

Burroughs
B 1700
SYSTEMS

System Software
OPERATIONAL GUIDE

\$5.00

Burroughs
B 1700 Systems
System Software

OPERATIONAL GUIDE



Burroughs Corporation

Detroit, Michigan 48232

\$5.00

COPYRIGHT © 1972, 1973, 1974 BURROUGHS CORPORATION
AA 370509, AA 401135

Burroughs Corporation believes the program described in this manual to be accurate and reliable, and much care has been taken in its preparation. However, the Corporation cannot accept any responsibility, financial or otherwise, for any consequences arising out of the use of this material. The information contained herein is subject to change. Revisions may be issued to advise of such changes and/or additions.

Correspondence regarding this document should be forwarded using the Remarks Form at the back of the manual, or may be addressed directly to Systems Documentation, Technical Information Organization, TIC-Central, Burroughs Corporation, Burroughs Place, Detroit, Michigan 48232.

TABLE OF CONTENTS

SECTION	TITLE	PAGE
	INTRODUCTION	ix
1	INTRODUCTION TO SYSTEM	1-1
	System Initialization	1-1
	Unit Mnemonics	1-1
	System Description	1-1
	Hardware Requirements	1-2
	Central Service Module	1-2
	Interpreters	1-3
2	MASTER CONTROL PROGRAM	2-1
	General	2-1
	MCP Disk Structures	2-1
	Disk Directory	2-1
	Disk-Pack-Identifier	2-2
	Main Directory File Name	2-2
	Sub-Directory File Name	2-2
	Main Directory Contents	2-2
	Sub-Directory Contents	2-2
	Directory Reference	2-3
	Multiple Pack Files	2-3
	Introduction	2-3
	Abbreviations	2-3
	Restrictions	2-3
	Base Packs	2-3
	Continuation Packs	2-3
	General Information	2-4
	Halts	2-4
	MCP Options	2-5
	BOJ	2-6
	CHRG	2-6
	CLOS	2-6
	DATE	2-6
	DUMP	2-6
	EOJ	2-6
	LAB	2-6
	LIB	2-6
	LOG	2-7
	MEM	2-7
	OPEN	2-7
	PBD	2-7
	PBT	2-7
	RMOV	2-7
	SCHM	2-7
	TERM	2-7
	TIME	2-7
	ZIP	2-7

TABLE OF CONTENTS (Cont)

SECTION	TITLE	PAGE
2 (Cont)	MCP-Operator Interface	2-8
	General	2-8
	MCP Communications	2-8
	Punched Cards	2-8
	Console Printer	2-9
	Zip	2-9
	General Terms	2-9
	Library Maintenance Instructions	2-11
	CHANGE	2-11
	{ADD }	
	{LOAD}	2-12
	{DUMP }	
	{UNLOAD}	2-13
	REMOVE	2-14
	Control Instructions	2-15
	COMPILE	2-15
	DYNAMIC	2-16
	EXECUTE	2-17
	MODIFY	2-18
	AFTER	2-19
	AFTER.NUMBER	2-20
	THEN	2-21
	CHARGE	2-22
	DYNAMIC.SPACES	2-23
	FILE	2-24
	File Attributes	2-24
	FREEZE	2-28
	HOLD	2-29
	INTERPRETER	2-30
	INTRINSIC.NAME	2-31
	INTRINSIC.DIRECTORY	2-32
	MEMORY	2-33
	PRIORITY	2-34
	SCHEDULE.PRIORITY	2-35
	UNFREEZE	2-36
	VIRTUAL.DISK	2-37
	File Parameter Instructions	2-38
	DATA	2-38
	END	2-39
	Keyboard Input Messages	2-40
	General	2-40
	Keyboard Entry Procedure	2-40
	AX	2-43
	BF	2-44
	CD	2-45
	CE	2-46
	CM	2-47
	CN	2-48
	CP	2-49
	CQ	2-50

TABLE OF CONTENTS (Cont)

SECTION	TITLE	PAGE
2 (Cont)	CS	2-51
	CT	2-52
	DM	2-53
	DP	2-54
	DR-DT	2-55
	DS	2-56
	ED	2-57
	FM	2-58
	FN	2-59
	FR	2-60
	FS	2-61
	FT	2-62
	GO	2-63
	HS	2-64
	HW	2-65
	IL	2-66
	JM	2-67
	KA	2-68
	KP	2-69
	LC	2-70
	LD	2-71
	LG-LN	2-72
	LT	2-73
	MP	2-74
	MR	2-75
	MX	2-76
	OF	2-77
	OK	2-78
	OL	2-79
	OU	2-80
	PB	2-81
	PD	2-82
	PG	2-84
	PM	2-85
	PO	2-86
	PR	2-87
	PS	2-88
	RB-RF	2-89
	RD	2-90
	RL	2-91
	RM	2-92
	RN	2-93
	RO	2-94
	RP	2-95
	RS	2-96
	RT	2-97
	RY	2-98
	SD	2-99
	SL	2-100
	SO	2-101
	SP	2-102

TABLE OF CONTENTS (Cont)

SECTION	TITLE	PAGE
2 (Cont)	ST	2-103
	SV	2-104
	TD	2-105
	TI	2-106
	TL	2-107
	TO	2-108
	TR	2-109
	UL	2-110
	WD	2-111
	WM	2-112
	WS	2-113
	WT	2-114
	WW	2-115
	WY	2-116
	MCP Output Messages	2-117
	General	2-117
	Syntax	2-117
	MCP Messages	2-117
3	SYSTEM SOFTWARE	3-1
	Disk Cartridge Initializer	3-1
	General	3-1
	Disk Initialization Instructions	3-1
	Disk Verification	3-2
	COLDSTART	3-3
	General	3-3
	Procedure	3-3
	Clear/Start and Memory Dump Procedure	3-4
	General	3-4
	Clear/Start Procedure	3-4
	Name Table	3-5
	Operating Environments	3-5
	Selecting Environments	3-7
	Temporary Environment Changes	3-7
	Memory Dump Procedure	3-8
	Firmware Detected Errors	3-8
	Disk File Copy	3-10
	General	3-10
	DISK/COPY Operating Instructions	3-10
	Specification Cards	3-10
	DMPALL	3-12
	General	3-12
	Printing	3-12
	Reproducing	3-12
	Operating Instructions	3-12
	Keyboard Console	3-12
	Cards	3-12
	Print Specifications	3-13
	Reproducing Specifications	3-14
	FILE/LOADER	3-18
	General	3-18

TABLE OF CONTENTS (Cont)

SECTION	TITLE	PAGE
3 (Cont)	Dollar Card	3-18
	Dollar Dollar Card	3-18
	Asterisk Card	3-18
	Error Messages	3-19
	FILE/PUNCHER	3-21
	General	3-21
	Error Messages	3-21
	SORT	3-22
	General	3-22
	Sort Execution Deck	3-22
	The File Statement	3-23
	File In	3-23
	Dp-Id	3-23
	File-Identifiers	3-23
	Card	3-23
	Tape	3-23
	Disk	3-23
	Records-Per-Area	3-24
	Record-Size	3-24
	Blocking-Factor	3-24
	Purge	3-24
	Default	3-24
	Multi	3-24
	Out	3-24
	The Key Statement	3-25
	Key	3-25
	Key-Location	3-25
	Key-Length	3-26
	Ascending or A	3-26
	Descending or D	3-26
	Alpha or UA	3-26
	Numeric or UN	3-26
	SA	3-26
	SN	3-26
	Sort Option Statements	3-27
	Noprint	3-27
	Memory	3-28
	Timing	3-28
	Bias	3-28
	Tagsort	3-28
	Inplace	3-29
	Syntax	3-29
	Comment	3-29
	Sort Reserved Words	3-30
4	PROGRAM PRODUCTS	4-1
	COMPILERS	4-1
	Introduction	4-1
	REPORT PROGRAM GENERATOR	4-1
	General	4-1
	Compilation Card Deck	4-1

TABLE OF CONTENTS (Cont)

SECTION	TITLE	PAGE
4 (Cont)	Dollar Card Specifications	4-2
	RPG Extensions	4-2
	Compiler-Directing Options	4-3
	RPG to COBOL Options	4-4
	Internal File Names	4-5
	RPG TO COBOL TRANSLATOR (COFIRS)	4-6
	General	4-6
	Execution of Translator	4-6
	COBOL COMPILER	4-7
	General	4-7
	Compilation Card Deck	4-7
	\$ OPTION Control Card	4-7
	SOURCE DATA Cards	4-9
	Internal File Names	4-11
	FORTRAN COMPILER	4-12
	General	4-12
	Compilation Card Deck	4-12
	\$ Option Card	4-12
	Options	4-13
	SOURCE DATA Cards	4-13
	Internal File Names	4-13
	BASIC COMPILER	4-14
	General	4-14
	Compilation Card Deck	4-14
	\$ Option Card	4-15
	Options	4-15
	SOURCE INPUT Cards	4-15
	Intrinsic Files	4-16
	Sample Compilation Deck	4-16
	Internal File Names	4-17

LIST OF ILLUSTRATIONS

FIGURE	TITLE	PAGE
3-1	DISK/COPY Control Deck	3-10
3-2	SORT Execution Deck	3-22
4-1	RPG Compilation Deck	4-1
4-2	COBOL Compilation Deck	4-7
4-3	FORTRAN Compilation Deck	4-12
4-4	BASIC Compilation Deck	4-14

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 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 1700 system.

The purpose of the B 1700 Systems System Software Operational Guide is to provide a general description of all Burroughs 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.

[=]

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	– 96-column card device
LPx	– Line printer
CRx	– 80-column card reader
CPx	– 80-column card punch
DPx	– Disk cartridge or pack
SRx	– Reader sorter
DISK	– Head-per-track Disk
MTx	– Magnetic tape
SPO	– Console Printer
CSx	– Magnetic Tape Cassette

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 (MCP II)

Both the MCP I and MCP II will service the following standard peripheral equipment:

- a. Console Printer
- b. 96-column Card Devices
- c. 80-column Card Devices
- d. Line Printers
- e. Magnetic Tape
- f. Disk Cartridges
- g. Disk Packs (MCP II)
- h. Head-per-Track Disk
- i. Magnetic Tape Cassette

In addition, MCP II will accommodate the MICR reader sorter as well as various data communications devices through a single line or a multi-line control.

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.

<u>Hardware Type</u>	<u>Usage</u>
Console Printer	Operator communication
Disk	Auxiliary storage
Card Reader	Control input

CENTRAL SERVICE MODULE

The Central Service Module (CSM) 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.

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

SECTION 2

MASTER CONTROL PROGRAM

GENERAL

The Master Control Programs (MCP's) are modular operating systems which assume complex and repetitive functions to make programming and operations more 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 by the MCP. 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.

The B 1700 System Software Operational Guide will make reference to both MCP I and MCP II, distinguishing between the functions of each where applicable. The basic difference between MCP I and MCP II is that MCP I is designed for minimum system configurations and does not have multiprogramming capabilities whereas MCP II is designed for larger system configurations with multiprogramming capabilities.

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, the users' 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 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 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.

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 will 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 will increase 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 addresses will 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. All COBOL files assigned to DISKPACK are assumed to be multiple pack files unless specified as SINGLE. For example: DISKPACK [SINGLE]

Abbreviations

MPF – Multiple Pack Files
BP – Base Pack
CP – Continuation Pack

Restrictions

There are some restrictions imposed on MPFs that may limit their selection for usage. They are as follows:

- a. The maximum number of packs that may be assigned to a MPF is 16, consisting of one BP and 15 CPs.
- b. There must be a minimum of two (2) disk drives present on the system (one system pack, one removable pack).
- c. Only removable disk packs may be used for a MPF. A system pack cannot be used for a MPF.
- d. All packs containing a MPF must have unique serial numbers. The disk pack-id is not the primary identifier for CPs.

NOTE:

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

Base Packs

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

The header on the BP retains all information concerning the file including the addresses of every area assigned to that file. For each area which is resident on a CP, the BP will contain the serial number of the CP. This allows the MCP (via the BP) to control all processing, and thereby avoids updating each CP as the file is processed.

Continuation Packs

An MPF may be contained on only one pack, the BP. When the data overflows or “continues” to additional packs, the term Continuation Pack (CP) is used. There may be up to a maximum of 15 CPs for one MPF, but a CP may be associated with only one BP. A CP may only contain continuation files, and all continuation files contained on a CP must be assigned to the same BP. It may contain no single pack files, and may not itself be a BP.

When space is needed for a MPF, the MCP will search for another CP that is associated with the same BP. If no CP is present on the system, the MCP will then search for a scratch pack (one that has just been INITIALIZED or PURGED and is of the same type, RESTRICTED or UNRESTRICTED, as the BP). If one is found the scratch pack is automatically assigned to the MPF.

General Information

DISK/COPY and DMPALL cannot be used to copy MPFs.

MPFs may be sorted (INPLACE sort only).

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

Random files are allowed for MPFs.

A system pack cannot be used as a BP or CP.

The CHANGE (CH) message is not allowed on a MPF; however a REMOVE (RE) message is legitimate.

A check is done by the MCP prior to opening an MPF to predict whether a duplicate file situation might exist. If so, the operator has the option of either removing the existing file at that time or waiting till CLOSE time to remove the duplicate file, and OKing the program.

A DS message will perform a normal close on a MPF.

A BP does not necessarily have to have any of its MPF data residing on it. In other words, as soon as the file is opened and the tables are built, the BP may 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.

HALTS

When certain conditions of the MCP have been violated, all processing may stop and a HEX value will be displayed in the "L" register. Recovering from a HALT state may usually be accomplished by performing a Clear/Start. The following list will explain the HALT codes and their meanings.

<u>Halt Code</u>	<u>Description of Halt</u>
1	Evaluation/Program Pointer Stack overflow.
2	Control Stack overflow.
3	Name/Value Stack overflow.
4	Cassette data error.
5	Invalid parameter passed to a procedure.
6	Invalid substring.
7	Invalid subscript.
8	Invalid value returned from a procedure.
9	Invalid case.

<u>Halt Code</u>	<u>Description of Halt</u>
A	Divide by zero.
B	Invalid index.
C	Memory parity. The "T" register will contain the address of the parity error. If the "T" register equals @FFFFFF@, the error was caused by an attempt to read outside the physical bounds of memory.
D	Invalid operator.
10	Console HALT. (INTERRUPT switch) To continue processing turn INTERRUPT switch off and press START.
11	This is a controlled HALT. The "T" register will contain a message from the MCP.

T REG

DESCRIPTION

@C3C1E2@	End-Of-Reel on Cassette "LC" LOAD.
TA ≥ @C@	Result Descriptor with Exception Bits Set.
ALL OTHERS	First Six Characters of MCP Source Sequence Number (Submit Memory Dump and Trouble Report).
12	Attempt to write outside of the MCP base or limit register.
22	Invalid service request.
25	Second operation complete bit is missing from the result status field. Result descriptor in T register.

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 option which is 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	LAB	PBT
CHRG	LIB	RMOV
CLOS	LOG	SCHM
DATE	MEM	TERM
DUMP	OPEN	TIME
EOJ	PBD	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.

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.

DUMP

The DUMP option must be set in order to dump memory. If the DUMP option is reset, SYSTEM/DUMPFIL 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.

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-identifier	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.

MEM

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.

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.

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.

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
```

TERM

The TERM option specifies that the MCP automatically discontinue (DS) processing 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 not 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.

ZIP

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

MCP-OPERATOR INTERFACE

General

The Master Control Program is directed to perform particular actions by the system operator through the use of Control Instructions. These control instructions apply to both the MCP I and the MCP II.

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

There are four major types of control instructions:

- (1) Library Maintenance Instructions
- (2) Control Instructions
- (3) Control Attributes
- (4) File Parameter Instructions

MCP Communications

PUNCHED CARDS

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

- a. Column 1 must contain an invalid character (80-column cards) or a question mark (96-column cards). An invalid character or question mark may not appear in any other column.
- b. The remainder of the card may contain control instructions in free-field format; the MCP ignores information in the last eight columns.
- c. If the special character percent (%) appears in a control card, all information following it is ignored for control purposes. This allows comments to be present in control cards.
- d. The appearance of the "less than" (<) sign in a control message will cause the MCP to backspace its pointer one position for each < sign in memory while scanning the control instruction. This allows correction of mistakes without requiring that the entire message be re-entered. Even though this is intended mainly for messages entered via the keyboard, it will work with control instructions entered on punched cards as well. The "less than" sign may not be used for any other purpose.
- e. Any program-name or file-identifier which contains the special characters listed below must be enclosed in quotes.

;	semicolon
,	comma
=	equal sign
/	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/# 00000001

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 does not need to be enclosed with quote marks.

- f. Control instructions may be contained on more than one card; however, words may not be split between cards. The card on which the information is to continue must contain an invalid character or question mark in column 1.

Example:

? EXECUTE ALPHA/BETA PRIORITY = 5 MEMORY
? = 16000 CHARGE = 123456 DATA CARDS

- g. All control instructions are described on the following pages under headings which would 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. It may be followed by another control instruction on the same card as the example above indicates.

CONSOLE PRINTER

Control instructions may be entered via the console printer as input to the MCP. The control statements are restricted to one line; there can be no continuation lines. When the END OF MESSAGE is pressed, the MCP assumes the end of the control instruction and processes the control statement.

ZIP

MCP control statements may be also passed 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.

GENERAL 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.

<u>unit-mnemonic</u>	<u>device</u>
CDx	96-column card device
CRx	80-column card reader
CPx	80-column card punch
LPx	Line Printer
MTx	Magnetic Tape
SRx	MICR Reader Sorter
PPx	Paper Tape Punch
PRx	Paper Tape Reader
DPx	Disk pack/cartridge
DISK	Head-per-Track disk
SPO	Console Printer
CSx	Magnetic Tape Cassette

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:

file-identifier
family-name/file-identifier
dp-id/family-name/file-identifier
dp-id/file-identifier/

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 } file-identifier-1 TO file-identifier-2 [ , file-identifier-3 TO file-identifier-4] . . .
    CH
```

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. For example:

```
? CHANGE
? A/B C/D,
? X Y , Z Q,
? ABC DEF;
```

Termination will occur when a semicolon (;) or “empty hopper” is detected.

**ADD
LOAD**

The ADD statement will cause a file 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 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 statements is:

[?] { ADD
LOAD } [TO disk-pack-identifier] FROM
library-tape-identifier file-identifier-1
[, file-identifier-2] . . . ;

Example:

? LOAD FROM SYSTEM COBOL,
? RPG, BASIC;

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

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

The format of the DUMP and UNLOAD statements is:

```
[?] { DUMP }  
      UNLOAD } [FROM disk-pack-identifier] TO library-tape-  
                identifier file-identifier-1 [ , file-identifier-2] . . . ;
```

Example:

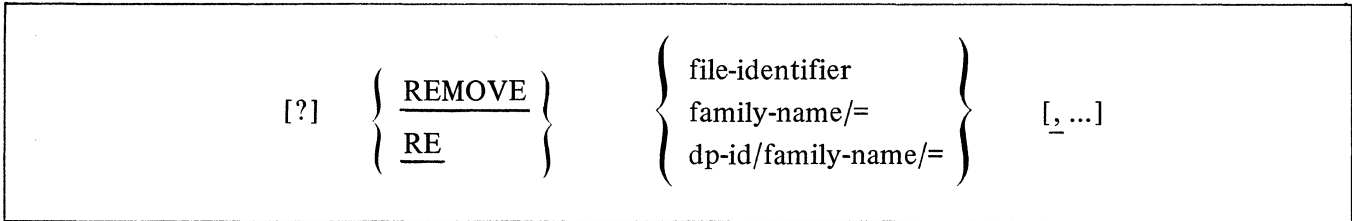
```
? DUMP TO A/B  
X/Y,  
Z/Q,  
AAA  
;
```

REMOVE

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.

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.

For Example:

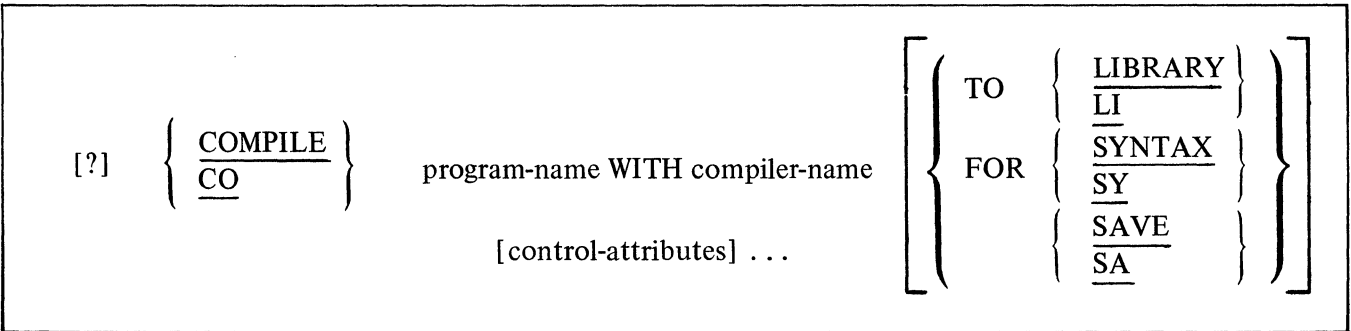
```
? REMOVE A/B
, X, Y,
Z;
```


CONTROL INSTRUCTIONS

COMPILE

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

The format for 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

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

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 can 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

MODIFY

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

The format of a MODIFY statement is:

[?]	$\left\{ \begin{array}{l} \underline{\text{MODIFY}} \\ \underline{\text{MO}} \end{array} \right\}$	program-name [control-attributes] ...
-----	--	---------------------------------------

The MODIFY control statement can 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
- PRIORITY
- SCHEDULE.PRIORITY
- UNFREEZE
- VIRTUAL.DISK

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 } program-name  
      { 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, or until FS-ed by the system operator.

AFTER.NUMBER

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 AN	job-number
-----	-------------------------	------------

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 }

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.

CHARGE

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:

$$[?] \left\{ \begin{array}{l} \underline{\text{CHARGE}} \\ \underline{\text{CG}} \end{array} \right\} [=] \text{ integer}$$

The control word CHARGE can 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 analyzation.

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

CONDITIONAL/UNCONDITIONAL

The flag "PROG.UNCONDITIONAL" has been added to the PPB. The default is false. It is accessed via the Program Parameter statements "CONDITIONAL" ("CA") and "UNCONDITIONAL" ("UC").

Given a program "SUCCESSOR" whose initiation is dependent on the termination of another job "PREDECESSOR", two possibilities have been identified.

"EX PREDECESSOR THEN EX SUCCESSOR UNCONDITIONAL" will cause "SUCCESSOR" to be initiated when "PREDECESSOR" terminates.

"EX PREDECESSOR THEN EX SUCCESSOR CONDITIONAL" will cause "SUCCESSOR" to be initiated only if "PREDECESSOR" terminates successfully.

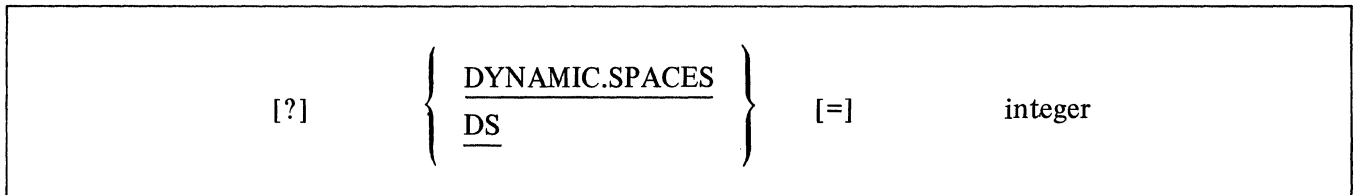
Another form:

"EX SUCCESSOR UC AFTER PREDECESSOR"

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:



Example:

DS = 5

FILE

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:

$$[?] \left\{ \begin{array}{l} \underline{\text{FILE}} \\ \underline{\text{FI}} \end{array} \right\} \text{internal-file-identifier file-attribute-1 [file-attribute-2] \dots ;$$

The control word FILE can 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 ETX. 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 file-identifier is to be changed for this run only, the FILE statement would be as follows:

? 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.

<u>FILE ATTRIBUTE</u>	<u>FUNCTION</u>
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.
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.
BACKUP.DISK	If the file is allowed to go to BACKUP, the output of the file will be allowed to go to disk backup.
BACKUP.TAPE	If the file is allowed to go to BACKUP, the output of the file will be allowed to go to tape backup.

<u>FILE ATTRIBUTE</u>	<u>FUNCTION</u>
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 an area.
BUFFERS[=] integer	The number of buffers assigned to the file will be altered to the value of the integer. The integer must be a positive number from 1 to 15.
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
	[?] $\left. \begin{array}{l} \text{FILE} \\ \text{FI} \end{array} \right\}$ internal-file-identifier-1
	COPY internal-file-identifier-2 <u>FROM</u>
	$\left. \begin{array}{l} \left\{ \begin{array}{l} \# \\ \text{JN} \\ \text{JOB.NUMBER} \end{array} \right\} \text{ job-number} \\ \text{ (working copy)} \end{array} \right\}$
	$\left. \begin{array}{l} \left\{ \begin{array}{l} \text{PN} \\ \text{PROGRAM.NAME} \end{array} \right\} \text{ program-name} \\ \text{ (original copy)} \end{array} \right\}$
CYLINDER.BOUNDARY	Each area of a disk file will start at the beginning of a CYLINDER when the file is directed to a disk pack or disk cartridge.
DEFAULT	Override the disk allocation declared and use the file header block and record sizes. (Input disk and tape files only.)
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.
EU[=] integer	Same as DRIVE.
EVEN	The file will be changed to even parity.
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.

FILE ATTRIBUTE

FUNCTION

INCREMENT.DRIVE

Each area of a disk file will start on the next system disk pack or disk cartridge drive. When the last system drive has been used it will start over from drive ZERO again.

INCREMENT.EU

Same as INCREMENT.DRIVE.

LABEL.TYPE[=] integer

The file will be processed as labeled (integer=0) or unlabeled (integer=1).

LOCK

The file will be LOCKED during file close or program termination time.

MAXIMUM.BLOCK.SIZE[=] integer

Fixed block size to be used for variable length records.

MULTI.PACK

The file will be considered a multi-pack file. (MPF)

NAME [=] file-identifier

The external file-identifier or dp-id will be changed to the value of file-identifier. If only the dp-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.

- a. BACKUP
- b. BACKUP.DISK
- c. BACKUP.TAPE
- d. FORMS
- e. ALLOCATE.AT.OPEN
- f. LOCK
- g. DEFAULT
- h. HARDWARE
- i. MULTI.PACK
- j. OPTIONAL
- k. VARIABLE
- l. CYLINDER.BOUNDARY
- m. INCREMENT.EU
- n. INCREMENT.DRIVE
- o. WORK.FILE

<u>FILE ATTRIBUTE</u>	<u>FUNCTION</u>
ODD	The file will be changed to ODD parity.
OPTIONAL	Select optional file (COBOL only).
PACK.ID[=] disk-pack-id	Alter the pack-id.
PSEUDO	Makes file a pseudo type.
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.
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.
VARIABLE	The file will be processed using variable length records.
WORK.FILE	Assigns this file as a work file used internally.

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

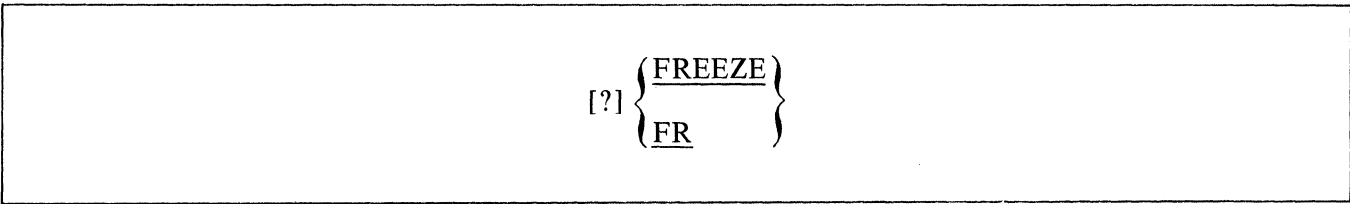
READER.96	REMOTE
READER.PUNCH	MFCU
READER.PUNCH.PRINTER	TAPE.7
READER.SORTER	TAPE.9
CARD.READER	TAPE.PE
CARD.PUNCH	TAPE
DISK	PAPER.TAPE.READER
DISK.PACK	PAPER.TAPE.PUNCH
DISK.FILE	PRINTER
DISK.CARTRIDGE	PUNCH.96
CASSETTE	PUNCH.PRINTER
QUEUE	

FREEZE

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:



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

INTERPRETER

INTERPRETER

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

The format of the INTERPRETER statement is:

[?]	{ <u>INTERPRETER</u> <u>IN</u> <u>INTERP</u>	[=]	{ interpreter-identifier <u>MCP/INTERP</u>
-----	---	-----	--

INTERPRETER may be abbreviated as IN or INTERP.

Examples:

? EXECUTE ALPHA/BETA INTERPRETER COBOL/INTERP001

? EX X/Y IN CCC/SDL/INTERP3

The reserved identifier MCP/INTERP causes the program to use whatever SDL interpreter is being used currently by the MCP. All system software (compilers, sorts, etc.) request MCP/INTERP.

INTRINSIC.NAME

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

The format of the INTRINSIC.NAME statement is:

$$[?] \left\{ \begin{array}{l} \underline{\text{INTRINSIC.NAME}} \\ \underline{\text{IT}} \end{array} \right\} [=] \text{ intrinsic-identifier}$$

The INTRINSIC.NAME may be abbreviated as IT.

The file-id portion of the intrinsics may not be changed.

For example:

? EXECUTE ALPHA/BETA INTRINSIC.NAME ZZZ.INTRIN

or

? EX ALPHA/BETA IT ZZZ.INTRIN

INTRINSIC.DIRECTORY

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:

$$? \left\{ \begin{array}{l} \text{INTRINSIC.DIRECTORY} \\ \underline{\text{ID}} \end{array} \right\} [=] \text{dp-id}$$

The INTRINSIC.DIRECTORY statement can 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:

$$[?] \left\{ \begin{array}{l} \underline{\text{MEMORY}} \\ \underline{\text{ME}} \end{array} \right\} [=] \text{ integer}$$

The control word MEMORY can 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
```

PRIORITY

PRIORITY

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

The format of a PRIORITY statement is:

$$[?] \left\{ \begin{array}{l} \text{PRIORITY} \\ \text{PR} \end{array} \right\} [=] \text{ integer}$$

The control word PRIORITY can 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.
- 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.

PROGRAM SWITCHES:

40 bits of the PPB have been designated as "PROGRAM SWITCHES". They may be accessed by any of the Program Parameter Control Statements (CO, EX, MO, DY, BI, QF, QP), as a group, or 4 bits at a time, by including a "SWITCH STATEMENT" in the Program Parameter Statement list:

SW<SWITCH.ID><VALUE>

SWITCH.ID may be "=" in which case the receiving field is all 40 bits, or it may be a number from 0-9 (followed by optional "=") which identifies which (from the left) of 10 four-bit groups is to act as the receiving field. Value is any control language literal. Storage rules are SDL bit.

eg.: EX GEORGE SW = "HELLO.PETER"

Note: The "H" is truncated

EX GEORGE SW 5 = @F@

Note: same as:

EX GEORGE SW 5 15

At BOJ time the 40 bits are transferred to the run structure nucleus. They may be written with the "SW" control statement, and read with the "TS" (type switches) control statement. While "SW" works exactly like the "SW" program parameter statement (with appended misc. no.), "TS" has no variants, but always types out all 40 bits in hex.

<MIX>SW<SWITCH.ID><VALUE>

<MIX>TS

Run time accessing is a matter for negotiation between the compiler in question, and the corresponding interpreter.

SCHEDULE.PRIORITY

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

The format of the SCHEDULE.PRIORITY statement is:

$$[?] \left\{ \begin{array}{l} \underline{\text{SCHEDULE.PRIORITY}} \\ \underline{\text{SC}} \end{array} \right\} [=] \text{ integer}$$

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.

UNFREEZE

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:

[?] { UNFREEZE }
 { UN }

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:

$$[?] \left\{ \frac{\text{VIRTUAL.DISK}}{\text{VI}} \right\} [=] \text{integer}$$

Integer must be eight digits or less.

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

DATA

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:

$$[?] \left\{ \begin{array}{l} \underline{\text{DATA}} \\ \underline{\text{DA}} \end{array} \right\} \text{file-identifier}$$

The control word DATA can be abbreviated as DA.

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

The file-identifier must be the same as the file-identifier assigned in the program or be file-equated to it in order for the MCP to automatically find the file.

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

The END control statement cannot be abbreviated.

When the END statement is used it must be the last card in that file. It signals the MCP to close the file, and makes the card reader available to the system.

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 will hold the card reader waiting for more data or a “? END” statement to be 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.

KEYBOARD INPUT MESSAGES

GENERAL

Information may be supplied to the MCP through the use of input messages entered through the console printer. These messages are referred to as keyboard input messages throughout this manual. The keyboard input messages are used by the system operator to communicate with the MCP. In order to make the operating system an effective and informative tool, the system operator should be familiar with all the keyboard input messages.

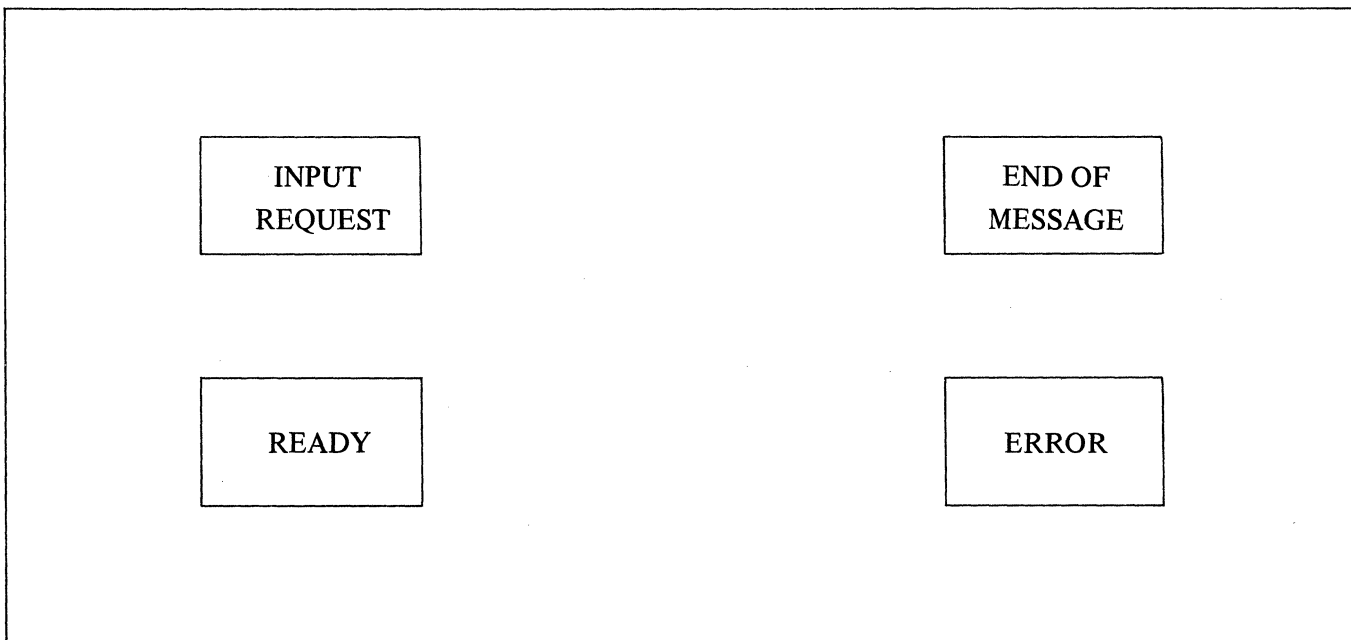
Keyboard input messages may be entered through a card reader by using the “?” or an invalid character in column one (1), followed by the input message. The last eight columns will be ignored as in a control card.

Keyboard Entry Procedure:

- a. Press INPUT REQUEST button.
- b. Wait for the READY indicator to light.
- c. Type in message.
- d. Depress END OF MESSAGE button (ETX) to terminate message.

If there are errors, press the ERROR button and retype the message. The MCP will print an exclamation point (!) at the end of error lines for ease of identification.

The (<) less than sign may also be used for error correction. See paragraph (d.) of MCP Communications.



Keyboard Input Messages

AX	(Response to ACCEPT)	LD	(Pseudo Load)
BF	(Display Backup Files)	{ LG LN	(Transfer/Print LOG)
CD	(List Card Decks)		
CE	(Change to Entry System Software)	LT	(Light Valid Types for FT)
CM	(Change MCP)	MC	(Assign System Charge Number)
CN	(Change to Non-Trace System Software)	MP	(List MPF Tables)
CP	(Compute)	MR	(Close Output File with Purge)
CQ	(Clear Queue)	MX	(Display MIX list)
CS	(Change to Standard System Software)	OF	(Optional File Response)
CT	(Change to Trace System Software)	OK	(Attempt to Continue Processing)
DM	(Dump Memory and Continue)	OL	(Display Peripheral Status)
DP	(Dump Memory and Discontinue)	OU	(Specify Output Device)
{ DR DT	(Change MCP Date)	PB	(Print/Punch Backup)
		PD	(Print Directory)
DS	(Discontinue Program)	PG	(Purge)
ED	(Execute Pseudo Deck)	PM	(Print Memory)
FM	(Response to Special Forms)	PO	(Power Off)
FN	(Display Internal File Names)	PR	(Change Priority)
FR	(Final Reel of Unlabeled Tape File)	PS	(PROD Schedule)
FS	(Force from Schedule)	{ RB RF	(Remove Backup Files)
FT	(Change File Type)		
GO	(Resume Stopped Program)	RD	(Remove Pseudo Card Files)
HS	(Hold in Waiting Schedule)	RL	(Relabel User Pack)
HW	(Hold in Waiting Schedule until Job EOJ)	RM	(Remove Duplicate Disk File)
IL	(Ignore Label)	RN	(Assign Pseudo Readers)
JM	(Jiggle Memory)	RO	(Reset Option)
KA	(Disk Analyzer)	RP	(Ready and Purge)
KP	(Print Disk Sements)	RS	(Remove Jobs from Schedule)
LC	(Load Cassette)	RT	(Remove MPF Table)

Keyboard Input Messages

RY	(Ready Peripheral)	TO	(Display Options)
SD	(Assign Additional System Drives)	TR	(Time Change)
SL	(Set LOG)	UL	(Assign Unlabeled File)
SO	(Set Option)	WD	(Display MCP Date)
SP	(Change Schedule Priority)	WM	(Display Current MCP and Interpreter)
ST	(Suspend Processing)	WS	(Display Schedule)
SV	(Save Peripheral Units)	WT	(Display MCP Time)
TD	(Display Time/Date)	WW	(List Contents of Name Table)
TI	(Time Interrogation)	WY	(Program Status Interrogation)
TL	(Transfer LOG)		

AX INPUT MESSAGE (Response to an ACCEPT Message)

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.

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 will be padded with trailing blanks. Longer messages will be truncated on the right.

BF

BF INPUT MESSAGE (Display Backup Files)

The BF input message lists disk backup files on the console printer.

The format of the BF message is:

<p><u>BF</u> [unit-mnemonic] <i>pack-id/</i></p>	$\left\{ \begin{array}{l} \text{PRN/=} \\ \text{PRT/=} \\ \text{PCH/=} \\ \text{=/=} \end{array} \right\}$
---	--

The PRT/= option will list all printer backup files on disk. The PCH/= option will list all punch backup files on disk.

The =/= option will list both the printer and punch backup 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 unit-mnemonic requests displaying the backup files on the designated removable disk drive. If it is omitted, the MCP will display the backup files resident on system disk.

Burroughs Corporation

INTER-OFFICE CORRESPONDENCE

CORPORATE UNIT Computer Systems Group	LOCATION Santa Barbara Plant	DEPT. Software Activity
NAME Software Activity Personnel		DATE 12 August 1974
FROM J. Chelini	DEPT. & LOCATION Operating Systems Development	

SUBJECT:

C.C.

NEW CONTROL MESSAGE

With MCP11 4.1 dated 8/1/74 or later a new message "BD" has been implemented.

Syntax 'BD <DIRECTORY NAME>'

DIRECTORY NAME := <PACK.ID> | <CARTRIDGE.ID> |
<SYSTEM.ID>

PACK.ID := NAME (10 CHARACTERS MAX)
CARTRIDGE := NAME (10 CHARACTERS MAX)
SYSTEM.ID := "blank"
blank := 1 to 10 blank characters

The purpose is to change the default disk backup device from the system unit to some other unit, or the converse. 'BD USERA' will cause all backup disk files to reside on that cartridge or pack named 'USERA'. This can be overridden at any time in the file statement (PACK.ID = NAME) or OU message. 'BD " "' will reset default to system disk. Since the backup default is changeable,

BF { =/= }
RB { PRT/= } will act on the default for the directory name
PB { PCH/= }
 { <number> }

BF <PACK.ID> / { =/= } will behave in the normal fashion,
RB { PRT/= } directly interrogating the stated
 and { PCH/= } directory.
PB <unit> { <number> }

J. Chelini

J. Chelini
Operating Systems Section
Operating Systems Development

gp

BB

BB INPUT MESSAGE (Set BACKUP BLOCKS per AREA)

The BB input message allows the user to specify the number of BLOCKS to assign to each AREA of a PRINTER or PUNCH BACKUP DISK file.

The BB message format is:

BB <INTEGER>

Each BLOCK is 900 bytes (5 segments) long, and a BACKUP file is always assigned 25 AREAS. 200 BLOCKS (1000 Segments) per AREA is assigned by COLDSTART as the default BB value.

BD INPUT MESSAGE (Set BACKUP DESIGNATION)

The BD input message allows the system operator to assign a default disk for backup files.

The BD message format is:

$BD \left\{ \begin{array}{l} \langle \text{PACK-ID} \rangle \\ \langle \text{" < 1 to 10 spaces >"} \rangle \end{array} \right\} (\text{SYSTEM DISK})$
--

Once set the BACKUP DESIGNATION may only be changed by another BD message or a COLDSTART. (IT will remain in effect after a CLEAR START). The SYSTEM DISK is assigned by COLDSTART as the default BACKUP DESIGNATION.

If the DESIGNATION is not the SYSTEM DISK a BF, RB, or RF message will generate the request as though it were a user version of a BF, RB, or RF message. A user version may be entered using the DESIGNATED PACK NAME.

To do a BF, RB, or RF of the SYSTEM DISK if the DESIGNATION is not the SYSTEM DISK the DESIGNATION must be set to the SYSTEM DISK. Suggest the use of PD or RE instead.

BT

BT INPUT MESSAGE (Back Track)

The BT input message allows the system operator to change a MK IV.1 code file back to a MK IV.0 code file.

The BT message format is:

BT <file-name>

Once a program is executed under MK IV.1 MCP control, its Program Parameter Block is modified. The BT message will allow the program to be executed again under control of MK IV.0 MCP.

CD INPUT MESSAGE (Lists Card Decks in Pseudo Readers)

The CD 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

The MCP displays the number of each pseudo deck 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.

CE

CE INPUT MESSAGE (Change to Entry (MCP I) System Software/Firmware)

The CE input message allows the operator to specify that during the next Clear/Start MCP I system software and firmware will be loaded on the system.

The format of the CE message is:

CE

CH

CHANGE

See page 2-11

2-46A

CL

CL INPUT MESSAGE (CLEAR UNIT)

The CL input message allows the system operator to "clear" a unit whose control may be hung.

The CL message format is:

CL <unit-mnemonic>

Any program using that unit will be DSed.

CM INPUT MESSAGE (Change System Software)

The CM message is used to change the MCP in control of the system.

The format of the CM message is:

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

The change will actually take place after the next Clear/Start.

See the Clear/Start procedure for an explanation of the CM system software-mnemonics used in the NAME TABLE.

CN

CN INPUT MESSAGE (Change to Non-Trace System Software)

The CN input message allows the system operator to change the operating environment to non-trace system software after the next Clear/Start.

The format of the CN message is:

CN

CAUTION

The CN input message is strictly for system software development and debugging. It should not be used in the standard operating environment.

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 operand-1 [operator operand-2] ... [=]
```

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

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

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

The CP message will calculate 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.

The response is displayed in both decimal and hexadecimal formats.

When a hexadecimal number is to be used in a calculation, it must be enclosed by @ signs. The valid hexadecimal digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. If A, B, C, D, E, or F, is entered without @ signs, the message is invalid.

Example:

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

response: CP: @0000F7@=247

CP @F@

CP: @000000@=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

CS INPUT MESSAGE (Change to Standard (MCP II) System Software/Firmware)

The CS input message allows the system operator to insure that during the next Clear/Start MCP II system software and firmware will be loaded on the system.

The format of the CS message is:

CS

CT

CT INPUT MESSAGE (Change to Trace System Software)

The CT input message allows the system operator to change the operating environment to trace system software after the next Clear/Start.

The format of the CT message is:

CT

CAUTION

The CT input message is strictly for system software development and debugging. It should not be used in the standard operating environment.

DD

DD INPUT MESSAGE (DATA MANAGEMENT DUMP)

The DD input message allows the system operator to cause a runtime trace of ~~active~~ DATA MANAGEMENT routines.

The DD message format is:

DD <option>

<option> Specifies which routines to trace. "DD WHAT" will give a list of options

This SERVICE is only available in a DEBUG MCP.

PROPRIETARY DATA

The information contained in this document is proprietary to Burroughs Corporation. The information of this document is not to be shown, reproduced, or disclosed outside Burroughs Corporation without written permission of the Patent Division.

THIS DOCUMENT IS THE PROPERTY OF AND SHALL BE RETURNED TO BURROUGHS CORPORATION, BURROUGHS PLACE, DETROIT, MICHIGAN 48232.

DF

DF INPUT MESSAGE (Date of File)

The DF input message allows the operator to obtain (printed on the SPO) the COMPILATION DATE and TIME (for CODE and INTREPETER files) or the CREATION DATE (for all other files).

The DF message format is:

Format 1

DF [dp-id/] =/=

Format 2

DF [dp-id/] <first.name> { /=
/<second.name> }

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 DM message format 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

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

$\left\{ \begin{array}{c} \underline{DR} \\ \underline{DT} \end{array} \right\}$ INPUT MESSAGE (Change MCP Date)

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

The DR message format is

$\left\{ \begin{array}{c} \underline{DR} \\ \underline{DT} \end{array} \right\}$ 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

DS INPUT MESSAGE (Discontinue Program)

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

The DS message format is:

```
mix-index  DS
```

The DS message can 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.

DY

NAMIC

See page 2-16

ED INPUT MESSAGE (Execute Pseudo Deck)

The ED input message will cause a specified pseudo deck to be executed.

The format for the ED message is:

ED integer

If a pseudo reader is not available, a new reader will be allocated for that deck.

When the deck has been processed the pseudo reader will be de-allocated.

EM

EM INPUT MESSAGE (ENTER MESSAGE TO ELOG)

The EM message allows the FIELD ENGINEER to enter a message in the Engineering LOG.

The EM message format is:

EM <any line of text>

PROPRIETARY DATA

The information contained in this document is proprietary to Burroughs Corporation. The information in this document is not to be shown, reproduced, or disclosed outside Burroughs Corporation without written permission of the Patent Division.
THIS DOCUMENT IS THE PROPERTY OF AND SHALL BE RETURNED TO BURROUGHS CORPORATION, BURROUGHS PLACE, DETROIT, MICHIGAN 48232.

ET

ET INPUT MESSAGE (ELOG TRANSFER)

The ET message allows the FIELD ENGINEER to transfer all information in the SYSTEM/ELOG to a unique file named ELOG /# <INTEGER> and will then print the LOG.

The ET message format is:

ET

PROPRIETARY DATA

The information contained in this document is proprietary to Burroughs Corporation. The information in this document is not to be shown, reproduced, or disclosed outside Burroughs Corporation without written permission of the Patent Division.

THIS DOCUMENT IS THE PROPERTY OF AND SHALL BE RETURNED TO BURROUGHS CORPORATION, BURROUGHS PLACE, DETROIT, MICHIGAN 48232.

FD

FD INPUT MESSAGE (FAST DUMP)

The FD input message allows the system operator to look at the current status of a program.

The FD message format is:

<MX> FD

Several register settings and program parameters will be displayed on the SPO.

FM

FM INPUT MESSAGE (Response to a SPECIAL FORMS REQUIRED)

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

The FM message format 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.

Example:

3 FM LPA

FN INPUT MESSAGE (Display Internal File Name(s))

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

The format of an FN input message is:

FN program-name external-file-identifier

The MCP will list 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

FR INPUT MESSAGE (Final Reel of an 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 an FR message is:

mix-index FR

The FR message is a response to the message:

mix-index NO FILE

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.

FS INPUT MESSAGE (Force from Schedule)

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

The format for a FS message is:

```
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.

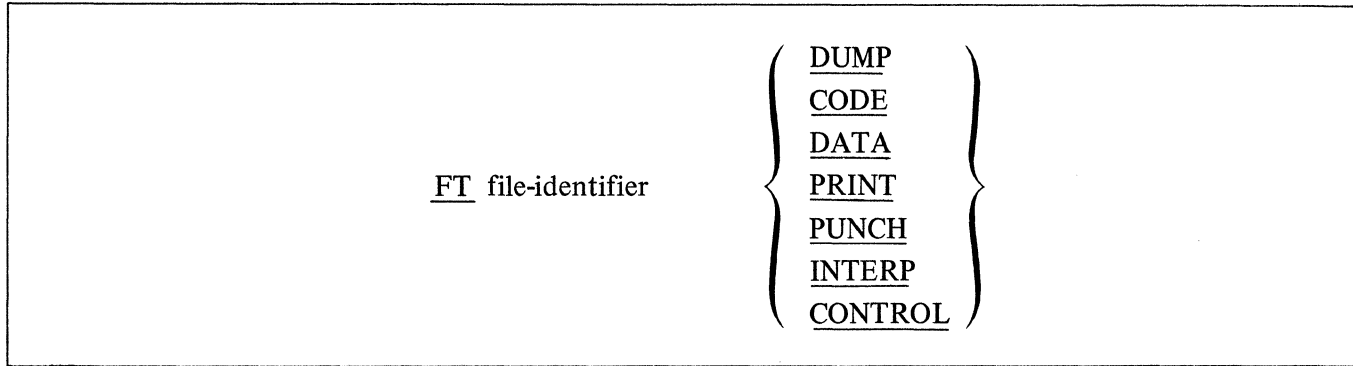
In order for a program to be in the mix, it must have gone to BOJ.

FT

FT INPUT MESSAGE (Change File Type)

The FT input message allows the operator to change the type of a disk file in the disk directory and file header.

The format of the FT message is:



By using the FT message the file type is the only change made to the file; the format of the file remains the same.

A CONTROL type file is a pseudo file or control deck.

A CODE file is the only type of file that an EXECUTE, MODIFY, COMPILE, or DYNAMIC statement may be valid as an operation.

FX

FX INPUT MESSAGE (Fast dump explanation)

The FX input message will cause an explanation of the abbreviations used in the FAST DUMP to be printed on the SPO.

The FX message format is:

FX

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 a 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.

GT

GT INPUT MESSAGE (GO TRACE)

The GT input message allows the system operator to request that a program's interpreter trace each opcode specified by the <INTEGER>.

The GT message format is:

<MX> GT <INTEGER>

<INTEGER> CAN BE

- 0 NO TRACE
- 1 TRACE BRANCH STATEMENTS
- 2 TRACE STORE STATEMENTS
- 3 TRACE ALL OTHER STATEMENTS

or the sum of any of the above.

PROPRIETARY DATA

The information contained in this document is proprietary to Burroughs Corporation. The information in this document is not to be shown, reproduced, or disclosed outside Burroughs Corporation without written permission of the Patent Division. THIS DOCUMENT IS THE PROPERTY OF AND SHALL BE RETURNED TO BURROUGHS CORPORATION, BURROUGHS PLACE, DETROIT, MICHIGAN 48232.

2-63A

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.

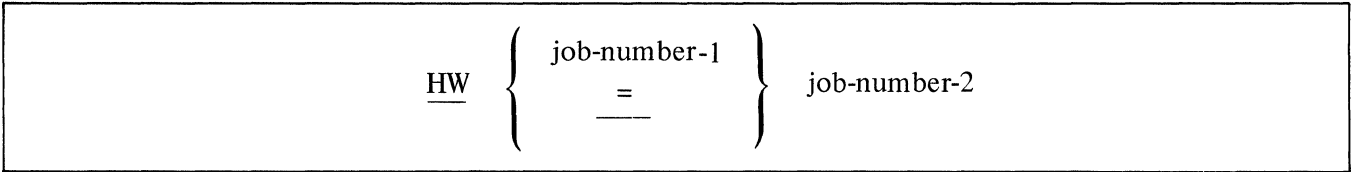
A job-number is assigned when a program is scheduled by the MCP.

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:



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 IL message format is:

mix-index IL unit-mnemonic

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.

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.

JM INPUT MESSAGE (Jiggle Memory)

The JM input message gives the operator the ability to try to get a job(s) running when memory is in a "checkerboard" condition.

The format of the JM message is:

JM

The JM message causes the MCP to attempt to reinstate all programs waiting for memory.

"Checkerboard" is a term used when programs have been rolled out leaving run structures scattered throughout memory, thus not allowing enough contiguous memory space for them to be rolled back in.

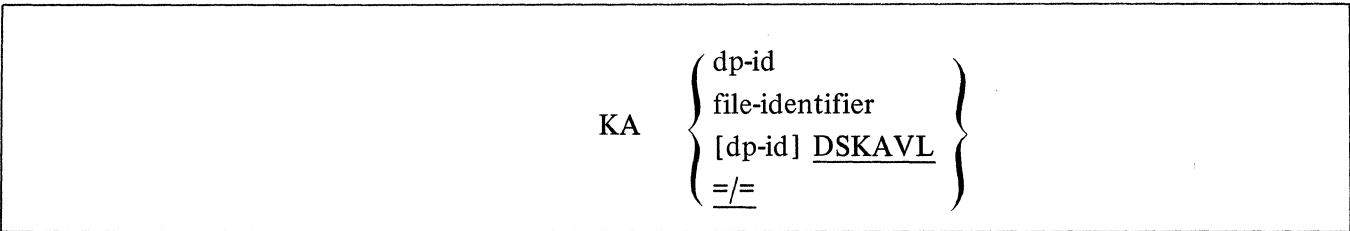
The JM message does not rearrange memory or alter it in any way.

KA

KA INPUT MESSAGE (Disk Analyzer)

The KA input message provides the system operator the means to analyze a disk directory's contents and the file area assignments.

The format for the KA message is:



The KA message prints a list of the disk areas available to be used, followed by a description of each file in the directory.

When the file-identifier is used with the KA, only the information concerning that file is printed.

The DSKAVL will list the available areas on the disk.

The =/= option lists all files and disk available space.

The dp-id option is used to obtain a disk directory on a removable disk pack.

Examples:

KA

KA DSKAVL

KA dp-id

KA file-identifier

KC

KC INPUT MESSAGE (Print Selected Disk Segments in Character)

The KC message function and format are the same as KP except that the disk is printed in character format rather than hexadecimal.

KI

KI INPUT MESSAGE (Read Disk in Hexadecimal)

PROPRIETARY DATA

The information contained in this document is proprietary to Burroughs Corporation. The information of this document is not to be shown, reproduced, or disclosed outside Burroughs Corporation without written permission of the Patent Division.

THIS DOCUMENT IS THE PROPERTY OF AND SHALL BE RETURNED TO BURROUGHS CORPORATION, BURROUGHS PLACE, DETROIT, MICHIGAN 48232.

2-68B

KO

KO INPUT MESSAGE (Write Disk in Hexidecimal)

PROPRIETARY DATA

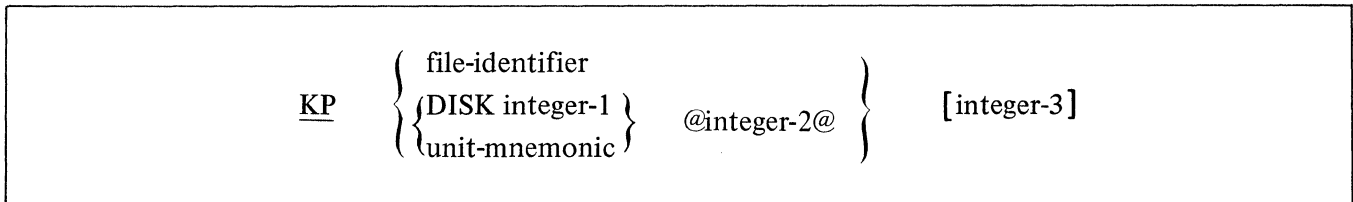
The information contained in this document is proprietary to Burroughs Corporation. The information or this document is not to be shown, reproduced, or disclosed outside Burroughs Corporation without written permission of the Patent Division. THIS DOCUMENT IS THE PROPERTY OF AND SHALL BE RETURNED TO BURROUGHS CORPORATION, BURROUGHS PLACE, DETROIT, MICHIGAN 48232.

2-6BC

KP INPUT MESSAGE (Print Selected Disk Segments)

The 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 KP message is:



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

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

Integer-2 is used to specify the disk address from which printing is to begin, and must be entered in hexadecimal format.

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 DISK 1 @ 200 @ 10 Print 10 segments on EU 1 from HEX LOC 200

LC

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 LC message format is:

LC [dp-id] [number of files]

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

The LC message will only load the first file on the cassette when the number of files option is omitted. Additional files may be loaded by re-entering the LC message every time the "file-id LOADED" message is displayed.

If the file that is needed is embedded within the cassette, all files up to and including the desired file must be loaded. In other words, there is no selective file loading other than the first file on the cassette.

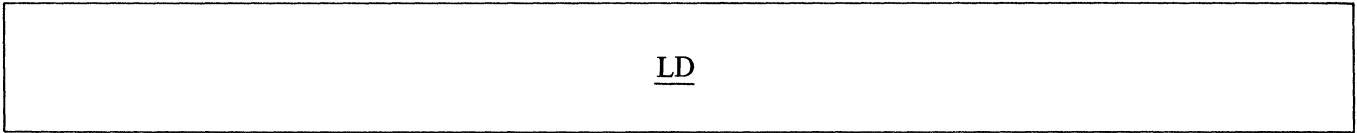
The LC message may be used to load a file that resides on more than one cassette. This is called a multiple cassette file. The following steps should be taken to load a multiple cassette file:

1. Place the first cassette into the cassette reader. (Cassette automatically rewinds.)
2. Enter the LC message.
3. When the first reel of the cassette finishes loading, the MCP will HALT. (RUN light goes off, L-register = @11@ and the T-register = @C3C1E2@.)
4. Load next cassette and press START. (DO NOT PRESS CLEAR.)
5. Repeat procedures 3 and 4 till the message "file-id LOADED" is displayed signifying End-of-Load.

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:



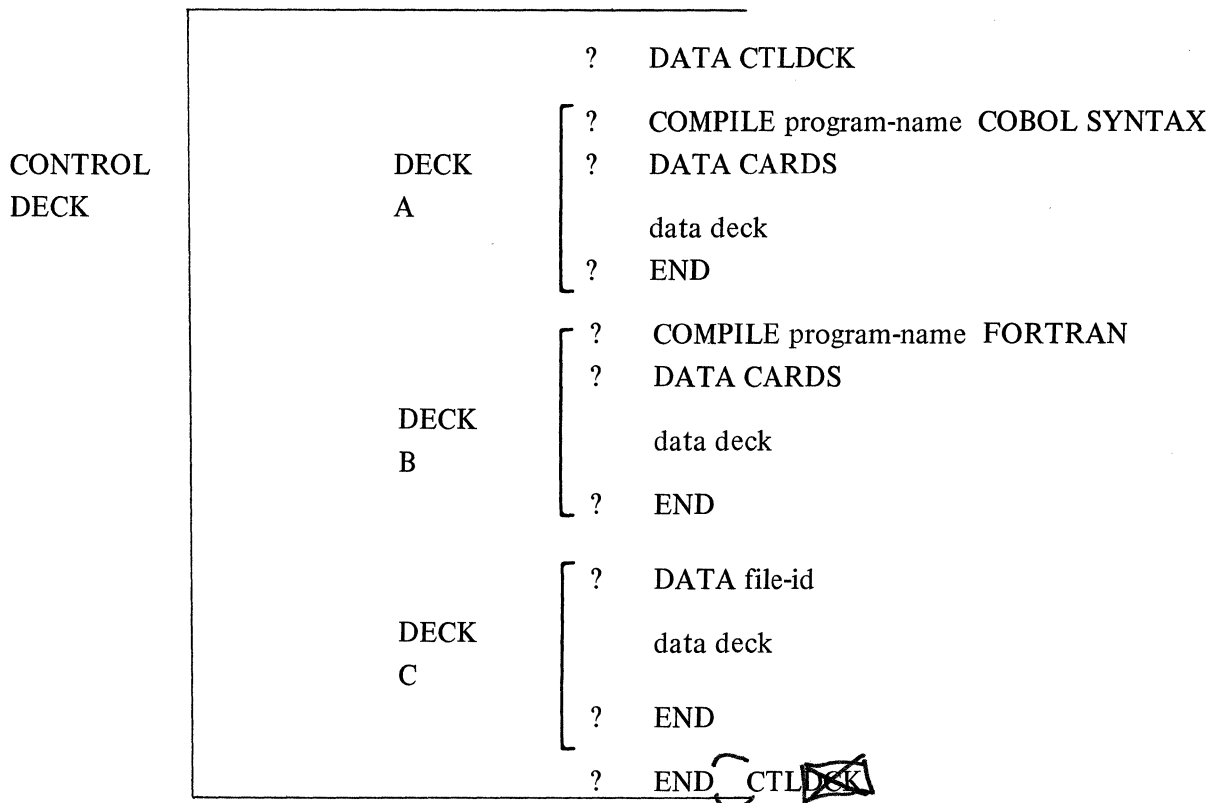
After receiving a LD message, the program, SYSTEM/LDCONTRL, looks for a “? DATA CTLDCK” control statement that initiates the read.

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 “? END CTLDCK” 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.



LG
LN

{ LG } INPUT MESSAGE (Transfer and Print LOG)
{ LN }

The LG or LN message permits the system operator to transfer the file SYSTEM/LOG to the file SYSTEM/LOG1 and print the LOG in a readable format.

The LG, LN message format is:

{ LG }
{ LN }

The SYSTEM/LOG1 is printed by the program SYSTEM/LOGOUT. SYSTEM/LOGOUT must be in the disk directory in order for this message to be accepted by the MCP.

LT INPUT MESSAGE (List File Types)

The LT input message will list the valid file types able to be changed by the FT message.

The format of the LT message is:

LT

MO

MODIFY

See page 2-18

MP

MP INPUT MESSAGE (List Multi-Pack File Tables Resident on System Disk)

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 INPUT MESSAGE (Close Output File with PURGE for Duplicate File Situations)

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

MX INPUT MESSAGE (Display MIX list)

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 MX message format 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 PR:04

program-name = 2 PR:05

END MX

NT

NT INPUT MESSAGE (program No Trace)

The NT input message allows the system operator to request a program's interpreter to stop tracing and close the trace file.

The NT message format is:

<MX> NT

The NT message is different from the GTO message in that the GTO message will not close the trace file. The system operator may request a trace again after the NT message has closed the trace with a GT message which will open the trace file.

PROPRIETARY DATA

The information contained in this document is proprietary to Burroughs Corporation. The information in this document is not to be shown, reproduced, or disclosed outside Burroughs Corporation without written permission of the Patent Division.

THIS DOCUMENT IS THE PROPERTY OF AND SHALL BE RETURNED TO BURROUGHS CORPORATION, BURROUGHS PLACE, DETROIT, MICHIGAN 48232.

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 optional file being requested is to be bypassed for this execution. Usage is restricted for input files that have been declared or label-equated as OPTIONAL.

OK

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 OK message format 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 INPUT MESSAGE (Display Peripheral Status)

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

The OL message format is:

```
      OL      { 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.

The following responses are generated:

- unit-mnemonic IN USE BY program-name file name
- unit-mnemonic LABELED file-name
- unit-mnemonic NOT READY
- unit-mnemonic UNLABELED

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

NULL unit-type-code TABLE

OU

OU INPUT MESSAGE (Specify Output Device)

The OU input message is a response to direct an output file to a specified output device. Within limits, the OU message may be used to change the hardware medium for a file.

The OU message format 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.

(pb control statement) ::= PB (input designator) (output unit) (option part)

(input designator) ::= (unit mnemonic) (pb number) / ~~(pb number)~~

(unit mnemonic) ::= (unit name) (unit) / (empty)

(unit name) ::= MT / DC / DP / DK

(unit) ::= A / B / C ...

(pb number) ::= (integer)

(output unit) ::= (empty) / LP(unit) / CP(unit) / CD(unit)

(option part) ::= (empty) / (associative option) / (general option) / (general option) (associative option)

(associative option) ::= (key declaration) (area specifier) / RECORD (record start) / RECORD (record start) (record stop)

(key declaration) ::= KEY (key specifier)

(key specifier) ::= COBOL / FORTRAN / RPG / BASIC / SDL / MIL / (column) (key length)

(column) ::= (integer) base 1 relative

(key length) ::= (integer)

(area specifier) ::= RANGE (start) (stop) / EQUAL (sequence)

(start) ::= "(ebcdic string)" / (integer) base 1 relative

(stop) ::= "(ebcdic string)" / (integer)

(sequence) ::= "(ebcdic string)" / (integer)

(record start) ::= (integer) base 1 relative

(record stop) ::= (integer)

(general option) ::= (general option part) / (general option) (general option part)

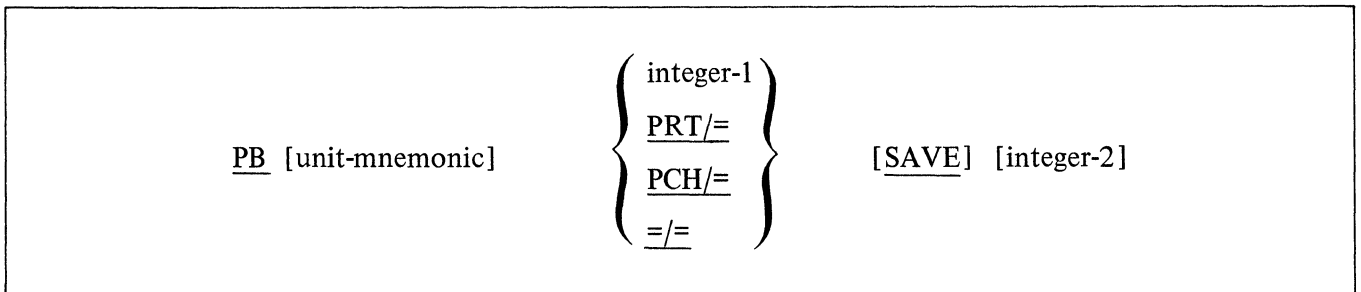
(general option part) ::= SAVE / COPIES (integer) / COPIES= (integer) / SINGLE / DOUBLE

PE = /
 PD (PB) = /
 PP = /
 PE (PB) = /

PB INPUT MESSAGE (Print/Punch Backup)

The PB input message permits the system operator to print and/or punch backup files.

The format of the PB message is:



The “integer-1” option is the number given to the file by the MCP when the backup was performed and is used to specify a single file for printing or punching.

The “PRT/=” and “PCH/=” options will either print or punch all printer or punch backup files on disk.

The “=/=” option will both print and/or punch all backup files on disk.

The “SAVE” option will prohibit the purging of the file(s) at close time.

The “integer-2” option is a counter to tell the MCP the number of copies of each file to be printed or punched for output. If this option is omitted, one (1) is assumed.

The unit-mnemonic option directs the MCP to a specific removable disk drive or magnetic tape unit.

PD

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:

<u>Format 1</u>	$\left\{ \begin{array}{l} \text{dp-id}/=/= \\ \underline{=}/= \end{array} \right\}$	(removable pack) (system pack)
<u>Format 2</u>	$\left\{ \begin{array}{l} \text{file-identifier} \\ \text{family-name}/= \\ \text{dp-id}/\text{family-name}/= \end{array} \right\} \dots$	

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

PG INPUT MESSAGE (Purge)

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

The message 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.

Example:

PG DPA 000456

PM INPUT MESSAGE (Print Memory)

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:

PM [integer [SAVE]]

A PM by itself will cause the execution of the MCPI/ANALYZER or MCPPI/ANALYZER program which will analyze and print the contents of SYSTEM/DUMPFIL. (System Memory)

The "integer" option will cause the execution of the DUMP/ANALYZER program which will analyze and print the contents of DUMPFIL/integer. (Program Memory)

The programs DUMP/ANALYZER and either MCPI/ANALYZER (MCPI) or MCPPI/ANALYZER (MCPPI) must be located on systems disk to perform a PM message.

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

PO

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 PO message format 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 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 PR message format is:

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

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

PS

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; however the PS message will cause the MCP to check the ACTIVE SCHEDULE when the message is entered.

PZ

PZ INPUT MESSAGE (Page size)

The PZ input message allows the system operator to specify the minimum amount of control memory which will be allocated to an interpreter when the sum of the active interpreters minimum requests exceeds the maximum amount of available control memory (contention mode).

The PZ message format is:

PZ <integer>

<Integer> is the minimum assignment in KBYTES.

Once set the PAGE SIZE may only be changed by another

PZ message or a COLDSTART. 2 K BYTES is assigned by COLDSTART as the default PAGE SIZE.

In contention mode the interpreter will be allocated at least the larger of the PAGE SIZE or the minimum requested by the interpreter.

PROPRIETARY DATA

The information contained in this document is proprietary to Burroughs Corporation. The information in this document is not to be shown, reproduced, or disclosed outside Burroughs Corporation without written permission of the Patent Division.

THIS DOCUMENT IS THE PROPERTY OF AND SHALL BE RETURNED TO BURROUGHS CORPORATION, BURROUGHS PLACE, DETROIT, MICHIGAN 48232.

QC

QC INPUT MESSAGE (Quit network Controller)

The QC input message allows the system operator to bring the network controller to an orderly halt ("Quit the controller").

The QC message format is:

QC

The network controller will not honor any requests from terminals but will finish any output to be sent to the terminals before going to EOD.

QF, QP

The Program Parameter Parser of the Control Language Processor in the MCP has been extended slightly in the MCPII 4.1 version.

"CO", "EX", "BI", "MO", "DY" control statements permit modification of Program parameters. "QF" and "QP" (standing for, but not substitutable by "QUERYFILE" and "Query Program") permit interrogation of Program Parameters.

QF<CODE.FILE.NAME><PROGRAM.PARAMETER.INTERROGATION.STATEMENT.LIST> will interrogate mother copy parameters.

QP<JOB.NUMBER><P.P.I.S.L> will interrogate working (log) copy parameters.

The <P.P.I.S.L> is much like the <P.P.MODIFICATION.S.L> employed in the five PROGRAM.PARAMETER.MODIFICATION.STATEMENTS, except: where a P.P.MODIFICATION was achieved by using a "NAME/VALUE" pair (e.g., INTERP=GLUMPH), the P.P.INTERROGATION will use only the NAME of the parameter. (e.g. "INTERP")

Where a P.P.MODIFICATION used only a value (e.g. "DISK"), the P.P.INTERROGATION used ANY POSSIBLE VALUE (e.g. "PAPER.TAPE.PUNCH").

Examples:

QF<FN> IN PR FIQ NAME DISK; ID INTERP

Will cause to be typed on the SPO (with appropriate preambles, etc.):

1. INTERP NAME
2. RUN TIME PRIORITY
3. FILE Q'S EXTERNAL NAME
4. FILE Q'S DEVICE TYPE
5. NAME OF INTRINSIC DIRECTORY
6. INTERP NAME (AGAIN)

as represented in the mother copy of code file "FN" on the disk.

"QP<JOB.NO>" will do the same thing but with respect to a "working copy" of program parameters associated with the particular instance of the CODE-File's execution.

Note:

Out of contest interrogations (e.g. "QFN LI") will cause the MCP to complain.

It makes sense to interrogate certain parameters which it does not make sense to modify. (e.g. "COMPILER NAME") hopefully the syntax to provide these will be added soon (but I doubt in 4.1).

QI

QI INPUT MESSAGE (Queue Insert)

The QI input message allows the system operator to insert a message into a queue.

The QI message format is:

QI <queue name> <message>

If no queue exists one will be created

QK

QK INPUT MESSAGE (Queue Kill)

The QK input message allows the system operator to remove a queue.

The QK message format is:

QK <queue name >

RE

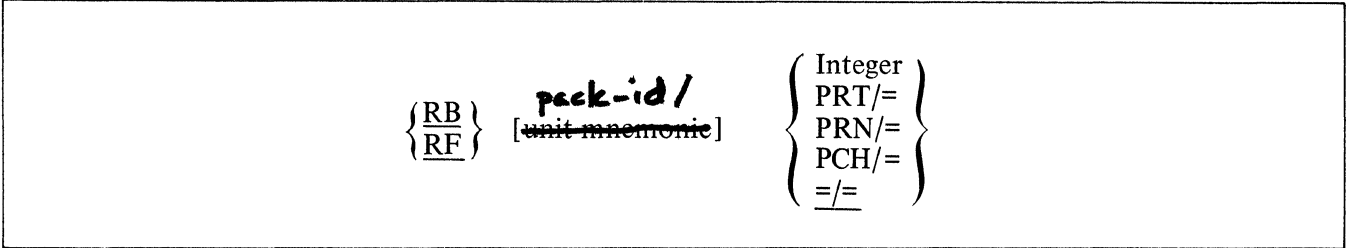
REMOVE

See Page 2-14

{RB}
{RF} INPUT MESSAGE. (Remove Backup Files)

The RB or RF input message gives the system operator the ability to remove backup files on disk.

The format of the RB, RF message is:



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

The PRT/= and PCH/= options will remove either all print backup files or all punch backup files respectively. PRN is equivalent to PRT.

The =/= option will remove all backup files from disk.

The unit-mnemonic option specifies that the backup files to be removed are on the designated removable disk.

RD

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:

$$\underline{\text{RD}} \left\{ \begin{array}{l} \text{integer} \\ \underline{=/=} \end{array} \right\}$$

RH

RH INPUT MESSAGE (Reset MCP conditional Haults)

The RH input message allows the system operator to reset all MCP conditional Haults.

The RH message format is:

RH

PROPRIETARY DATA

The information contained in this document is proprietary to Burroughs Corporation. The information in this document is not to be shown, reproduced, or disclosed outside Burroughs Corporation without written permission of the Patent Division.
THIS DOCUMENT IS THE PROPERTY OF AND SHALL BE RETURNED TO BURROUGHS CORPORATION, BURROUGHS PLACE, DETROIT, MICHIGAN 48232.

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 $\left\{ \begin{array}{c} \underline{R} \\ \underline{U} \end{array} \right\}$

RM

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 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:

<u>RN</u> 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

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 RO message format is:

```
RO option-name-1 [option-name]...
```

The MCP replies with a verification that the option has been reset after each RO input message.

Example:

request: RO EOJ

response: EOJ = 0

The LOG and CHRG options cannot be reset. The MCP message LOG LOCKED or CHRG LOCKED will be displayed when an attempt has been made to reset these options.

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.

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 [unit-mnemonic] . . .

The RP message can be used for tape only.

RS

RS INPUT MESSAGE. (Remove Job(s) 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:

$$\underline{\text{RS}} \left\{ \begin{array}{l} \text{job-number-1} \\ \underline{\quad} \end{array} \quad [\text{job-number-2}] \quad \dots \right\}$$

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 INPUT MESSAGE (Remove multi-pack file table from disk)

The RT input message allows the operator to remove an entry from the multi-pack file table on the system disk.

The format of the RT message is:

RT file-identifier

RY

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).

Example:

RY DPB	(ready second drive on system)
RY DPC	(ready third drive on system)
RY LPA	(ready line printer on system)

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 message format for the SD message is:

<u>SD</u> 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

SE INPUT MESSAGE (SWITCH ENABLE)

The SE input message permits the system operator to allow classes of programs to SENSE CONSOLE SWITCHES.

The SE message format is:

SE <integer>

Program classes are:

- 1 MCP
- 2 Normal State
- 4 Interpreters
- 8 CSM (Gismo)

or any sum of the above.

Applies only to 1720's.

SH

SH INPUT MESSAGE (Set MCP conditional Hault)

The SH input message allows the system operator to set a MCP conditional Hault.

The SH message format is :

SH <MCP sequence number>

If this mcp code is executed the system will Hault with
L = @000011@ and T = <MCP sequence number>.

PROPRIETARY DATA

The information contained in this document is proprietary to Burroughs Corporation. The information of this document is not to be shown, reproduced, or disclosed outside Burroughs Corporation without written permission of the Patent Division. THIS DOCUMENT IS THE PROPERTY OF AND SHALL BE RETURNED TO BURROUGHS CORPORATION, BURROUGHS PLACE, DETROIT, MICHIGAN 48232.

SL

SL INPUT MESSAGE (Set LOG)

The SL input message gives the operator the ability to set the LOG option, and allocate the area required.

The format of the SL message is:

SL integer-1 [integer-2]

The integer-1 entry is the size of each area to be assigned to the LOG and cannot be less than 100 or greater than 1000 disk segments.

The integer-2 option is the maximum number of these areas desired and must be between 1 and 105, inclusive. Default is 25.

The MCP will respond with the following message when an SL message has been entered.

LOG NOW SET—CLEAR START REQUIRED

or

NO SPACE TO BUILD LOG

SL 0 will cause the LOG option to be reset.

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 SO message format 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.

The LOG option cannot be set with an SO message. The MCP message "LOG LOCKED" will be displayed when an attempt has been made to set LOG with an SO message.

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 message 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

SR

SR INPUT MESSAGE (SECTOR RELOCATE)

The SR input message allows the system operator to relocate an assumed defective sector on the overflow track.

The SR message format is:

SR DP<x> <sector>

The sector will be verified to be defective then the sector will be placed on one of the overflow tracks. The SR message applies only to diskpack and not head per track disk or disk cartridge.

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 message format of the ST message 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, 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 GO message is issued, the MCP will not restart the program.

Example:

3 ST

SV

SV INPUT MESSAGE (Save Peripheral unit(s))

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 SV message format 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 (SWitch)

The SW input message allows the system operator to set the value of program switches.

The SW message format is:

$\langle MX \rangle SW \left\{ \begin{array}{l} \langle \text{switch.id} \rangle [=] \langle 4 \text{ bit assignment} \rangle \\ = \langle 40 \text{ bit assignment} \rangle \end{array} \right\}$
--

$\langle \text{switch.id} \rangle$ may be 0-9.

If no $\langle \text{switch.id} \rangle$ is included the entire 10 switches are set to the given value. See SWITCHES page 2-34A

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 TD message format 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 message format for 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 message permits the system operator to transfer all information in the SYSTEM/LOG to the file SYSTEM/LOG1. The TL message does not print the LOG. (See LG message.)

The TL message format is:

TL

TO

TO INPUT MESSAGE (Display Options)

The TO input message allows the system operator to interrogate the status of the MCP options.

The message format of the TO input message is:

TO [option-name] . . .

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

LOG = 1

or:

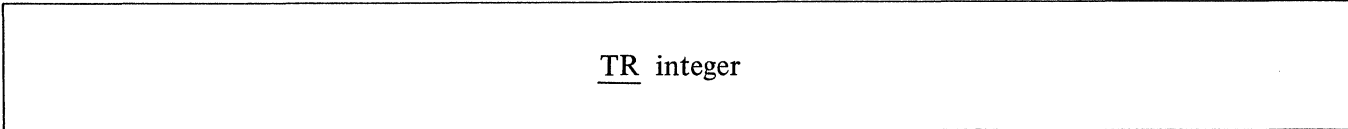
TO

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 message format of the TR message is:



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 (TYPE SWITCHES)

The TS input message will cause all 40 bits of the programs switches to be typed on the SPO in hexadecimal.

The TS message format is:

`<MX>TS`

See SWITCHES page 2-34A.

UL

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 for 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 mix-index must be used to identify the program to which the file is to be assigned.

If integer is used, it must not have a value greater than 99. When this option is used, the MCP spaces forward "integer" blocks or until a tape mark is read prior to reading the first data block into the object program. 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 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

WM INPUT MESSAGE (Display Current MCP and Interpreter)

The WM input message allows the system operator to inquire which MCP and Interpreter are currently being used since there can be more than one MCP and Interpreter residing on the system pack.

The format for the WM message is:

WM

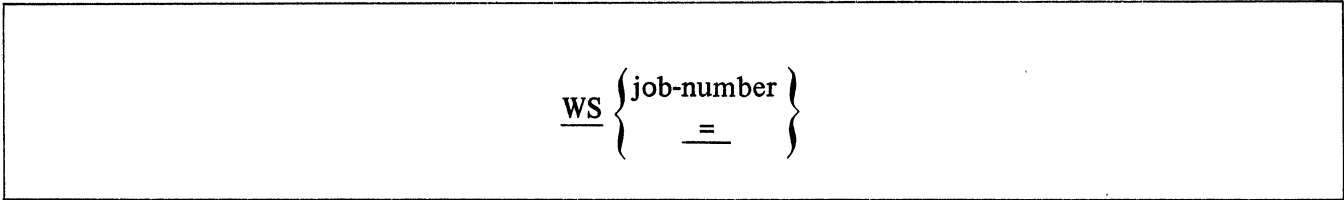
The reply to the WM message is in the following format:

MCP = mcp-name INTERP = interpreter-name GISMO = gismo-name INIT = initializer-name

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:



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

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.

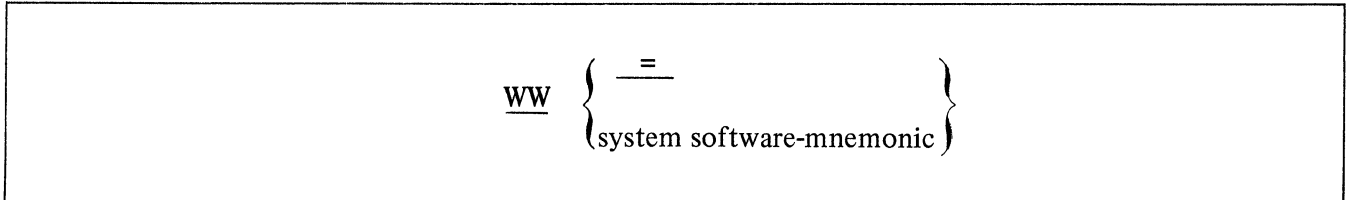
The WT input message format is:

WT

WW INPUT MESSAGE (What entries are in the 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:



See the Clear/Start procedure for an explanation of the system software-mnemonics used in the NAME TABLE.

WY

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 program status of the entire MIX.

The MCP response to the WY message is:

program-name = mix-index . . . status message . . .

Example:

1 WY

ALPHA = 01 WAITING OPERATOR ACTION

WY

ALPHA = 01 WAITING OPERATOR ACTION

BETA = 02 EXECUTING

DELTA = 03 AX-WAITING FOR KEYBOARD INPUT

ECHO = 04 IN COMMUNICATE QUEUE

CORPORATE UNIT Computer Systems Group	LOCATION Santa Barbara Plant	DEPT. Operating Systems
NAME G. Hammond		DATE 8 January 1974
FROM P. Woo	DEPT. & LOCATION Operating Systems Development	
SUBJECT:		C.C.

KEYBOARD INPUT MESSAGES XCARD XD

Format: { XC } { Dm }
 { XD } { DISK n } <segment-address> <number-of-segments>

where: m may be any alphabet so that Dm is the unit-mnemonic of some pack or cartridge.

n may be a number <16, designating a specific head-to-track disk.

<segment-address> is the address of a disk segment, expressed as a number, usually in hex form.

<number-of-segments> is a number.

Function: A minimum number of tracks, including all the specified number of disk segments beginning from the specified segment address, are assumed defective and to be deleted from the Working Available Directory on the disk. If the tracks are not entirely contained in one available block, the function will not be performed, but the console will display "Requested Segments not entirely in Available."

After deleting the tracks XC will enter them into Temporary Directory on the disk, so that they may automatically be restored to the Working Available Directory by the next clear start. XD differs from XC in that the deleted tracks are not entered into Temporary Directory, but rather deleted also from the Master Available Directory on the disk. Such tracks cannot be recovered unless the disk is reinitialized.

When to Use: When uncorrectable disk read or write occurs, the affected tracks can be disabled from all future use by XC or XD. However, if they contain useful data, such data must be discarded, or regenerated by other means. (For example, a whole file may be deleted from the disk). When the tracks' space is returned to Working Available Directory, then XC or XD may be entered.

Implementation: XD and XC are two lex.level 0 procedures in the MCP which calls the procedures SCANNER and GITNUMBER to scan and verify the input parameters. If the parameters are meaningful, then the procedure REMOVE.DISK is called. REMOVE.DISK may call either one of two lex.level zero procedures which may be useful for many other purposes: DELETE.ENTRY.IN.AVL.DIRECTORY and INSERT.ENTRY.IN.AVL.DIRECTORY.

G. Hammond
8 January 1974
Page 2

DELETE.ENTRY.IN.AVL.DIRECTORY needs two formal parameters: segment address of a disk segment structured like a typical available directory segment, and an entry number ≤ 22 specifying the particular entry in the segment to be deleted. After deletion, the gap will be shrunk by pulling up subsequent entries, even chasing down the chain of segments via the pointer AVL.FOR.LINK.

INSERT.ENTRY.IN.AVL.DIRECTORY needs one more parameter besides the above two, namely, the 60 bit entry itself to be inserted. The entry number may range from 0 to 22 indicating an entry in front of which will the new entry be inserted. Insertion causes bumps on all the subsequent entries. The last entry (the 22nd) in the segment will overflow to become the 1st entry of the next segment (chained by AVL.FOR.LINK), and so on, even to the extent of invoking GET.DISK to build a new available segment, if necessary.



P. Woo
Operating Systems Section
Operating Systems Development

gp

2-116B

MCP OUTPUT MESSAGES

GENERAL

The MCP communicates to the system operator via 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 messages are indented one space by the MCP, in order for the operator to easily distinguish them from input messages.

Numbers enclosed in at signs (@) indicate hexadecimal; all others are decimal.

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:

[compiler-name:] program-name = mix-index . . .

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. 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-info message is:

: NXT INSTR(HEX) = SEG @nnn@ DISP @nnnnnn@ (nnnn) (nnnn) DS OR DP

The entries in parentheses are the SEG (Segment) and DISP (Displacement) in decimal format.

MCP MESSAGES

job-specifier-ABORTED

job specifier-ACCEPT

job-specifier-ACCESS PPB TARGET OUT OF RANGE terminal-reference

job-specifier-ATTEMPT TO READ UNASSIGNED DISK AREA ON file-identifier

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— $\left. \begin{array}{l} \text{BOJ.} \\ \text{EOJ.} \\ \text{DS-ED.} \end{array} \right\} = \text{job-number PR} = \text{nn} [\text{integer SYNTAX ERRORS}] \text{ TIME} = \text{hh:mm:ss.}$

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 } FROM { DISK ADDRESS@nnnn@ }
{ WRITE } { 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 } FILE file-identifier CLOSED { RELEASE }
{ INPUT/OUTPUT } { 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 } JLN DT = yy/ddd
 { PM }
 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 $\left\{ \begin{array}{l} \text{CHANGE} \\ \text{REMOVE} \end{array} \right\}$ COMMA IS REQUIRED FOR MORE THAN ONE $\left\{ \begin{array}{l} \text{CHANGE} \\ \text{REMOVE} \end{array} \right\}$

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 $\left\{ \begin{array}{l} \text{RESTRICTED} \\ \text{INTERCHANGE} \end{array} \right\}$ 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 $\left\{ \begin{array}{l} \text{LABELED . . .} \\ \text{UNLABELED} \end{array} \right\}$ 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 $\left. \begin{array}{l} \text{LOCKED} \\ \text{REMOVED} \end{array} \right\}$ 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 $\left. \begin{array}{l} \text{PARITY ERROR} \\ \text{ACCESS ERROR} \end{array} \right\}$ — 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-mnemonic 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 = $\left\{ \begin{array}{l} \text{PURGED LABEL} \\ \text{file-identifier [REEL nnnnnn]} \\ \text{UNLABELED} \end{array} \right\}$

unit-mnemonic READ CHECK

job-specifier-READ OUT OF BOUNDS terminal-reference

unit-mnemonic RELABELED pack-id $\left\{ \begin{array}{l} \text{R} \\ \text{U} \end{array} \right\}$

job-specifier-REQUESTED A $\left\{ \begin{array}{l} \text{CODE} \\ \text{DATA} \end{array} \right\}$ SEGMENT OF LENGTH ZERO terminal-reference

job-specifier-REQUESTED A CORE SPACE NEXT 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 $\left\{ \begin{array}{l} \text{READ} \\ \text{WRITE} \\ \text{SEEK} \end{array} \right\}$ ON CLOSED FILE

$\left\{ \begin{array}{l} \text{RO} \\ \text{SO} \end{array} \right\}$ 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 $\left\{ \begin{array}{l} \text{SAVED} \\ \text{TO BE SAVED} \end{array} \right\}$

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 $\left. \begin{array}{l} \text{SEND TO} \\ \text{RECEIVE FROM} \end{array} \right\}$ “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 $\left. \begin{array}{l} \text{NOT READY} \\ \text{JAM} \\ \text{MISSORT} \end{array} \right\}$

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

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.

Disk Initialization Instructions

The Disk 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.

Information will be supplied to the initializer through the card reader. There must be one input card for each disk cartridge to be initialized followed by an ? END card. The following is a description of the Initialization input card.

<u>Card Columns</u>	<u>Description</u>
1	Drive Number (usually drive zero)
2	"V" = Verify blank = initialize
3-8	Disk Cartridge serial number
10-19	Label
21	TYPE of cartridge S = System U = Unrestricted R = Restricted I = Interchange
23	Julian date (YYDDD)
29-42	Remarks (Owner's Name)

The initializer program is contained on a cassette tape and its operations are explained in the paragraphs that follow.

- a. Place the DISK 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. Place input cards in the card reader. One card for each cartridge to be initialized or verified followed by the ? END card.
- d. 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.
- e. Set the system MODE switch to the RUN position and press START (DO NOT PRESS CLEAR). This will load and execute the initializer.

- f. When the cassette tape has been read, the following message will be displayed on the console printer.

B 1700 DISK CARTRIDGE INITIALIZER – MARK level-number

Note: When a disk cartridge is initialized, all previous data is lost and must be reloaded if needed.

Example:

The following message would be displayed if a successful initialization had been completed.

```
ID = UNRESTRICTED SER# = 222001 000000 BAD SECTORS  
INITIALIZATION COMPLETE DRIVE 0
```

The disk cartridge is now ready to be used on the system.

Disk Verification

Any disk pack/cartridge previously initialized may be verified by using the same control card and placing a V in column 2.

Each segment is tested using the same criteria for the verification as is used for an initialization.

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 if head-per-track.
- b. Loads the MCP from magnetic tape to system disk.
- c. Loads the SDL Interpreters for both the 1710 and 1720 series of computers from magnetic tape to the system disk.
- d. Loads the CSM firmware for both the 1710 and 1720 series of computers from magnetic tape to system disk.
- e. Loads the System Initializer from magnetic tape to system disk.
- f. Loads SYSTEM/LOAD.DUMP and FILE/LOADER from magnetic tape to system disk.
- g. Makes appropriate entries in the NAME TABLE for all system software and firmware loaded.
- h. Constructs the COLDSTART VARIABLES on system disk.
- i. 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 on drive 0, (if not a head-per-track system).
- b. Set MODE switch to TAPE.
- c. Place the COLDSTART cassette in the cassette reader. Cassette is automatically rewound.
- d. Press CLEAR, then START.
- e. Cassette will read a few feet and the system will HALT.
- f. Set MODE switch to RUN, press START.
- g. Cassette will continue to read. If the system HALTS with @ 4 @ 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.

During COLDSTART execution one message is displayed requiring action by the system operator. This message and its response is as follows:

WHERE IS THE MCP-MT (X) Respond with the tape unit with
the MTx input message.

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.

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 the Central Service Module (CSM) 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 all A's. 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 system and with either the MCP I or the MCP II.

Name Table

The NAME TABLE is built during COLDSTART and resides on 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
- b. A SDL Interpreter for both the B 1710 and B 1720 series of computers
- c. A CSM for both the B 1710 and B1720 series of computers
- d. A System Initializer
- e. SYSTEM/LOAD.DUMP
- f. FILE/LOADER

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.

<u>NAME TABLE Entry Number</u>	<u>System Software Mnemonic</u>	<u>Meaning</u>
0	N	Standard System Initializer
1	NE	Entry System Initializer
2	NX	Experimental System Initializer
3	G1	1710 Central Service Module
4	G2	1720 Central Service Module
6	GE	Entry Central Service Module
7	G1T	1710 Trace Central Service Module

<u>NAME TABLE</u> <u>Entry Number</u>	<u>System</u> <u>Software</u> <u>Mnemonic</u>	<u>Meaning</u>
8	G2T	1720 Trace Central Service Module
10	GET	Entry Trace Central Service Module
11	GX	Experimental Central Service Module
12	I1	1710 MCP Interpreter
13	I2	1720 MCP Interpreter
14	IE	Entry MCP Interpreter
15	I1T	1710 MCP Trace Interpreter
16	I2T	1720 MCP Trace Interpreter
17	IET	Entry MCP Trace Interpreter
18	IX	Experimental MCP Interpreter
19	M	Standard MCP II
20	ME	Entry MCP (MCP I)
21	MT	Trace MCP
22	MET	Entry Trace MCP
23	MX	Experimental MCP
24	SD	Stand-Alone Memory Dump
25	SDE	Stand-Alone Entry Memory Dump
26	SDD	Stand-Alone Disk Dump
27	SDL	Stand-Alone SDL Program
28	SIO	Stand-Alone I/O Debug
29	SL	Loader for Stand-Alone SDL Program
30	SX	Stand-Alone MIL Program

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 four general environments which can be selected as a basis for operation:

- a. Standard MCP (MCP II)
- b. Standard MCP with Trace
- c. Entry MCP (MCP I)
- d. Entry MCP with Trace

The operator may select one of these by making two choices:

- a. STANDARD vs. ENTRY
- b. TRACE vs. NON-TRACE

The following input messages are used to make the above choices.

<u>Input message</u>	<u>Description</u>
CE	Use Entry MCP/firmware
CS	Use Standard MCP/firmware
CT	Make Trace Available
CN	Non-Trace

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.

Bits on the control panel are numbered from LEFT to RIGHT.

<u>Bits</u>	<u>Description</u>
0	Dump Memory
1	Run a stand-alone program (see, below, bits 8-11)
2	Switch MCPs, I vs. II
3	Switch TRACE vs. NON-TRACE
4	Run with experimental MCP
5	Run with experimental System Initializer
6	Run with experimental Interpreter

<u>Bits</u>	<u>Description</u>										
7	Run with experimental CSM										
8-11	When bit 1 is set, the following programs will be run.										
	<table border="1"> <thead> <tr> <th><u>Value</u></th> <th><u>Identification</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>SX</td> </tr> <tr> <td>1</td> <td>SDD</td> </tr> <tr> <td>2</td> <td>SIO</td> </tr> <tr> <td>3</td> <td>SDL, using SL to load with interpreter</td> </tr> </tbody> </table>	<u>Value</u>	<u>Identification</u>	0	SX	1	SDD	2	SIO	3	SDL, using SL to load with interpreter
<u>Value</u>	<u>Identification</u>										
0	SX										
1	SDD										
2	SIO										
3	SDL, using SL to load with interpreter										
12-23	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.

<u>Bits</u>	<u>Value</u>
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.

Firmware Detected Errors

Errors detected during Clear/Start will cause a halt with an error message in the L register identifying the error and the program that found it.

<u>L Register Value Bits 0-15</u>	<u>Program Identification</u>
@ 0000@	Central Service Module
@ 000F @	SYSTEM/INIT
@ 00F0 @	CLEAR/START
@ 0F00 @	MEM/DUMP
<u>L Register Value Bits 16-23</u>	<u>Error Description</u>
@ 01 @	No device on the designated I/O channel.
@ 02 @	I/O device on channel is not disk. (See T register)

L Register Value
Bits 16-23

Error Description

@ 03 @	Disk is not idle. (See T register for status)
@ 04 @	Time-out while waiting for service request .
@ 05 @	Bad reference address. (X = good, Y = bad)
@ 06 @	Bad status count after service request. (See T register)
@ 07 @	Bad result status from I/O control. (See T register)
@ 08 @	Seek time-out (timed by system software)
@ 09 @	Memory parity error in I/O descriptor.
@ 0A @	Memory parity error in I/O data.
@ 0B @	Time-out waiting for I/O operation to complete.
@ 0C @	Exception condition after 15 retries. (See T register)
@ 0D @	Exception on test I/O operation.
@ 0E @	Designated port and channel is not disk.
@ 0F @	No disk on system.
@ 10 @	Designated port is invalid.
@ 11 @	Designated channel is invalid.
@ 12 @	Not enough memory for this program.
@ 13 @	Memory parity after CSM overlay.
@ 14 @	Parity error somewhere in memory.
@ 15 @	NAME TABLE entry (number in T register) is zero or blank.
@ 16 @	Memory dumpfile port not equal to 7.
@ 17 @	Memory dumpfile address equal zero.
@ 18 @	Disk address in INITIALIZER IPB equal zero.
@ 19 @	MCP type field in HINTS is zero.
@ 1A @	Invalid stand-alone program specified.
@ 1B @	Stand-alone SDL file not available.
@ 1C @	No console printer on system

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.

Cards are used as input for the DISK/COPY routine. 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.

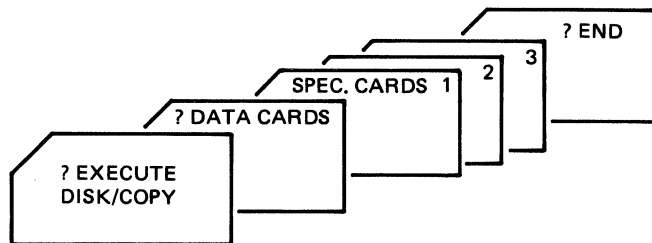


Figure 3-1. DISK/COPY Control Deck

Specification Cards

There may be multiple specification cards processed with a single execution of DISK/COPY, but each specification card is limited to one file.

Specification cards are free-form. Each card 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. See the REMOVE control instruction for further syntax explanation.

If the file-identifier is to be retained when copying to another 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/

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 a 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

KEYBOARD CONSOLE

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 replies to the ACCEPT message by entering an AX message containing the specifications needed to perform the DMPALL operation.

CARDS

The DMPALL execute control deck has the following format:

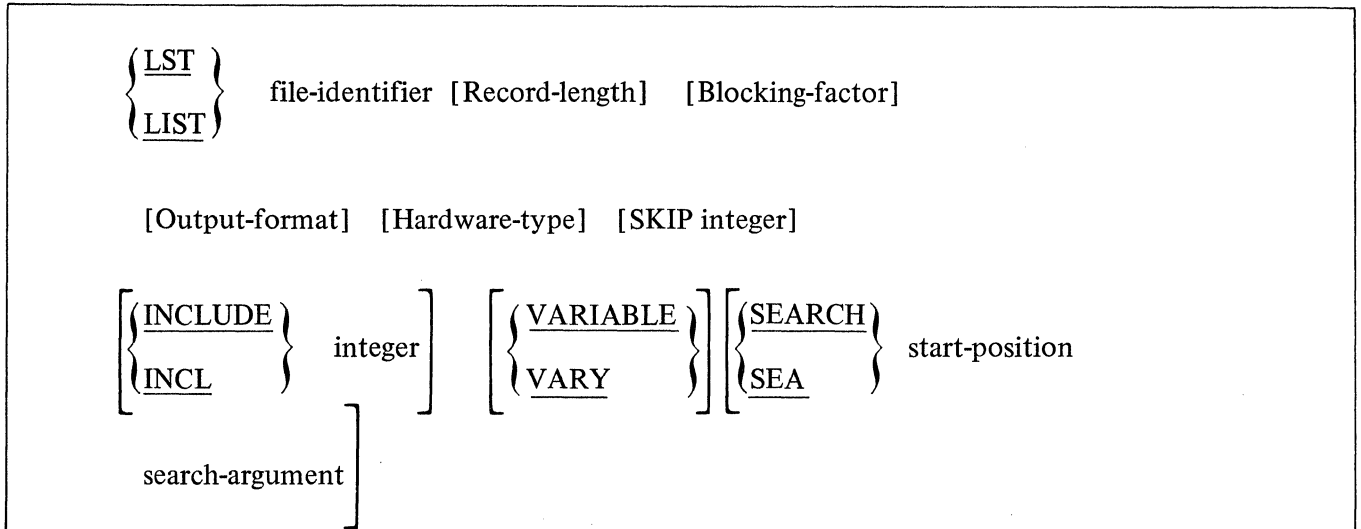
- ? EXECUTE DMPALL FILE SPEC NAME specification-file-identifier
- ? DATA specification-file-identifier
(specification cards)
- ? END

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:



The file-identifier entry must immediately follow the LIST or LST 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 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 a disk file when both the record length and blocking factor entry are omitted, the blocking factor with which the file was created will be used.

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 be one of the following:

- a. Card files: CRD or CARD
- b. Magnetic tape files: MTP or TAPE

- c. Paper tape files: PPT or PAPER
- d. Disk files: DSK, DISK, or the entry may be omitted.
- e. 96-col. card files: C96 or CARD96

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.

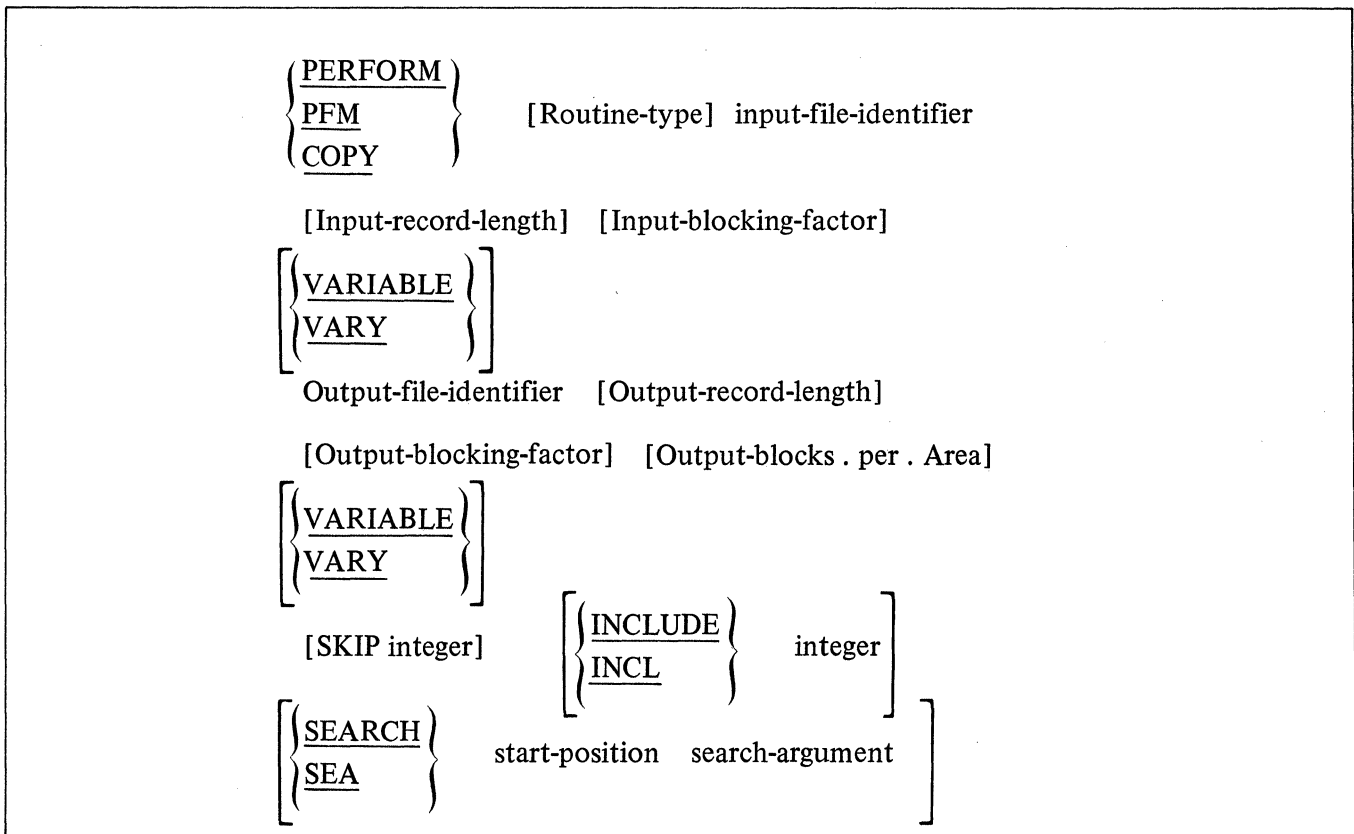
The VARIABLE or VARY entry may be used to specify tape or disk files having variable length records.

The SEARCH or SEA entry may be used to specify that printing should begin with the first record containing the value of the specified search-argument at the specified start-position (byte-number) in the record. The first byte in the record is relative position 1.

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 the names of two of the following media:

- a. Card files: CARD
- b. Magnetic tape files: TAPE
- c. Paper tape files: PAPER
- d. Disk files: DISK or the entry may be omitted.
- e. 96-col. card: CARD96.
- f. Binary 80-col. card reproduction: BINBIN

The short hand form uses a combined abbreviation format.

O U T P U T D E V I C E S

From	To	Card	Mag. Tape	Paper Tape	Disk	96-col. CARD
Card		CRDCRD	CRDMTP	CRDPPT	CRDDSK	CRDC96
	Mag. Tape	MTPCRD	MTPMTP	MTPPPT	MTPDSK	MTPC96
	Paper Tape	PPTCRD	PPTMTP	PPTPPT	PPTDSK	PPTC96
	Disk	DSKCRD	DSKMTP	DSKPPT	DSKDSK	DSKC96
	96-col. card	C96CRD	C96MTP	C96PPT	C96DSK	C96C96

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 MCP control instructions.

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.

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. All output disk files have 25 areas. 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 VARIABLE or VARY entry may be used after the output identifier to indicate variable length input records with variable length records being produced.

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 entry may be used to specify that copying should begin with the first record containing the value of the specified search-argument at the specified start-position in the record. The first relative location in the record is one.

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  
  LISK 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.

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 distinguish 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.

<u>Column</u>	<u>Description</u>
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:

<u>Column</u>	<u>Description</u>	
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 nnnnnn-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.

SSLOAD/MAKCAS

? EXECUTE SSLOAD/MAKCAS

? FILE CARDS NAME=CARDS;

? DATA CARDS

L <LOADER FILE-ID>

G <GISMO FILE-ID>

I <INTERP FILE-ID>

S <SDL CODE FILE-ID>

? END CARDS

<MX>AX <FILE-ID> % for MIL Standalone programs

SORT

General

The SORT is a system program that provides the user with a means to arrange a file of records. It processes specification cards that describe the input and output files, the keys by which the file will be arranged, and various options.

A parameter table is generated by the SORT and a sort intrinsic is invoked. The sort intrinsic may also be invoked from within a language (RPG or COBOL), and the manual for that language contains a description of its sort statement.

The sort intrinsic does the actual sorting of the file in either an ascending or descending sequence according to a designated key or keys.

There are two sort intrinsics, referred to as the vector replacement and the INPLACE technique. The intrinsic using the vector replacement technique is normally the one invoked.

The intrinsic using the INPLACE technique is invoked when the user includes an optional INPLACE specification card in the SORT source deck. This option should be used when a minimum of disk space is available for sorting.

SORT reserved words and characters appear in uppercase type throughout the SORT text. A list of the SORT reserved words appears at the end of the SORT text.

SORT Execution Deck

The SORT execution deck consists of specification cards and control cards. See figure 3-2.

Three of the specification cards are required: FILE IN, OUT, and KEY. Other specification cards are optional and allow modification and optimization of the sort.

A description of each of the SORT specification cards (statements) appears in the following pages.

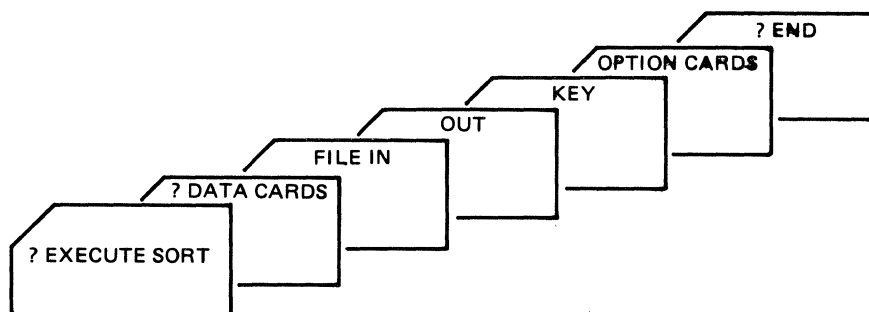


Figure 3-2. Sort Execution Deck

The FILE Statement

The FILE statement is comprised of two parts which describe the input file to be sorted and the output file to be produced. The first part must be the FILE IN statement and the second part is the OUT statement which must immediately follow FILE IN.

FILE IN

The FILE IN statement describes the input file to be sorted, and is one of the three specification cards that are required. The parameters following the file-identifier must be enclosed in parentheses and separated by a space. The FILE IN statement has the following format:

FILE IN file-identifier

$$\left(\begin{array}{l} \underline{\text{CARD}} \\ \underline{\text{TAPE}} \\ \underline{\text{DISK}} \end{array} \begin{array}{l} \\ \\ (\text{records-per-area}) \end{array} \right) \text{ record-size } [\text{blocking-factor}] \left[\begin{array}{l} \underline{\text{PURGE}} \\ \underline{\text{DEFAULT}} \\ \underline{\text{MULTI}} \end{array} \right])$$

DP-ID

The dp-id is the name of the disk pack or disk cartridge that the file is to be read from or written to. If dp-id is omitted on input, the file is assumed to reside on the systems disk. If it is omitted on output, the file is written on the systems disk.

FILE-IDENTIFIERS

File-identifiers are standard.

When the INPLACE sort option is specified and the file-identifiers are the same for both the FILE IN and OUT statements, the original file will be altered during the sorting process and the output of the sort will occupy the same space when the files are on disk.

If the file-identifiers are different, the input file will not be disturbed and a new output file will be created.

When the PURGE option is used, the input file-identifier will be removed from the disk directory at the completion of the sort intrinsic.

CARD

The word CARD specifies that the input file is on cards.

TAPE

The word TAPE specifies that the input file is on magnetic tape.

DISK

The word DISK specifies that the input file is on disk.

RECORDS-PER-AREA

When the file is on disk the records-per-area must be supplied and enclosed within parentheses. The records-per-area must be calculated by the user.

RECORD-SIZE

The record-size is a required entry and is the actual record size in bytes (characters) associated with the file. When the DEFAULT option is used a record-size must be specified but need not be correct.

BLOCKING-FACTOR

The blocking-factor is optional and specifies the number of logical records in a block. When this entry is omitted the blocking-factor default of one (1) will apply.

PURGE

This option will result in the input file-identifier being removed from the disk directory at the completion of the sort.

DEFAULT

This option allows the user to sort a file when he doesn't know anything about the file except the file-identifier. If the file is not on the system pack the user must also supply the disk pack name.

This option applies only to disk files.

MULTI

This option allows the user to sort a multi-pack disk file. If MULTI is designated on the IN specification, it must also be designated on the OUT specification.

OUT

The OUT statement describes the output file to be created, and is one of the three specification cards that are required.

The OUT statement must immediately follow the FILE IN statement. The parameters following the file-identifier must be enclosed in parentheses and separated by a space.

The OUT statement has the following format:

OUT file-identifier

$$\left(\begin{array}{l} \text{CARD} \\ \text{DISK} \\ \text{TAPE} \\ \text{PRINTER} \end{array} \right. \begin{array}{l} \\ \text{(records-per-area)} \\ \\ \end{array} \left. \right) \text{record-size [blocking-factor][MULTI] } \underline{\quad}$$

The elements of the OUT statement have the same function as they do in the FILE IN statement except that they describe the desired output file.

Examples:

```
FILE IN CARDX ( CARD 80 )
  OUT LINE ( PRINTER 80 )
```

```
FILE IN CARDX(CARD 80) OUT LINE(PRINTER 80)
```

Both of the above examples will produce the same result.

```
FILE IN RANDOM (DISK(1000) 100 10)
  OUT SORTED ( DISK (1000) 100 10 )
```

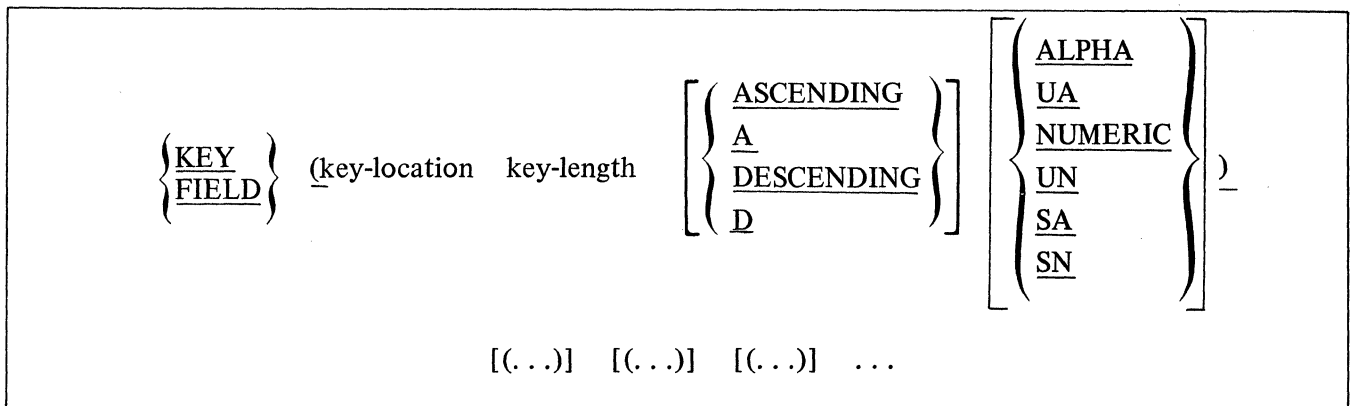
Note that in the above example parentheses serve as delimiters between parameters so that additional spaces are permitted but not required.

The Key Statement

KEY

The KEY statement defines the field or fields within a record that will determine the order in which the file is to be arranged. It is one of the three specification cards that are required.

The format of the KEY statement is:



Multiple key descriptions are allowed and must be enclosed in parentheses. The first key is the major key and any additional keys are minor keys of decreasing significance. Each succeeding minor key is subordinate to any preceding minor or major key.

The maximum number of keys is 30 unsigned keys, 15 signed keys, or any combination not exceeding 30 where each signed key is counted as two unsigned keys.

KEY-LOCATION

The key-location specifies the relative position of the most significant byte or digit (alpha or numeric) of the field from the beginning of the record.

The first byte or digit in a record is relative position one (1). The position is counted in the number of units applicable to the data type for that key. This permits all possible data types to appear within a record. Additional information describing position will be found in following paragraphs concerning data types (ALPHA, NUMERIC, etc.).

For signed fields the key-location is specified as the most significant byte or digit of the key itself, and not the position of the sign. The sign location is the left-most or high order position of the field.

KEY-LENGTH

The key-length specifies the number of significant bytes or digits in the key. It should not include the length of the sign when the key is signed.

ASCENDING or A

Ascending sequence does not have to be specified as it is the default. The file will be arranged with the record having the smallest key appearing first, followed by records with increasingly larger keys.

DESCENDING or D

The use of this option will result in the sorted file being arranged with the record having the largest key appearing first, followed by records with succeeding smaller keys.

ALPHA or UA

ALPHA or UA (unsigned alpha) 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. ALPHA or UA need not be specified as they are 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 alpha) indicates that the data is alphanumeric and that some or all of the keys may contain a minus sign.

The key-location is specified as the most significant byte of the key itself and not the position of the sign.

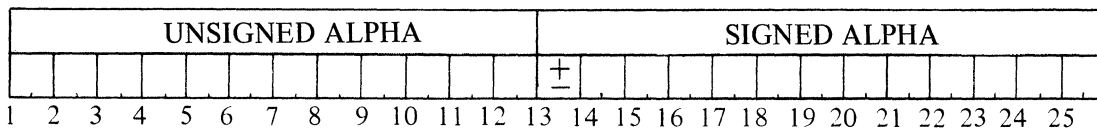
The minus sign is represented as a hexadecimal D in the most significant four bits of the first byte in the field.

SN

SN (signed numeric) indicates that the data is 4-bit numeric and that some or all of the keys may contain a minus sign. The key-location is specified as the most significant digit of the key itself, and not the position of the sign.

The minus sign is represented as a hexadecimal D and will occupy the most significant digit of the field.

Examples to illustrate several key descriptions follow:



The method of referencing the key-location and key-length of ALPHA and SA data may be illustrated with the use of the above illustration which represents a record twenty-five bytes (8-bit) in length. The first twelve bytes are type ALPHA and the following thirteen bytes are type SA. The thirteenth byte of the record contains the minus sign if the field is a negative value.

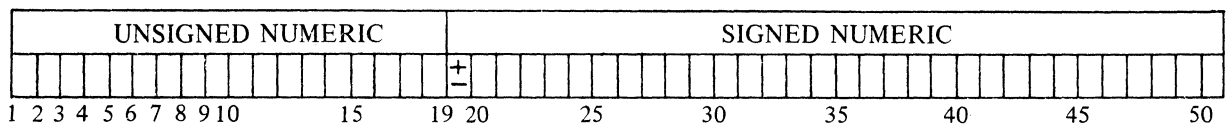
KEY (5 2) describes the field that starts with the fifth byte and is two bytes long. The data is type ALPHA and the output sequence is to be ascending order.

KEY (5 2 A UA) explicitly names the options A and UA and will have the same result as the above description.

KEY (14 12 SA) describes the signed field starting at byte fourteen and continuing to the end of the record.

KEY (14 1 D SA) describes the one byte field at byte fourteen. The output sequence will be in descending order so that all positive keys will appear in the output before any of the negative fields.

KEY (1 12) (14 6 SA) describes an unsigned major key field twelve bytes long that starts in the first byte of the record, and a signed alpha field starting in byte fourteen that is six bytes long.



The method of referencing the key-location and key-length of NUMERIC and SN data is illustrated by the above example which represents a record fifty digits (4-bit) long. The first eighteen digits are type unsigned numeric, and the remaining thirty-two digits are type SN. The nineteenth digit from the beginning of the record is the sign location.

KEY (1 18 A UN) describes all of the eighteen digit unsigned numeric field. The reserved word A could be omitted since ascending sequence is the default option.

KEY (18 1 UN) describes the last digit of the unsigned numeric portion of the record.

KEY (20 5 SN) describes the left-most five digits of the signed numeric part of the record. Digit nineteen is the sign location.

Data of the type NUMERIC or SN could only be associated with disk or tape files because of its "packed" nature.

SORT Option Statements

The purpose of the sort option statements is to allow the user to optimize the sort and add comments to the SORT specification card deck.

There are eight option cards as described below.

NOPRINT

The NOPRINT option will inhibit the printing of the sort specifications on the line printer. This allows the sort to be executed when the printer is in use, and also results in less execution time.

The NOPRINT statement must be the first entry in the sort specifications.

The TIMING option is not affected by the use of the NOPRINT option.

MEMORY integer

The MEMORY option can be used to allocate more memory to the sort than the 6000 bytes assigned by default.

Increasing the memory available to the sort will usually make the sort run faster, until an optimum memory size is reached. Increasing the memory size beyond this optimum will result in a slower sort. The optimum size is dependent on file size (record size and number of records).

Example: MEMORY 15000

integer RECORDS

The user may furnish an estimate of the number of records in the input file, which helps to optimize the execution of the sort. If this option is omitted the default is 20,000 records.

Example: 12500 RECORDS

TIMING

The TIMING option may be used when the vector replacement sort intrinsic is used, and furnishes an estimate of the number of merge passes that will be required during execution of the sort. The estimate and some other information that may be useful for debugging will be printed on the line printer.

This option does not apply to the INPLACE sort.

BIAS integer [%]

This option is used to estimate how ordered or sequenced a file is in relation to the keys the file is to be sorted on. The estimate is used to optimize the execution of the sort intrinsic.

The number entered may be from zero (0) to ninety-nine (99), where a fifty (50) indicates completely random data and is the default if no BIAS statement is included. A zero (0) would indicate that the file is in reverse order in relation to the keys to be sorted on. A ninety-seven (97) would suggest that the file is nearly in the desired sequence.

Example: BIAS 60%

The percent sign is optional and may be omitted.

The BIAS option does not apply when the INPLACE option is used.

TAGSORT

TAGSORT is a means of sorting a file that leaves the original file intact, and creates a new file containing indices pointing to the relative locations of records within the original file.

Input files can be of any type.

The output file (tagfile) must be defined as four characters per record. It consists of eight decimal digit indices pointing to the input file's records.

TAGSORT cannot be used with INPLACE sort.

In COBOL the access method would be RANDOM using tagfile as the ACTUAL KEY.

In RPG the access method would be INDEXED using the relative record number as delivered in the tagfile to DIRECTly access the original file.

INPLACE

This option may be used when a minimum of disk space is available for sorting. The vector replacement sort produces work files approximately two-and-one-third times the size of the input file. The INPLACE sort requires work file space equal to the input file space, unless the input and output file identifiers are the same. In the latter case, no work file space is required but the input file is replaced by the output file during the sorting process.

SYNTAX

The SYNTAX option should be used when the SORT specification cards are to be checked for errors only. The sort intrinsic will not be executed, even when no errors are detected in the specifications.

COMMENT

Any non-reserved word or character.

This option allows explanations or notes to be interspersed between SORT statements. SORT control (reserved) words may not be used in the text of the comment.

Specification cards for a typical sort might be as shown below:

```
FILE IN SRT/AAA/ (DISK(500) 100 1)
      OUT XYZ/BBB/ (DISK(500) 100 10)
2500 RECORDS
MEMORY 12000
BIAS 50%
KEY (7 12) ( 1 6)
```

The above specification cards could also appear in different format as shown below and produce the same results.

```
FILE IN SRT/AAA/ (DISK(500) 100 1) OUT XYZ/BBB/ (DISK(500) 100 10) 2500 RECORDS
MEMORY 12000 BIAS 50 KEY (7 12) (1 6)
```

The disk pack names SRT would contain the file AAA and the sort intrinsic would order the file, with the output file (sorted) BBB getting written on disk pack XYZ. The sort intrinsic using the vector replacement method would be used in both cases.

```
FILE IN CARDX (CARD 80)
      OUT LINE (PRINTER 80)
KEY (1 10)
INPLACE
1900 RECORDS
BIAS 60
```

The above SORT specification cards would result in the INPLACE sort intrinsic being invoked to do the sorting. The input file CARDX in the card reader would be read in, the records sorted according to the single ten-byte key, and the sorted file would be printed on the line printer. The BIAS estimate would be meaningless since that option has no effect when the INPLACE option is used. The example below shows a tagsort execution deck:

```
? EXECUTE SORT
  FILE IN A/B (DISK (10) 200)
  OUT A/BTAGFILE (DISK (20) 4)
  KEY (10 8 UA)
  TAGSORT
? END
```

SORT RESERVED WORDS

(PACK
)	PAPER
,	PARTITION
%	PRINTER
/	PURGE
A	RECORDS
ALPHA	RESTART
ASCENDING	
ASSEMBLE	SA
	SAVEBLOCK
BIAS	SB
	SN
CARD	SYNTAX
CARDS	
COMP	TAGSEARCH
COMPILE	TAGSORT
COMPLEMENT	TAPE
CORE	TIMING
D	UA
DEFAULT	UN
DESCENDING	USERBLOCK
DISK	
DISTRIBUTE	V
E	WAIT
EVEN	
EXECUTE	ZIP
FIELD	
FIELDS	
FILE	
GENERATE	
IDENT	
IN	
INPLACE	
KEY	
KEYS	
MEMORY	
MERGE	
NOPRINT	
NUMERIC	
O	
ODD	
OUT	

SECTION 4

PROGRAM PRODUCTS

COMPILERS

INTRODUCTION

The B 1700 computer system recognizes four compilers: RPG, COBOL, FORTRAN and BASIC. Each of these compilers, which generate executable object programs from a programmer's source program, 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 COBOL 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 COBOL S-Language which is then ready for execution.

The program generated by the RPG compiler is executed under control of the MCP using the COBOL interpreter.

Following is an example of an RPG compilation deck.

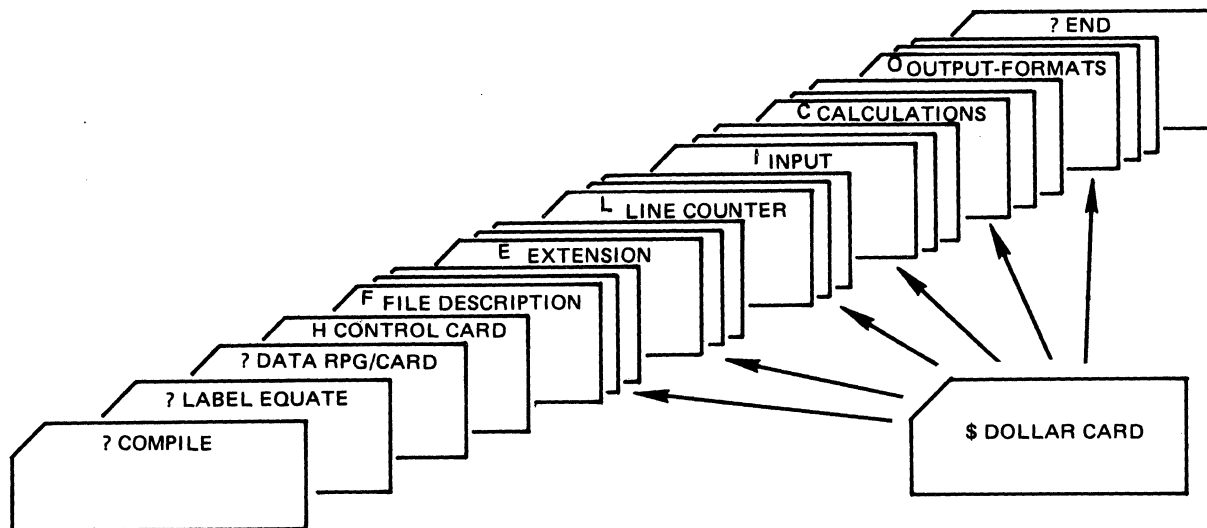


Figure 4-1. RPG Compilation Deck

Dollar Card Specifications

Dollar Card Specifications allow the RPG Compiler or Translator 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:

<u>Columns</u>	<u>Description</u>
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".

- a. \$ 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.
- b. \$ 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.
- c. \$ 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.
- d. \$ 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 40, right-justified, leading zeros optional. The default value assigned is 40, unless specified otherwise.

- e. \$ 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 500 unless specified otherwise.
- f. \$ OPEN—explicit open allows for all files to be opened at Beginning-of-Job. Default is an implicit open when the files are actually called for.
- g. \$ CLOSE—explicit close allows all input serial files to remain opened until End-of-Job. Default is the implicit close of files at End-of-File.
- h. \$ AAOPEN—is a file time option used to set a bit in the MCP file parameter block and allocate all disk space areas at the beginning of the program.
- i. \$ ONEPAK—specifies that this particular file must be contained on one disk.
- j. \$ CYL—allocates file areas starting on an integral cylinder boundary.
- k. \$ 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.
- l. \$ REFORM—input and update disk files are assumed to have the block and record length declared on the file header unless the \$ REFORM option is used. However, on input or update chained indexed file specifications “data keys in core” option, it may be desirable to also use \$ REFORM to indicate to the compiler that it may juggle the blocking factor to optimize the speed of chaining. Under this condition, the blocking-record-length specified on the File Description Specifications must be the same as when the file was outputted. This combination will produce the fastest chaining possible.
- m. \$ 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.

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 COBOL-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 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.
- \$ SEG** Orders the compiler to begin placing code in an overlayable segment identified by the integer in the VALUE field (right-justified, between 0 and 7 inclusive). Segmentation is an automatic function of the RPG compiler and optimized for its best uage. When the SEG option is used, automatic segmentation is not suppressed.
- \$ 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.
- \$ 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.
- \$ XREF** The XREF option must be placed at the beginning of the RPG source program, prior to the first File Specification. 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.

RPG to COBOL Options

The following options may appear prior to the first source statement in the RPG program to direct the compiler to terminate prior to generation of the code file. The intermediate work files in the disk directory may then be used as input to COFIRS.

- a. **\$ XLATE**—specifies termination of the compiler prior to generation of the code file.
- b. **\$ XLIST**—specifies a single-spaced listing of the COBOL source language will be produced during the execution of COFIRS.

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.
SOURCE	RPG/CARD	Input file from the card reader.
TABCRD	RPG/VECTOR	Input file for TABLES from the card reader.

RPG Internal File Names

RPG TO COBOL TRANSLATOR (COFIRS)

General

The RPG to COBOL translator converts the intermediate disk file, previously created from the RPG compiler through the \$ XLATE option, to a COBOL source language file on disk (SOLD file). This source file is then acceptable input to the B 1700 COBOL compiler. The flexibility of this translator allows for any RPG source statement, acceptable to the B 1700 RPG compiler, to be translated to COBOL with little or no loss of run-time efficiency of the object program.

Execution of Translator

As a preliminary step to the execution of the translator, the RPG program must be compiled with the RPG compiler using the \$ XLATE option. An additional dollar card, \$ XLIST, may also be included in the RPG source deck if a listing of the generated COBOL source file is desired during the execution of the translator.

Example:

```
?   COMPILE program-name RPG LIBRARY

?   DATA RPG/CARD

      $ XLATE
      [$ XLIST] optional
      (RPG SOURCE cards)

?   END
```

Once the program has been compiled, the intermediate disk work file will be locked, prior to generating the COBOL S-Language file. This file is then used as input to the translator.

The following is an example of the execute statement:

```
?   EXECUTE COFIRS
```

At end of job, COFIRS will lock a COBOL source file named RPGCOB in the disk directory. This file may then be used as input to the B 1700 COBOL compiler.

The following is an example of the RPGCOB file used as input to the COBOL compiler:

```
?   COMPILE program-name COBOL LIBRARY

?   FILE SOURCE NAME RPGCOB;

?   DATA CARDS

      $ MERGE

?   END
```

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.

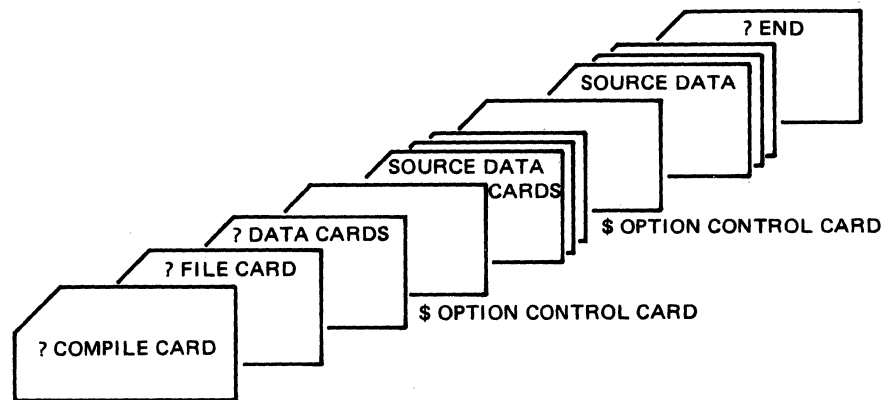


Figure 4-2. COBOL Compilation Deck

\$ OPTION Control 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 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:

\$ OPTION-1 [OPTION-2] . . .

OPTIONS

The options available for the COBOL compiler are listed below:

- a. **CARD**—input is from the source language cards or paper tape. This option is for documentation only.
- b. **LIST**—creates a single-spaced output listing of the source language input, with error and/or warning messages, where required.
- c. **SINGLE**—causes the output listing to be printed in a single-spaced format.
- d. **DOUBLE**—causes the output listing to be printed in a double-spaced format.
- e. **CODE**—list object code following each line of source code from the point of insertion.
- f. **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.

- g. **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.

- h. **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.
- i. **SUPPRESS**—suppresses all warning messages except sequence error messages. The sequence error message can be suppressed with the NO CHECK option.
- j. **SPEC**—if syntax ERRORS occur, this option negates the control and LIST option and causes only the syntax errors and associated source code to be printed. Otherwise the CONTROL and LIST options remain in effect.

- k. “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.
- l. 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.
- m. 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.
- n. 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.
- o. 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.
- p. NO SEQ—terminates the SEQ option and resumes using the sequence number in the source statement as it is read in.
- q. CONTROL—prints the \$-option control cards on the output listing. The LIST option must be on.
- r. NO—when the NO option precedes one of the above options, with the exception of MERGE which cannot be terminated, it will terminate the function of that option.
- s. REFERENCE—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.
- t. ANSI—when used, will inhibit the EXTENSION of AT END . . . ELSE, and during compilation will flag them as syntax errors.
- u. 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.
- v. 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: The VOID card, and the CHANGE or addition card.

- 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 1700.

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.

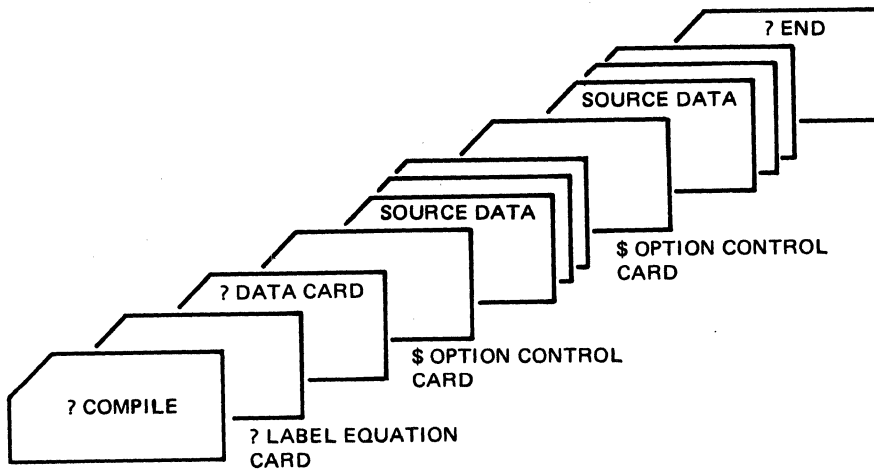


Figure 4-3. FORTRAN Compilation Deck

\$ 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" are assumed.

The format for the FORTRAN \$ Option control card is:

```
$ option-1 ... .. option-n
```

The FORTRAN \$ Option control card has the following characteristics:

- a. Column one must be a \$ sign.
- 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:

- a. **CARD**—symbolic input is from source language cards. This is a default option.
- b. **LIST**—creates a single-spaced output listing of the source program with error and/or warning messages. This is a default option.
- c. **SINGLE**—causes the output listing to be printed in a single-spaced format. This is a default option.
- d. **DOUBLE**—causes the output listing to be printed in a double-spaced format.
- e. **CODE**—will list the object code following each source statement from the point of insertion into the source deck.
- f. **NO**—used in conjunction with any of the above options will terminate or reset the function of that option. When an option is preceded by **NO**, there must be at least one space between the word **NO** and the option to be reset.

SOURCE DATA Cards

The Source data cards, commonly referred to as the source deck, are the statements comprising the source program.

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.

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 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.

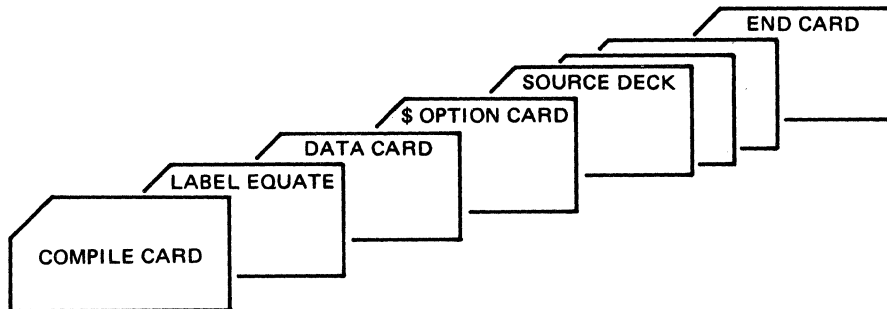


Figure 4-4. Compilation Deck

\$ 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 \$ option-1 . . . 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 a default option. |
| 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. |

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 (identified by the name BAS.INTRIN/#nnnnnnnn) 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	LINE	Source output listing to the line printer.

BASIC Internal File Names

BURROUGHS CORPORATION
DATA PROCESSING PUBLICATIONS
REMARKS FORM

TITLE: B 1700 SYSTEMS
SYSTEM SOFTWARE
OPERATIONAL GUIDE

FORM: 1068731
DATE: 6-74

CHECK TYPE OF SUGGESTION:

ADDITION

DELETION

REVISION

ERROR

GENERAL COMMENTS AND/OR SUGGESTIONS FOR IMPROVEMENT OF PUBLICATION:

FROM: NAME _____
TITLE _____
COMPANY _____
ADDRESS _____

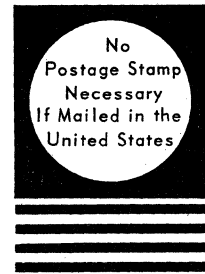
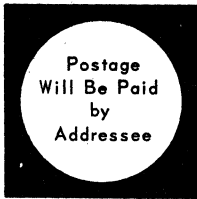
DATE _____

STAPLE

FOLD DOWN

SECOND

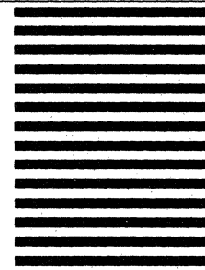
FOLD DOWN



BUSINESS REPLY MAIL
First Class Permit No. 817, Detroit, Mich. 48232

Burroughs Corporation
Burroughs Place
Detroit, Michigan 48232

attn: Systems Documentation
Technical Information Organization, TIC-Central



FOLD UP

FIRST

FOLD UP



*Wherever There's
Business There's*



Burroughs