used incorrectly.

The ⟨password⟩ may be up to ten (10) characters in length; it may also be null (" "), indicating that no password is required for the associated ⟨usercode⟩.

Multiple entries with the same ⟨usercode⟩ are allowed; however, each must have a unique ⟨password⟩. In addition, multiple entries with the same ⟨usercode⟩ must all specify the same default ⟨pack-id⟩ and security (PUBLIC or PRIVATE).

Examples:

```
US=USERA PW=X CHG=123456 PRI=4 PACK=USERPACK *PRIV

CHG=1000 PRI=8 US=USERB PW=ME

US=USERA PW=" " PACK=USERPACK

US=SITE PW=" " CHG=999999 PRI=15 *PRIV PUBLIC

US=USERB PW=PW3 PACK=" "
```

Commands

The syntax, semantics, and examples of actual input commands are contained in this section. The commands are presented in alphabetical order.

ADD COMMAND

The ADD command allows entries to be added to the (SYSTEM)/USERCODE file.

The syntax of the ADD command is as follows:

```
{ ADD } option-1  [option-2 . . . ]
{ AD  }
```

The usercode (US) and password (PW) options are both required; any other options described for the input record format are allowed, with the exception of the *PRIV option.

Examples:

```
1AX ADD US=NEWUSER PW=NEWPW CHG=1234 PRI=5 PACK=USER

1AXAD CHG=50 US=SITE PW=" "

1AX AD US=SITE PW=PRIV PRI=15 % SYSTEM DISK IS DEFAULT

1AXADD US=FINANCE PW=VP % ALL OTHER OPTIONS DEFAULT
```

CHANGE COMMAND

The CHANGE command allows modification of attributes in the (SYSTEM)/USERCODE file.

The syntax of the CHANGE command is as follows:

```
{CHANGE}
{CH    }  usercode-specifier TO option-1 [option-2. . . ]
```

The usercode-specifier may be one of the following:

⟨usercode⟩/⟨password⟩

⟨usercode⟩/" "

⟨usercode⟩/=

The first two forms are used to change a specific entry in the (SYSTEM)/USERCODE file (the " " option is for usercodes having a null password). The "=" option is used to change all entries having the specified ⟨usercode⟩.

Only the attributes specified in the CHANGE command are modified. Allowable options in the CHANGE command are as follows:

PW=

PACK=

CHG=

PRI=

The password (PW) option is not allowed if the "/=" form of the usercode-specifier is used.

Examples:

1AX CHANGE FINANCE/VICEPRES TO PW=VP CHG=1234 PRI=10

1AXCH SITE/= TO CHG=55555

1AXCHANGE NEWUSER/" " TO PW=NEWPW PACK=" "

COPY COMMAND

The COPY command controls listing of all SPO input and output messages on the line printer.

The syntax of the COPY command is as follows:

```
COPY {ON }
     {OFF}
```

The ON option is the default case, and causes all SPO messages to be listed on the line printer. The OFF option deactivates listing of SPO messages.

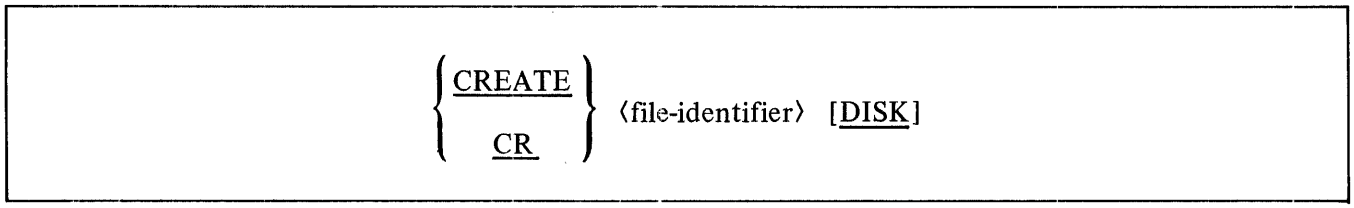
Example:

2AXCOPY OFF

CREATE COMMAND

The CREATE command allows creation of a new (SYSTEM)/USERCODE file.

The syntax of the CREATE command is as follows:



If the ⟨file-identifier⟩ is not included, it is assumed to be CARD. The input file is also assumed to be a card file, unless the key word DISK is specified.

The format of the input records was described previously in the section entitled Input Record Format.

SYSTEM/MAKEUSER enters the new (SYSTEM)/USERCODE file into the "US" slot of the NAME TABLE. If a (SYSTEM)/USERCODE file already exists, it is removed from disk and replaced by the new (SYSTEM)/USERCODE file.

The CREATE command should be performed only when no programs are using an existing (SYSTEM)/ USERCODE file, or unpredictable results may occur.

Examples:

1AXCREATE

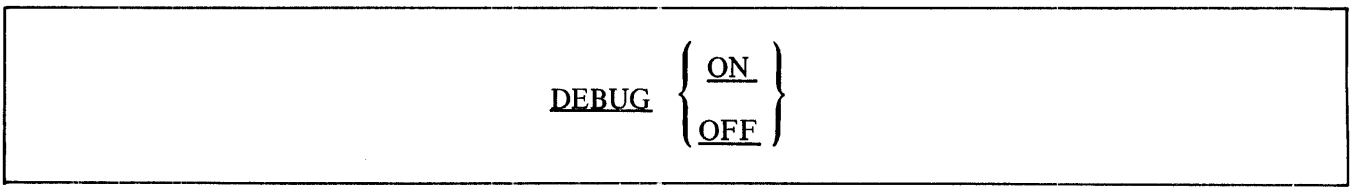
1AX CREATE CARDFILE

3AXCR USERDISK/USER/CODEFILE DISK

DEBUG COMMAND

The DEBUG command controls listing of debugging output on the line printer.

The syntax of the DEBUG command is as follows:

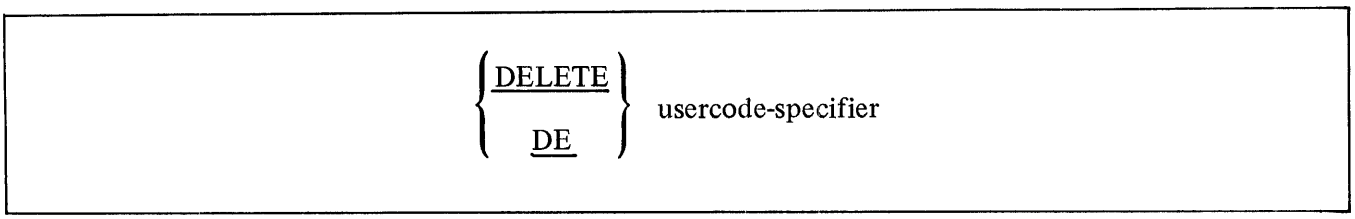


This message is intended for system software development and debugging, and should not be used in normal operation.

DELETE COMMAND

The DELETE command allows removal of existing entries from the (SYSTEM)/USERCODE file.

The syntax of the DELETE command is as follows:



The usercode-specifier is specified in the same manner as is shown for the ADD command. The "=" option deletes all entries with the specified ⟨usercode⟩.

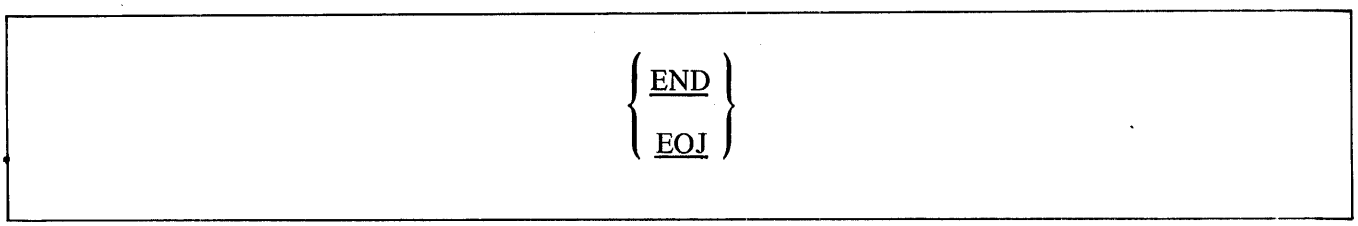
Examples:

```
1AXDELETE USER1/PASS1

1AX DE USER2/=

1AXDE USER3/" "
```

END/EOJ COMMAND

The END/EOJ command causes SYSTEM/MAKEUSER to go to End-of-Job.

The syntax of the END/EOJ command is as follows:



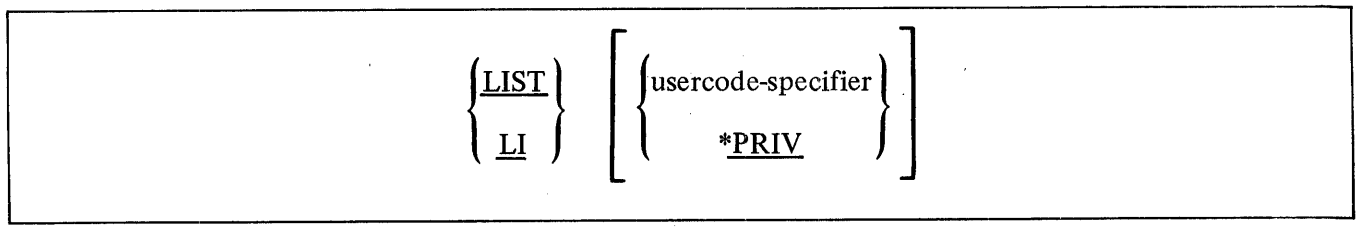
Example:

```
2AX END % THIS IS THE SAME AS "EOJ"
```

LIST COMMAND

The LIST command allows listing of all or part of the existing (SYSTEM)/USERCODE file on the line printer.

The syntax of the LIST command is as follows:



If no options are specified, the entire (SYSTEM)/USERCODE file is listed on the line printer. The usercode-specifier option allows listing of only the designated entries. If *PRIV is specified, only the entries for usercodes marked as PRIVILEGED are listed.

Examples:

1AX LIST

1AXLI USER1/=

1AXLIST SITE/PRIV

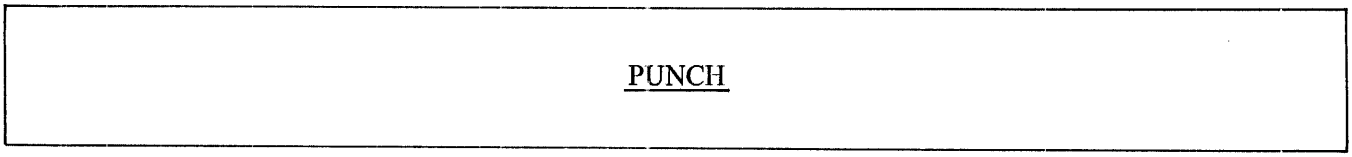
1AX LI USERA/" "

1AXLIST *PRIV

PUNCH COMMAND

The PUNCH command allows the current (SYSTEM)/USERCODE file to be punched into cards, in a form suitable for the CREATE function.

The syntax of the PUNCH command is as follows:



The output card file is labeled SYSTEM/USER.CODES, and is acceptable as input to the CREATE command (either explicit or automatic).

Example:

1AX PUNCH

Error Messages

UNKNOWN COMMAND "⟨command⟩", TRY ONE OF "CHANGE, ADD, DELETE, CREATE, LIST, DEBUG, PUNCH, COPY, END, OR EOJ."

TOO MANY PARAMETERS FOR THIS COMMAND.

PARAMETER REQUIRED AND NOT FOUND FOLLOWING ⟨input parameter⟩.

REQUIRED PARAMETERS WERE OMITTED FOR THIS COMMAND.

NUMBER FIELD FOR "CHARGE NUMBER" OR "PRIORITY" TOO LARGE.

"(SYSTEM)/USERCODE" FILE NOT ON DISK, COMMAND IGNORED.

"(SYSTEM)/USERCODE" FILE LOCKED, COMMAND IGNORED.

MAXIMUM FILE SIZE OF 1024 ENTRIES EXCEEDED, COMMAND TERMINATED.

INVALID USERCODE "⟨usercode⟩" ENTRY DISCARDED.

INVALID PACK SPECIFICATION FOR USERCODE "⟨usercode⟩" ENTRY DISCARDED.

FILE NAME MUST BE SPECIFIED WITH "DISK" OPTION, COMMAND IGNORED.

PACK NAME IS INVALID FOR CARD FILES, COMMAND IGNORED.

INPUT FILE SPECIFIED IS NOT ON DISK, COMMAND IGNORED.

NO USERCODE FILE PRESENT, COMMAND IGNORED.

SPECIFIED ENTRY DOES NOT EXIST, COMMAND IGNORED.

NUMERIC FIELD CONTAINS NON-NUMERIC CHARACTERS.

INVALID DELIMITER "⟨delimiter⟩", COMMAND IGNORED.

CHANGE TO PACKNAME REQUIRES "⟨usercode⟩/=."

"⟨usercode⟩/⟨password⟩" ALREADY EXISTS.

COBOL CROSS REFERENCE UTILITY PROGRAM (COBOL/XREF)

General

The COBOL Cross Reference Utility Program accepts as input a syntactically correct COBOL source program, and depending on the option selected, outputs a cross reference listing, or a program source listing only, or both a cross reference and a program source listing.

Operating Instructions

The source program may reside on disk, magnetic tape, or punched cards. The figure below is an example of a COBOL/XREF execution card deck.

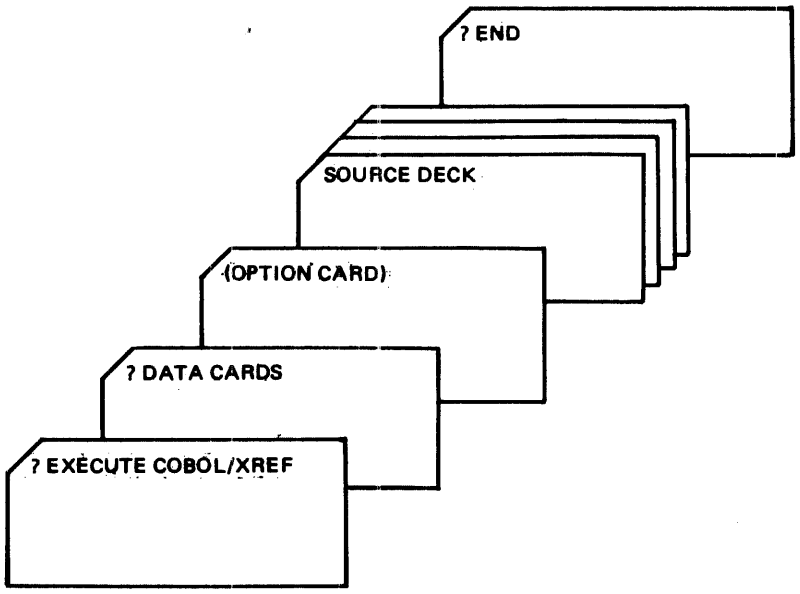
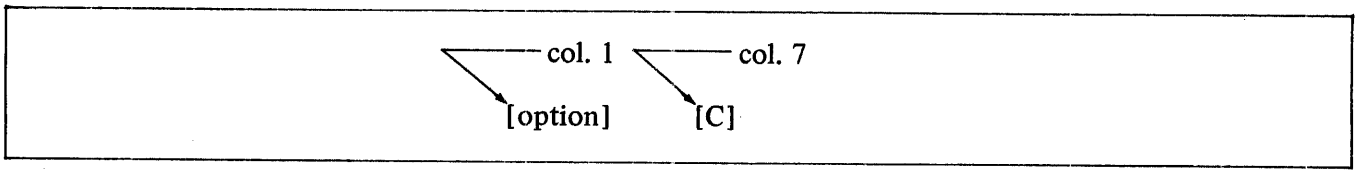


Figure 3-4. COBOL/XREF Execution Deck

Option Cards

The option entry specifies the location of the source medium and the output desired. Below is the format of the COBOL/XREF option card.



The option entry must begin in column 1.

The letter C denotes that the COBOL COPY verb is used within the source program. C, when used, must be in column 7. Requested library files must reside in the disk directory. The library sequence numbers within a source program are indicated by a L to the left of the sequence number.

The options and their descriptions are as follows:

CARD — The input source program is punched cards. Produces a cross reference listing.

CARD is the default input. If the option entry is omitted the input is assumed to be cards.

CDLIST — The input source program is punched cards. Produces both a source program listing and a cross reference.

DISK — The input source program having a file identifier COBOLW/SOURCE resides on disk. Produces a cross reference listing.

DKLIST — The input source program having a file identifier of COBOLW/SOURCE resides on disk. Produces both a program source listing and a cross reference.

LISTCD — The input source program is punched cards. Produces a source program listing only.

LISTDK — The input source program is on disk. Produces a source program listing only.

LISTTP — The input source program having a file identifier of SOLT is on magnetic tape. Produces a source program listing only.

TAPE — The input source program having a file identifier of SOLT is on magnetic tape. Produces a cross reference listing.

TPLIST — The input source program having a file identifier of SOLT is on magnetic tape. Produces both a source program and cross reference listing.

Internal File Names

Internal File Identifiers	External File Identifiers	Description
CARDS	CARDS	Input Source Cards
TAP-S	SOLT	Input Source Tape
DISKS	COBOLW/SOURCE*	Input Source Disk
CBXPRT	XREFER	Printer Output

*Refer to the COBOL Compiler Option NEW for the creation of this file.

Examples:

Card:

```
? EXECUTE COBOL/XREF
? DATA CARDS
CARD C
    (source deck)
? END
```

Disk: ? EXECUTE COBOL/XREF

? DATA CARDS

ÐISK

? END

NOTE

If more than one cross reference file could exist, the FILE statement must be used to make each file identifier unique.

Tape: ? EXECUTE COBOL/XREF

? DATA CARDS

TAPE

? END

Certain options of the COBOL/XREF program are controlled by programmatic switches entered by the system operator. Refer to the SW control instruction attribute and the SW INPUT MESSAGE keyboard input message for details. The options and the programmatic switches and values required to control the options follow:

Switch	Value	Option
SW1	0	Do not cross reference literals.
SW1	1	Cross reference literals.
SW2	0	Vertical spacing for line printer is to be 6 lines per inch, 56 lines per page.
SW2	1	Vertical spacing for line printer is to be 8 lines per inch, 76 lines per page.
SW3	0	No copy files required.
SW3	1	Copy files required.
SW5	0	CARD.
SW5	1	TAPE.
SW5	2	DISK.
SW5	3	TPLIST.
SW5	4	DKLIST.
SW5	5	LISTTP.
SW5	6	LISTDK.
SW6	0	Double space.
SW6	1	Single space.

LOGCONVERT

General

The LOGCONVERT program extracts information from the file LOG/#<n> and creates a new file in COBOL/RPG readable format. The new file is named NEW.LOG/#<n>, where n is an eight-digit number corresponding to the eight-digit number of the LOG/#<n> file.

Execution

Before the LOGCONVERT program can be executed, the SYSTEM/LOG file must be transferred to the LOG/#<n> file with either the LG or TL system Control Instruction. The number (n) of the LOG/#<n> file to be converted is entered with an ACCEPT message.

NEW.LOG/#<n> File Format

The NEW.LOG/#<n> file created by the LOGCONVERT program can contain the following three types of records:

a. Clear/Start record – one for each Clear/Start performed on the system since the last LOG transfer.

b. Program Parameter Block (PPB) record – one for each program scheduled or executed.

c. File Parameter Block (FPB) record – one for each file declared in each program. The FPB record(s) follow the PPB record to which they are associated.

Each record type is 180 bytes.

The NEW.LOG/#<integer> file is blocked 5 records per block.

Additional information concerning the format of the Clear/Start record, Program Parameter Block record, and File Parameter Block record is provided in Tables 4-1, 4-2, and 4-3.

Table 4-1. Clear/Start Record Format

FIELD NAME	FIELD SIZE	DATA TYPE	FORMAT OR VALUE
Record Type	2	N	1 = Clear/Start Record
Filler	10	A	
MCP Family Name	10	A	
MCP File Name	10	A	
Filler	10	A	
Interpreter Family Name	10	A	
Interpreter File Name	10	A	
MCP Version	6	A	MARK.LEVEL.PATCH
Main Memory Size	10	N	Size in bits
Clear/Start Year	4	N	
Clear/Start Julian Day	4	N	
Clear/Start Time	4	N	Counter value
Filler	208	N	

The DATA TYPE column in the tables contains a letter indicating whether the data is alphanumeric or numeric. The letter A represents alphanumeric data or eight-bit characters. The letter N represents signed numeric, 4-bit digits. For signed numeric fields, the FIELD SIZE indicator includes the sign as part of the field length.

Table 4-2. Program Parameter Block Record Format

FIELD NAME	FIELD SIZE	DATA TYPE	FORMAT OR VALUE
Record Type	2	N	2 = PPB Record
Job Number	8	N	
Program Pack-ID	10	A	
Program Family Name	10	A	
Program File Name	10	A	
Interpreter Pack-ID	10	A	
Interpreter Family Name	10	A	
Interpreter File Name	10	A	
Execute Priority	4	N	
Static Memory	10	N	Size in bits
Dynamic Memory	10	N	Size in bits
Total Memory	10	N	Size in bits
Filler	10	N	
Number of Files Declared	4	N	
Charge Number	8	N	
Schedule Priority	4	N	
Virtual Disk	10	N	Size in segments
Execute Type	2	N	1 = Execute 2 = Compile and Go 3 = Compile for Syntax 4 = Compile to Library 5 = Compile and Save 6 = Ex of Compile & Go 7 = Ex of Compile & Save
EOJ Type	2	N	0 = Normal EOJ 1 = DS-ed 2 = Program Error 3 = Aborted (Clear/Start)
Year Compiled	4	N	
Julian Day Compiled	4	N	
Time Compiled	8	N	System counter value
Mix Number	4	N	
Schedule Year	4	N	
Schedule Julian Day	4	N	
Schedule Time	8	N	System counter value
BOJ Year	4	N	
BOJ Julian Day	4	N	
BOJ Time	8	N	System counter value

Table 4-2. Program Parameter Block Record Format (Cont)

FIELD NAME	FIELD SIZE	DATA TYPE	FORMAT OR VALUE
EOJ Year	4	N	
EOJ Julian Day	4	N	
EOJ Time	8	N	System counter value
Processor Time	10	N	System counter value
Object Pack-id	10	A	(Compilations only)
Object Family Name	10	A	(Compilations only)
Object File Name	10	A	(Compilations only)
Code Overlays	8	N	
Data Overlays	8	N	
Filler	2	N	

Table 4-3. File Parameter Block Record Format

FIELD NAME	FIELD SIZE	DATA TYPE	FORMAT OR VALUE
Record Type	2	N	3 = FPB Record
Job Number	8	N	
File Number	4	N	
Internal File Name	10	A	
Pack-id	10	A	
Family Name	10	A	
File Name	10	A	
Hardware Type	4	N	000 = Invalid device 001 = 80-Col. Data Recorder 002 = 80-Col. punch 003 = Invalid device 004 = Invalid device 005 = 96-Col. Data Recorder 006 = Paper tape reader 007 = Paper tape reader 008 = Printer 009 = Invalid device 010 = MICR reader/sorter 011 = Head-per-track disk 012 = Head-per-track disk 013 = Disk cartridge (DCC-2) 014 = Disk cartridge (DCC-1) 015 = Disk pack 016 = Disk pack or cartridge 017 = Disk 018 = Invalid device 019 = 96-Col. reader 020 = Paper tape punch 021 = 80-Col. reader 022 = Invalid device 023 = Invalid device

Table 4-3. File Parameter Block Record Format (Cont)

FIELD NAME	FIELD SIZE	DATA TYPE	FORMAT OR VALUE
			024 = 9-Track mag. tape (NRZ) 025 = 7-Track mag. tape (NRZ) 026 = 9-Track mag. tape (PE) 027 = Mag. tape 028 = 9-Track mag. tape 029 = Invalid device 030 = Cassette 031 = Printer
Number of Buffers	10	N	
Record Size	10	N	Size in bits
Records Per Block	10	N	
Maximum Block Size	10	N	Size in bits (variable length records only)
Save Factor	10	N	
Access Type	2	N	0 = Serial 1 = Random (disk files only) 2 = I/O Sequential (disk files only)
Number of Areas	6	N	(Disk files only)
Blocks Per Area	10	N	(Disk files only)
Last time opened	10	N	System counter value
First time opened	10	N	System counter value
Record Count	10	N	
Block Count	10	N	
Number of Opens and Closes	8	N	
Cumulative Time Open	10	N	System counter value
Number of Errors	10	N	
Filler	126	N	

COBOL Record Format

The following are COBOL declarations that show the format of the Clear/Start, PPB, and FPB records in the NEW.LOG/#⟨integer⟩ file when utilizing the output of the LOGCONVERT program.

Format of Clear/Start Record:

```
01 CLEAR-START-RECORD.
   02 CS-REC-TYPE          PC S9     CMP.
   02 CS-MCP-NAME          PC X(30).
   02 CS-INTERP-NAME       PC X(30).
   02 CS-VERSION           PC X(6).
   02 CS-S-MEM-SIZE        PC S9(9) CMP.
   02 CS-YEAR              PC S9(3) CMP.
   02 CS-JDAY              PC S9(3) CMP.
   02 CS-TIME              PC S9(7) CMP.
```

Format of PPB Record:

```
01 PPB-RECORD.
   02 PPB-REC-TYPE                PC S9     CMP.
   02 PPB-JOB-NUM                 PC S9(7) CMP.
   02 PR-NAME                     PC X(30).
   02 PR-INTERP-NAME              PC X(30).
   02 PR-PRIORITY                 PC S9(3) CMP.
   02 PR-STATIC-CORE              PC S9(9) CMP.
   02 PR-DYNAMIC-CORE             PC S9(9) CMP.
   02 PR-TOTAL-CORE               PC S9(9) CMP.
   02 FILLER                      PC X(5).
   02 PR-FILES                    PC S9(3) CMP.
   02 PR-CHARGE-NUMBER            PC S9(7) CMP.
   02 PR-SCHED-PRIORITY           PC S9(3) CMP.
   02 PR-VIRTUAL-DISK             PC S9(9) CMP.
   02 PR-EXECUTE-TYPE             PC S9     CMP.
   02 PR-EOJ-TYPE                 PC S9     CMP.
   02 PR-YEAR-COMPILED            PC S9(3) CMP.
   02 PR-JDAY-COMPILED            PC S9(3) CMP.
   02 PR-TIME-COMPILED            PC S9(7) CMP.
   02 PR-MY-MIX                   PC S9(3) CMP.
   02 PR-SCHED-YEAR               PC S9(3) CMP.
   02 PR-SCHED-JDAY               PC S9(3) CMP.
   02 PR-SCHED-TIME               PC S9(7) CMP.
   02 PR-BOJ-YEAR                 PC S9(3) CMP.
   02 PR-BOJ-JDAY                 PC S9(3) CMP.
   02 PR-BOJ-TIME                 PC S9(7) CMP.
   02 PR-EOJ-YEAR                 PC S9(3) CMP.
   02 PR-EOJ-JDAY                 PC S9(3) CMP.
   02 PR-EOJ-TIME                 PC S9(7) CMP.
   02 PR-PROCESS-TIME             PC S9(9) CMP.
   02 PR-OBJECT-NAME              PC X(30).
   02 PR-CODE-OVLY                PC S9(7) CMP.
   02 PR-DATA-OVLY                PC S9(7) CMP.
```

Format of FPB Record:

```
01 FPB-RECORD.
   02 FPB-REC-TYPE                PC S9     CMP.
   02 FPB-JOB-NUM                 PC S9(7) CMP.
   02 FILE-NUM                    PC S9(3) CMP.
   02 FP-FILE-NAME                PC X(10).
   02 FP-NAMES                    PC X(30).
   02 FP-HDWR                     PC S9(3) CMP.
   02 FP-BUFFERS                  PC S9(9) CMP.
   02 FP-RECORD-SIZE              PC S9(9) CMP.
   02 FP-RECORDS-PER-BLOCK        PC S9(9) CMP.
   02 FP-MAX-BLOCK-SIZE           PC S9(9) CMP.
   02 FP-SAVE                     PC S9(9) CMP.
   02 FP-ACCESS                   PC S9     CMP.
   02 FP-AREAS                    PC S9(5) CMP.
   02 FP-BLOCKS-AREA              PC S9(9) CMP.
   02 FP-OPEN                     PC S9(9) CMP.
   02 FP-1ST-OPEN                 PC S9(9) CMP.
   02 FP-RECORD-COUNT             PC S9(9) CMP.
   02 FP-BLOCK-COUNT              PC S9(9) CMP.
   02 FP-NO-OPENS-AND-CLOSES      PC S9(7) CMP.
   02 FP-CUMULATIVE               PC S9(9) CMP.
   02 FP-ERRORS                   PC S9(9) CMP.
```

RPG Record Format

The following are RPG declarations that show the format of the Clear/Start, PPB, and FPB records in the NEW.LOG/#⟨integer⟩ file when utilizing the output of the LOGCONVERT program

Format of Clear/Start Record:

```
LOGFILE NS    01    1 D1
                                  P    1    10RCTYP1
                                       2    31 MCP
                                      32    61 INTERP
                                      62    67 VERSON
                                  P   68   720MEMSIZ
                                  P   73   740CSYR
                                  P   75   760CSDA
                                  P   77   800CSTI
```

Format of PPB Record:

```
          NS    02    1 D2
                                  P    1    10RCTYP2
                                  P    2    50JOBNUM
                                       6    35 PRNAME
                                      36    65 INTPNA
                                  P   66   670PRIOR
                                  P   68   720STCORE
                                  P   73   770DYCORE
                                  P   78   820TOCORE
                                      83    87 FILLER
                                  P   88   890FILES
                                  P   90   930CHARGE
                                  P   94   950SCHDPR
                                  P   96  1000VIRDSK
                                  P  101  1010EXTYPE
                                  P  102  1020EOJTYP
                                  P  103  1040YRCOMP
                                  P  105  1060DACOMP
                                  P  107  1100TICOMP
                                  P  111  1120MIX
                                  P  113  1140SCHYR
                                  P  115  1160SCHDA
                                  P  117  1200SCHTI
                                  P  121  1220BOJYR
                                  P  123  1240BOJDA
                                  P  125  1280BOJTI
                                  P  129  130EOJYR
                                  P  131  1320EOJDA
                                  P  133  1360EOJTI
                                  P  137  141PROCTI
                                     142  171 OBJNAM
                                  P  172  1750CODOVY
                                  P  176  1790DATOVY
```

Format of FPB Record:

```
NS    03    1 D3

P     1     10RCTYP3
P     2     50JBNUM
P     6     70FILNUM
      8   17  INTNAM
     18   47  EXTNAM
P    48   490HDWR
P    50   540BUFFS
P    55   590RECSIZ
P    60   640RECBLK
P    65   690MAXBLK
P    70   740SAVE
P    75   750ACCESS
P    76   780AREAS
P    79   830BLKARE
P    84   880OPEN
P    89   930FRSTOP
P    94   980RECCNT
P    99  1030BLKCNT
P   104  1070NOOPCL
P   108  1120CUMUL
P   113  1170ERR
```

DISK/DUMP

General

DISK/DUMP provides the capability of copying data from disk packs to disk packs, disk cartridges to disk packs, and disk cartridges to disk cartridges (except 200 TPI to 100 TPI cartridges). The label is first checked for validity, and then the data is copied on a sector-for-sector basis. For drive zero system disks and user disks, termination of the dump occurs beyond the end of valid data, thereby reducing run time in many cases. All input and output specifications are entered from the console printer. When the number of input errors exceeds 10, the user can specify the number of retries desired on an "as-occurs" basis. If more than 10 output errors occur on a given area, DISK/DUMP is terminated.

Operating Instructions

DISK/DUMP does not operate under MCP control, and must be loaded from the cassette reader of the system console. Execute DISK/DUMP in the following manner:

a. Place the DISK/DUMP cassette in the cassette reader. The BOT light must be lit at this time.

b. Place the console printer on-line.

c. Set the system MODE switch to the TAPE position, press CLEAR, then START. This procedure loads the bootstrap loader from the cassette tape and halts the processor. The L register must be equal to @AAAAAA@ at this time.

d. Set the MODE switch to the RUN position, press START. (Do not press the CLEAR button.) This loads the DISK/DUMP program.

When the cassette tape has been read, DISK/DUMP begins operation with the following message displayed on the console printer:

DISK DUMP MARK <level-number>

ENTER INPUT DRIVE - <DC? or DP?>

After a correct response, the following message is displayed:

ENTER OUTPUT DRIVE - <DC? or DP?>

After a correct response, the data on the disk is copied and compared. Upon completion, without errors, the following message is displayed:

DUMP COMPLETE FROM <input drive mnemonic> TO <output drive mnemonic>

ENTER INPUT DRIVE - <DC? or DP?> OR BLANK TO TERMINATE

If the END OF MESSAGE is pressed at this time, DISK/DUMP terminates and the following message is displayed:

END DISK DUMP

The following error messages can occur during execution.

Error Messages

a. DISK ERROR - RESULT IN "T" (Observe T register to determine type of error. Press START for retry).

b. DISK NOT READY ⟨input or output drive mnemonic⟩ CORRECT AND HIT START

c. PARITY ERR N ⟨input drive mnemonic⟩ ENTER DESIRED NUMBER OF RETRIES OR BLANK TO RESTART

d. TIMEOUT ON ⟨input drive mnemonic⟩ ENTER DESIRED NUMBER OF RETRIES OR BLANK TO RESTART

e. INVALID RESPONSE - TRY AGAIN

f. I/O ERROR ⟨input or output drive mnemonic⟩ ⟨disk address⟩ RESULT = ⟨result⟩

g. ⟨integer⟩ RETRIES ON PARITY

h. ⟨integer⟩ RETRIES ON TIMEOUT

i. TEMP TABLE FILLED ⟨input drive mnemonic⟩ (Clear/Start this pack and try again).

j. COMPARE ERROR ⟨disk address⟩ HIT START TO RETRY

k. INPUT SIZE LARGER THAN OUTPUT

l. NO CART OR PACKS ON SYSTEM

SYSTEM/DISK.DUMP

General

The SYSTEM/DISK.DUMP program runs under the control of the MCP, and has the ability to copy cartridges or packs to cartridges or packs of the same or larger capacity. A disk dump is not allowed if either the input or the output drive is in use by any other programs. This includes the system disk when the MCP is running.

There are two methods used by SYSTEM/DISK.DUMP to accomplish a copy. One is a segment-by-segment copy, which is the same method used by the stand-alone DISK/DUMP program. The second is a file-by-file copy which results in a squashed output disk. The file-by-file option is normally intended for user disks only, but allows a disk with a bad label to be copied. In this way, a single drive system disk with a bad label may be dumped, but the resulting disk will be a user disk.

When the dump is performed segment-for-segment, the detection of an error condition while attempting to read or write causes appropriate retries to be made. Failure of the retries results in termination of the dump. In a file-by-file copy, unsuccessful recovery of an error condition results in the error data being copied to the output pack, and the resulting address being displayed in an error message. If an irrecoverable error occurs in the disk directory or in a file header, the associated file will not be copied.

Operating Instructions

SYSTEM/DISK.DUMP is executed from the console printer or console display. Upon successful execution, the following message is displayed:

SYSTEM/DISK.DUMP = ⟨mix-index⟩ ENTER INPUT DRIVE ⟨DC? OR DP?⟩

If the input drive is either a user disk or the label is bad, the following message is displayed:

SYSTEM/DISK.DUMP = ⟨mix-index⟩ IS FILE BY FILE COPY DESIRED? ⟨YES OR NO⟩

A YES response causes the disk to be copied in a file-by-file manner, without verification. A NO response causes the disk to be copied segment-for-segment with an optional verify pass.

The output drive is selected by responding to the following message:

SYSTEM/DISK.DUMP = ⟨mix-index⟩ ENTER OUTPUT DRIVE ⟨DC? OR DP?⟩

Optional verification, if applicable, is provided by an affirmative response to the following:

SYSTEM/DISK.DUMP = ⟨mix-index⟩ IS VERIFICATION REQUIRED? ⟨YES OR NO⟩

After the copy is complete, the beginning of the verify pass, if applicable, is indicated by the appearance of the following:

SYSTEM/DISK.DUMP = ⟨mix-index⟩ VERIFICATION BEGINS

After completion of the copy, or copy and compare, assuming no errors, the following messages are displayed:

SYSTEM/DISK.DUMP = ⟨mix-index⟩ DUMP COMPLETE ⟨input-drive⟩ ⟨serial-number⟩
TO ⟨output-drive⟩ ⟨serial-number⟩

SYSTEM/DISK.DUMP = ⟨mix-index⟩ ENTER INPUT DRIVE ⟨DC? OR DP?⟩

A blank response terminates the program.

DISK ERROR - RESULT = ⟨result-status⟩
IS RETRY DESIRED? ⟨YES OR NO⟩

A YES response causes the operation to be retried, and a NO causes termination of the program.

ENTER DESIRED NUMBER OF RETRIES

Follows an affirmative response to the previous message. The entry of a blank or zero terminates the dump.

INVALID RESPONSE - TRY AGAIN

The response to a message is other than Y, N, DCn, or DPn, or the requested drive is not available.

TEMP TABLE FILLED - ⟨input-drive⟩

Input drive must be CLEAR/START-ed.

DISK NOT READY

WRITE LOCKOUT

REMOVED SECTORS ON DRIVE ⟨drive⟩

A cartridge with removed sectors cannot be copied except in the file-by-file mode.

PACK LABEL BAD - ⟨drive⟩

If message is for input drive, a file-by-file copy must be requested. If the message is for an output drive, the disk must be initialized.

I/O ERROR - ⟨drive⟩

Parity, timeout, address parity, extended result descriptor error, or address error has occurred and all retries have failed.

RESULT = ⟨result-status⟩

Follows previous message to indicate type of error.

⟨multi-file-id⟩ ⟨file-id⟩ IS A MULTI PACK FILE AND CANNOT BE COPIED

This message appears if a multi-pack file is encountered and the serial numbers of the input and output packs are not the same.

BAD DIRECTORY ⟨output drive⟩

Cannot read the new directory, dump must be restarted with another output disk.

BAD MAIN DIRECTORY ⟨disk-address⟩ ⟨drive⟩

Invalid input directory at indicated address. All files within that segment are not copied.

BAD SUB DIRECTORY ⟨disk-address⟩ ⟨drive⟩

Invalid input sub-directory at indicated address. All files within that segment are not copied.

BAD HEADER ⟨multi-file-id⟩ ⟨file-id⟩ ⟨disk-address⟩ ⟨drive⟩

Invalid header, file is not copied.

BAD SECTOR ⟨multi-file-id⟩ ⟨file-id⟩ ⟨disk-address⟩ ⟨drive⟩

The data contained in this sector of the input disk could not be read without the occurrence of an exception condition. The data has been written to the output disk as is, and must be manually verified.

⟨file-name⟩ IS A MULTI PACK FILE AND CANNOT BE COPIED

If the serial numbers of the input and output packs do not agree, a multi-pack file cannot be copied.

DISK PACK INTERCHANGE PROGRAMS

General

The Burroughs standard structure of interchange media, as defined for disk packs, is compatible across several of the Burroughs product lines. The B 1800/B 1700 systems interface to this disk format through either of the following two utility programs:

DISKPACK/INTERCHANG

STANDALONE/INTERCHANG

DISKPACK/INTERCHANG is an SDL program that runs under control of the MCP, and STANDALONE/INTERCHANG is a stand-alone SDL program which is loaded and executed through the system console. The operation of these two programs is virtually identical; therefore, they are hereinafter referred to collectively as the INTERCHANGE PROGRAM.

The INTERCHANGE PROGRAM has two basic modes of operation:

a. The forward mode, which creates a disk pack in standard interchange format from a B1800/B 1700 user disk pack.

b. The reverse mode, which creates a B 1800/B 1700 user disk pack from a disk written in standard interchange format.

In the forward mode, the INTERCHANGE PROGRAM reads a B1800/B 1700 user disk pack and writes a disk pack in the standard interchange format. The following files are not copied:

a. Multi-pack files.
b. Files with family-names.
c. Files with file-ids longer than six (6) characters.
d. Files with variable-length records.

The output disk pack must have been previously initialized with INTERCHANGE ("I") specified as the pack type.

In the reverse mode, the INTERCHANGE PROGRAM reads a disk pack previously written in the standard interchange format, and creates a B 1800/B 1700 user disk pack. The following files are not copied:

a. Multi-pack files.
b. Files with more than 105 disk areas.

The output disk pack must have been either a newly initialized or "purged" user disk. A user disk with all the files removed is not considered as scratch ("purged"); the PG message must be used to purge a disk pack.

NOTE

The input disk pack is not altered in any way in either the forward or the reverse mode.

Operating Instructions

If the forward mode is desired, mount the input B 1800/B 1700 user disk pack in one disk pack drive, and mount a disk pack initialized as INTERCHANGE in another disk pack drive.

If the reverse mode is desired, mount the previously written interchange disk pack in one disk pack drive, and mount a newly initialized or purged B 1800/B 1700 user disk pack in another disk pack drive.

Execute the INTERCHANGE PROGRAM in the appropriate manner (refer to DISK/DUMP and SYSTEM/DUMP.DUMP for instructions on execution of STANDALONE/INTERCHANGE and DISKPACK/INTERCHANG, respectively). Messages from the INTERCHANGE PROGRAM are displayed on the SPO. Responses are also entered on the SPO (using ACCEPT (AX) messages for DISKPACK/INTERCHANG). The messages displayed and the responses expected are as follows:

WHICH PACK (SOURCE OR DESTINATION) IS B 1700 FORMAT? (ANSWER S OR D)

An "S" response causes the forward mode to be selected, thus creating an interchange disk pack.

A "D" response causes the reverse mode to be selected, thus creating a B 1800/B 1700 user disk pack.

ENTER SOURCE PACK DRIVE (A, B, C, ETC.)

Enter the unit-mnemonic of the input disk drive.

ENTER DESTINATION PACK DRIVE (A, B, C, ETC.)

Enter the unit-mnemonic of the output disk drive.

Neither drive specified may be a system drive, nor may the source and destination unit-mnemonics specify the same drive.

Error Messages

Due to differences between the B 1800/B 1700 user pack format and the standard interchange format, it is not possible to copy certain files. As these incompatibilities are encountered during conversion, they are indicated by the error message listed below. The INTERCHANGE PROGRAM continues to copy the remaining valid files.

FILE ⟨file-name⟩ CONTAINS MULTI-PACK LINKS

Files containing multi-pack areas are not copied.

FILE ⟨file-name⟩ EXCEEDS 105 AREAS

Files with more than 105 disk areas are not copied (reverse mode only).

FILE ⟨file-name⟩ EXCEEDS 6 CHARACTERS

Files with file-ids longer than six (6) characters are not copied (forward mode only).

FILE ⟨file-name⟩ IS MULTI-FILE NAME

Files with family-names are not copied (forward mode only).

The following messages may appear due to disk I/O or format errors:

DISK ERROR AT ⟨sector-address⟩ IN FILE ⟨file-name⟩

An irrecoverable I/O error occurred on the input disk pack. The file containing the error is not copied.

TOO MANY ERRORS AT SECTOR <sector-address>

An irrecoverable I/O error occurred on the output pack. The sector in error is removed from the Master Available Table on the output pack, and processing continues.

DISK ERROR AT <sector-address> ON INPUT DRIVE
DISK ERROR AT <sector-address> ON OUTPUT DRIVE

An irrecoverable I/O error occurred in the directory of the specified pack. The INTERCHANGE PROGRAM terminates immediately.

COULD NOT FIND SOURCE PACK
COULD NOT FIND DESTINATION PACK

Indicates that the specified pack is (1) not on the proper drive, (2) not ready, or (3) in-use.

DESTINATION PACK NOT AN INITIALIZED PACK

The destination pack is not a scratch pack (reverse mode only) or has not been properly initialized.

ERROR IN PACK LABEL, CAN'T CONTINUE

Indicates that one of the following has occurred:

a. The source and destination packs are reversed (the copy is being attempted in the wrong direction).

b. The source pack is not a B 1800/B 1700 user pack or a pack written in the standard interchange format.

CHECK/LOAD.DUMP

General

CHECK/LOAD.DUMP compares the files on a system library tape (created by SYSTEM/LOAD.DUMP) to those files of the same name on disk. A listing of the files on the tape is produced indicating:

a. Relative file number
b. File-identifier
c. File type
d. Record size (bytes)
e. Blocking factor
f. EOF pointer
g. Creation/compilation date

The listing also contains a message indicating the type of comparison error, when the files fail to compare.

In addition, CHECK/LOAD.DUMP provides the following capabilities:

a. Allows specification for comparison of only certain files on the library tape, rather than the entire tape.

b. Ability to interrogate the current status of the program, including file-name, file-number, and cumulative error count.

c. Ability to request detailed analysis of errors on a separate printed listing.

Operating Instructions

To execute CHECK/LOAD.DUMP for default comparison of an entire library tape, enter the following control message:

EX CHECK/LOAD.DUMP FILE T NAME= ⟨library-tape-id⟩ ;

If the FILE clause is not included in the EXECUTE string, the tape can be IL-ed when requested by the program. If the files to be compared are located on a user disk rather than systems disk, include the following FILE clause in the EXECUTE string:

FILE D PACK.ID= ⟨user-pack-id⟩ ;

If the comparison is desired on only certain files rather than on the entire tape, execute CHECK/LOAD.DUMP with switch 0 set to any non-zero value. For example:

EX CHECK/LOAD.DUMP SW0=1

In this case, CHECK/LOAD.DUMP requests the names of the files to be compared. These are entered through ACCEPT messages with one name per message. " ⟨family-name⟩ /=" and "=/ ⟨file-id⟩ " forms are allowed. Also, "=/=" is allowed and will cause the entire tape to be compared, When all desired file-names have been specified, enter a blank ACCEPT to terminate the loop. For example:

```
EX CHECK/LOAD.DUMP SW0=1 FILE T NAME=LIBRARY;
 CHECK/LOAD.DUMP =1  BOJ. ...
 % CHECK/LOAD.DUMP =1 ENTER FILE NAMES
 CHECK/LOAD.DUMP  =1  ACCEPT.
!AXSDL/=
!AXCOBOL
!AX=/INTERP1M
!AX
```

The listing produced only contains information for the files specified in the ACCEPT messages.

Once file comparison has started, CHECK/LOAD.DUMP can be interrogated for its current status. Enter.

```
⟨mix⟩ AXSTATUS   (or)
⟨mix⟩ AXST
```

The response includes the file-name and number of the file currently being compared, as well as the cumulative error count for the entire tape.

It is also possible to request a separate listing which gives a more detailed analysis of the reason for a library tape error. If CHECK/LOAD.DUMP is executed with switch 1 set to a non-zero value (for example, SW1=1), this separate error listing is produced.

For HEADER COMPARE ERRORS, the relevant fields from both the disk header and the tape header are printed. For FILE COMPARE ERRORS, both the disk and tape buffers are dumped in hexadecimal format. Each 4-bit character that does not compare is flagged with an asterisk.

DISKMAP/UTILITY

General

DISKMAP/UTILITY is a program which accesses the MCP disk directory and available tables, and provides a number of optional forms of output. These options are as follows:

CHECK	Sort disk information by address; check for disk integrity errors.
MAP	Causes listing to be printed during disk integrity check. Sets CHECK option by default.
ALPHA	Sort disk information alphabetically by file-name; print in a format similar to a KA listing.
SAVE	Create a file on disk containing the data printed during ALPHA, in a format accessible by UPL/COBOL programs.
DECK	Punch a deck of cards containing the file names of all files on disk, one to a card.

DISKMAP/UTILITY requires approximately 3000 segments of systems disk for work file space and sorting if either the MAP or CHECK options are specified. Disk requirements for the files used by the other options vary, depending upon the number of files contained on the disk being accessed.

Operating Instructions

Two ACCEPT messages are generated at program BOJ. The first of these requests the unit-mnemonic of the disk to be mapped. Entering the unit-mnemonic of any systems disk in a multiple systems disk configuration causes all systems disks to be accessed. The second requests the options for the run. These messages are entered as shown in the following example:

```
DISKMAP/UTILITY =1 BOJ. ...
% DISKMAP/UTILITY =1 ENTER ⟨UNIT-MNEMONIC⟩ OF DISK TO BE MAPPED.
DISKMAP/UTILITY =1 ACCEPT.
1AXDPA
% DISKMAP/UTILITY =1 ENTER OPTIONS.
DISKMAP/UTILITY =1 ACCEPT.
1AXMAP ALPHA DECK
```

If no options are entered, MAP, CHECK, and ALPHA are set by default.

The program switches can also be used to specify the options, and thus avoid the ACCEPT messages. If switch 0 is set to a non-zero value (for example, SW0=1), the first ACCEPT message is not used and systems disk is assumed as the unit-mnemonic to be accessed. Note that to access a user disk, switch 0 must be zero and the ACCEPT message must be used.

Switches 1 thru 4 control the program options; if any of these switches are non-zero, the second ACCEPT message is not used and the options are set as follows:

Switch	Value	Options Set
1	1	CHECK
1	2-15	MAP, CHECK
2	1-15	ALPHA
3	1-15	SAVE
4	1-15	DECK

The DISKMAP/UTILITY program executes in the following three phases:

Phase 1

Access the directory, available tables, and headers on the specified disk; build temporary work files for use in the other phases.

During phase 1, nothing can be allowed to disturb the disk being mapped, or the output may be in error. This means that no library maintenance instructions (REMOVE or CHANGE) can be performed, nor any message that affects the disk directory or available tables. (For example, XC, XD, TL, SL.) Also, no programs may be scheduled or executed while mapping systems disk. If mapping a user disk, it is sufficient that no programs which may be running attempt to open any file on that disk.

Errors detected during phase 1 are listed on the line printer. Each error specifies the disk address read by DISKMAP/UTILITY, the item being read (DIRECTORY, DISK FILE HEADER, and so forth), the name of the field in error (AVL.SELF, DFH.AREA.ADDRESS, and so forth), and the value contained by the field.

Phase 2

Phase 2 is invoked only if MAP or CHECK is specified. Phase 2 consists of sorting the disk information obtained during phase 1 by address, then checking for integrity errors. If MAP is specified, a listing of the disk areas by address is also performed.

The listing produced by the MAP option contains one line for each logical "chunk" of disk space, giving the starting and ending addresses, the size in disk segments, a description of the "chunk" plus the area number and file-id for all files. Any errors detected are identified by a descriptive error message, as follows:

MISSING DISK AREAS	The specified "chunk" of disk is unaccounted for in all MCP directories and available tables. Note that the "chunk" may have been explicitly removed by an "XD".
OVERLAPPING DISK AREAS	The specified "chunk" of disk is described by more than one header, directory, or available table entry.
ADDRESS BEYOND DISK CAPACITY	The specified "chunk" of disk begins at a disk address greater than any shown by the MASTER AVAILABLE TABLE.

Areas of disk described as TEMPORARY appear on the listing when mapping systems disk; these are work areas in-use by DISKMAP/UTILITY and are returned at program EOJ.

Phase 3

Phase 3 is invoked if ALPHA, SAVE, or DECK is specified. It consists of sorting the disk header information by file name, then producing the type of output requested. Any one or all of the three types of output can be produced during a single run.

The listing printed by the ALPHA option may contain asterisks to the right of some of the SAVE FACTORS printed. This signifies an expired file; that is, a file where the LAST ACCESS DATE plus the SAVE FACTOR is less than the current date maintained by the MCP. This is merely an indication that the file may not have been accessed for an extended period of time, or may not have any SAVE FACTOR specified in the DISK FILE HEADER (refer to the SAVE attribute of the FILE statement for syntax to specify the SAVE FACTOR). Such files should be examined as possible candidates for removal from disk; however, the MCP does not automatically remove an expired file.

The file produced by the SAVE option has the following attributes:

internal-file-name:	"USER"
external-file-name:	"USER"
record size:	180 bytes
records per block:	5
areas:	25
blocks per area:	100

The information for each file on the disk occupies from 1 to 4 records in this file, depending upon the number of disk areas in use by the file. The record format (in COBOL notation) is as follows:

```
01  DATA-RECORD.
      02  FILE-IDENT                 PC X(20).
      02  RECORD-COUNT               PC 9      CMP.
      02  HEADER-DISK-ADDRESS        PC 9(12)  CMP.
      02  RECORD-SIZE                PC 9( 4)  CMP.
      02  RECORDS-PER-BLOCK          PC 9( 4)  CMP.
      02  BLOCKS-PER-AREA            PC 9( 6)  CMP.
      02  SEGMENTS-PER-AREA          PC 9( 6)  CMP.
      02  AREAS-DECLARED             PC 999    CMP.
      02  AREAS-IN-USE               PC 999    CMP.
      02  EOF-POINTER                PC 9( 7)  CMP.
      02  FILE-TYPE                  PC 99     CMP.
      02  CREATION-DATE              PC 9( 5)  CMP.
      02  LAST-ACCESS-DATE           PC 9( 5)  CMP.
      02  SAVE-FACTOR                PC 9( 4)  CMP.
01  AREA-ADDRESSES REDEFINES DATA-RECORD.
      02  AREA-ADDRESS OCCURS 30     PC 9(12)  CMP.
```

The RECORD-COUNT field contains a count of how many records are required for the file description. For example, if a file has less than 16 areas in use (AREAS-IN-USE), it only requires 1 record; for 16-45 areas in use, it requires 2 records; and so forth.

Disk addresses are stored in the following format:

```
01  DISK-ADDRESS-FORMAT.
      02  SERIAL-NUMBER-FLAG         PC 9      CMP.
      02  PORT-NUMBER                PC 9      CMP.
      02  CHANNEL-NUMBER             PC 99     CMP.
      02  UNIT-NUMBER                PC 99     CMP.
      02  DISK-ADDRESS               PC 9( 6)  CMP.
```

If the SERIAL-NUMBER-FLAG is 1, PORT-NUMBER, CHANNEL-NUMBER, and UNIT-NUMBER are not used; DISK-ADDRESS contains a disk cartridge/pack serial number (for multi-pack files). If the SERIAL-NUMBER-FLAG is 0, the other fields contain decimal values describing the disk address.

Note that the entire AREA-ADDRESS field is zero for an unallocated area.

The AREA-ADDRESSES are stored using the DISK-ADDRESS-FORMAT 30 to each 180-byte record (except for the first record). The first 15 AREA-ADDRESSES are stored in the second half of the first record, as shown below:

Area Number	Record Number	Subscripts
1 - 15	1	16 - 30
16 - 45	2	1 - 30
46 - 75	3	1 - 30
76 - 105	4	1 - 30

Thus, the address for area number 3 is located in AREA-ADDRESSES(18) of record 1; the address for area number 30 is located in AREA-ADDRESS(15) of record 2; and so forth.

SYCOPY

General

SYCOPY provides a means of duplicating, comparing, or merging system library tapes (that is, tapes created by SYSTEM/LOAD.DUMP). Up to seven output tapes may be produced.

Operating Instructions

Execution and control of SYCOPY is through the console printer. FILE statements can be used to supply names to any of the up to seven output tapes produced. The internal names of the output tape files are TAPE.1 through TAPE.7. The default names, when there are none specified, are OUT.SYS.

Upon execution, SYCOPY displays the following message:

ENTER ACTION: CPY, CMP, CCM, MRG AND ⟨tape-count⟩

A response to the above message must have the following format:

⟨mix-index⟩ AX ⟨option⟩ [⟨integer⟩]

The allowable options and their functions are as follows:

Option	SYCOPY Action
CPY	Prepare ⟨integer⟩ copies of a library tape with no comparison.
CMP	Compare (only) ⟨integer⟩ library tapes.
CCM	Prepare ⟨integer⟩ copies of a library tape and compare all copies against the original. If any copied tape is not identical to the original, that tape is purged and the system operator is notified by a SPO message. SYCOPY continues to compare the remaining tapes.
MRG	Merges the files from two library tapes to one tape with ⟨integer⟩ number of copies. The first input tape opened (primary tape) has an expected ⟨file-identifier⟩ of SYSTEM, and the second input opened (secondary tape) has an expected ⟨file-identifier⟩ of UPDATE. Both tapes can be "IL-ed" if the tape identifiers are other than default. Files which appear on both tapes result in the file from the secondary tape being written on the output tape.

From one to seven tapes can be specified as output; if the ⟨integer⟩ entry is omitted, one output tape is assumed by default.

If SYCOPY is executed with switch 9 set to 1, the input commands are obtained from a card file labeled CARDS. The ACCEPT messages are not generated, and no operator action is required.

Although SYCOPY requests an input tape with a ⟨file-identifier⟩ of SYSTEM, any library tape can be "IL-ed" when the NO FILE message is displayed. All output file-identifiers are made identical to the input (primary tape) file-identifier.

At the completion of processing, SYCOPY displays the following message:

⟨integer-1⟩ TAPES ⟨function⟩ , (⟨integer-2⟩ DELETED)

⟨integer-1⟩ specifies the number of copies upon which the requested function (COPIED, COMPARED, COPIED AND COMPARED, or MERGED) was successfully completed. ⟨integer-2⟩ indicates the number of tapes in error (and purged), if any.

TAPECOPY

General

TAPECOPY provides a means of duplicating, comparing, merging, or concatenating non-library, multi-file or single-file tapes. It allows selective deletion of any of the files processed or production of multiple copies of a copied, merged, or concatenated tape. TAPECOPY accepts input files from tape or disk.

Note that ANSI standards do not allow a blank file-id on a multi-file tape. However, a blank file-id is allowed on a single-file tape.

TAPECOPY does not accept library tapes (those produced by SYSTEM/LOAD.DUMP). To copy, merge, or compare such tapes, SYCOPY should be used.

Operating Instructions

Execution and control of TAPECOPY is through the console keyboard. Upon execution, TAPECOPY displays the following message:

ENTER ⟨ACTION: CPY, CMP, CCM, MRG, OR CAT⟩
⟨DEVICE: T(APE) or D(ISK)⟩ ⟨TAPE COUNT⟩

The response to the above message must have the following format:

⟨mix index⟩ AX ⟨action⟩ [⟨device⟩] [⟨integer⟩]

There are five possible actions:

CPY — Prepare ⟨integer⟩ tape copies of the input, with input files coming from a tape labeled SOURCE or disk files with a ⟨family-name⟩ of SYS. The input tape may be IL-ed if the name is other than SOURCE. If the ⟨family-name⟩ of the disk files is other than SYS, modify TAPECOPY as follows:

MODIFY TAPECOPY FI DISK.FILE NAME ⟨family-name⟩;

CMP — Compare (only) ⟨integer⟩ copies with the tape or disk files. The tapes to be compared are expected to have names of SOURCE and OUT.SYS, but they may be IL-ed. Disk files are expected to have a ⟨family-name⟩ of SYS, but this may be changed by modifying TAPECOPY as shown previously. Files with different record sizes can be compared; in such a case the shorter record is padded on the right with blanks.

CCM — Create ⟨integer⟩ copies of the source tape or disk files, and compare all copies with the original. If any tape is not identical with the original, that tape is purged and the system operator is notified by a SPO message. TAPECOPY continues to compare the remaining tapes.

MRG — Merges the files from a source tape with the files on a second tape or a family of disk files, and creates ⟨integer⟩ copies. The source tape is expected to be labeled SOURCE, and the update tape is expected to be labeled UPDATE. Both tapes may be IL-ed if the tape identifiers are other than the default. If the update files are on disk, they are assumed to have the same ⟨family-name⟩ as the label of the source tape. TAPECOPY may be modified as shown above to change the ⟨family-name⟩, if necessary. The update file takes precedence in the case of duplicate files, and is the file placed on the output tape.

CAT Concatenates the input files onto one tape file, with ⟨integer⟩ copies. The output tape is given the same name as the first input file. Subsequent input tapes are expected to be labeled NEXTFILE, but may be IL-ed if necessary. Only the first file on each tape is used. To terminate the concatenation, enter the following message when a new tape is requested:

⟨mix-index⟩ OF

Input disk files are expected to have a ⟨family-name⟩ of SYS, but this may be changed by modifying TAPECOPY as shown above. The message

ENTER FILE ID OF NEXT FILE

appears on the SPO for each disk file to be concatenated. A blank (null) ACCEPT terminates the requests. If TAPECOPY is executed with switch 2 set to 1, the ⟨file-id⟩ s are read from a card file labeled TCARDS, with one ⟨file-id⟩ per card.

From one (1) to seven (7) tapes may be specified by ⟨integer⟩. If ⟨integer⟩ is omitted, one (1) tape is assumed by default. If ⟨device⟩ is omitted, TAPE is assumed.

The disk files associated with all functions are expected to be on the system disk with a ⟨family-name⟩ of SYS. Both the ⟨pack-id⟩ and the ⟨family-name⟩ may be changed by using the FILE statement. The ⟨internal-file-name⟩ of the disk file is DISK.FILE.

If switch 0 is set to 1, TAPECOPY prints a summary of the actions taken and the files accessed on the line printer.

If switch 1 is set to 1 or 2, files can be deleted. If switch 1 is set to 1, the ⟨file-id⟩ s are read from a card file labeled TCARDS, one ⟨file-id⟩ to a card. A blank card or the end of the card deck terminates specifiation of the ⟨file-id⟩ s to be deleted. If switch 1 is set to 2, the ⟨file-id⟩ s are entered on the SPO.

At the completion of processing, TAPECOPY displays the following messages:

⟨integer⟩ TAPE(S) ⟨action taken⟩

⟨integer⟩ TAPE(S) FAILED TO COMPARE/PURGED

The second message is displayed only if any tapes failed to compare or were purged.

Optional Features

Normally, TAPECOPY is executed and controlled through the console keyboard (SPO). However, if TAPECOPY is executed with switch 3 set to 1, all commands are read from a card file labeled TCARDS. If any card is incorrect, TAPECOPY aborts with an appropriate message.

CODE/ANALYZER

General

The function of CODE/ANALYZER is to evaluate a code file, list information about its files, data and code segments, and calculate an estimate of memory required for the program to run. The code file and any intrinsic files used thereby must be present on disk.

NOTE

The memory estimate calculated by CODE/ANALYZER is a minimum memory requirement, not a "working-set." The program may use more memory if it is available, or less if not all of the files are open at the same time.

Operating Instructions

Upon execution from the console printer, CODE/ANALYZER displays the following message:

CODE/ANALYZER = ⟨mix-index⟩ ENTER PROGRAM NAME

A valid response has the following format:

⟨mix-index⟩ AX ⟨file-name⟩

An error message is displayed if the file is not a code file or is not on disk. Upon completion of the generated listing, a request for an additional program name is displayed. A blank response terminates the program.

If CODE/ANALYZER is executed with switch 0 set to 1, the ACCEPT messages are not generated, and a card file labeled CARDS is used for the file-name specifications. Only one file-name per card is allowed.

Switch 9 controls printer spacing and skipping; setting switch 9 to 1 causes single-spacing and suppression of all skips to heading.

QWIKLOG

General

QWIKLOG is designed to provide a convenient and compact analysis of job information contained in the SYSTEM/LOG file. SYSTEM/LOG is a disk file of system activity maintained by the MCP when the LOG option is set (refer to the SL message). Data is presented by QWIKLOG in two distinct formats: a tabular summary of selected information about each job, and a "chronograph" which graphically indicates schedule and mix activity at any point in time.

Operation

By default, QWIKLOG analyzes the currently active log file, SYSTEM/LOG. Note that the log may be transferred at any time by the operator (refer to the TL message), or automatically by the MCP upon reaching capacity. This action renames the log file LOG/# <integer> and creates a new SYSTEM/LOG file.

QWIKLOG will analyze a transferred log file if executed with the following FILE statement:

EXECUTE QWIKLOG FILE LOGFILE NAM LOG/# <integer>;

As a convenience, if the file LOGFILE is file-equated with the single name of LOG, QWIKLOG requests the log number (<integer>) via an ACCEPT message. All ACCEPT message input to QWIKLOG is free-format.

QWIKLOG normally prints both parts of the analysis described above for all jobs within the log file. If, however, switch 8 is set to one (1), QWIKLOG accepts three options to control its actions. The messages which appear are as follows:

JOB SUMMARY? (Y/N)

Enter Y, YES, N, or NO.

GRAPH? (Y/N)

Enter Y, YES, N, or NO.

JOB NUMBER RANGE: "ALL" OR <FROM> [<TO>]

This allows selection of a specific range of jobs for analysis. ALL specifies analysis of all jobs in the log file (as is done in the default case); otherwise the first integer (<FROM>) is used as the lower bound for the job number range to analyze. Analysis continues for the remainder of the log file, unless a second integer (<TO>) is provided to indicate an upper bound.

If the analysis is done on SYSTEM/LOG, QWIKLOG stops after the job immediately preceding itself, unless further restricted in range as described previously.

Two other switches affect the operation of QWIKLOG. Switch 0 is used to specify the width of the line printer being used; a zero (0) indicates 132 print positions and a one (1) indicates 120 print positions. This may be permanently set by using the following MODIFY control instruction:

MODIFY QWIKLOG SW 0=1

If switch 9 is set to one (1), job summary information is not printed across page perforations. Owing to the particular formatting used by QWIKLOG, this option wastes a large portion of each page.

Description of Output

The following information is provided in the job summary portion of the QWIKLOG output. Irrelevant fields for a particular job are left blank (for example, a job that is RS-ed has no BOJ or EOJ date/time).

a. Job number. An asterisk preceding the job number indicates that the job was part of a job stream (refer to the THEN, AFTER, and AFTER.NUMBER control instruction attributes).

b. Program Name.

c. Compiler name (if the execute type is COMPILE). For a CLEAR/START, this field contains the MCP version.

d. The CHARGE number and MIX number.

e. Execute type. Items that may appear are EXECUTE, COMPILE & GO, COMPILE & SAVE, COMPILE TO LIBRARY, GO PART OF COMPILE & GO, and GO PART OF COMPILE & SAVE.

f. Schedule priority and run-time priority.

g. Interpreter family-name.

h. Schedule date and time.

i. BOJ date and time.

j. EOJ date and time.

k. Elapsed execution time.

l. Processor time consumed.

m. Job termination type. Items that may appear are NORMAL, DS-ED (includes DP-ed), SYNTAX ERRORS, ABORTED (by a CLEAR/START), RS-ED, and RUNNING (implies that the job was still in the mix when QWIKLOG was executed or the log was transferred).

n. Static memory assigned (in bits).

o. Dynamic memory assigned (in bits).

p. Total memory assigned (in bits), defined as STATIC.MEMORY + DYNAMIC.MEMORY + RUN.STRUCTURE.NUCLEUS. Note that this figure does not include structures such as file buffers (for a complete analysis of the memory requirements for a particular program, CODE/ ANALYZER should be used). In the entry for a CLEAR/START, the TOTAL MEMORY field contains the main memory size of the system (in K bytes).

q. The number of files declared within the program.

r. The number of code overlays (program, interpreter, and MICRO.MCP) caused by execution of the program.

s. The number of data overlays performed during execution.

t. VIRTUAL.DISK assigned (in sectors).

The following mnemonics are used within the chronograph:

*	CLEAR/START
S	Program Scheduled
E	Program Execution
C	Compiler Execution
R	Program Running

The R mnemonic indicates that a program was still running at the time of analysis or when the log was transferred. Thus no EOJ time is available to properly graph with the C or E mnemonic.

SYSTEM/ELOGOUT

General

SYSTEM/ELOGOUT is a system utility program designed to list the contents of the SYSTEM/ELOG file, containing I/O error descriptors, maintained by the MCP. For each I/O error, SYSTEM/ELOGOUT lists the following information:

a. Time of error
b. Unit-mnemonic of device on which the error occurred
c. Hardware address of device (port and channel)
d. Hardware device-id (control-id in hexadecimal)
e. Number of retries attempted
f. The actual I/O error descriptor and its memory address
g. The file-id or disk pack/cartridge label, if any
h. Magnetic tape or disk pack/cartridge serial number, if any
i. Magnetic tape reel number
j. Program job number, if any
k. Disk pack Extended Result Descriptor, if any

In addition, CLEAR/START records and Operator Messages (refer to the EM input message) are listed when they occur. The system I/O configuration is listed following the CLEAR/START record any time the configuration is changed.

A summary of all errors by unit-mnemonic is printed at the end of the detailed listing, giving totals for each device and type of I/O error.

Operating Instructions

The ET input message is used to transfer the SYSTEM/ELOG file, creating a file labeled ELOG/#⟨integer⟩ and a new SYSTEM/ELOG file. SYSTEM/ELOGOUT is executed automatically following the transfer, and is file-equated to the transferred ELOG file.

If it is necessary to print a previously-transferred ELOG file, the following control statement may be used:

```
EXECUTE SYSTEM/ELOGOUT FILE ELOG.FILE NAME ELOG/#⟨integer⟩;
```

Two options are available in SYSTEM/ELOGOUT, both of which are specified using the program switches. The first option permits automatic removal of the ELOG file at end-of-job, and is specified by setting switch 0 to 1. For example,

```
MODIFY SYSTEM/ELOGOUT SW0=1
```

The second option allows the system number to be printed on the summary listing, as is specified as follows:

```
MODIFY SYSTEM/ELOGOUT SW=⟨integer⟩
```

If both options are desired, the system number option must be specified first, followed by the automatic removal option. For example:

```
MODIFY SYSTEM/ELOGOUT SW=⟨integer⟩  SW0=1
```

Summary Report

The summary report generated by SYSTEM/ELOGOUT contains the following information:

Column Name	Information Reported
DEVICE	Unit-mnemonic
READ	Read parity errors
WRITE	Write parity errors
MEM	Memory parity errors
ADDR	Address parity errors
TRANSM	Transmission parity errors
READ CHECK	Card read check errors
PUNCH CHECK	Card punch check errors
TIMEOUT	Timeout errors
CRC CORR.	CRC correctable errors
CONSOLE	B 1800 Console Cassette parity error
OTHERS	Any other errors
REMARKS	Further explanation

The following table indicates the maximum number of peripherals which the summary will recognize and the unit-mnemonic associated with each:

Peripheral Device	Maximum	Unit-mnemonic
Data Recorder	2	CDA-CDB
Card Punch	2	CPA-CPB
Card Reader	2	CRA-CRB
Cassette	4	CSA-CSD
Magnetic Tape	16	MTA-MTP
Disk Cartridge	8	DCA-DCH
Disk Pack	16	DPA-DPP
Head-per-track disk	4	DKA-DKD
Line Printer	4	LPA-LPD
Paper Tape Reader	2	PRA-PRB
Paper Tape Punch	2	PPA-PPB
MICR Reader-Sorter	4	SRA-SRD

SYSTEM/BUILDTRAIN

General

SYSTEM/BUILDTRAIN is a system utility program used to create translate tables for the B 1247-4 Train Printer Control (control-id @3E@). The generated translate tables are contained in a file on system disk labeled SYSTEM/TRAINTABLE, which should be created with all character sets required by an installation.

Refer to the LT input message for further information on specifying translate tables for the train printers.

Translate Table Format

Each translator generated by SYSTEM/BUILDTRAIN consists of 256 one-byte elements. Each element contains the hexadecimal value of the link position on the train module for the corresponding graphic. For example, byte 193 (zero-relative) in a translator contains the link position for the graphic "A" (internal EBCDIC representation @C1@).

Thus, the 96-character EBCDIC translator (train-id number 016) contains link position values from @00@ through @60@ in the byte positions corresponding to the internal representations of the graphics on the train module. All other positions in the translator contain a code representing the INVALID.CHARACTER.

The INVALID.CHARACTER code is represented by a hexadecimal character corresponding to the link position of the graphic to be printed plus @80@. For example, the graphic usually printed as the INVALID.CHARACTER is the question mark ("?"). On the 96-character EBCDIC print module, the question mark graphic is located in link position 17 (@11@). Therefore, the INVALID.CHARACTER code for the 96-character EBCDIC translator is @91@ (@11@+@80@).

If a link position value is greater than any existing on the train module, a PRINT CHECK exception condition occurs. Thus, for the 96-character EBCDIC print module link positions @61@-@7F@ and @E1@-@FF@ result in a PRINT CHECK exception, and should not be specified in the translator.

Operating Instructions

SYSTEM/BUILDTRAIN is controlled by the program switches. The default case (switch 1 set to zero) causes generation of a new SYSTEM/TRAINTABLE file, followed by a summary listing of the generated translators. During a "generate" run, if switch 3 is set to one, the input file is also listed on the line printer; if switch 4 is set to one, the input file is punched into cards.

If SYSTEM/BUILDTRAIN is executed with switch 1 set to one, a summary listing of the current SYSTEM/TRAINTABLE file is printed. New translate tables are not generated.

For example, the following control instruction causes a new SYSTEM/TRAINTABLE file to be generated, and the input file to be punched into cards:

```
EXECUTE SYSTEM/BUILDTRAIN SW3=1
```

The following control instruction causes only a summary listing of the current SYSTEM/TRAINTABLE file to be produced:

```
EXECUTE SYSTEM/BUILDTRAIN SW1=1
```

Input Record Format

The input file for a generate run is labeled INPUT/PC5.TABLES, and is expected to be located on system disk. If it is necessary to designate a different file-identifier or hardware device for the input records, the following FILE statement may be used:

```
? EXECUTE SYSTEM/BUILDTRAIN
? FILE INPUT NAME= <file-identifier>  <device> ;
```

For each translate table to be included in SYSTEM/TRAINTABLE, a set of sixteen (16) input records is required. The format of each of these records is as follows:

Columns	Description
1-20	Train name
22-24	Train-id number
26-28	Character set size
30-31	Printer type (00=400/750 LPM, 01=1100/1500 LPM)
33-64	Link positions (16 per record)
66-67	Sequence number (01-16)
70-80	Optional date (format: 01 JAN 1977)

The train name may be any identification desired. The train-id number must be the identification generated by the train module for 1100/1500 LPM printers (all less than 128). For 400/750 LPM printers, the train-id number may be any value desired that is greater than 127 and less than 256.

Each link position consists of two hexadecimal characters that describe the location on the print train module of the graphic representing the internal EBCDIC configuration. Thus, record # 01 gives the link positions for EBCDIC characters @00@ through @0F@, record # 02 gives the link positions for EBCDIC characters @10@ through @1F@, and so forth.

Link positions equal to or greater than @80@ are used to specify unprintable characters. If an internal EBCDIC character translates to a link position equal to or greater than @80@, an INVALID.CHARACTER exception result descriptor is returned from the print operation. The actual graphic printed is the link position specified minus @80@.

It is possible to print a graphic other than the question mark as the INVALID.CHARACTER by specifying the link position of the graphic desired plus @80@. For example, to print the space graphic (which has a link position of @00@) as the INVALID.CHARACTER on the 96-character EBCDIC train module, a link position of @80@ (@00@ + @80@) is substituted for every occurrence of the @91@ link position in the input record set.

It is also possible to print a legitimate character in place of the INVALID.CHARACTER, thus avoiding the exception result descriptor. For example, to print the question mark graphic as the INVALID.CHARACTER (and suppress reporting of the INVALID.CHARACTER result descriptor), substitute @11@ for every occurrence of @91@ in the input record set.

If the resulting link position is greater than the number of characters on the train module, a PRINT CHECK exception result descriptor is returned from the print operation.

Standard Translate Tables

A set of standard printer translate tables is supplied with SYSTEM/BUILDTRAIN in a disk file labeled INPUT/PC5.TABLES. SYSTEM/TRAINTABLE may be generated from these standard tables directly by executing SYSTEM/BUILDTRAIN, or individual tables may be modified to suit installation requirements.

The standard tables supplied are as follows:

1100/1500 LPM Train Printer

Train Name	Train-id Number	Description
EBCDIC18	001	18-character EBCDIC
FORTRAN48	002	48-character FORTRAN
B300.B50048	003	48-character B300/B500
EBCDIC48	004	48-character EBCDIC
EBCDIC72	005	72-character EBCDIC
BCL72	012	72-character BCL
ASCII72	015	72-character ASCII
EBCDIC96	016	96-character EBCDIC
EBCDIC96-UPPER.CASE	016	96-character EBCDIC
EBCDIC96.UPPER.CASEB	016	96-character EBCDIC
EBCDIC96-LOWER.CASE	016	96-character EBCDIC
EBCDIC96.LOWER.CASEB	016	96-character EBCDIC
EBCDIC.A72	018	72-character Alphatized EBCDIC
EBCDIC.N72	019	72-character Numericized EBCDIC
RPG48	020	48-character RPG
OCR.A72	021	72-character OCR-A

The five versions of the 96-character EBCDIC translate table are included as examples of the manner in which specific tables can be generated and tailored to individual installation requirements. EBCDIC96 is the standard 96-character EBCDIC translator, having both upper and lower case graphics. It prints the question mark graphic for the INVALID.CHARACTER. EBCDIC96-UPPER.CASE and EBCDIC96-LOWER.CASE also print the question mark graphic for the INVALID.CHARACTER; however, EBCDIC96-UPPER.CASE prints all alphabetic characters in upper case and EBCDIC96-LOWER.CASE prints all alphabetic characters in lower case. EBCDIC96.UPPER.CASEB and EBCDIC96.LOWER.CASEB function in a similar manner; however, they both print the space graphic (" ") for the INVALID.CHARACTER.

Multiple translate tables with the same train-id number (but unique train names) may be contained in the same SYSTEM/TRAINTABLE file. The most commonly used version should be the first one specified in the input file. It is the table loaded automatically by the MCP when the printer first goes ready following a CLEAR/START, or if the train-id number is specified in the LT message. For example:

```
LT LPA 016
```

A specific version of such multiple translate tables may be designated by using the train name in the LT message. For example:

```
LT LPA EBCDIC96.UPPER.CASEB
```

Such a translate table will remain loaded until the next CLEAR/START or until explicitly changed by another LT message.

NOTE

It is not possible to designate a translate table for the 1100/1500 LPM printers where the train-id number does not match the identification number contained in the train module.

Train Name	Train-id Number	Description
FORTRAN48	130	48-character FORTRAN
B300/B500.48	131	48-character B300/B500
EBCDIC3.48	132	48-character EBCDIC-3
RPG48	140	48-character RPG
EBCDIC96	144	96-character EBCDIC
EBCDIC3.16	254	16-character EBCDIC-3
EBCDIC3.64	255	64-character EBCDIC-3

The 400/750 LPM train printers do not have automatic train identification, and the proper translate table must be explicitly specified with an LT message.

SECTION 4
PROGRAM PRODUCTS

COMPILERS

Compilers generate executable code from a programmer's source statements. Each compiler has various options and operational techniques which affect its output. The following pages discuss each compiler and its individual operating procedures.

The COMPILE card, DATA card, and the Label equate (FILE) cards are standard for all compilers and are not discussed in detail for each compiler concerned. See the Control Instruction section for their particular usage and syntax.

REPORT PROGRAM GENERATOR

General

The Report Program Generator (RPG) enables the user to obtain comprehensive reports from existing files with a minimum time involved in source coding. An object program produced from RPG source coding is in the RPG S-Language format.

Compilation Card Deck

A program written in Burroughs RPG, called a source program, is accepted as input by the RPG compiler. The compiler has two major functions: (1) verify all syntax rules outlined in the RPG Program Manual, and (2) convert the source program language into RPG S-Language which is then ready for execution.

The program generated by the RPG compiler is executed under control of the MCP using the RPG interpreter.

Following is an example of an RPG compilation deck.

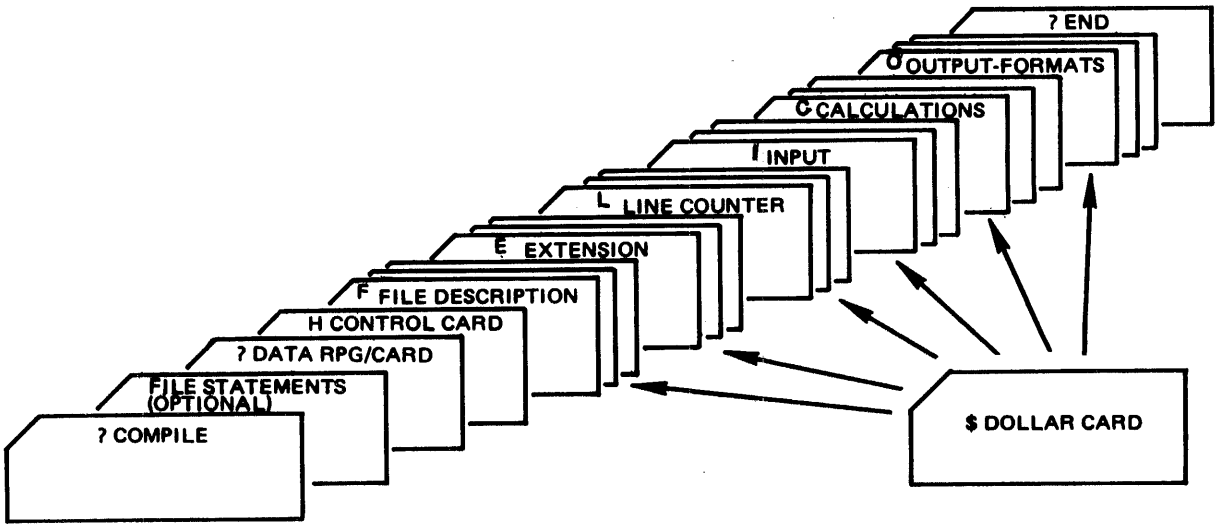


Figure 4-1. RPG Compilation Deck

Dollar Card Specifications

Dollar Card Specifications allow the RPG Compiler to accommodate various extensions to other manufacturers RPG and RPG II languages, which cannot be handled on the other specification forms. Dollar Cards also allow certain compiler-control options to be set or reset during compilation.

Dollar cards may appear anywhere within the source deck, as required. Only one option can be entered on a card and must be in the following format:

Columns	Description
1-5	Page and Line Sequence Number
6	This field may be left blank or contain the form type to align with the associated form that the $ option was inserted in.
7	A $ sign must appear in this field.
8	This field is used to specify that the option entered in the KEY WORD field is set ON or OFF. (Blank = ON, N = OFF).
9-14	KEY WORD: This field is used to name the option that is to be used. The option must be left-justified.
15-24	VALUE: This field is used to specify a value to be associated with the option. All values in alphanumeric form must be left-justified, numeric form must be right-justified.
25-74	COMMENTS: This field is available for comments and documentary remarks.
75-80	Program Name

RPG Extensions

The following options may appear only within the file description specifications, and must immediately precede the specification line describing the file to which they apply.

NOTE

None of the following operations may be "reset".

PACKID Specifies the pack name of a disk file. Similar to $ FAMILY and $ FILEID, default of blank dp-id name and the MCP will assume systems pack. This entry should be included to ensure correct handling of files by the MCP.

FAMILY Specifies the external family name (MFID) associated with the file. The VALUE field contains the name which is one to ten characters, left-justified.

FILEID Specifies the external file identification (FID) associated with the file. The VALUE field contains the name which is one to ten characters, left-justified.

AREAS Specifies the maximum number of areas to be allocated for the file (disk files only). The VALUE field contains an integral value, 1 to 105, right-justified, leading zeros optional. The default value assigned is 25, unless specified otherwise.

RPERA Specifies the maximum number of logical records that will be written in each disk area. The VALUE field contains an integral value, right-justified, leading zeros optional. The default value assigned is the first multiple of the records-per-block that is equal to or greater than 500.

OPEN The use of this option results in the explicit open of a file at Beginning-of-Job (BOJ). A $OPEN card must appear before the File Specification card for each file that is to be explicitly opened at BOJ. The default is as follows:

At BOJ, the primary and all secondary files are opened, as well as all input or update indexed files. Demand and output files are implicitly opened upon the first read or write to them.

CLOSE The use of this option forces a serial input file to remain open until the program reaches End-of-Job (EOJ). A $CLOSE card must appear before the File Specification for each file that is to remain open until EOJ. The default is for each serial input file to be explicitly closed upon encountering End-of-File (EOF).

AAOPEN Is a file option used to set a bit in the MCP file parameter block and allocate all disk space areas at the beginning of the program.

ONEPAK Specifies that this particular file must be contained on one disk.

CYL Allocates file areas starting on an integral cylinder boundary.

DRIVE Allocates a physical drive to that particular file. VALUE field must be 0-15. Option may not be reset and is not related to PACKID.

REFORM Input and update disk files are assumed to have the record and block sizes specified in the disk file header, regardless of those specified in the File Specification card. The use of the REFORM option will cause the record and block sizes declared in the File Specification card to override those in the disk file header (that is, the DEFAULT bit in the File Parameter Block is reset). A $REFORM card must appear before each input or update disk file where it is necessary to override the record and block sizes specified in the disk file header.

REORG Specifies a specialized method of sorting indexed files will be invoked at End-of-Job. The REORG feature only sorts the additions and then merges them, in place, into the master file. This method of sorting should decrease the sort time and the temporary disk area required. The VALUE field contains the external file identifier of the indexed file including disk pack-id. The use of this option will result in the following program executions at EOJ:

Indexed File - RPG/REORD
"B" Indexed File - RPG/REORG

NONEPACK

File may be multipack.

Compiler-Directing Options

LIST Specifies that the compiler produce a single spaced output listing of the source statements with the error or warning messages. This option is set "on" by default. Resetting to "off" will not inhibit the errors or warning messages from printing.

LOGIC Specifies that the compiler produce a single-spaced listing of each source specification line followed immediately by an intermediate code used to generate RPG-S code. The listing is produced after the NAMES listing (if the NAMES option is set), and does not include addresses or bit configurations, but only the opcodes and logical operands of the program.

MAP — Specifies that the compiler produce a single-spaced listing detailing the program's memory utilization. The MAP listing is produced after the LOGIC listing (if the LOGIC option is set).

NAMES — Specifies that the compiler is to produce a single-spaced listing of all assigned indicators file names, and field names. The attributes associated with each file and field are also listed. The NAMES listing is produced immediately after the normal source input listing.

RSIGN — Indicates to the compiler, the location of the sign in numeric data items. When set, all signs are assumed to be right-justified; when reset, all signs are assumed to be left-justified. This option may be set and reset at different points in the File, Extension, Input, and Output-Format Specifications, allowing different fields to have different sign positions. If the option is used, it will override the sign position specified in the Control Card Specifications.

SUPR — Specifies that the Compiler is to suppress all warning messages from the source program listing. (Error messages still print.)

XMAP — Specifies that the compiler print a single-spaced listing of all the code generated, complete with actual bit configurations and addresses. Combined with the listing produced by the LOGIC option, complete information about the generated code of the program is available. The XMAP listing is produced after the MAP listing if the MAP option is set.

STACK — Due to infrequent stack overflow conditions during program execution, the user may now change the stack size of the resultant program. This should only be used when a STACK overflow condition has occurred. The default stack size is 313 bits which will allow 8 entries in the stack. To increase the stack size add 39 bits, for each additional stack entry, to the default size of 313.

BAZBON — This specifies that if an indicator is assigned to a field to test for ZERO or BLANK in the Input or Calculation Specifications and the same field is used in the Output Specifications with a BLANK AFTER designation, that indicator will be turned ON after the field is blanked during the output operations. Should a N (not) be specified in column 8 the indicator will be turned OFF, overriding the original RPG I or RPG II specifications. The defaults are as follows:

RPGI - BAZBON is SET
RPGII - BAZBON is RESET.

ZBINIT — This specifies that all ZERO BLANK indicators are initialized ON at Beginning-of-Job or if a N (not) is specified in column 8 they will be initialized OFF regardless of the specifications for RPG II or RPG I. The defaults are as follows:

RPGI - ZBINIT is SET
RPGII - ZBINIT is RESET

XREF — The XREF option must be placed at the beginning of the RPG source program, prior to the first File Specification and prior to H card if present. This option allows the RPGXRF file to be created during compilation for use as input to the RPG/XREF program. At the completion of the compilation it is necessary to manually execute the RPG/XREF program in order to obtain the cross reference listing.

PARMAP — Produces a single-spaced listing of the compiler-generated paragraph names, source statement numbers, and actual segment displacements of the emitted code. This listing may be used to relate to the LOGIC listing.

Internal File Names

The RPG Compiler's internal file-identifiers and external file-identifiers for use in file statement are as follows:

Internal	External	Description
LINE	RPG/LIST	Source output listing to the line printer.
CARDS	RPG/CARD	Input file from the card reader.
TABCRD	RPG/VECTOR	Input file for TABLES from the card reader.

RPG Internal File Names

COBOL COMPILER

General

The COBOL compiler is designed in accordance with the COBOL standard as specified by the American National Standards Institute (ANSI). The COBOL compiler can function with any system that runs under the control of the MCP.

The COBOL compiler in conjunction with the MCP allows for various types of actions during compilation which are explained in the following paragraphs.

Compilation Card Deck

Control of the COBOL source language input is derived from presenting the compilation card deck to the MCP. See figure 4-2.

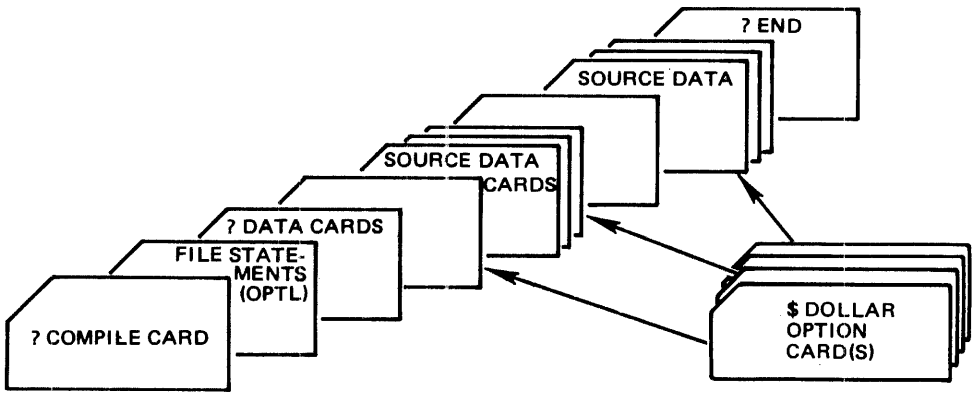


Figure 4-2. COBOL Compilation Deck

Dollar Option Card

The third card, excluding file statement cards, is the COBOL $ Option card. This card is used to notify the compiler which options are desired during a compilation. Without the $ Option Card, $ CARD LIST CHECK SINGLE CONTROL will be assumed.

The $ Option card has the following characteristics:

a. A $ sign must appear in column 7.

b. There must be at least one space separating options on a card.

c. There may be more than one option per card.

d. The options may be in any order.

e. Any number of $ cards may be used and may appear anywhere in the source deck. The option will be set or reset from that point on.

f. Columns 1 - 6 are used for sequence numbers

The format of the $ Option card is as follows:

$ [NO] option-1 ··· [NO] option-n

OPTIONS

The options available for the COBOL compiler are listed below:

CARD — Input is from the source language cards or paper tape. This option is for documentation only.

LIST — Creates a single-spaced output listing of the source language input, with error and/or warning messages, where required.

LISTP — Lists the source images during the first compilation pass, and prints the error messages as they occur.

SINGLE — Causes the output listing to be printed in a single-spaced format.

DOUBLE — Causes the output listing to be printed in a double-spaced format.

CODE — List object code following each line of source code from the point of insertion.

MERGE — Primary input is from a source other than a card reader and may be merged with a patch deck in the card reader. It is assumed to be from a disk file, with a file-id of COBOLW/SOURCE, by default.

If it is desired to change the input file-id or change the input device from disk to tape, a LABEL EQUATION CARD must be used. The NEW option may be used with the MERGE option to create a new output source file plus changes.

NEW — Creates a NEW output source file with changes, if any, entered through the use of the MERGE option, but does not include compiler option cards which must be merged in from the card reader when compiling from disk or tape.

The output file will be created on disk by default with the file-ID of COBOLW/SOURCE.

If it is desired to change the output file-id or change the output device from disk to tape, a LABEL EQUATION CARD must be used.

CHECK — This option will cause the compiler to check for sequence errors and print a warning message for each sequence error. The CHECK option is set on by default at the beginning of each compile, but may be terminated with the NO CHECK option.

SUPPRESS — Suppresses all warning messages except sequence error messages. The sequence error message can be suppressed with the NO CHECK option.

SPEC — If syntax errors occur, this option negates the CONTROL and LIST options, and causes only the syntax errors and associated source code to be printed. Otherwise the CONTROL and LIST options remain in effect.

"Non-numeric literal"
Is inserted in columns 73-80 of all following card images when creating a new source file and/or listing. This option can be turned off or changed by a subsequent control card with the area between the quote marks containing blank characters.

SEQ Starts resequencing, the output listing and the new source file if applicable, from the last sequence number read in and increments the sequence number by ten or by last increment presented in a previous $-option card. When resequencing starts at the beginning of the program source statements the sequence will start with 000010.

SEQ nnnnnn
Starts resequencing the output listing and new source file if applicable from the sequence number specified by nnnnnn and increments the sequence numbers by ten.

SEQ +nnnnnn
Starts resequencing the output listing and new source file if applicable from the last sequence number read in and increments by the number specified by +nnnnnn. When resequencing starts at the beginning of the program source statements, the sequence will start with 000010.

SEQ nnnnnn +nnnnnn
Starts resequencing the output listing and new source file if applicable from the sequence number specified by nnnnnn and increments by the value of +nnnnnn.

NO SEQ Terminates the SEQ option and resumes using the sequence number in the source statement as it is read in.

CONTROL Prints the $-option control cards on the output listing. The LIST option must be on.

NO When the NO option precedes one of the options, with the exception of MERGE and NEW which cannot be terminated, it will terminate the function of that option.

REF
During debugging additional monitoring can be done to see the effect upon variables specified in the MONITOR declaration and referenced in a statement that does not change its value.

ANSI When used, will inhibit the EXTENSION of AT END . . . ELSE, and during compilation will flag them as syntax errors.

STACK [integer]
Is used to increase the program stack by "integer" bits. The default size, when at least one PERFORM statement is used, is 1000 bits.

NOCOP When used will generate COP entries in the code instead of a COP table causing more memory to be utilized but faster program execution.

The NEW option does not have to be included when operating with a tape or disk source input, thus allowing temporary source language alterations without creating a new source output file.

The MERGE option without the NEW option allows a disk or tape input file to be referenced and to have external source images included from the card reader on the output listing and in the object program. A new output file will not be created.

Columns 1 - 6 of the Compiler Option Control card may be left blank when compiling from cards. A sequence number is required when compiling from tape or disk when the insertion of the $ option is requested within the source input.

Source Data Cards

The Source Data cards follow the $ Option control cards. These cards have two functions: (1) to update and create a newer version of a program, and (2) cause temporary changes to the tape or disk source program.

The following two paragraphs outline the Source Data Cards that are available to use with the COBOL Compiler:

a. VOID Patch Card. Punch the beginning sequence number in card columns 1-6 followed by a $ sign in column 7 with the word VOID starting in column 8, and terminate with the optional ending sequence number. This will delete the source statements beginning with the 6-digit sequence number through the ending 6-digit sequence number. For example:

    nnnnnn $VOID [nnnnnn]

    If the ending sequence number is omitted, only the source statement associated with the beginning sequence number will be deleted. For example:

    nnnnnn $VOID

b. CHANGE or Addition Patch Card. Punch the 6-digit sequence number in card columns 1-6 of the card that is to be changed or added, followed by the data to be input in their applicable columns. These cards must be arranged in the sequential order of the source program in order to be MERGED correctly into the program.

The COBOL Compiler has the capability of merging inputs from punched cards or paper tape, either of which may be merged with magnetic tape or disk.

The output listing will indicate any inserts and/or replacements when in the MERGE mode.

The following are examples of a COBOL compile deck.

Example 1:

```
?   COMPILE ALPHA WITH COBOL FOR SYNTAX
?   DATA CARDS
    $ CARD LIST DOUBLE
    . . . source program deck . . .
?   END
```

Example 2:

```
?   COMPILE ALPHA WITH COBOL SAVE
?   DATA CARDS
    $ CARD NO CHECK DOUBLE
    . . .source program deck. . .
?   END
```

Internal File Names

The COBOL compiler's internal file-identifiers and external file-identifiers for use in Label Equation are as follows:

Internal File-name	External File-ID	Description
CARDS	CARDS	Input file from the card reader. If $ MERGE is used, this file will be merged with the input file on disk or tape. The default input is from the card reader.
SOURCE	COBOLW/SOURCE	Input file from disk or tape when the MERGE option is used. The default input is from disk.
NEWSOURCE	COBOLW/SOURCE	Output file to disk or tape for a NEW source file when the NEW option is used. The default output is to disk.
LINE	LINE	Source output listing to the line printer.

COBOL Internal File Names

FORTRAN COMPILER

General

FORTRAN (FORmula TRANslation) was designed for writing programs concerned with scientific and engineering applications in mathematical-type statements. The FORTRAN compiler translates these statements into object code which can be executed by the B 1800/B 1700.

B 1800/B 1700 FORTRAN is designed to be compatible with FORTRAN IV, Level H, and to contain ANSI Standard FORTRAN as a subset.

Compilation Card Deck

Control of the FORTRAN source program is derived by presenting to the MCP the FORTRAN compilation card deck. See figure 4-3.

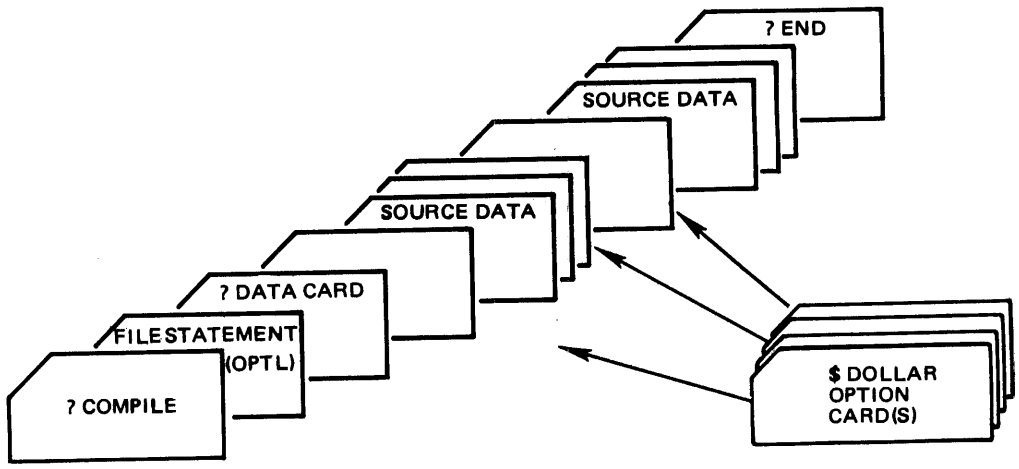


Figure 4-3. FORTRAN Compilation Deck

Dollar Option Card

The third card, excluding Label equation cards, and the standard COMPILE and DATA cards, is the FORTRAN compiler $ Option control card. This card is used to notify the compiler as to which options are required during the compilation. By omitting the $ Option card, the options CARD LIST SINGLE BIND INITIAL are assumed.

The format for the FORTRAN $ Option control card is:

$ [NO] option-1 · · · [NO] option-n

The FORTRAN $ Option control card has the following characteristics:

a. A $ sign may appear in column 1 or 2. When placed in column 2, the $ option card will be included in the new output source file if such a file is generated.

b. There must be at least one space between each item.

c. Options may be in any order.

d. Columns 73-80 are reserved for sequence numbering.

e. Any number of option cards may appear within the source deck.

OPTIONS

The options that are available for the FORTRAN compiler are as follows:

BIND	Causes the intermediate code files to be bound into an executable code file. This is a default option; if BINDING is not desired then NO BIND should be used.
CARD	Input is from source language cards. This is a default option.
CODE	Lists the object code for each source code line from the point of its insertion into the source deck.
CONTROL	NO CONTROL prevents $-option cards from appearing on the compilation listing. The default is CONTROL; $-cards are listed by default.
DATAMAP	Causes virtual memory mapping information to be included on the compilation listing.
DOUBLE	Causes output source listing to be double spaced.
DYNAMIC integer	This specifies the size in words to be assigned for an object program's dynamic memory. If the number of words specified is greater than the total size of all data pages, the total data page size is used instead. The compiler by default will assign the dynamic memory in one of two ways: (1) if the data pages are less than 10, it assigns a size equal to the sum of the data pages, or (2) if the data pages exceed 10, then the size of the 10 largest data pages are used.
ERRORTRACE	Provides a FORTRAN level trace of subprogram and statement usage prior to the detection of a run-time error. The ERRORTRACE option must be placed before the first executable statement of the main program or any subprogram. Once set it may reset at any point by the NO ERRORTRACE option.
INITIAL	Provides a default initialization for variables which are not explicitly initialized. If a variable is never used, the compile-time warning "⟨variable⟩ NOT INITIALIZED" will occur, but program execution will not be aborted. If an array element is assigned a value in a DATA statement, the rest of the array will be "blanked" (if strings were assigned) or "zeroed" (if numbers were assigned). A data item is considered uninitialized if it does not occur in one of the following cases: a. As the left side of a replacement statement. b. As the object of an ASSIGN statement.

c. As an element in an input list.

d. As a parameter in a CALL statement.

e. As a dummy parameter in a subprogram declaration.

f. As an element in a COMMON statement.

g. As an element in an EQUIVALENCE statement.

h. As an element in a DATA statement.

If an array element occurs in any of the above cases, the entire array is considered to be initialized.

INITIAL is the default option; if initialization is not desired, NO INITIAL must be specified.

{INTERPRETERPACK / INTERPPACK} ⟨pack-id⟩ — Modifies the ⟨pack-id⟩ of the FORTRAN interpreter for the object program. Default is system disk.

INTRINPACK ⟨pack-id⟩ — Modifies the ⟨pack-id⟩ of the FORTRAN intrinsic file (FOR.INTRIN). Default is system disk.

LIST — Creates a single spaced output listing of the source statements with error and/or warning messages. This is a default option.

MAP — Prints symbolic reference information about variables, subprograms, and labeled statements. If MAP is set, uninitialized data items are noted in the symbol table dump at the end of each program unit. In all other cases, warning messages are printed for each uninitialized data item at the end of the program unit.

The following information is generated by MAP:

a. A class of UNSPEC (unspecified) occurs when a variable is declared but not referenced.

b. Addresses of parameters are Return Control Word (RCW) relative, and are printed as "RCW+nn" or "RCW-nn", where nn is a word displacement.

c. Variables in COMMON are called GLOBAL in the notes portion of the listing.

d. System routines do not have their number of arguments printed, and "." routines are all given a class of "SUBRTN" regardless of their actual class. The correct number of arguments is printed on the BINDER listing.

e. Dummy parameters in statement function declarations do not appear in the listing.

f. In a function subprogram unit, the function name appears as a scalar.

g. Labels with a type of CONTROL are those to which transfers of control might be made, as opposed to FORMAT and DO-END (a specialized control label).

MERGE

The MERGE option allows source input from disk or tape (disk by default, file-identifier SOURCE) to be merged with source statements from a card reader. The NEW option must be used with the MERGE option to create a new output source file. When the NEW option is not used, both the output listing and the object code file will reflect the merged statements but a new output source file will not be created.

NEW

Creates a new source output file having a file-identifier of NEWSOURCE. The new output file will include any changes made by the use of the MERGE option and any compiler option statements that have the dollar sign in column two.

NEWINTRINSIC

This compile-time option allows testing of potential intrinsics without placing them in the FORTRAN intrinsic file (FOR.INTRIN). If this option is specified, the compiler first attempts to locate the intrinsic on disk by name before searching the FOR.INTRIN file. If a pack-id was specified on the COMPILE control card, the intrinsic must be located on the specified user pack rather than system disk.

NO

When used in conjunction with the following options, it will negate or put them in a reset condition. There must be a space between NO and the option.

BIND
CODE
DOUBLE
LIST
SEQERR
SAVEICM
ERRORTRACE
PROFILES
TRACEC
TRACEF

PAGE

Causes the output listing to eject at that point and start a new page.

PROFILES

Is an optimization aid that indicates to the user those areas of a program that can be optimized to improve program performance. At run time the following data will be output by using the PROFILES option:

a. Frequency of subprogram usage.

b. Time spent in each subprogram.

c. Use of individual statements within a subprogram.

d. Use of each statement during program execution.

The PROFILES option must be placed before the first executable statement of the main or subprogram. To reset the option use the $ NO PROFILES at any point within the program.

SAVEICM

Causes the intermediate code files for each syntax-error-free program part to be made a permanent disk file at the end of the compilation.

SEQ

Causes resequencing of the output listing and the new source file, if applicable, starting with the default number 00001000 and incrementing sequence numbers by 1000.

SEQ nnnnnnnn [nnnnnnnn]

Causes resequencing of the output listing and the new source file if applicable. SEQ is followed by either an eight digit number which is the starting sequence number, or two eight digit numbers with the first number being the starting sequence number and the second the resequencing increment value. The default resequence increment is 1000.

SEQERR — Causes a warning message to be printed for statements out of sequence.

SINGLE — Causes the output listing to be printed in single spaced format. This is a default option.

STACKSIZE integer — Specifies the size in words to be allocated for the object program Evaluation stack. Default size is 100; maximum size is 4096.

TRACEC — Causes tracing of compiler scanning and symbol table information.

TRACEF — Causes a FORTRAN level trace to be printed for each FORTRAN statement executed in the program. This option may be inserted anywhere within the program. Once set, it remains set until reset by using NO TRACEF. Each TRACEF line contains the name of the current subprogram, the compiler-generated line number of the current statement in the subprogram, and an identification of the statement.

VOID — Causes the source input image corresponding to the sequence number of the VOID card to be deleted from the input disk file.

VOIDnnnnnnnn — Causes a series of source images to be deleted starting from the sequence number in the sequence number field (73-80) through and including the sequence number of the VOID option.

XREF — Causes the compiler to produce a cross-reference listing after the source program listing and before the CODE AND DATA MAPPING table. This listing consists of two parts: the first is a list of label numbers followed by a subprogram name (in parentheses) and the sequence numbers where the label appears in the subprogram. The second part of the XREF listing contains a list of variable names followed by a subprogram name (in parentheses) and the sequence numbers where the variable appears in the subprogram. Each sequence number in both parts is immediately followed by one of the following keys:

Key	Description
#	Specification statement or explicit label.
space	Identifier or label reference.
=	Left side of replacement or assignment statement.
*	Variable passed by address.

Internal File Names

The FORTRAN Compiler's internal file-identifiers and external file-identifiers for use in Label Equation are as follows:

Internal File-name (file-number)	External File-ID (Label)	Description
CARDS	CARDS	Input file from the card reader.
LINE	LINE	Source output listing to the line printer.
SOURCE	NEWSOURCE	When $ NEW is used the output file will go to disk or tape. The default output is to disk (80 character records, blocked 2).
SOURCE	SOURCE	Input file is from disk or tape when $ MERGE is used. The default input is from disk and assumed to be 80 character records, blocked 2.

FORTRAN Internal File Names

BASIC COMPILER

General

BASIC is a problem-oriented language designed for a wide range of applications and may be easily applied to business, commercial, engineering and scientific processing tasks. The BASIC language is designed for use by individuals who have little previous knowledge of computers, as well as individuals with considerable programming experience. A distinct advantage of BASIC is that its rules of form and grammar are quite easily learned.

B 1800/B 1700 BASIC includes the capabilities of the original Dartmouth College BASIC plus extensions provided for compatibility with the General Electric MARK II® BASIC language.

The BASIC compiler, in conjunction with the Master Control Program, enables source programs to be compiled through the use of a card reader or a card device. Compilation of the BASIC source language input is achieved by presenting the compilation card deck to the MCP. Control cards included in the compilation deck are of two general types: (1) MCP control cards, and (2) compiler $ Option control cards. The structure of the BASIC compilation deck is discussed in the text that follows:

Compilation Card Deck

The entities comprising the structure of the BASIC compilation deck and the order of their occurrence are shown in figure 4-4 below.

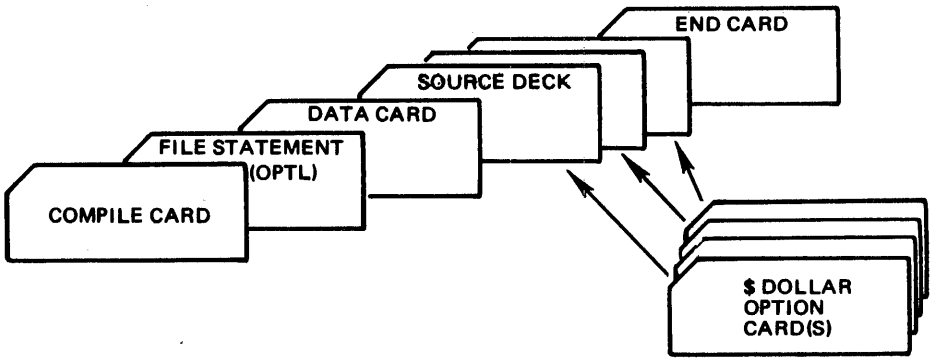


Figure 4-4. BASIC Compilation Deck

Dollar Option Card

The third card, excluding the optional Label Equation cards and the standard COMPILE and DATA cards, is the BASIC $ Option card. This card is used to notify the compiler which options are desired during a compilation. By omitting the $ Option card, the options "CARD LIST SINGLE" are assumed.

The $ Option cards for the BASIC compiler have the following characteristics:

a. All option cards are in a free-form format.

b. A line-number,which is required to be sequential within the program, cannot be greater than five digits and must precede the $ sign.

c. The $ sign may appear anytime after the line-number and before the first option.

d. All options listed on the card may appear in any order.

e. There must be at least one space between each option.

f. $ cards may be used anywhere within the source deck to either set or reset an option.

The format of the $ Option card is:

line-number $ [NO] option-1 · · · [NO] option-n

OPTIONS

The following options are available for the BASIC compiler.

CARD	Symbolic input is from source language cards. At the present time, this option is for documentation purposes only.
LIST	Creates a compilation output listing of the source language input, with error and/or warning messages, where required. LIST is the default option for batch BASIC compilations. NO LIST is the default option for remote BASIC. If a listing is required in remote BASIC, the LIST option will cause the source input to be listed, with error messages, on the CANDE terminal.
SINGLE	Causes the compilation output listing to be printed in a single-spaced format. SINGLE is a default option.
DOUBLE	Causes the compilation output listing to be printed in a double-spaced format.
CODE	Lists the object code generated for a source statement from the point of insertion into the source deck.
NO	Each of the above options may be preceded with NO. This enables the options to be set for selected program parts and then reset as desired. When an option is preceded by NO, there must be at least one space between the word NO and the option to be terminated.

STRINGSPACE data-pages

Allows specification of the number of data pages available for string concatenation, input, and other operations which generate new strings. Each data page has a capacity of 512 words, and the compiler sets the STRINGSPACE to eight data pages by default. The maximum number of data pages that may be specified is 128.

STACK stack-size

Allows specification of the number of words available for the NUMERIC and STRING STACKS. The compiler sets the STACK size to 100 words by default.

Source Input Cards

The source program cards have the following characteristics:

a. Each card is taken as a different line and can contain only one statement. If the 96-column cards are used, the source statement must be contained in the first 80 columns.

b. There can be no continuation cards.

c. Each card between the ? DATA card and the ? END card must contain a line-number.

d. A line-number starts in column 1 and can be a length of 5 digits.

e. The first non-numeric character will terminate the line-number when less than 5 digits.

f. The line-number is used both as a statement label and sequence number.

g. Each statement is sequence checked by the BASIC compiler as it is read in.

h. Spaces or blanks have no significance within a source statement except for information contained in string constants. Spaces can be used to make a program more readable.

Intrinsic Files

The BASIC intrinsic files must be present on disk when a compiled BASIC program is executed; however, they are not needed when compiling the BASIC program. The intrinsic files contain input/output routines and intrinsic functions provided by the BASIC language. If the intrinsic files reside on a user pack, the INTRINSIC.DIRECTORY control instruction must be used to identify the user pack, otherwise, the intrinsics are assumed to reside on the system pack.

Example:

```
? EXECUTE program-name
? INTRINSIC.DIRECTORY dp-identifier
? END
```

Sample Compilation Deck

In the following example, a BASIC program is to be compiled to LIBRARY and the object program, EXAMPLE/PROGRAM, is to be entered in the disk directory of a removable disk cartridge labeled BAS. In addition, the BASIC compiler resides on the removable disk, BAS. A $ card is enclosed to cause the compilation output listing to be printed in a double-spaced format. The options CARD and LIST being default options are not required, but are included on the $ card for documentation purposes only.

```
?   COMPILE BAS/EXAMPLE/PROGRAM  BAS/BASIC/LIBRARY
?   DATA CARDS
10  $ CARD LIST DOUBLE
20  INPUT X, Y, Z
30  PRINT "X="; X, "Y="; Y, "Z="; Z
40  END
?   END
```

In the next example the compiled program EXAMPLE/PROGRAM is ready for execution. The compiled program as well as the BASIC intrinsic files and the BASIC interpreter reside on the removable disk pack labeled BAS. The card file labeled INPUT is required during execution of this program.

```
?   EXECUTE BAS/EXAMPLE/PROGRAM
?   INTRINSIC.DIRECTORY = BAS
?   INTERPRETER = BAS/BASIC/INTERP2
?   END
?   DATA INPUT
    12, 32, 56
?   END
```

Internal File Names

The BASIC Compiler's internal file-identifiers and external file-identifiers for use in Label Equation are as follows:

Internal File-name	External File-ID	Description
CARDS	CARDS	Input file from the card reader.
LINE	LINES	Source output listing to the line printer.

BASIC Internal File Names

UPL COMPILER

General

The User Programming Language (UPL) is a problem oriented language developed for writing B 1800/B 1700 system software. The UPL Compiler is a single pass compilation that transforms the programmer's source statements into object code. Figure 4-5 illustrates the generation of an UPL object program.

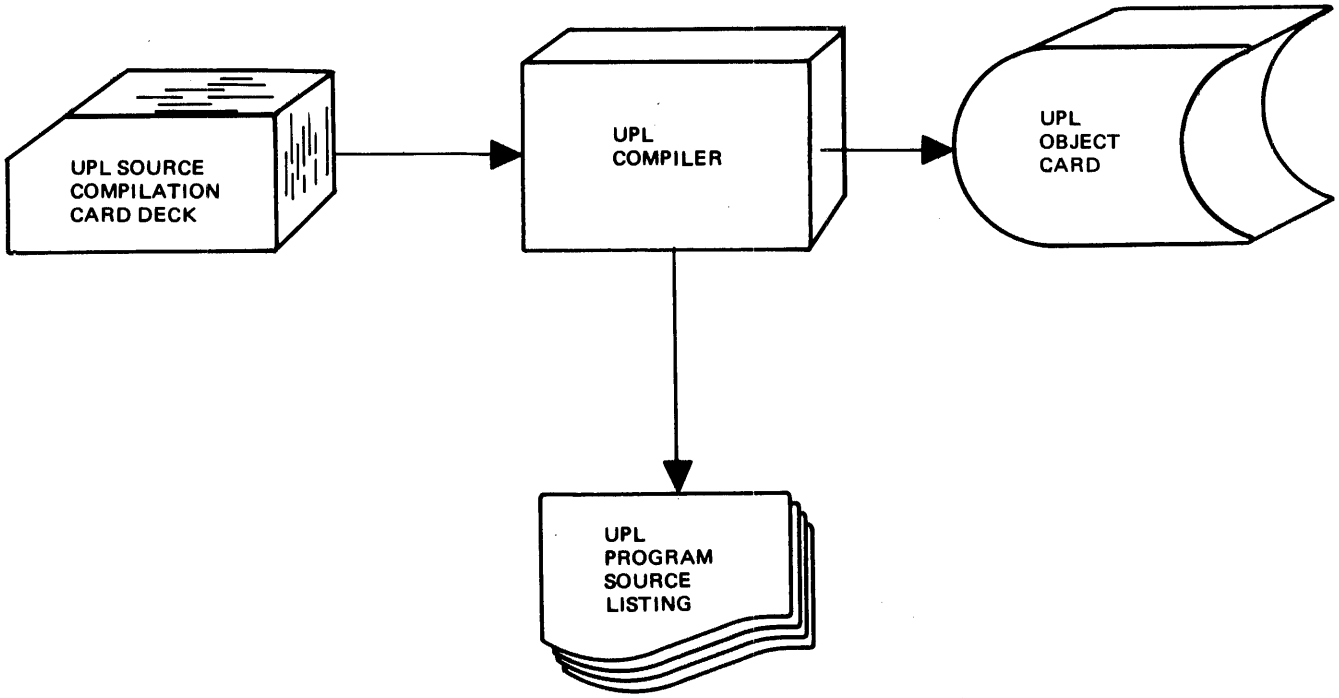


Figure 4-5. UPL Compilation Process

Compilation Card Deck

Figure 4-6 contains an example of a UPL Compilation deck.

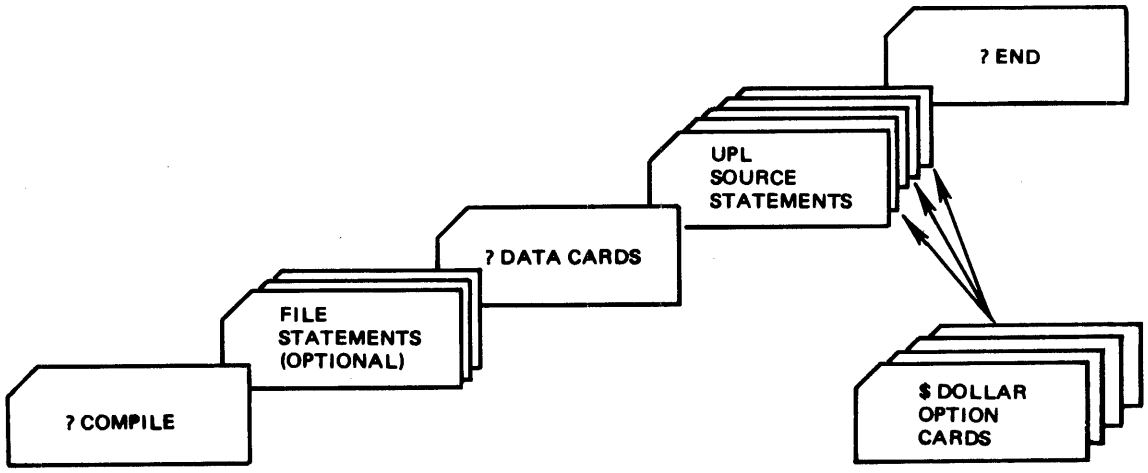


Figure 4-6. UPL Compilation Card Deck

Compiler Options

The UPL Compiler has certain options available through a Dollar card ($) that gives the operator the ability to override some of the standard compiler functions, alter stack sizes, suppress error and/or warning messages, and merge and create a new source file from an existing file. Dollar options are either in a set or reset condition. The UPL Compiler is preset with the following options: AMPERSAND, CHECK, LIST, and SINGLE.

The UPL Compiler option card has the following format:

```
$ [NO] option-1 · · · [NO] option-n
```

The UPL Compiler Dollar option card has the following characteristics:

a. Column one must contain a $ sign.

b. There must be at least one space between options on the same card.

c. Options may be set or reset in any order.

d. Columns 73-80 are reserved for sequential numbering.

e. There is no limit as to the number of options being set or reset during the compilation.

f. The option NO when appearing before any other option resets or negates that option.

The UPL Compiler options and their description are as follows:

ADVISORY	Prints advisory messages on the printer listing. Default is on.
AMPERSAND	Prints those ampersand cards that are examined. Default is on.
CHECK	Checks the source input file for sequence errors. Default is on.
CODE	Print the SDL object code generated for each source statement.
CONTROL	Prints all compiler option cards from that point. If the option control word CONTROL is required to be printed, the $ CONTROL (space) CONTROL format must be used.
CREATE.MASTER	This option must be the first card in the compilation deck and causes the compiler to perform the following functions: a. Dump information to the master information files. b. Create a new source file. c. Create a new code file.
CSSIZE integer	Assigns the number of entries in the Control stack represented by integer and overrides the compiler estimate.
DEBUG	Compiler debug only.
DETAIL	Causes the compiler to list the expansion of all define invocations.

DOUBLE	Double space listing.
DYNAMICSIZE integer	Assigns the value of the integer as the estimated memory size allocated for paged arrays. Integer is expressed in bits.
ERROR.FILE	A separate error file is produced containing only error and warning messages and the source images to which they apply.
ESSIZE integer	Assigns the number of entries in the EVALUATION stack by integer and overrides the compiler estimate.
EXPAND.DEFINES	Causes define expansions to be cross-referenced (used in conjunction with XREF or XREF.ONLY).
FORMAL.CHECK	The checking of the actual parameters passed to each procedure during execution against the TYPE and LENGTH specifications of their corresponding formal declarations. Also, the values returned from function procedures will be checked against the TYPE and LENGTH in the procedure head statement. Lack of correspondence is a run time error.
INTERPRETER file-identifier	Causes the program when executed to use the assigned Interpreter rather than the compiler default interpreter.
INTRINSIC file-identifier (family-name only)	Causes the program when executed to use those intrinsics with the assigned family-name rather than the compiler assigned family-name.
LIBRARY.PACK	Assumes that all library files are on the pack specified.
LIST	Prints the source input that was compiled. Default is on. The NO option when invoked with LIST will reset the LISTALL option also.
LISTALL	Prints all source input regardless if conditionally excluded. The LISTALL option sets the LIST option, but NO LISTALL does not reset LIST.
MERGE	Indicates to the compiler that the source file is on tape or disk and there are cards to be merged during the current compilation.
MONITOR	Refer to Appendix IX, SDL MONITORING FACILITY, in the B 1800/ B 1700 System Software Development Language (SDL) Reference Manual, Form 1081346.
NEW	Creates a new primary source file.
NO	The presence of the NO option immediately before any other option causes that option to be reset from that point on during the compilation.
NO.DUPLICATES	The newly-declared identifier is not checked to determine if it is unique. The programmer must guarantee that there are no duplicates before using this option. The NO.DUPLICATES option reduces compile time for large programs only.
NO.SOURCE	Program source images are not saved, thereby shortening the compiler work file. No source listing is possible when this option is specified. This option is to be used with long programs only.
NSSIZE integer	Assigns the number of entries expressed by integer to the Name stack thereby overriding the compiler's estimated size.

PAGE	Ejects page.
PASS.END	The total elapsed time and the number of errors are printed at the end of each pass.
PPSIZE integer	Assigns the number of entries expressed by integer to the Program Pointer stack thereby overriding the compiler's estimated size.
RECOMPILE	Refer to Appendix VIII, THE SDL RECOMPILATION FACILITY, in the B 1700 System Software Development Language (SDL) Reference Manual, Form 1081346.
RECOMPILE.TIMES	The start and stop times of each of the phases of the "bind" pass of a CREATE.MASTER or RECOMPILE are printed on the listing.
SEQ beginning-sequence-number increment	Causes the output file to be resequenced beginning with the number used with SEQ.
SINGLE	Single space listing.
SIZE	Outputs at the end of the listing, the code segment names and their sizes.
SUPPRESS	Causes all warning messages to be suppressed. To suppress sequence error messages invoke the NO CHECK option.
VOID sequence number	Causes all records in the primary source file (as in the case of the MERGE) to be removed from the sequence number of the VOID card itself through the sequence number entered with the VOID option. The VOID option has the following restrictions: a. Must be the only compiler option on the card. b. Cannot be preceded by the NO option. c. Must contain a sequence number in columns 73-80.
VSSIZE integer	Assigns the number of entries expressed by integer to the size of the VALUE stack thereby overriding the compiler estimated size.
XMAP	Causes an extended SDL object code MAP file to be created showing the relative displacement of object code per source card sequence number, per Code Segment.
XREF XREF.ONLY	The XREF options may be used in one of the two following modes: a. A $XREF card at the beginning of the source deck will cause the compiler to build and XREF file, then ZIP SDL/XREF to sort and print the file at the end of the pre-pass. The compilation will continue. b. A $XREF.ONLY card at the beginning of the source deck will cause the compilation to be terminated at the end of the pre-pass after the SDL/XREF program has been ZIPPED.

Internal File Names

The UPL Compilers internal and external file identifiers are as follows:

Internal	External	Description
CARDS	CARDS	Card source input file.
SOURCE	SOURCE	Primary source input file if MERGE option used.
NEWSOURCE	NEWSOURCE	Updated source output file if NEW option used.
LINE	LINE	Line printer file.

NDL COMPILER

General

The Network Definition Language (NDL) is a high level language for data communication and provides a means of generating a B 1800/B 1700 Network Controller. The B 1800/B 1700 NDL Compiler translates the input source code and outputs a NDL program listing, a Network Controller code file, and the Network Information File (NIF). Figure 4-7 below illustrates the NDL generation process.

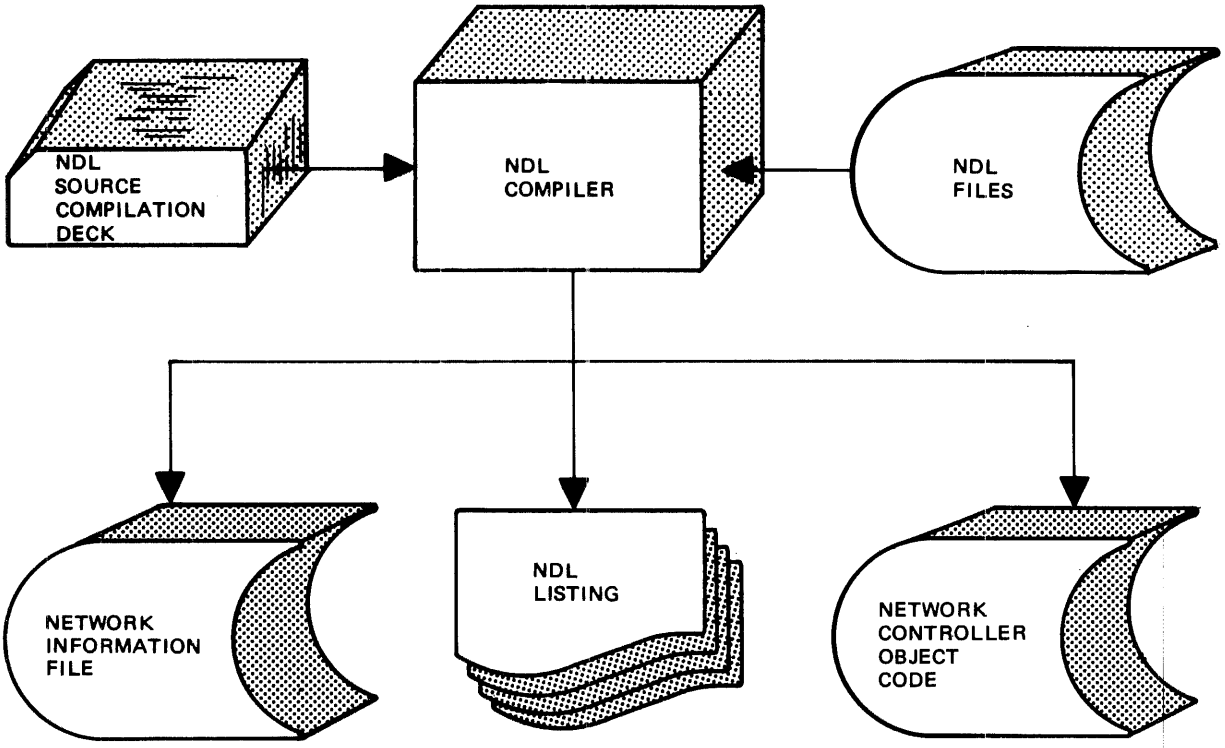


Figure 4-7. NDL Generation Process

Compilation Card Deck

Figure 4-8 contains an example of a NDL compilation card deck used to compile a NDL program.

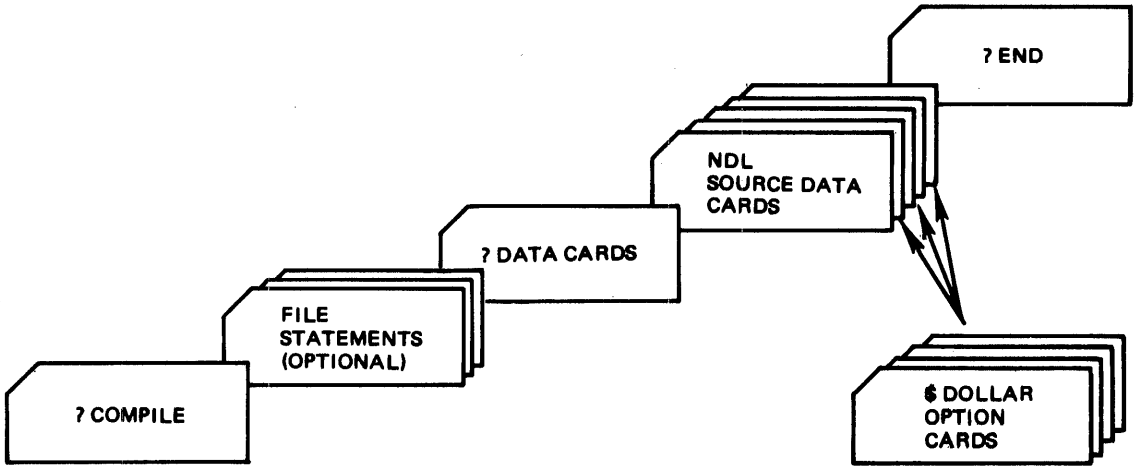


Figure 4-8. NDL Compilation Card Deck

Compiler Options

There are various options available that when invoked affect the compilation process. The options cover areas such as list format, error and warning message handling, maintenance, changing stack sizes, and merging source code.

An option is either in a set or reset condition. The NDL compiler is preset with the following options: LIST, CHECK, and DOUBLE. All other options must be invoked using the dollar option card at compile time. The NDL dollar option card has the following format:

$ [NO] option-1 . . . [NO] option-n

The dollar symbol ($) must be in column one with one or more spaces separating each option specified. With the one exception LIBRARY, there may be multiple options per card.

The available options and an explanation of their functions appear alphabetically as follows:

Option	Description
CHECK	This option causes the compiler to print warning messages for sequence errors in the source language input. A sequence error will occur when the sequence number of the last card is greater than or equal to the current sequence number.
CODE	The generated SDL code (S-operators) will be listed on the line-printer.
CONTROL	The dollar ($) option cards will be output on the object program listing.

CREATE LIBRARY	If this card is used, the NDL compiler must be called by an EXECUTE control instruction rather than by a COMPILE. Following the CREATE LIBRARY card, there must be all the requests and controls to be included in the new library file, which are given the name NDL/NEWLIBRARY.
CSSIZE integer	This option is used to alter the Control stack size to integer entries.
DOUBLE	Double space listing.
DYNAMICSIZE integer	Sets the Network Controller's dynamic memory size to integer bits.
ESSIZE integer	The Network Controller's Evaluation stack size may be set to integer entries.
FORGETERRORS	Directs the compiler to generate the object Network Controller despite syntax errors.
LIBRARY	The NDL source code specified by standard identifier is retrieved from the NDL source/standards and inserted in the user's program following the $ LIBRARY card. The LIBRARY option may not be included on a card containing other options. When the LIBRARY option is used to access standard REQUEST and CONTROL routines, the standard REQUESTS must precede the standard CONTROLS.
LIST	The source code will be listed.
LST	The source code will be listed.
MERGE	This option is used to merge the primary input with the secondary input.
NEW	A new source file will be created for use later as secondary input when this option is specified.
NIF	This option allows the creation of a new Network Controller in about half the time required for a total compilation. The old requests and line control code must remain unchanged.
NSSIZE integer	The Network Controller's Name stack size may be set to integer entries.
NO	Options may be reset by specifying $ NO followed by the name of the option to be reset. This allows options to be set and reset at the user's discretion. NO does not affect the VOID or LIBRARY options.
PAGE	Causes a page eject of the compiler output listing.
PPSSIZE integer	The Network Controller's Name stack size may be set to integer entries.

SEQ	The source may be sequenced by supplying a beginning sequence number and an increment. The numbering will begin at SEQ BASE and will be incremented by SEQ INCRMT. A plus sign (+) is used to separate SEQ BASE and SEQ INCRMT which are both integers. `$ SEQ SEQBASE + SEQ INCRMT` If only $ SEQ is specified thereby omitting SEQ BASE and SEQ INCRMT, the numbering will start with 00000000 and increment by 100.
SGL	Single space listing.
SINGLE	Single space listing.
SUPPRESS	Prohibits the syntax warnings to be printed on the object program listing.
UPDATE LIBRARY [new-library-name]	If this card is used, the NDL compiler must be called by an EXECUTE control instruction rather than by a COMPILE. The request and control patches, with appropriate sequence numbers, must follow the UPDATE LIBRARY card. If not specified, the default name of the new library file is NDL/NEWLIBRARY.
VSSIZE integer	The Network Controller Value stack size may be set to integer bits.
VOID	When VOID is used in conjunction with $ MERGE, it eliminates certain unwanted secondary source records from the new new source file being created. By specifying $ VOID, the secondary source record with the current sequence number is skipped by the compiler. $ VOID may also be followed by an eight character integer which instructs the compiler to skip all secondary source records beginning at the current sequence number and continuing until a secondary source record is read that has a sequence number higher than the eight character integer specified.

Internal File Names

The NDL's internal and external file identifiers are as follows:

Internal	External	Description
CARDS	CARDS	Input file from card reader.
LINE	LINE	Source output listing to line printer.
SOURCE	SOURCE	Input file from disk or tape when the MERGE option is invoked.
NEWSOURCE	NEWSOURCE	Output file to disk or tape when the new option is invoked. Default is to disk.
NIF	NDL/NIF	Network Information File
ADDRESS	NDL/ADDRESS	Network Controller Address File
MACRO	NDL/MACRO	Skeletal Network Controllers
LIBRARY	NDL/LIBRARY	Library

MIL COMPILER

General

The Micro Implementation Language (MIL) is a symbolic coding technique that makes available all the capabilities of the B 1800/B 1700 processor. A MIL program contains a set of micro instructions that are directly executable upon the B 1800/B 1700 hardware. MIL assumes interpretive or indirect processing of information contained in main memory.

Compilation Card Deck

Figure 4-9 contains an example of a MIL compilation deck.

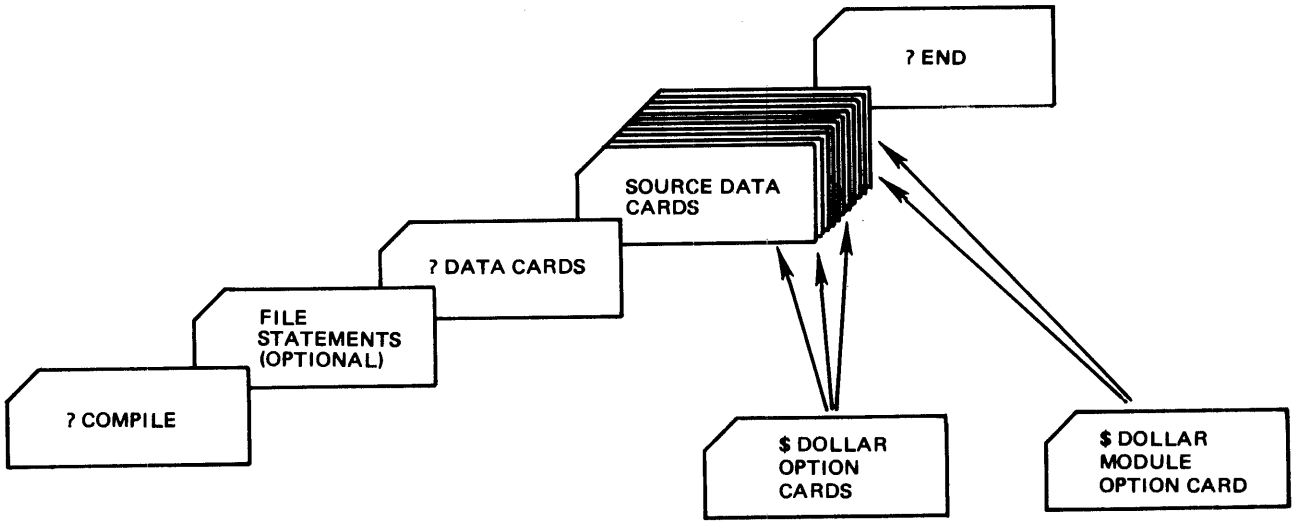


Figure 4-9. MIL Compilation Card Deck

Compiler Options

The $ Option Card is used to notify the MIL Compiler as to which options are required by the programmer during compilation.

The $ Option card for the MIL Compiler has the following format:

$ [NO] option-1 . . . [NO] option-n

The MIL $ Option Card has the following characteristics:

a. Column one must be a $ sign.

b. There must be at least one space between options.

c. Options may be in any order.

d. Columns 73-80 are reserved for sequence numbering.

e. Any number of $ Option Cards may appear anywhere within the source deck.

f. The optional word NO appearing before any option RESETS that option.

The MIL Compiler is preset with the following options: LIST, ALLCODE, SINGLE, AMPERSAND, and CHECK.

ALLCODE	Lists all codes generated by each MIL statement.
AMPERSAND	Prints all ampersand (&) cards.
CHECK	Checks for sequence errors.
DEBUG	Debugs compiler only.
DECK	Punches an object deck.
DOLLAR	Prints all dollar ($) cards.
DOUBLE	Double spaces listing.
EXPAND	Prints all statements within a macro invocation.
FORCE	Outputs all files regardless of syntax errors.
HEADINGS	Prints headings and titles on top of each page; does not affect line count.
LINES.PER.PAGE	Specifies the number of lines to be put on the page of a listing.
LIST	Lists all MIL source input that is compiled.
LISTALL	Lists all MIL source input regardless whether conditionally excluded.
MERGE	Merges the secondary source of input with the file SOURCE. When a duplicate sequence number exists the record from the card file will be used.
NEW	Creates a new source file.
NO	Resets option.
PAGE	Ejects page of listing at that point.
PAGE.NUMBERS	Numbers the pages of the listing and maintains a count.
PARAMETER.BLOCK	Used in conjunction with DECK and causes a parameter block to be punched with the deck. Used primarily with interpreters that are to be run with the MCP.
SEQ	Resequences and outputs NEWSOURCE file and listing.
SINGLE	Single spaces program listing.
SUBSET	Generates code for the B 1710 series processors.

SUPPRESS	Suppresses all warning messages except sequence error messages.
VOID	Voids those images from the secondary input file SOURCE which have sequence fields less than or equal to the terminating sequence field. If the terminating sequence field is missing, then the only image voided is the first one with the same sequence field as the VOID card.
XREF	Produces a listing of all user specified names and labels with each identifier associated with its sequence number for each declaration and invocation.
XREF.LABELS	Produces a cross reference of labels only.
XREF.NAMES	Produces a cross reference of user specified names only.

Module Option Dollar Card

The module option dollar card ($) is used to set or reset user defined toggles used in conjunction with IF statements in the conditional inclusion of source statements. It may be used anywhere within the source deck, and each module option dollar card affects only those user defined toggles which are referenced on that card. A user defined toggle can only be referenced by an IF statement when declared (set or reset) on a module option dollar card.

Example:

```
$ SET SYSTEM1, RESET SW2, SET SW4, SET SW5
```

Internal File Names

The MIL Compiler's internal and external file identifiers are as follows:

Internal	External	Description
CARDS	CARDS	Input file from the card reader.
LINE	LINE	Source output listing to the printer.
SOURCE	SOURCE	Input file from disk or tape when the MERGE option is invoked.
NEWSOURCE	NEWSOURCE	Output file to disk or tape when the NEW option is invoked. Default is to disk.

Object Code Deck Format

The DECK option causes the object code to be output to punched cards. The cards have the following format with all fields except the program identifier in hexadecimal format.

Card Columns	Description
1-6	24-bit control memory address.
8-9	8-bit count of the number of bits of data on this card.
11-70	Contains up to 240 bits of data, left justified.
72-80	Program identifier, used for documentation only.

Compiler Restrictions

a. The only source of input is the card reader, unless otherwise specified by the MERGE option. Once the MERGE option has been invoked, card only input is not possible.

b. When dollar cards ($) are not included in the compilation deck, the default options will prevail.

c. Options may be reset only by using the NO option. A space must separate NO and the option being reset.

d. Comments may appear on dollar cards only if preceded by either an asterisk (*) or a percent (%) sign.

e. Dollars cards are not included as part of the NEWSOURCE file when the option NEW is specified.

SDL COMPILER

General

The Software Development Language (SDL) was developed specifically for writing B 1800/B 1700 system software. SDL is a high-level, procedure oriented language. All programs written in SDL source language must be processed by the SDL Compiler. The SDL Compiler transforms the source statements into S-Code to be interpreted by a set of micro-instructions called firmware.

Compilation Card Deck

Figure 4-10 contains an example of a SDL compilation card deck.

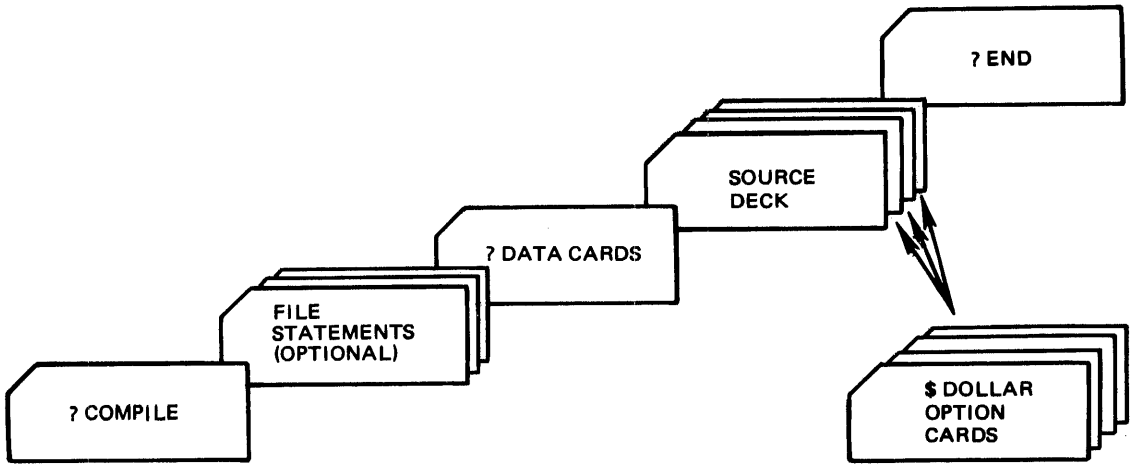


Figure 4-10. SDL Compilation Card Deck

Compiler Options

The SDL Compiler has certain options that are available to the operator or programmer that may be implemented at the time of compilation. These options are input by card along with the source deck and have the following format:

$ [NO] option-1 . . . [NO] option-n

The SDL Dollar ($) Options have the following characteristics:

a. Column one must contain a $ sign.

b. There must be at least one space between options.

c. Options may be in any order.

d. Columns 73-80 are reserved for sequence numbering.

e. Any number of options may appear anywhere within the source deck.

f. The option NO appearing before any other option resets or negates that option.

The following is a list of the SDL Compiler options and their definitions.

AMPERSAND	Prints those ampersand cards that are examined.
CHECK	Checks the source input file for sequence errors.
CODE	Prints the SDL object code generated for each source statement.
CONTROL	Prints all Compiler Dollar Option cards from that point. If the option word CONTROL is to be printed, $ CONTROL (space) CONTROL format must be used.
CREATE.MASTER	When used, this option must be the first card in the compilation deck and causes the compiler to perform the following functions: a. Dump information to the master information files. b. Create a new source file. c. Create a new code file.
CSSIZE integer	Assigns the number of entries in the Control stack represented by integer and overrides the compiler estimate.
DEBUG	Debugs compiler only.
DETAIL	Causes the compiler to list the expansion of all define invocations.
DOUBLE	Double spaces listing.
DYNAMICSIZE integer	Assigns the value of the integer as the estimated memory size allocated for paged arrays. Integer is expressed in bits.
ESSIZE integer	Assigns the number of entries in the Evaluation stack by integer and overrides the compiler estimate.
FORMAL.CHECK	Causes the checking of the actual parameters passed to each procedure during execution against the TYPE and LENGTH specifications of their corresponding formal declarations. Also, the values returned from function procedures will be checked against the type and length in the procedure head statement. Lack of correspondence is a run time error.
INTERPRETER file-identifier	Causes the program when executed to use the assigned interpreter rather than the compiler default interpreter.
INTRINSIC file-identifier (family-name only)	Causes the program when executed to use those Intrinsics with the assigned family-name rather than the compiler assigned family-name.

LIST	Prints the source input that was compiled. The NO option when invoked with LIST will reset the LISTALL option also.
LISTALL	Prints all source input regardless if conditionally excluded. The LISTALL option sets the LIST option on, but NO LISTALL does not reset LIST.
MERGE	Indicates to the compiler that the source file is on tape or disk and there are cards to be merged for the current compilation.
NEW	Creates a new primary source file.
NO	The presence of NO immediately before any other option negates that option.
NSSIZE integer	Assigns the number of entries expressed by integer to the Name stack thereby overriding the compiler estimated size.
PAGE	Ejects page.
PPSIZE integer	Assigns the number of entries expressed by integer to the Program Pointer stack thereby overriding the compiler estimated size.
SEQ beginning-sequence-number increment	Causes the output file to be resequenced beginning with the number used with SEQ.
SINGLE	Single spaces listing.
SIZE	Outputs at the end of the listing the code segment names and their sizes.
SUPPRESS	Causes all warning messages to be suppressed. To suppress sequence error message invoke the NO CHECK option.
VOID sequence number	Causes all records in the primary source file (as in the case of the MERGE) to be removed from the sequence number of the the VOID card itself through the sequence number entered with the VOID option. The VOID option has the following restrictions: a. Must be the only compiler option on the card. b. Cannot be preceded by the NO option. c. Must contain a sequence number in columns 73-80.
VSSIZE integer	Assigns the number of entries expressed by integer to the size of the Value stack thereby overriding the compiler estimated size.
XMAP	Causes an extended SDL object code MAP file to be created showing the relative displacement of object code per source card sequence number, per Code Segment.

Internal File Names

The SDL Compiler's internal and external file identifiers are as follows:

Internal	External	Description
CARDS	CARDS	Card source input file.
SOURCE	SOURCE	Primary input source file if MERGE option used.
NEWSOURCE	NEWSOURCE	Updated source output file if NEW option used.
LINE	LINE	Line printer file.

SDL Recompilation

The recompilation of an SDL program creates a Master Information File.

The create master must take place once and then may be followed by successive recompilations. Both the create master and the recompilation may be performed at the same time. In addition it is possible to perform successive regular compilations without invoking the recompilation facility.

CREATING MASTER INFORMATION FILES

In order to create Master Information Files, the first card of the compilation source file must be a $ CREATE.MASTER option card. This option causes the SDL Compiler to perform the following functions:

a. Save information needed for the recompilation into master files.

b. Create a new source file about which the information is to be used.

c. Use the Master Information Files and the new source file to create a new output code file.

The following files contain the information to be saved and used in the recompilation process.

NEWSOURCE

NEW.INFO.FILE

NEW.BLOCK.ADDRESS.FILE

NEW.SECONDARY.FILE

NEW.FPB.FILE

The following information is contained in the master information files: the input source images, Lexic Level one procedure boundaries for both the source file and object file, Lexic Level zero symbol tables, a record of all code addresses that have been emitted, the object code from which code addresses that have been emitted, the object code from which code addresses have been excised, the File Parameter Blocks, and SCRATCHPADS. (Refer to B 1700 Master Control Reference Manual, dated June, 1974, and B 1700 System Reference Manual, dated December, 1973.)

CREATE MASTER RESTRICTIONS

The create master operation has the following restrictions:

a. $ CREATE.MASTER must be the first card of the compilation source card file.

NOTE

This is to include any ampersand cards; they sould be sequenced.

b. $ NEW is not needed for this operation.

c. $ SEQ should be used if any input source images do not contain sequence numbers.

RECOMPILING

Recompilation is performed on a Lexic Level one procedural basis. That is, the outermost procedure containing a recompilation source card is the procedure which is recompiled. The code that is produced by the recompilation will be merged into, and in some cases replace some of the information created during the create master process.

The recompilation is invoked by including as the first card of the recompilation source deck a $ RECOMPILE.

The $ RECOMPILE causes the compiler to use the recompilation source deck (usually referred to as "patches") and the master information files to locate the Lexic Level one procedures and generate the same information for them as was generated for the entire program in the create master operation. This information is then combined, procedure by procedure, with the Master Information Files to produce the final form of the program that is turned into a new code file.

RECOMPILATION RESTRICTIONS

The recompilation process has the following restrictions:

a. The $RECOMPILE must be the first card of the recompilation source deck (patch deck).

b. The recompilation source deck may contain dollar cards, and ampersand (SET and RESET) cards, followed by the patch cards.

c. Lexic Level zero code cannot be patched. This includes all global data, Lexic Level one procedure headings, and the main program.

d. Neither $ SEQ or $ MERGE options may be invoked while using $ RECOMPILE process.

e. The source file that is input during the recompilation must be on disk in order that it may be accessed randomly.

CREATE MASTER AND RECOMPILE OPERATION PERFORMED TOGETHER

Both the create master and the recompilation process may be performed at the same time. Simply adhere to the rules for each separate operation and use $ RECOMPILE CREATE.MASTER as the first card of the source deck. It should be noted, however, that this procedure updates some of the information in the file MASTER.INFO.FILE. Therefore, the file must be saved because if any subsequent recompilations are desired, they must be performed against the saved master file.

GENERAL INFORMATION

1. The only information which may be listed during a recompilation is that which is being recompiled.

2. Both the source file used with $ CREATE.MASTER and the file created by $ CREATE.MASTER may be on tape, but the new source file must be placed on disk prior to any recompilations.

3. Because of the disk space required for recompilation, it is advantageous to keep source files on tape until needed.

4. The source image file created by the create master process contains no information other than the source images. Therefore it may be used in a regular compilation.

SDL COMPILATION DECK EXAMPLES

Compile and Create Master

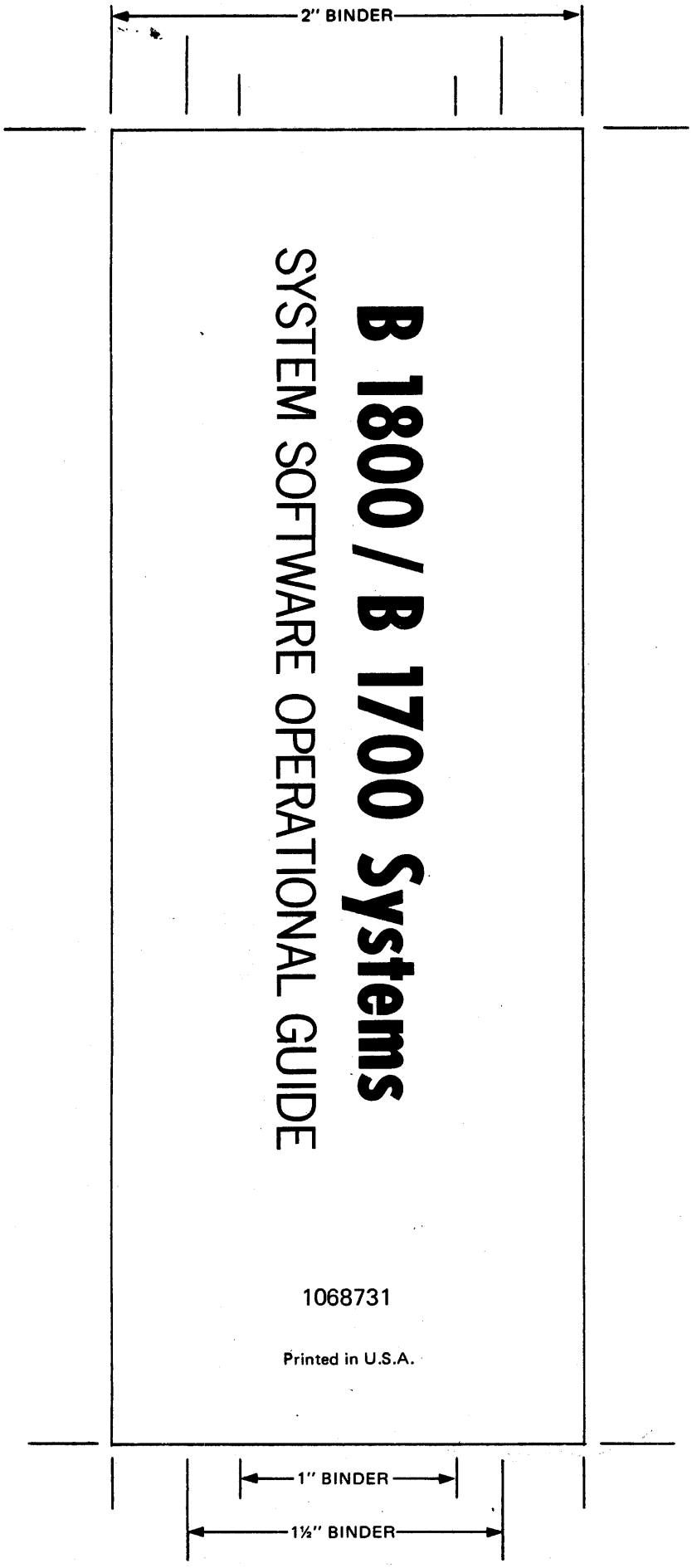
```
? COMPILE SA SDL LIBRARY
? FILE SOURCE NAME SA0206/SOURCE TAPE;
? FILE NEWSOURCE NAME SA0410/SOURCE TAPE;
? FILE NEW.INFO.FILE NAME SA0410/INFO;
? FILE NEW.BLOCK.ADDRESS.FILE NAME SA0410/BLOCK.ADDRESS;
? FILE NEW.SECONDARY.FILE NAME SA0410/SECONDARY;
? FILE NEW.FPB.FILE NAME SA0410/FPB;
? DATA CARDS
$ CREATE.MASTER
$ MERGE LIST SINGLE SIZE SEQ
[PATCH CARDS]
[99999999 CARD]
? END
```

Recompile

```
? COMPILE SA SDL LIBRARY
? FILE SOURCE NAME SA0410/SOURCE;
? FILE MASTER.INFO.FILE NAME SA0410/INFO;
? FILE MASTER.BLOCK.ADDRESS.FILE NAME SA0410/BLOCK.ADDRESS;
? FILE MASTER.SECONDARY.FILE NAME SA0410/SECONDARY;
? FILE MASTER.FPB.FILE NAME SA0410/FPB;
? DATA CARDS
$ RECOMPILE
$ LIST SINGLE SIZE
$ VSSIZE 10000 NSSIZE 100
[PATCH CARDS]
[99999999 CARD]
? END
```

Recompile and Create Master

```
? COMPILE SA SDL LIBRARY
? FILE SOURCE NAME SA0410/SOURCE;
? FILE MASTER.INFO.FILE NAME SA0410/INFO;
? FILE MASTER.BLOCK.ADDRESS.FILE NAME SA0410/BLOCK.ADDRESS;
? FILE MASTER.SECONDARY.FILE NAME SA0410/SECONDARY;
? FILE MASTER.FPB.FILE NAME SA0410/FPB;
? FILE NEWSOURCE NAME SA0411/SOURCE TAPE;
? FILE NEW.INFO.FILE NAME SA 0411/INFO;
? FILE NEW.BLOCK.ADDRESS.FILE NAME SA0411/BLOCK.ADDRESS;
? FILE NEW.SECONDARY.FILE NAME SA0411/SECONDARY;
? FILE NEW.FPB.FILE NAME SA0411/FPB;
? DATA CARDS
$ RECOMPILE CREATE.MASTER
$ VSSIZE 10000 NSSIZE 100
$ LIST SINGLE SIZE
[PATCH CARDS]
[99999999 CARD]
? END
```



Printed in U.S.A. August 1977 1068731