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

Printed in U.S. America

7-73

1068731

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 Page

Section Page

Section Page

TABLE OF CONTENTS (cont)

Section Page

Section Page

4 COMPILER (cont)

LIST OF TABLES

INTRODUCTION

The productivity of a computer facility is largely dependent on an operator's experience and knowledge of the system. When the programs produced for the installation have been refined and are ready for use, the successful results obtained are largely due to the expertise of the operator. Therefore, some concept of the B1700 MCP and a knowledge of its peripherals 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 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.

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

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

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

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

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

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

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

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.

This publication supersedes and replaces the B 1700 System Software Operational Guide (Preliminary Edition), form 1057171 dated October, 1972.

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 COLD START, 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
MTX – Magnetic tape
SPO – Console Printer

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. Multi-programming (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 (MCP II)
f. Disk Cartridges
g. Disk Packs (MCP II)
h. Head-per-Track Disk (MCP II)

In addition, MCP II will accommodate the MICR reader sorter as well as various data communication 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.

Hardware Type	Usage
Console Printer	Operator communication
Disk	Auxiliary storage
Card Reader(s)	Control input

GISMO

GISMO (Generalized Interface and Supervisor for Multitasking Operations) is a microcoded routine which performs the following functions in an equivalent hard-wired machine:

a. Interrupt Detection and Handling.
b. Passes control to/from the MCP, usually on an interrupt.
c. Controls all I/O activity, such as:
    1. I/O Initialization
    2. Data Transfers
    3. I/O Termination
d. Manages Interpreter Activity.

INTERPRETERS

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

SECTION 2
MASTER CONTROL PROGRAM

GENERAL

The Master Control 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 System Software Operational Guide will make reference to both an 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 multi-programming capabilities whereas MCP II is designed for larger system configurations with multi-programming capabilities.

A detailed description of the MCP is presented in the B1700 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. Dictionary Maintenance
   Users need only to specify changes 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.

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 which the MCP received a permanent disk directory entry request.

Disk Pack Id

The disk pack-id is the name that is assigned to a disk pack 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 name 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 address.
3. Type of File:
   0 = MCP
   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. If there are more than twelve names in the sub-directory the MCP will increase the size by one segment for each twelve additional names.

The sub-directory entry is identical to the main directory entry with one exception, 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 cartridge or disk pack identifier. The removable disk directories and system disk directories are the same format.

HALTS

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

Halt Code	Description of Halt
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 sub-string.
7	Invalid sub-script.
8	Invalid value returned from a procedure.
9	Invalid case.
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.
E	Unused.
F	Attempt to trace the MCP with a Non-Trace version of Clear/Start.
10	Console HALT. (interrupt Switch)

Halt Code	Description of Halt
11	This is a controlled HALT. The "T" register will contain a message from the MCP.
12	Attempt to write outside of the MCP base or limit register.
17	Bad I/O descriptor. System port and channel in "X" register; address of I/O descriptor in "Y" register.
19	Irrecoverable I/O error. Result descriptor in "T" register.
1A	Insufficient memory available for Clear/Start.
22	Invalid service request.
25	Second operation complete bit is missing from the result status field. Result descriptor in "T" register.
31	Invalid command to soft I/O. Can COLD START only.

MCP OPTIONS

The MCP will perform certain functions based on the status of its available options. The system operator can use the SO input message to set (turn on) an option, or the RO message to reset (turn off) an option except in the case of LOG and CHRG which are independently set with SL and MC respectively.

At COLD START all of the MCP options with the exception of DATE, TIME, BOJ, and EOJ are set "off", and must be set "on" 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	SCHM	PBT
DATE	TERM	PBD
EOJ	TIME	CHRG
LIB	LAB	CLOS
OPEN	LOG	RMOV

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.

DATE

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

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 when "on" causes the MCP to display a tape label-name when a BOT is read.

LIB

The LIB option causes the MCP to display the affected file-id and the resultant library maintenance actions performed on that file. 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

OPEN

The OPEN option specifies that a "file-identifier OPENED . . . " message be displayed on the SPO 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 when "on" 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 program scheduled message when a program is placed in the schedule awaiting exection. The MCP message has the following format:

SCHEDULED: program-name = job-no. PR = integer TIME = (seconds)

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 "on" at COLD START time and specifies that the "** TR PLEASE" message be displayed on the SPO at Clear/Start time. When this option is set and the "** TR PLEASE" message is displayed, the system operator must enter the time with the TR input message before program execution may begin.

CLOS

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

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 printing the log and/or starting a new log. The LOG option once set by the SL message cannot be reset (turned off).

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

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, except for the last eight columns, 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.

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
<	less than sign

Any special characters not contained in the above list do not require quote marks to enclose the identifier.

Examples:

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

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

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

In the last example the pound sign (#) is not listed as a special character and therefore 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 because if the text of one control instruction is delimited by a space then this is considered to indicate the "logical end" of that control instruction and may be followed by another control instruction on the same card as the example above indicates.

h. If control instructions are communicated to the MCP through keyboard input messages, the entire instruction must fit on one line. There is no continuation of control instructions allowed with keyboard input.

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. Program control and program parameter statements must be entered in the same statement as the COMPILE, EXECUTE, or MODIFY statements.

Example:

```
EXECUTE program-name PRIORITY = 9 MEMORY = 8000
```

ZIP

MCP control statements may be 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 additional 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, and special characters in any combination.

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

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

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

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

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

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

unit–mnemonic	device
CDX	96–column card device
CRX	80–column card reader
CPX	80–column card punch
LPX	Line Printer
MTX	Magnetic Tape
CSX	Cassette Tape (peripheral unit)
SRX	MICR Reader Sorter
PPX	Paper Tape Punch
PRX	Paper Tape Reader
DPX	Disk pack/cartridge
DISK	Head–per–Track disk
SPO	Console Printer

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 pack: The name given to 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.

i. removable disk pack: The name given to a disk pack or cartiridge 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 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:

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

The control word CHANGE may be abbreviated as CH.

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

The CHANGE statement will cause the MCP to change the file-identifier of specified disk files from one name to the other. If the file-identifier in the CHANGE statement is residing 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/
```

A new sub-directory is created any time the main directory file name is changed. This is due to the structure of the directory where more than one main directory file may not point to the same sub-directory file.

If the CHANGE statement is entered into the system 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) . . .
```

If the CHANGE statement is accepted by the MCP and the change is made the following message will be displayed on the console printer:

```
file-identifier CHANGED TO file-identifier
```

CHANGE

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 (;) is detected.

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:

```
? { REMOVE } { file-identifier          }
  { RE     } { family-name/=            }
             { dp-id/family-name/=      }
```

The control statement REMOVE may be abbreviated as RE.

The "/=" option 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.

In the REMOVE statement when the pack is not referenced, 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.

COMPILE

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:

```
? { COMPILE } program-name [WITH] compiler-name [ { LIBRARY } ]
  { CO      }                                    [ { SAVE    } ]
                                                 [ { SYNTAX  } ]
    [control attributes]
```

The COMPILE statement may be abbreviated as CO.

The compiler control statement must be the first statement in a set of compile 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 program for execution. The program-name 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 until an EXECUTE statement for that program-name is issued to the MCP.

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 to be executed after an error-free compilation. The program remains available for re-scheduling using an EXECUTE statement.

The COMPILE [FOR] SYNTAX provides a source 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 a debugging tool, first time compilation, or a new source listing.

EXECUTE

EXECUTE

The EXECUTE statement instructs the MCP to call a program from the disk system software 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 control statement in a set of control statements pertaining to the execution of a program. This does not, however, prevent the use of a CHANGE or REMOVE statement prior to the EXECUTE statement in order to prevent a duplicate file situation and reduce operator intervention. The CHANGE and REMOVE statements do not directly pertain to the execution itself.

If the program referenced in the EXECUTE statement resides on a removable disk cartridge or disk pack, the disk-pack-id must precede 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 called out of the system software library on disk and executed. One of the files in the program TEST assigned as a card file is identified by the DATA control card. If the program does not require a card file, only the EXECUTE control statement is necessary and can be entered through the card reader with the "? EXECUTE TEST" or the console printer with the "EX TEST" command.

MODIFY

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

The format of a MODIFY statement is:

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

The MODIFY control statement 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
FILE
FREEZE
HOLD
INTERPRETER
INTRINSIC.NAME
INTRINSIC.DIRECTORY
MEMORY
PRIORITY
UNFREEZE
VIRTUAL.DISK

AFTER

AFTER

The AFTER statement is used to conditionally schedule a program "after" the End-of-Job of another program.

The format of the AFTER statement is:

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

The control word AFTER may be abbreviated as AF.

AFTER.NUMBER

The AFTER.NUMBER statement is used to conditionally schedule a program in relation to a program already in the schedule.

The format of the AFTER.NUMBER statement is:

```
    ⎧ EXECUTE ⎫                ⎧ AFTER.NUMBER ⎫
  ? ⎨         ⎬ program-name   ⎨              ⎬ job-number   [control-attributes]
    ⎩ EX      ⎭                ⎩ AN           ⎭
```

The AFTER.NUMBER can be abbreviated as AN.

Example:

EXECUTE ALPHA AFTER.NUMBER 7

THEN

THEN

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

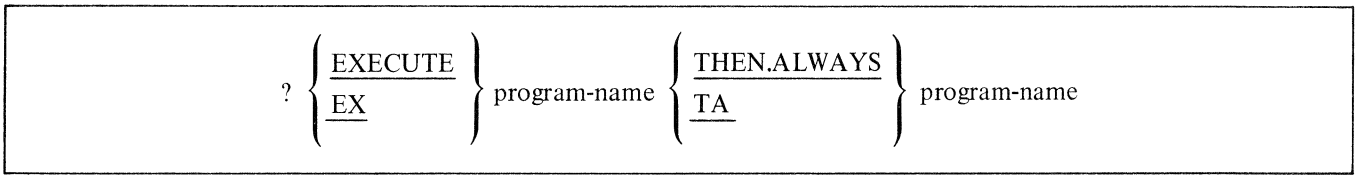
The format for the THEN statement is:

? { EXECUTE / EX } program-name { THEN / TH } program-name

THEN.ALWAYS

The THEN.ALWAYS statement is used to unconditionally schedule a program for execution which is independent of the other program's outcome.

The format for the THEN.ALWAYS statement is:



CHARGE

CONTROL INSTRUCTION ATTRIBUTES

CHARGE

The CHARGE instruction is used to insert a charge number into the MCP log for any or all programs.

The format of a CHARGE statement is:

```
? { CHARGE } [=] integer
  { CH     }
```

The control word CHARGE can be abbreviated as CH.

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.

FILE

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

The format of the FILE statement is:

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

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 or EXECUTE 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-identifier 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
```

Changes may be made permanent by using the MODIFY statement.

OBJ

When compiling, the OBJ option within a FILE statement must be included if the file information to be changed applies to the object-program. In such a case, the FPB of the object-program being compiled will be permanently changed to include such file information without the use of a MODIFY statement.

By omitting the OBJ option, the MCP assumes the statement is to apply to the compiler and will not affect the object-program.

FILE

FILE ATTRIBUTES

The FILE statement provides a vehicle by which to modify the original file-attributes at execution time without having to recompile or permanently alter the source program. For instance, a program written to output a paper tape file can be changed to output the same data to a punched card file, thus avoiding a recompilation.

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

FILE ATTRIBUTE	FUNCTION
ALLOCATE.AT.OPEN	All of the areas requested by this file will be allocated at the time the file is opened.
AREAS[=] integer	The number of areas assigned to the file at compile time will be altered to the value of the integer.
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.
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 devices 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.
CYLINDER.BOUNDARY	Each area of a disk file will start at the beginning of a CYLINDER when the file is directed to disk pack or disk cartridge.
DEFAULT	Override the disk allocation declared and use the file header block and record sizes. (Input disk files only.)

FILE ATTRIBUTE	FUNCTION
DRIVE[=] integer	The file will be directed to the drive specified by the integer. The drive must be one of the system disk(s). 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 halt and the MCP will display a message for the operator to load special forms in the device (printer or punch) when the file is opened.
HARDWARE	A printer or punch file will be allowed to go to the hardware device assigned.
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	A file to be processed as labeled (integer = 0) or unlabeled (integer = 1).
LOCK	The file will be LOCKED at program close or termination time.
MAXIMUM.BLOCK.SIZE[=] integer	Fixed block size will be used for variable length records.
NAME file-identifier	The external file-name or dp-id will be changed to the value of file-identifier. If only the dp-id is to be changed it must include the complete file-identifier also. If the file is opened input the program will expect the new dp-id and/or file-identifier to be in the system.
{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 to backup only 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.

FILE

FILE ATTRIBUTE	FUNCTION
	a. BACKUP b. BACKUP.DISK c. BACKUP.TAPE d. FORMS e. ALLOCATE.AT.OPEN f. LOCK g. DEFAULT h. HARDWARE i. PACK.SINGLE j. OPTIONAL k. VARIABLE l. CYLINDER.BOUNDARY m. INCREMENT.EU n. INCREMENT.DRIVE
ODD	The file will be changed to ODD parity.
OPTIONAL	Select optional file (COBOL only).
PACK.ID[=] disk-pack-id	Alter the pack-id.
PACK.SINGLE	A file will be restricted to one pack.
PSEUDO	Makes file a pseudo type.
RANDOM	The file will be changed to a RANDOM access file.
RECORDS.BLOCK[=] integer	The number of 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 reel number for the first read on a file that has a multiple reel file.
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.

The following list of HARDWARE TYPE ATTRIBUTES may be used to change the input or output device originally assigned to a file. When one of the hardware types is used the MCP will direct the file to that device when the file is opened.

READER.96
READER.PUNCH
READER.PUNCH.PRINTER
READER.SORTER
CARD.READER
CARD.PUNCH
DISK
DISK.PACK
DISK.FILE
DISK.FILE.1
DISK.FILE.2
DISK.CARTRIDGE
DISK.PACK.10
MFCU
TAPE.7
TAPE.9
TAPE.PE
TAPE
PAPER.TAPE.READER
PAPER.TAPE.PUNCH
PRINTER
PUNCH.96
PUNCH.PRINTER

FREEZE

FREEZE

The FREEZE control statement allows the system operator to prohibit rolling a program out to disk during its execution, thereby remaining in memory regardless of the situation until End-of-Job.

The format of the FREEZE control statement is:

```
? FREEZE
```

HOLD

The HOLD control statement allows the system operator to place a program into the schedule, but temporarily prohibiting its execution until it is forced (FS'ed) to execute.

The format of the HOLD message is:

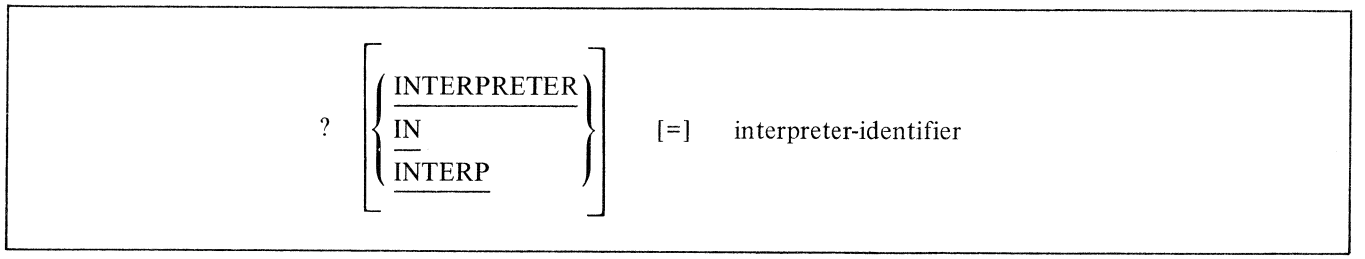
```
? HOLD
```

INTERPRETER

INTERPRETER

The INTERPRETER statement allows the system operator to select a different interpreter for use by a program.

The format of the INTERPRETER statement is:



INTERPRETER may be abbreviated as IN or INTERP.

Examples:

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

INTRINSIC.NAME

The INTRINSIC.NAME statement gives the system operator the ability to change the family-name of the intrinsics to be requested by a program.

The format of the INTRINSIC.NAME statement is:

? { INTRINSIC.NAME / IT } [=] 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 statement gives the system operator the ability to reference intrinsic files from a selected removable disk pack.

The format of the INTRINSIC.DIRECTORY statement is:

```
? { INTRINSIC.DIRECTORY } [=] dp-id
  { ID                  }
```

The INTRINSIC.DIRECTORY statement can be abbreviated as ID.

Example:

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

MEMORY

The MEMORY statement allows the system operator to override the dynamic memory assigned by the compiler for a given program at execution time.

The format of a MEMORY statement is:

```
? { MEMORY } [=] integer
  { ME     }
```

The control word MEMORY can be abbreviated as ME.

The program will terminate if there is not enough memory assigned to execute the program.

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 50000 bits of memory for the compiler. The following example will assign 50000 bits of memory for the execution of a program.

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

PRIORITY

PRIORITY

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

The format of a PRIORITY statement is:

```
? { PRIORITY } [=] integer
  { PR       }
```

The control word PRIORITY can be abbreviated as PR.

The PRIORITY statement gives the system operator 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 fifteen is specified, the following action occurs in a multiprogramming mode:

a. If necessary, jobs which are running and which have a lower priority than fifteen will be "rolled-out" from memory to disk to create space for the priority fifteen job. This action called "crashout" is the same as when the STOP instruction is introduced by the system operator.

b. A PRIORITY fifteen job entered in the schedule will not automatically suspend any other priority fifteen job running in memory, however, the system operator may STOP them providing they meet the other criteria for suspension.

c. Upon termination of the PRIORITY fifteen job, the suspended programs will be automatically reinstated to memory.

A PRIORITY of fourteen will cause a "crashout" in the same manner as a PRIORITY fifteen except it will not interfer with the execution of any PRIORITY fifteen already in the MIX.

UNFREEZE

The UNFREEZE statement 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
```

VIRTUAL.DISK

VIRTUAL.DISK

The VIRTUAL.DISK statement gives the operator the ability to expand the number of disk segments assigned for saving data overlays during execution.

The format of the VIRTUAL.DISK statement is:

? { VIRTUAL.DISK / VI } [=] integer

Integer must be eight digits or less.

FILE PARAMETER INSTRUCTIONS

DATA

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

The format of the DATA control instruction is:

? { DATA / DA } 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 label-equated to it in order for the MCP to automatically find the file.

It is the responsibility of the system operator which input device is to be used for the card 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 make 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

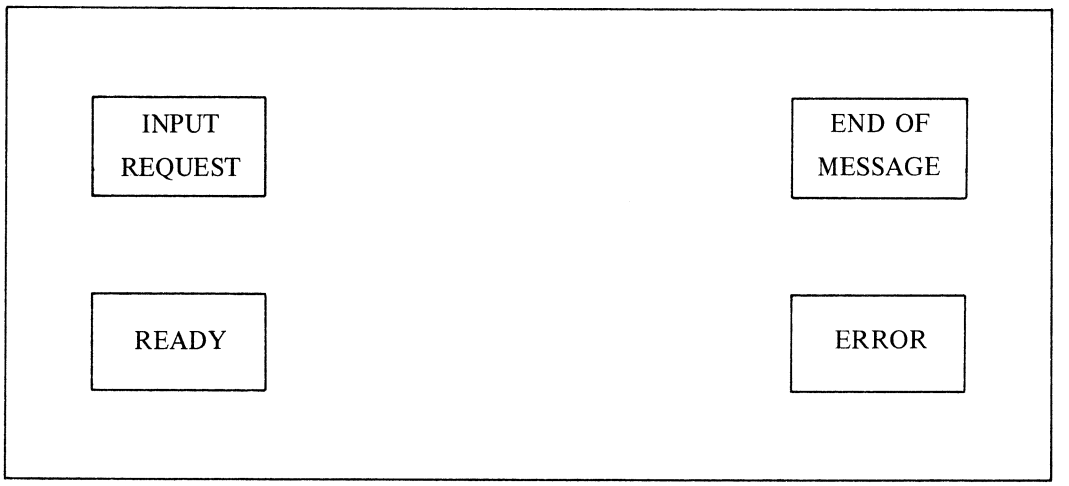
Information may be supplied to the MCP through the use of input messages entered through the console printer. These messages are referred to as keyboard input messages throughout this manual. The keyboard input messages are used by the system operator to communicate with the MCP. In order to make the 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 or the message is to be ignored, press ERROR prior to the END OF MESSAGE and retype the message.



Keyboard Input Messages

AX	(Response to ACCEPT)
BF	(Display Backup Files)
CD	(List Card Decks)
CI	(Change Interpreter)
CM	(Change MCP)
CN	(Display Physical Tape No.)
CQ	(Clear Queue)
DM	(Dump Memory and Continue)
DP	(Dump Memory and Discontinue)
DR DT	(Change MCP Date)
DS	(Discontinue Program)
ED	(Execute Psuedo Deck)
FM	(Response to Special Forms)
FN	(Display Internal File Names)
FR	(Final Reel of Unlabeled Tape File)
FS	(Force from Schedule)
GO	(Resume Stopped Program)
HS	(Hold Schedule)
IL	(Ignore Label)
KA	(Disk Analyzer)
KP	(Print Disk Sements)
LC	(Load Cassette)
LD	(Pseudo Load)
LG LN	(Transfer/Print LOG)
MC	(Assign System Charge Number)
MX	(Display MIX list)
OF	(Optional File Response)
OK	(Attempt to Continue Processing)
OL	(Display Peripheral Status)
OU	(Specify Output Device)
PB	(Print/Punch Backup)
PD	(Print Directory)
PG	(Purge)
PM	(Print Memory)
PO	(Power Off)
PR	(Change Priority)
PS	(PROD Schedule)
RB RF	(Remove Backup Files)
RD	(Remove Pseudo Card Files)
RM	(Remove Duplicate Disk File)
RN	(Assign Pseudo Readers)
RO	(Reset Option)
RP	(Ready and Purge)
RS	(Remove Jobs from Schedule)
RY	(Ready Peripheral)
SD	(Assign Additional System Drives)
SL	(Set LOG)
SO	(Set Option)
SP	(Change Schedule Priority)

Keyboard Input Messages

ST	(Suspend Processing)	UL	(Assign Unlabeled File)
SV	(Save Peripheral Units)	WD	(Display MCP Date)
TD	(Display Time/Date)	WM	(Display Which MCP and Interpreter)
TI	(Time Interrogation)	WS	(Display Schedule)
TL	(Transfer LOG)	WT	(Display MCP Time)
TO	(Display Options)	WY	(Program Status Interrogation)
TR	(Time Change)		

AX

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 INPUT MESSAGE (Display Backup Files)

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

The format of the BF message is:

BF [unit-mnemonic] { PRN/= | PRT/= | PCH/= | =/= }

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 card 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.

CD

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.

CI INPUT MESSAGE (Change Interpreter)

The CI input message gives the operator the ability to change the interpreter the MCP will use.

The format for a CI message is:

CI interpreter-identifier

The change will actually take place after the next CLEAR/START.

CM

CM INPUT MESSAGE (Change MCP)

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

The format of the CM message is:

```
CM  mcp-identifier
```

The change will actually take place after the next CLEAR/START.

CN INPUT MESSAGE (Display Physical Tape Number)

The CN input message will display the five-character physical tape number contained in the tape label.

The CN message format is:

CN unit-mnemonic

The unit specifier must reference a magnetic tape unit.

CQ

CQ INPUT MESSAGE (Clear Queue)

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

The CQ message format is:

CQ

DM INPUT MESSAGE (Dump Memory and Continue)

The DM input message allows the system operator to dump the entire contents of a program's memory space to disk for subsequent analysis by the 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. See the "PM" message.

Example:

```
2 DM (Dump the memory contents of a program with a mix-index of 2)
```

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, 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.

DR
DT

DR INPUT MESSAGE (Change MCP Date)
DT

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

The DR message format is

$$\left\{ \begin{matrix} \underline{\text{DR}} \\ \underline{\text{DT}} \end{matrix} \right\} \text{mm/dd/yy}$$

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 all program execution and return the memory the program occupied to the system. Any files not previously entered into the disk directory before the DS message are lost and the disk area returned to the disk available table.

The DS message causes the following message to be displayed on the console printer:

program-name DS-ED (time)

Example:

```
03 DS
```

ED INPUT MESSAGE (Execute Psuedo 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.

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 mix-index must be the same mix-index number that appeared in the request for the special forms action.

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

The message:

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

is output on the console printer requiring that a FM message be submitted by the system operator.

Example:

```
03 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 is at End-of-File.

The format of a 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-File; 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 reenter (force) jobs into the mix.

The format for a FS input message is:

FS { job-number / = }

The equal sign (=) option will force all jobs in the schedule for execution.

See the HS message for placing a job in a hold condition.

GO

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.

HS INPUT MESSAGE (Hold 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 schedule.

The format for the HS message is:

$$\text{HS} \left\{ \begin{array}{c} \underline{\text{job-number}} \\ \underline{=} \end{array} \right\}$$

The equal sign (=) option will place all jobs in the schedule in a hold condition.

The schedule number is assigned to the job when it is entered into the schedule by the MCP.

IL

IL INPUT MESSAGE (Ignore Label)

The IL input message allows the system operator either of two functions: (1) to designate the unit on which a particular input file is located, or (2) to assign a different disk drive for a disk file at the time the file is opened.

The IL message format is:

mix-index IL unit-mnemonic

The mix-index must be used to identify the program. In a multiprograming 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 file on 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.

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:

```
KA  [ { dp-id                  }
      { file-identifier        }
      { [dp-id] DSKAVL         } ]
```

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

The KA message entered by itself will print the disk directory of the system disk only.

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

The DSKAVL will print the areas available on the disk.

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

Examples:

```
KA
KA  DSKAVL
KA  dp-id
KA  file-identifier
```

KP

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 KP message format is:

```
KP { file-identifier   } [number of segments]
   { @disk-address@    }
```

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

The file-identifier options will print an entire file, or when used with the number of segments option will print just that number of segments of the file.

When the disk-address option is used, it must be the hexadecimal starting address of the area desired to be printed. If the number of segments option is omitted, only the data for that address will be printed.

If a disk file contains multiple areas, only the first area will be printed.

LC INPUT MESSAGE (Load Cassette)

The LC message is used to load system programs (compilers, interpreters, object code, system software) from a cassette 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 free-standing 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 everytime 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 on a load, not at End-of-Tape.)

2. Enter the LC message.

3. When the cassette finishes loading, the MCP will HALT. (RUN light goes off, L-register = Hex 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-Job.

LD

LD INPUT MESSAGE (Pseudo Load)

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

The LD message format is:

```
LD
```

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

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

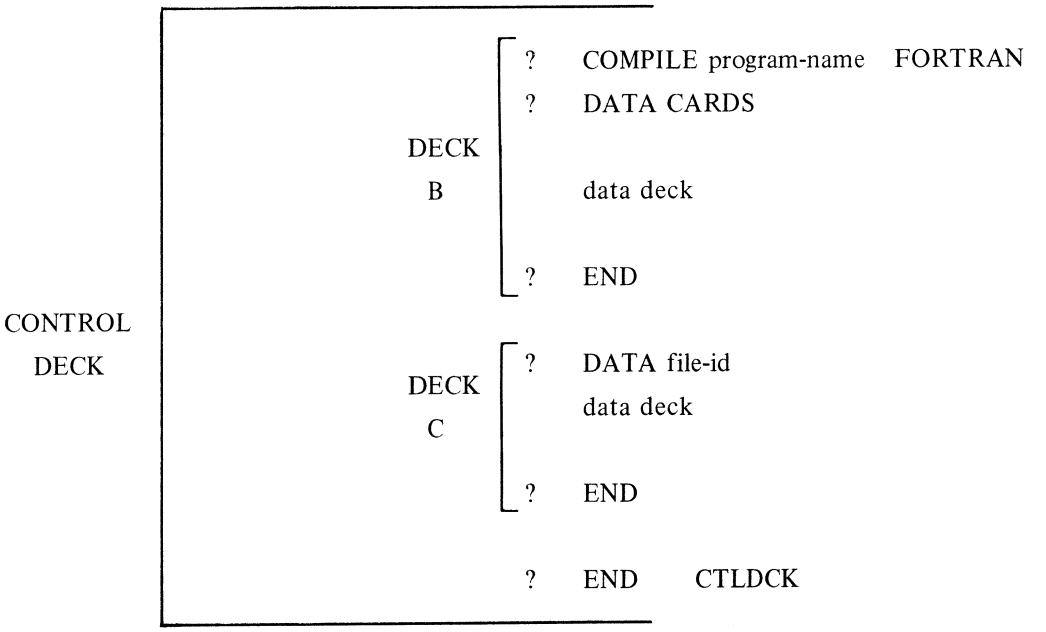
Terminating the LD function requires a "? END CTLDCK" control statement immediately following the last data deck that is to be read.

A DS may also be entered to terminate the function providing a "? END" card has been read previously,

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.

```
                                    (Load cards)
                                ?   DATA  CTLDCK

                                ?   COMPILE program-name   COBOL SYNTAX
CONTROL                     ?   DATA CARDS
DECK                    A

                                    data deck
                                ?   END
```



Enter LD

Enter RN digit (number of pseudo readers to be used)

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 to transfer the LOG.

MC INPUT MESSAGE (Assign System Charge Number)

The MC input message allows the system operator to set the CHRG option ON and to assign a six-digit charge number to all system-type programs.

The format of the MC message is:

```
MC integer
```

The MC message sets the CHRG option requiring that all programs executed have a charge number. Once the MC message has been entered, the charge option cannot be turned off without performing a COLD START.

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
PR:   4 program-name  =  01
PR:   4 program-name  =  02
END MIX
```

OF INPUT MESSAGE (Optional File Response)

The OF input message is used in response to the "NO FILE" or when a "duplicate file" situation exists.

The format for the OF message is:

```
mix-index OF
```

When a duplicate file situation exists, the OF allows the system operator to close the new output file with PURGE, and continue processing leaving the previous file in the disk directory.

The OF message in response to the "NO FILE" message indicates to the MCP the optional file being requested is to be bypassed for this execution. Usage in this context is subject to input files only.

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 f-id 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 input 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:

. . . IN USE BY program-name file name

. . . LABELED file-name

. . . NOT READY (The unit is assigned to tape and has a purged tape, with a write ring installed, mounted on the tape drive.)

. . . UNLABELED (The unit is assigned to tape and has an unlabeled tape, without a write ring installed, mounted on the tape drive.)

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

NULL type-code-entered 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:

```
04 OU DPC
```

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

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:

```
                                  { integer-1 }
PB [unit-mnemonic]                { PRT/=     }          [SAVE] [integer-2]
                                  { PCH/=     }
                                  { =/=       }
```

The "integer-1" option is the number given to the file by the MCP when the backup was performed and is used to access a single file for processing.

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:

Format 1

```
PD  { dp-id/=/= }   (removable pack)
    { =/=       }   (system pack)
```

Format 2

```
PD  {  file-identifier                }  . . .
    {  family-name/=                  }
    {  dp-id/family-name/=            }
```

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 specifier 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 compiler 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 non-system 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
            PD = 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 RESTRICTED or SYSTEM disk cartridge/pack that is purged will be redefined 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.

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 will cause the execution of the MCPI/ANALYZER or MCPII/ANALYZER program which will analyze and print the contents of SYSTEM/DUMPFILE. (Total Memory)

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

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

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

If a pack is removed from the system prior to a PO message, the MCP will retain the label of the removed pack regardless of the label of the new pack when mounted. If this does occur, a PO message may be entered followed by an RY message. This procedure, however, may cause loss of process time and operational confusion.

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 bring additonal programs into the MIX before the normal EOJ of a program.

The message format of the PS message is:

PS

The normal function of the MCP checks the schedule at each EOJ; however the PS message will cause the MCP to check the schedule when the message is entered.

{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:

{RB / RF} [unit-mnemonic] {PRN/= / integer / PRT/= / PCH/= / =/=}

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

The PRN,PRT/= and the PCH/= options will remove either all print backup files or all punch backup files respectively.

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:

```
RD { integer }
   { =/=    }
```

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 an output file with the same name as a file already in the directory. This causes the program to go into a wait state. The RM message will remove the old file, close the new file, enter it in the directory, and continue processing.

Example:

```
1 RM
```

RN

RN INPUT MESSAGE (Assign Pseudo Readers)

The RN message is used by the system operator to assign a specific number of pseudo card readers.

The format of the RN message is:

```
RN integer
```

The RN message can be entered either before or after the creation of pseudo files.

It is the responsibility of the operator to determine the optimum number of pseudo readers in relation to the number of pseudo files to be processed.

By entering RN 0 (zero) all pseudo card readers will be closed as soon as they are finished processing the file that they are presently reading.

The psuedo card readers may also be closed by performing a Clear/Start.

RO INPUT MESSAGE (Reset Option)

The RO message allows the system operator to reset the options used to direct or control some of the MCP functions.

The RO message format is:

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

An option in a set condition is equal to one (1). An option in a reset condition is equal to zero (0).

The options which may be reset with the RO message are:

BOJ	SCHM	PBD
DATE	TERM	PBT
EOJ	TIME	LAB
LIB	CLOS	RMOV
OPEN		

The TO message displays a complete list of the MCP options and their status (set (1) or reset (0)).

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

Example:

request: RO EOJ (Reset the EOJ option)

response: EOJ = 0

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

RP

RP INPUT MESSAGE. (Ready and Purge)

The RP message entered by the system operator will set a tape unit in "READY" status and "PURGE" the tape.

The format of the RP message is:

RP unit-mnemonic [unit-mnemonic] . . .

The RP message can be used for tape only.

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 message format for the RS message is:

```
RS { job-number-1   [job-number-2] ... }
   {      =                            }
```

The RS message can remove one or more jobs from the schedule.

The schedule number is the number assigned to the job by the MCP when it is entered into the schedule.

The job-number will be displayed by the MCP when the job is entered into the schedule if the SCHM option is set. The WS message will display the jobs in the schedule by 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

program-name 33 RS-ED
program-name 34 RS-ED
program-name 35 RS-ED
36 NULL SCHEDULE              (job 36 not in schedule)
```

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 message format for 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, and the locked or saved files made accessible to the MCP. After the unit has been made ready, the MCP attempts to read a file label (input devices only).

Examples:

```
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:

SD unit-mnemonic serial-number

The SD message, after verification of the serial-number, will PURGE the pack, and add it to the system packs already on the system.

At COLD START, 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 COLD START.

The following message is displayed when the new system drive is linked to the system.

DP serial-number IS NOW A SYSTEM PACK–CLEAR START REQUIRED

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 the area which is to be assigned to the LOG and cannot be less than 100 or greater than 1000.

The integer-2 option is the maximum number of these areas desired. The default is 40.

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

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 [option-name] ...
```

The options which may be set with the SO message are:

BOJ	SCHM	PBD
DATE	TERM	PBT
EOJ	TIME	LAB
LIB	CLOS	RMOV
OPEN		

The option indicator equals one when set and zero when reset.

To determine which options are set at any given time, the TO input message may be entered. The MCP will then display a complete list of the MCP options and their status.

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

The LOG and CHRG options cannot be set with an SO message. The MCP message "LOG LOCKED" or "CHARGE LOCKED" will be displayed when an attempt has been made to set these with an SO message.

SP

SP INPUT MESSAGE (Change Schedule Priority)

The SP input message provides a means for the system operator to change the priority of a program currently in the schedule.

The message format of the SP message is:

SP job-number integer

The "Schedule Priority" is not the same thing as 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 15 where zero is the lowest priority and 15 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
```

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 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.

There is no MCP acknowledgment that a program has been suspended.

Example:

```
3 ST
```

SV

SV INPUT MESSAGE (Save Peripherial 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
```

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 seconds.

Example:

request: 4 TI

reply: program-name = mix-index CPU TIME 000073.6 SEC

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:

```
TR integer
```

The time specified by the integer is designated according to a 24-hour clock.

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

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 01 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:

```
01 UL MTA
```

If the first three blocks on the tape are not desired, they can be skipped with the following UL message:

```
01 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 Which MCP and Interpreter)

The WM input message allows the system operator to inquire which MCP and Interpreter is 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:

CURRENT MCP IS mcp-name USING interpreter-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:

```
WS { job-number }
   {    =     }
```

The job-number is assigned by the MCP as it 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 number of seconds the program has been in the schedule.

Example:

```
WS 4

ALPHA = 4 NEEDS 8 KB PR = 4 IN FOR 000789.9 SEC
```

WT

WT INPUT MESSAGE (Display MCP Time)

The WT input message permits the system operator to request the time used by the MCP. The reply is in the twenty-four hour clock format.

The WT input message format is:

```
WT
```

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 indentifies the program in the MIX that is to be checked and its status displayed on the SPO. 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  SUSPENDED BY OPERATOR

ALPHA  = 01  SUSPENDED BY OPERATOR
BETA   = 02  EXECUTING
DELTA  = 03  AX-WAITING FOR KEYBOARD INPUT
ECHO   = 04  IN COMMUNICATE QUEUE

5 WY
INVALID MIX NUMBER
```

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.

Some messages have been omitted mainly because of their simplicity or their self-explanatory nature.

Suggestions and/or directions to the system operator as to what actions to take in regard to the messages have been omitted, mainly because there is usually a choice of two or more alternatives available for a given situation, as well as prior instructions as to what course an operator should take.

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. Any optional type entries are to be ignored and not considered as part of the key. The job-specifier portion of the message is also to be ignored.

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 option is only printed when the executing program is in a compiler.

Terminal-info: The phrase "terminal-info" 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 MCP 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 last two entries 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-info

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 # nnnnnnNOT REMOVED–NOT ON DISK

BACKUP TAPE NOT FOUND–"ṘY" unit-mnemonic

BATCH COUNT COMMUNICATE ISSUED WHILE SORTER FLOWING terminal-info

job-specifier–BEGINNING DATA OVERLAY ADDRESS = nnnn,WHILE BR = nnnn terminal-info

job-specifier–BOJ # = job-number PR = nn TIME = hh:mm:ss.s

job-specifier–CANNOT ACCEPT "[IL'UL'OF'FR'FM'OU'OK'RM' or QT]" 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:ss.s

***CLEAR/START REQUIRED

CLEAR/START REQUIRED–SYSTEM/PRINTCHAIN MISSING

COMPILE program-name CTRL RCD ERR: . . .

job-specifier–CONTROL STACK OVERFLOW terminal-info

job-specifier–"CONVERT" ERROR terminal info

COULD NOT CHANGE THE MCP

job-specifier–CPU TIME = nnnnnn.n (seconds)

CURRENT MCP IS identifier USING interpreter-id

job-specifier–DATA OVERLAY RELATIVE DISK ADDRESS = nnnn, WHILE SIZE OF AREA = nnnn terminal-info

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 FROM MEMORY ADDRESS @nnnn@ }

job-specifier–DISK FILE DECLARED SIZE EXCEEDED ON file-identifier terminal-info

job-specifier–unit-mnemonic DISK PARITY @nnnn@

job-specifier–nnnnDISK SEGMENTS REQUIRED FOR AREA OF file-identifier

job-specifier–DIVIDE BY ZERO terminal-info

**DR PLEASE

job-specifier–DS-ED TIME = hh:mm:ss.s

job-specifier– DUPLICATE INPUT FILES file-identifier

END BF

END MX

END PD

job-specifier–ENDING DATA OVERLAY ADDRESS = nnnn, WHILE BR = nnnn terminal-info

"=" 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-info

EXECUTE program-name CTRL RCD ERR: . . .

job-specifier–EXPONENT OVERFLOW terminal-info

job-specifier–EXPONENT UNDERFLOW terminal-info

job-specifier–EXPRESSION OUT OF RANGE terminal-info

job-specifier { INPUT / OUTPUT / INPUT/OUTPUT } FILE file-identifier CLOSED { RELEASE / PURGE / REMOVE / CRUNCH / NO REWIND / CODE / LOCK / CONDITIONAL / ROLLOUT / TERMINATE }

job-specifier–FILE internal-file-identifier LABELED . . . REEL nnnnnn NOT PRESENT

job-specifier–FILE internal-file-identifier NEEDS nnnn BITS TO OPEN, WHICH I COULDN'T FIND–"OK" WILL TRY AGAIN, ELSE "DS"

file name "file-identifier" REQUESTED BY "FN" NOT FOUND

FN = "internal-file-identifier"

FREE UP SOME DISK AND CLEAR/START

GOOD MORNING, TODAY IS name-of-day, hh:mm:ss.s { AM / PM } JLN DT = yy/ddd

unit-mnemonic HAS nnnn USERS

unit-mnemonic HAS BEEN PURGED

job-specifier–unit-mnemonic HOPPER EMPTY

INVALID BIT CHARACTER– . . .

INVALID BIT SPECIFIER– . . .

INVALID CHAR COL nn

INVALID CHARACTER . . .

INVALID CHANGE–PACK-IDS DO NOT AGREE . . . .

job-specifier–INVALID CASE terminal-info

job-specifier–INVALID COMMUNICATE IN USE ROUTINE terminal-info

unit-mnemonic INVALID CONTROL CARD

INVALID DECK NUMBER . . .

INVALID ED MESSAGE DECK NUMBER

job-specifier–INVALID INDEX terminal-info

INVALID JOB NUMBER

INVALID MC-CHARGE OPTION ALREADY SET

INVALID MIX NUMBER

INVALID MNEMONIC . . .

job-specifier–INVALID LINK terminal-info

job-specifier–INVALID OPERATOR terminal-info

INVALID PACK.ID OR TAPE MNEMONIC FOR PB . . .

job-specifier–INVALID PARAM TO VALUE DESC terminal-info

job-specifier–INVALID PARAMETER terminal-info

INVALID PG

job-specifier–INVALID RETURN terminal-info

INVALID SD–SERIAL NUMBER REQUIRED

INVALID SERIAL NUMBER

INVALID SL–LOG ALREADY SET

job-specifier–INVALID SUBSCRIPT terminal-info

job-specifier–INVALID SUBSTRING terminal-info

INVALID SYNTAX FOR CHANGE OR REMOVE, COMMA IS REQUIRED FOR MORE THAN ONE CHANGE

unit-mnemonic INVALID TYPE CODE . . .

INVALID unit-mnemonic

INVALID UNIT MNEMONIC FOR FN, MUST BEGIN WITH ALPHA

"IL" REQUIRES A PARAMETER

file-identifier IN USE

job-specifier–INSUFFICIENT MEMORY TO OPEN file-identifier

job-specifier IS EXECUTING

pack-id IS ALREADY A SYSTEM DRIVE

pack-id IS A NONREMOVABLE SYSTEM PACK OR IS ALREADY OFF LINE

pack-id IS AN INTERCHANGE PACK

unit-mnemonic IS NOT A USER PACK

pack-id IS NOT INITIALIZED

job-specifier IS NOT STOPPED

pack-id IS { RESTRICTED / INTERCHANGE } PACK

job-specifier IS SUSPENDED

job-specifier–INTEGER OVERFLOW terminal-info

INTRINSIC "intrinsic-name" REQUESTED BY program-name = job-number IS NOT IN DIRECTORY–RS–ED

INV OPTION option-name

unit-mnemonic LABELED . . . . REEL nnnnnn

unit-mnemonic LABELED . . . . [S,R,U, or I] SERIAL.NO = nnnnnn

unit-mnemonic { LABELED . . . / UNLABELED } IN USE BY J-S . . .

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

MC REQUIRES 6-DIGIT MCP CHARGE NUMBER

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

MC REQUIRES NULL MIX

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 | BLANK 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-info

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-info

job-specifier–NO PROVISION FOR END OF FILE ON file-identifier terminal-info

NO PSEUDO DECKS ON DISKS

job-specifier–NO ROOM TO OPEN FILE file-identifier

file-identifier NOT IN DIRECTORY

file-identifier NOT IN DISK DIRECTORY

"=" NOT PERMITTED IN PROGRAM NAME FOLLOWING "FN"

file-identifier NOT LOADED–IN USE BY SYSTEM

file-identifier NOT { LOCKED / REMOVED } INVALID PACK-ID pack-id

file-identifier NOT ON DISK

pack-id NOT ON LINE

unit-mnemonic NOT READY

NULL SCHEDULE

NULL . . . TABLE

NUMBER OF PSEUDO READERS CHANGED TO nnnnnn

unit-mnemonic OFF LINE

OUT OF MEMORY SPACE

job-specifier–OUTPUT UNIT NOT AVAILABLE FOR BACKUP

job-specifier–unit-mnemonic { PARITY ERROR / ACCESS ERROR } –NO RECOVERY

PM CANNOT FIND DUMPFILE/integer FOR DUMP/ANALYZER

job-specifier–POCKET LIGHT COMMUNICATE REQUESTED WHILE SORTER FLOWING terminal-info

job-specifier–PRIORITY CHANGED TO new-priority-number

job-specifier–unit-mnemonic PRINT CHECK

PRINTER NOT READY

job-specifier–PROGRAM ABORTED terminal-info

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

mag-tape-id = { PURGED LABEL / TAPE-ID file-identifier [REEL# nnnnnn] / UNLABELED }

unit-mnemonic READ CHECK

job-specifier–READ OUT OF BOUNDS terminal-info

pack-id RELABELED now-pack-id [S,R,U, or I]

job-specifier–REQUESTED A [CODE or DATA] SEGMENT OF LENGTH ZERO terminal-info

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-info

program-name REQUESTED BY "FN" NOT IN DIRECTORY

program-name REQUESTED { READ / WRITE / SEEK } ON CLOSED FILE

{ RO / SO } REQUIRES THREE OR FOUR CHARACTERS

device-mnemonic REQUIRED FOR REEL # nnnnnn file-identifier

REQUIRES MIX NO.

program-name # job-number RS-ED

unit-mnemonic REWINDING

unit-mnemonic { SAVED / TO BE SAVED }

SD REQUIRES NULL MIX

SCHEDULED: program-name # = Job-number PR = nn hh:mm:ss.s

job-specifier–SEEK REQUESTED ON SERIAL FILE file-identifier terminal-info

nnnn SEGS REQ FOR SYSTEM DUMP FILE

SERIAL NUMBER REQUIRED

SPACE REQUIRED BEFORE " or @ . . .

job-specifier–STACK OVERFLOW terminal-info

job-specifier–SUPERFLUOUS EXIT terminal-info

SYSTEM/LOGOUT NOT IN DIRECTORY

job-specifier–TANK OVERFLOW terminal-info

3 DISK SEGMENTS NEEDED FOR SYSTEM/PRINTCHAIN

THERE ARE NO ENTRIES IN LOG . . . NO TRANSFERS OCCURRED

THERE ARE NO RELEVANT BACKUP FILES–PB IGNORED

***THERE IS NO BACKUP PRINT OR PUNCH FILE WITH NUMBER nnnnnn [ON PACK-ID]

job-specifier–unit-mnemonic TIMEOUT @nnnnnn@

TOKEN TOO LONG–REQUEST IGNORED . . . .

job-specifier–TOO LONG IN USE ROUTINE

TOO MANY "=" IN NAME . . . TRY AGAIN

TOO MANY "/"-S IN NAME . . . TRY AGAIN

job-specifier–TRIED TO INITIALIZE A GLOBAL BLOCK LARGER THAN ENTIRE STATIC SPACE
REQUESTED STATIC = nnnn GLOBAL = nnnn –RS-ED

job-specifier–TRIED TO { SEND TO / RECEIVED FROM } "program-name" WHICH IS NOT RUNNING

**TR PLEASE

job-specifier–UNDEFINED RUN TIME ERROR terminal-info

job-specifier–UNEXPECTED POCKET SELECT terminal-info

job-specifier–UNINITIALIZED DATA ITEM terminal-info

mag-tape-id UNIT PURGED

job-specifier–unit-mnemonic { NOT READY / JAM / MISSORT }

UNIT-MNEMONIC MUST START WITH ALPHA

unit-menemonic UNLABELED

pack-id WRITE–LOCKOUT

job-specifier–WRITE REQUESTED ON INPUT FILE file-identifier terminal-info

ZIPPED AN INVALID CONTROL CARD

SECTION 3
SYSTEM SOFTWARE

DISK CARTRIDGE INITIALIZER

General

A disk cartridge must be initialized before it can be presented to the system for operation. The purpose of the disk initialization is twofold. One, it checks to see what segments, if any, are unusable (cannot be read from or written to). If any flaws occur in track ZERO or ONE, the entire pack is considered faulty and cannot be used on the system. Two, it assigns addresses to the appropriate segments to be used by the MCP.

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.

Card Columns	Description
1	Drive Number (usually drive zero)
3-8	Disk Cartridge serial number
10-19	Label
21	TYPE of cartridge S = System U = Unstricted 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 initialized 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.

B1700 DISK CARTRIDGE INITIALIZER

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 used on the system.

SYSTEM LOADING PROCEDURES

General

The Cold Start routine is used to initialize the system. The routine is furnished on a cassette tape and is loaded via the control panel cassette reader .

The following actions occur while performing a Cold Start:

a. The Cold Start routine is loaded from cassette.

b. The Cold Start routine is executed and performs the following:

1. Constructs and initializes the disk directory on the system's disk(s).

2. Loads the MCP from cassette or magnetic tape to system disk.

3. Loads the SDL Interpreter from cassette or magnetic tape to the system disk.

4. Displays a message on the console printer instructing the operator to perform a Clear/Start.

NOTE

When a Cold Start 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 Cold Start routine.

The first section of the Cold Start loads the bootstrap loader. Once loaded it then loads GISMO, an abbreviated version of the SDL Interpreter, and the Cold Start routine. The run structure is then switched to Cold Start and that routine begins.

Cold Start creates a table containing the Cold Start variables. This table is used by the MCP for system control. It contains information regarding the MCP disk addresses, MCP name, Options, etc. This table is primarily used on Clear/Start.

Cold Start

The Cold Start operating procedures are as follows:

a. Mount a system pack on drive 0.

b. Set MODE switch to TAPE.

c. Place the Cold Start cassette in the cassette reader. Cassette is automatically rewound.

d. Press CLEAR, then START buttons.

e. Cassette will read a few feet and the system will HALT.

f. Set MODE switch to RUN, press START button.

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 the Cold Start will begin execution.

During execution of the Cold Start, a number of messages are displayed, and certain actions are required of the system operator. These messages and their responses are as follows:

Message	Response
DRIVE 0 IS INTERCHANGE OR IS NOT SYSTEM PACK	The system will halt, and must be re-started using a disk cartridge that has been initialized for system use.
WHAT MCP	Type in MCP name and press "End of Message"
NO DISK AVAILABLE	The addresses on the disk have been destroyed. The disk must be re-initialized.
MOUNT MCP CASS - START WHEN AT BOT	Mount reel 1 of MCP. Press "Start" when the "BOT" light is lit.
MOUNT 2ND CASS - START WHEN AT BOT	Mount reel 2 of MCP. Press "Start" when the "BOT" light is lit.
WHAT INTERPRETER	Enter Interpreter name then "End of Message".
MOUNT INTERPRETER CASS - START WHEN AT BOT	Mount Interpreter cassette Press "Start" when the "BOT" light is lit.
WRONG CASSETTE - TRY AGAIN	Insert correct cassette and press "Start".
CLEAR START REQUIRED	Perform a Clear/Start.

The system disk created by COLD START 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 at any time. Also, the LOG option may be set "on" at any time, and the SYSTEM/LOG file created on disk.

CLEAR/START

GENERAL

A Clear/Start is used by the system operator to restore the system to an operable state. A Clear/Start must be performed when any of the following conditions are present: (1) power-up of the system, (2) the SDL interpreter is

changed, (3) an unscheduled halt, (4) an uninterruptable program loop, or (5) the MCP is changed.

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. Reinitializes the MCP.

e. Returns control to the MCP.

f. Marks all peripherals unassigned.

g. Displays the "Clear/Start" message on the console printer.

If the processor is running at the time a Clear/Start is to be performed, the INTERRUPT switch should be used to bring the system to an orderly halt.

CLEAR/START PROCEDURE

a. Place Clear/Start cassette in cassette reader.

b. Set MODE switch to TAPE position.

c. Press HALT.

   Press CLEAR.

   Press START. (This loads bootstrap from cassette and processor will come to a halt.)

d. Set MODE switch to the RUN position.

e. Press START.

f. Cassette will complete the read and place the system under control of the MCP.

DISK FILE COPY

General

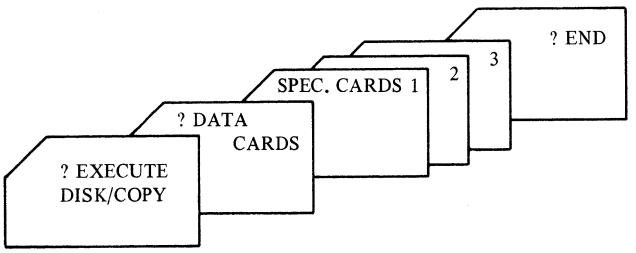
The DISK/COPY program will copy one or more disk files from one disk to another or to another location on the same disk.

All files, data or program, can be copied except those files marked as "ABSOLUTE MCP" or "LOG".

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.



DISK/COPY Control Deck
Figure 3 - 1

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/

MEMORY DUMP

The memory dump routine will dump the contents of memory to a disk file called SYSTEM/DUMPFILE to be analyzed with MCPI/ANALYZER or MCPII/ANALYZER.

Memory Dump Procedure

a. Record the contents of the following registers: X, Y, T, L, and A.

b. Set MODE switch to TAPE.

c. Load MEMORY DUMP cassette in cassette reader.

d. Press CLEAR.
   Press START.

e. Cassette will read for about ten seconds, then it will HALT with @AAAAAA@ in the L register.

f. Set MODE switch to RUN.

g. Press START.

h. Memory will be dumped to disk, after which the system will HALT with @AAAAAA@ in the L register. If the contents of memory have been destroyed to the point where a dump is not possible the system should hang. However, this may not always be true.

i. Perform a Clear/Start.

j. Enter PM on the console printer.

DMPALL

General

The program DMPALL has two separate functions: (1) printing the contents of files, and (2) reproducing data from hardware device to hardware device. Execution may be from either a keyboard console 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 device.

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 Insturctions

KEYBOARD CONSOLE

DMPALL executed from the keyboard console responds with the following three messages:

a. DMPALL = mix-index BOJ.
b. DMPALL = mix-index ENTER SPECS.
c. DMPALL = mix-index ACCEPT.

The operator replys to the ACCEPT message by entering an AX message containing the specifications needed to perform the DMPALL operation. There cannot be continuation lines with the AX message.

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 period will 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, and may be separated by either a space or a comma, or a combination thereof.

Multiple Specification Files

There may be two or more specification files executed with only one call to DMPALL. When the specifications have been acted upon for the first specification-file, DMPALL will call for another DATA card and its specifications.

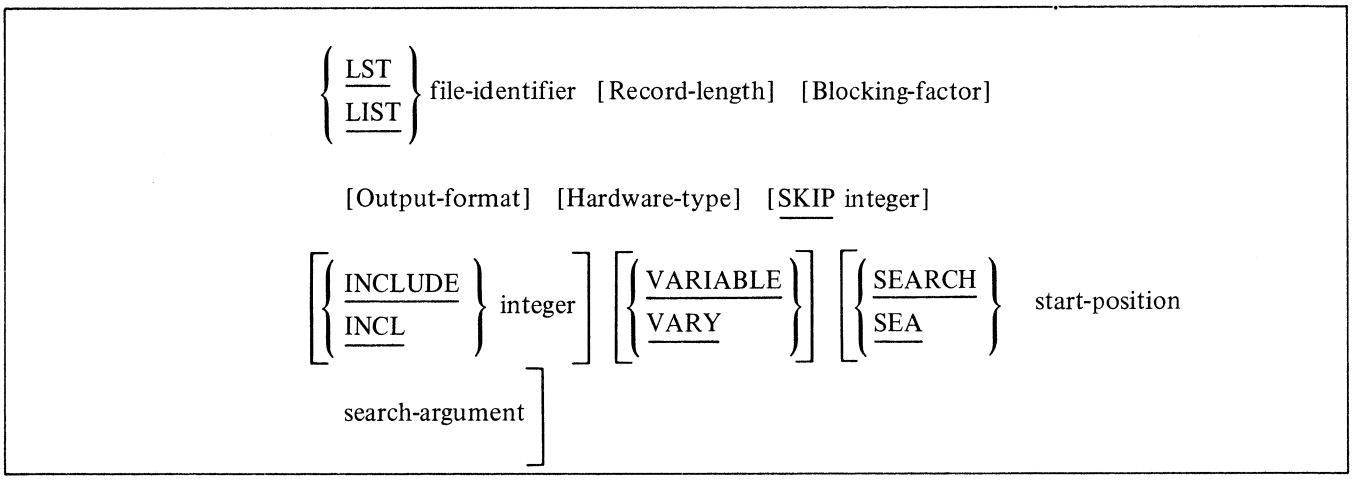
When the last set of specifications have been executed, the DMPALL program will terminate by reading a DATA card and then an END card.

For example:

```
? EXECUTE DMPALL FILE SPEC NAME SPECS

? DATA SPECS
  (specification cards)

? DATA SPECS
  (specification cards)

? DATA SPECS

? END
```

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.

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.

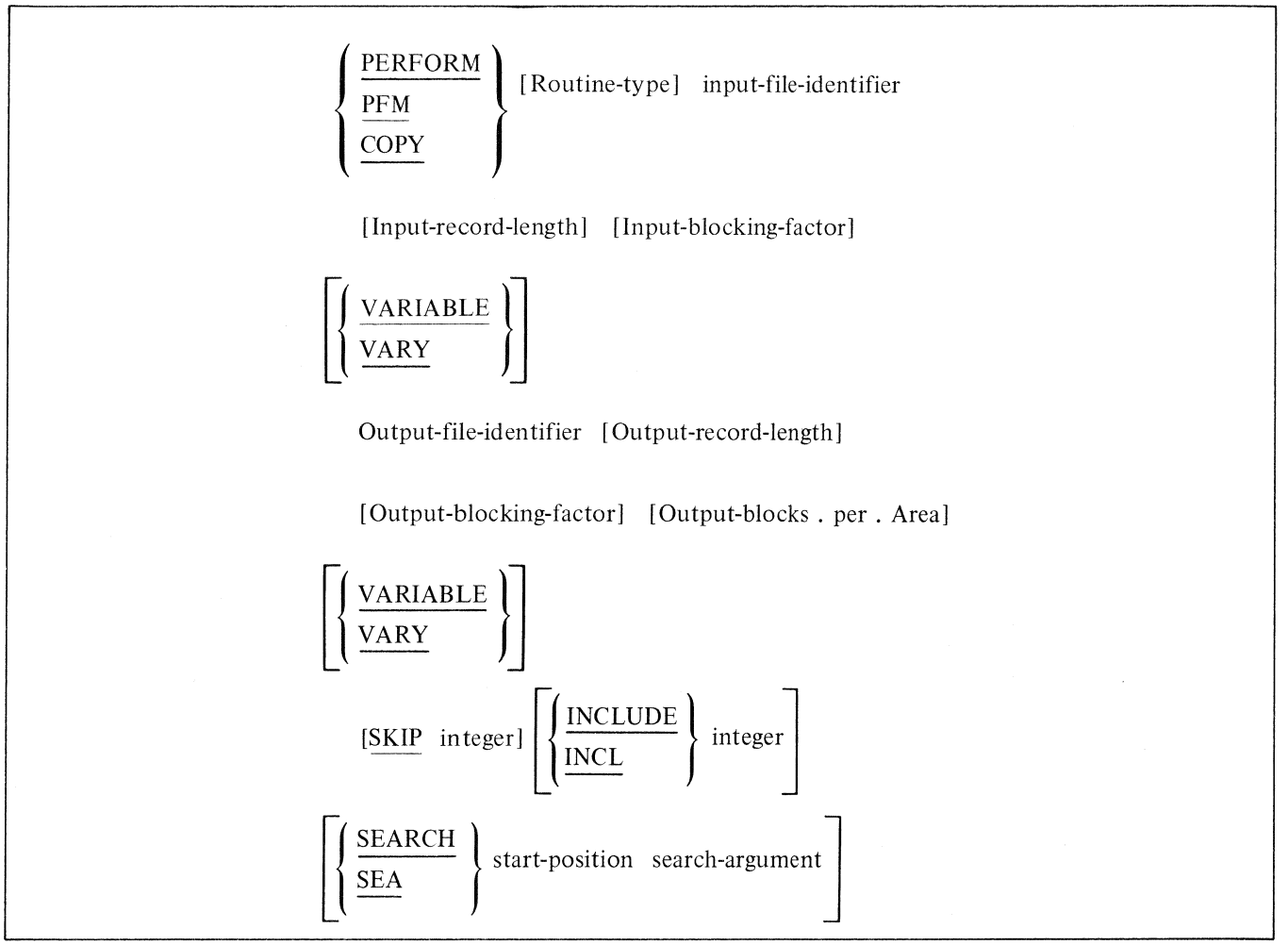
The VARIABLE or VARY entry may be used to specify tape or disk files having variable length records.

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 has the number of 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.

The short hand form uses a combined abbreviation format.

OUTPUT DEVICES

INPUT DEVICES: From↓ To→	Card	Mag. Tape	Paper Tape	Disk
Card	CRDCRD	CRDMTP	CRDPPT	CRDDSK
Mag.Tape	MTPCRD	MTPMTP	MTPPPT	MTPDSK
Paper Tape	PPTCRD	PPTMTP	PPTPPT	PPTDSK
Disk	DSKCRD	DSKMTP	DSKPPT	DSKDSK

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 40 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 output 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 the specifications via a card reader. DMPALL will look for a card file with the file-identifier SPECCARDS.

```
? EXECUTE DMPALL FILE SPEC NAME SPECCARDS
? DATA SPECCARDS
LIST XXX A CRD
? DATA XXX
  (card data deck)
? END
```

The specifications will cause the card file XXX to be 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.

Asterisk Card

The asterisk card is used to input the values for the file which is being loaded to disk. This card is output 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 format of the FILE/LOADER asterisk card is:

Column	Description	
1	"*" Asterisk Sign	
3	File Type 0 Invalid for FILE/LOADER 1 LOG 2 Invalid for FILE/LOADER 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
? END
```

Error Messages

MISSING "$" IN COLUMN ONE

The first card of the input deck does not have a "$" in column one.

MISSING file-identifier

The first card of the input deck has a "$" in column one, but is otherwise blank.

SEQUENCE ERROR FOLLOWING #nnnnnnn–file-identifier NOT LOADED

The card following the card number specified is out of sequence.

RECORD.SIZE SPECIFIED nnnn–file-identifier NOT LOADED

AREAS SPECIFIED = 0 – file-identifier NOT LOADED

RECORDS.BLOCK SPECIFIED = 0 – file-identifier NOT LOADED

SEGMENTS.AREA SPECIFIED = 0 – file-identifier NOT LOADED

EOF.POINTER SEPCIFIED = 0 – file-identifier NOT LOADED

INVALID FILE TYPE SPECIFIED–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 card with "$$" specified in columns 1 and 2 is invalid.

file-identifier LOADED

After each file is loaded, the above message will be output.

FILE/PUNCHER

General

The purpose of FILE/PUNCHER is to output disk files to cards in hexadecimal format. 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.

Error Messages

file-identifier NOT ON DISK

The file-identifier requested for output cannot be located by the MCP.

CANNOT PUNCH AN ABSOLUTE MCP

An ABSOLUTE MCP file cannot be output by the program FILE/PUNCHER.

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.

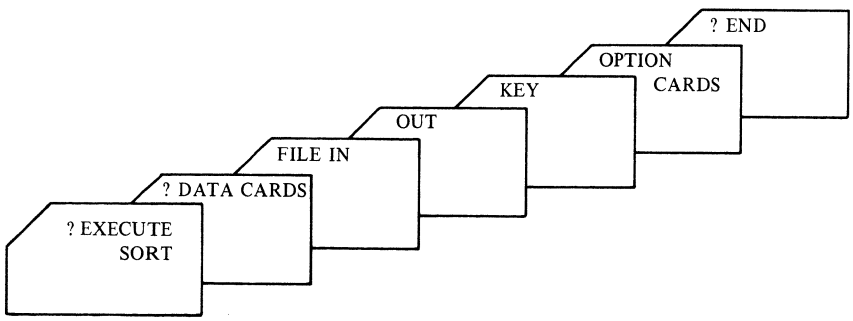
There are two sort intrinsics hereafter referred to as the vector replacement and INPLACE techniques. The intrinsic using the vector replacement technique is normally the one invoked.

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.



Sort Execution Deck

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.

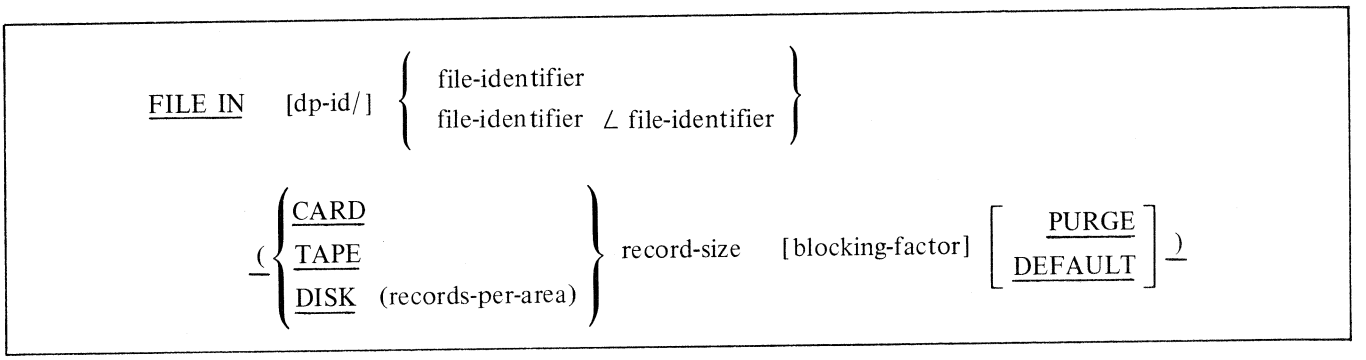
A description of each of the SORT specification cards (statements) appears in the following pages.

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:



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 file names as described in the MCP portion of this manual.

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.

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   [dp-id/] { file-identifier                   }
               { file-identifier ∠ file-identifier }

      ( { CARD                       }
        { DISK                       }
        { TAPE    (records-per-area) }  record-size  [blocking-factor]  )
        { PRINTER                    }
```

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 )
```

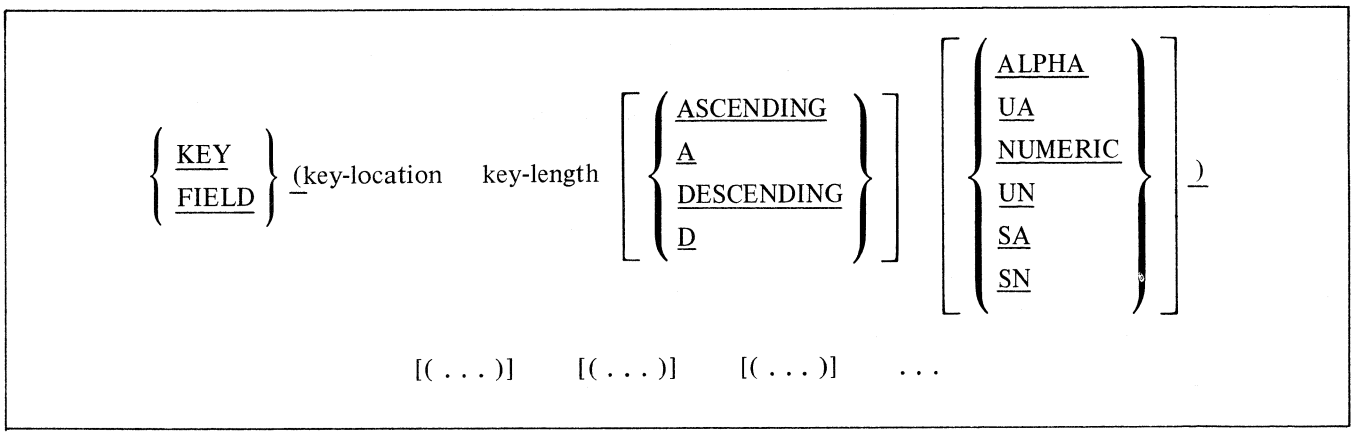
Note that in the above example parentheses serve as delimiters between parameters so that additional spaces are permitted but not required.

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 forty unsigned keys, twenty signed keys, or any combination not exceeding forty 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 succeedingly smaller keys.

The following SORT reserved words are used to describe the type of data within the key fields of the records to be sorted.

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.

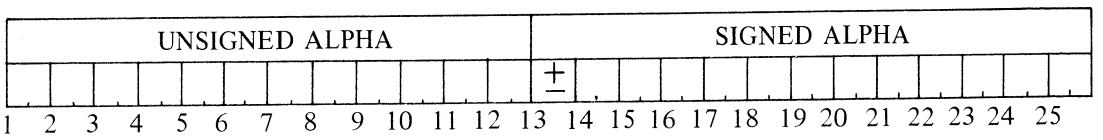
The minus sign is represented as a hexadecimal D in the most significant four bits of the first byte in the field.

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.

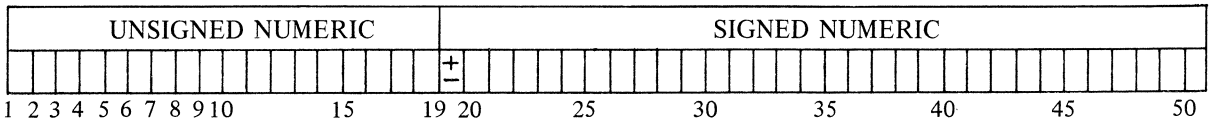
KEY (5 2) describes the field starting with the fifth byte that is two bytes long. The data is type ALPHA and the output sequence is to be ascending order.

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 number

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

number 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 number[%]

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.

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 named 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.

SORT Reserved Words

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

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

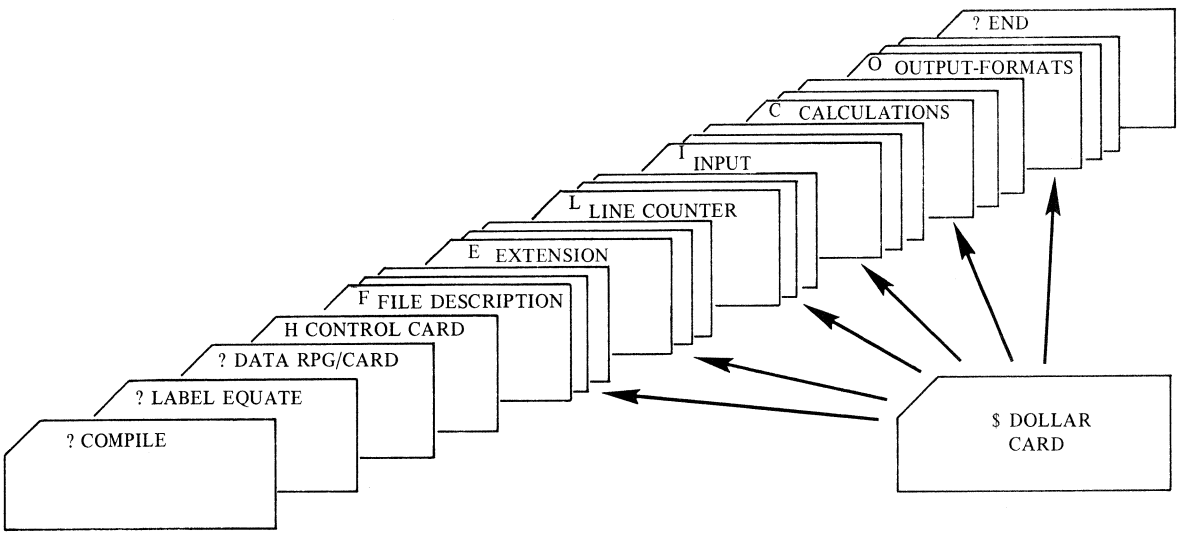
The Report Program Generator (RPG) enables the user to obtain comprehensive reports from existing files with a minimum time involved in source coding. An object program produced from RPG source coding is in the COBOL S-Language format.

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.



RPG Compilation Deck
Figure 4-1

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:

Columns	Description
1-5	Page and Line Sequence Number
6	This field may be left blank or contain the form type to align with the associated form that the $ option was inserted in.
7	A $ sign must appear in this field.
8	This field is used to specify that the option entered in the KEY WORD field is set ON or OFF. (Blank = ON, N = OFF).
9-14	KEY WORD: This field is used to name the option that is to be used. The option must be left-justified.
15-24	VALUE: This field is used to specify a value to be associated with the option. All values in alphanumeric form must be left-justified, numeric form must be right-justified.
25-74	COMMENTS: This field is available for comments and documentary remarks.
75-80	Program Name

RPG Extensions

The following options may appear only within the file description specifications, and must immediately precede the specification line describing the file to which they apply.

NOTE

None of the following operations may be "reset".

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-Job.

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 Directed 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 usage. 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 operation. 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 Label Equation 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
Figure 4-2

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

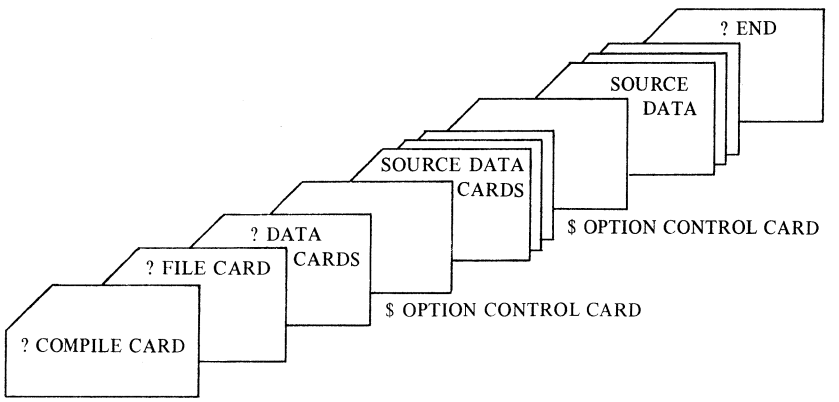
General

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.



COBOL Compilation Deck
Figure 4-3

$ OPTION Control Card

The third card, excluding label equation 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 (on) or reset (off) from that point on.

f. Columns 1 - 6 are used for sequence numbers.

The format of the $ OPTION card is as follows:

```
? OPTION [OPTION . . .]
```

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 re-sequencing, 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 re-sequencing starts at the beginning of the program source statements the sequence will start with 000010.

m. SEQ nnnnnn–starts re-sequencing 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 re-sequencing 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 re-sequencing starts at the beginning of the program source statements, the sequence will start with 000010.

o. SEQ nnnnnn +nnnnnn–starts re-sequencing 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 a 1000 bits.

v. NOCOP–when used will generate COP enteries 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
Figure 4-4

FORTRAN COMPILER

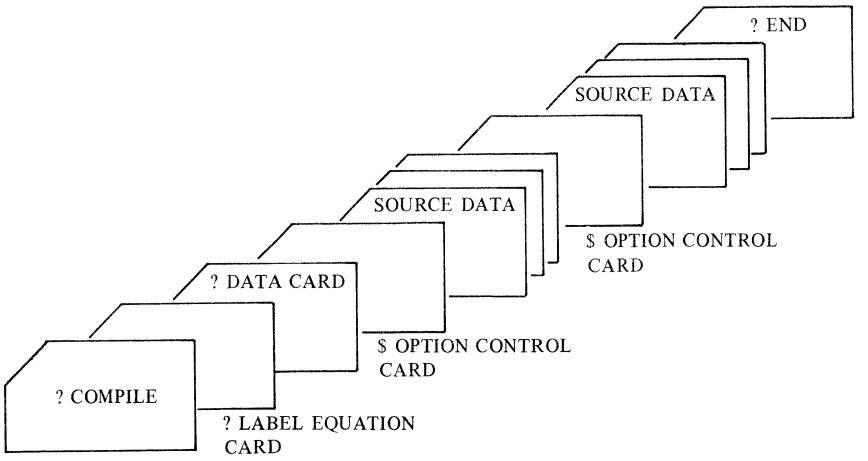
General

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.



FORTRAN Compilation Deck

Figure 4-5

$ 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 characterics:

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

Figure 4-6

BASIC COMPILER

General

BASIC is a problem-oriented language designed for a wide range of applications and may be easily applied to both 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 programing 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 the figure below.

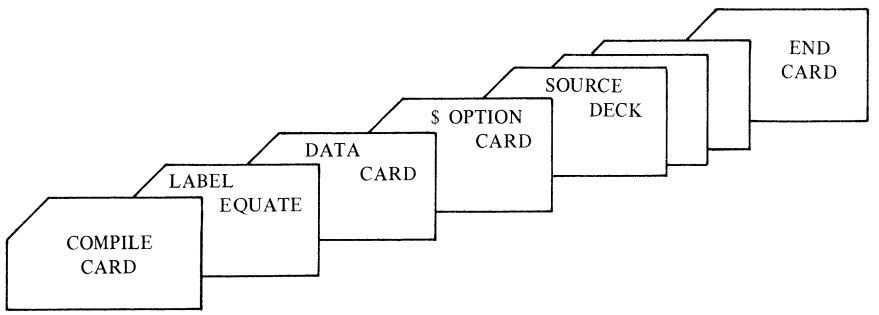


Figure 4-7

MCP control cards are made distinguishable from other cards by an invalid character in column 1 for 80 column cards, or a valid question mark (?) for 96 column cards. An invalid character is represented by a "?" for clarity in this manual. MCP control information is punched in a free-form format in columns 2 through 72. The presence of

MARK II® is a registered trademark of General Electric Corporation.

a percent sign in a MCP control card terminates the control information on that card and any information following the period is treated as a comment by the MCP.

$ 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 characterics:

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 turned "on" for selected program parts and then turned "off" 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 BAS.INTRIN/ #00000000, BAS.INTRIN/ #00000001, through BAS.INTRIN/ #00000038 must be present on disk when a compiled BASIC program is executed, they are not however, 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
Figure 4-8

BURROUGHS CORPORATION
DATA PROCESSING PUBLICATIONS
REMARKS FORM

TITLE: B 1700 SYSTEMS
SYSTEM SOFTWARE
OPERATIONAL GUIDE

FORM: 1068731
DATE: 7-73

CHECK TYPE OF SUGGESTION:

☐ ADDITION ☐ DELETION ☐ REVISION ☐ ERROR

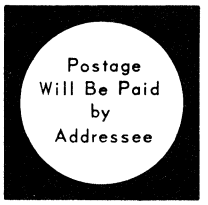
GENERAL COMMENTS AND/OR SUGGESTIONS FOR IMPROVEMENT OF PUBLICATION:

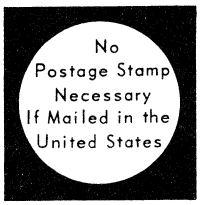
FROM: NAME _____ DATE _____
TITLE _____
COMPANY _____
ADDRESS _____
_____

cut along dotted line

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

1068731

7-73

Printed in U.S. America