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 3-75 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

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

LIST OF ILLUSTRATIONS

INTRODUCTION

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

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

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

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

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

EXECUTE

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

FOR

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

file-identifier

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

{ EXECUTE / EX }

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

[=]

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

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

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

```
[?] LOAD
```

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

```
@0CF3@
```

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

decimal – any valid decimal character or characters.

```
452
```

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

```
@A22F@
```

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

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

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

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

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

SECTION 1
INTRODUCTION TO SYSTEM

SYSTEM INITIALIZATION

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

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

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

UNIT MNEMONICS

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

CDx	– 96-Column Card Device
CPx	– 80-Column Card Punch
CRx	– 80-Column Card Reader
CSx	– Magnetic Tape Cassette
DCx	– Disk Cartridge
DISK	– Head-per-Track Disk
DPx	– Disk Pack
LPx	– Line Printer
MTx	– Magnetic Tape
SPO	– Console Printer
SRx	– Reader Sorter

NOTE

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

SYSTEM DESCRIPTION

The following functions are controlled by the MCP:

a. Loading

b. Interrupt handling

c. I/O control

d. Selection and initiation of programs

e. I/O error handling

f. System log maintenance

g. Storage allocation–memory and disk

h. Overlay functions – data and code

i. Multiprogramming

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. Head-per-Track Disk (MCP II)

h. Magnetic Tape Cassette (MCP II)

In addition, MCP II will accommodate the MICR Reader Sorter, Disk Pack, and Paper Tape Reader/Punch, as well as various data communications devices through a single line or a multi-line control.

HARDWARE REQUIREMENTS

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

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

CENTRAL SERVICE MODULE

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

a. Interrupt Detection and Handling.

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

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

   1. I/O Initialization

   2. Data Transfers

   3. I/O Termination

d. Manages Interpreter Activity.

INTERPRETERS

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

SECTION 2
MASTER CONTROL PROGRAM

GENERAL

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

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

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

MCP DISK STRUCTURES

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

Areas handled by the MCP include:

a. Directory Maintenance

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

b. Disk Allocation

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

c. File Assignment

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

d. Record Addressing

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

e. Paging

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

DISK DIRECTORY

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

Disk-Pack-Identifier

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

Example:

AAA/program-name/

AAA is the disk-pack-id

Main Directory File Name

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

Example:

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

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

Sub-Directory File Name

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

Main Directory Contents

The main directory entry contains:

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

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

Sub-Directory Contents

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

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

Directory Reference

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

MULTIPLE PACK FILES

Introduction

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

Abbreviations

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

Restrictions

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

a. The maximum number of packs that may be assigned to a MPF is 16, consisting of one BP and 15 CPs.

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

c. Only removable disk packs may be used for a MPF. A system pack cannot be used for a MPF.

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

NOTE:

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

Base Packs

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

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

Continuation Packs

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

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

General Information

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

MPFs may be sorted (INPLACE sort only).

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

Random files are allowed for MPFs.

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

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

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

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

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

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

HALTS

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

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 substring.
7	Invalid subscript.
8	Invalid value returned from a procedure.
9	Invalid case.

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

T REG	DESCRIPTION
TA ≥ @C@	Result Descriptor with Exception Bits Set.
ALL OTHERS.....	First Six Characters of MCP Source Sequence Number (Submit Memory Dump and Trouble Report).

12	Attempt to write outside of the MCP base or limit register.
22	Invalid service request.
25	Second operation complete bit is missing from the result status field. Result descriptor in T register.
29	Memory parity error during I/O operation.

MCP OPTIONS

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

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

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

The following is a list of the available MCP options:

BOJ	DUMP	MEM	RMOV
CHRG	EOJ	OPEN	SCHM
CLOS	LAB	PBD	TERM
DATE	LIB	PBT	TIME
DBM	LOG	PWS	ZIP

The MCP options are defined in the following paragraphs.

BOJ

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

CHRG

The CHRG option requires that all program executions be accompanied by a charge number which will be entered in the log.

CLOS

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

DATE

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

DBM

The DBM (Data Base Management) option pulls into memory an overlay disk segment containing a search operator to be bound into the CSM for disk search at Clear/Start time. The DBM option must be set to cause the presence of the overlay and must be used for Data Base Management, or when a RPG program uses the Index Sequential filing technique. A Clear/Start is required by the MCP after the option is set or reset.

DUMP

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

EOJ

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

LAB

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

LIB

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

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

LOG

The LOG option will request the MCP to keep a log of all program executions on disk. See the LG, SL, and TL input messages for actions pertaining to the LOG.

MEM

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

OPEN

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

PBD

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

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.

NOTE

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

PWS (MCPI only)

The PWS (Program Working Set) option allows the MCP to minimize the amount of memory it has to swap when going from the user program to the MCP and back again. The PWS option will only benefit a system with 32 K or greater. The working set of a program is defined to be those code segments needed in memory to run the program efficiently.

RMOV

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

SCHM

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

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

TERM

The TERM option specifies that the MCP automatically discontinue (DS) processing of a program when an error condition is encountered. If an error condition occurs and it is necessary to obtain a memory dump of the program, the TERM option should not be set.

TIME

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

ZIP

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

MCP-OPERATOR INTERFACE

General

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

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

There are four major types of control instructions:

(1) Library Maintenance Instructions

(2) Control Instructions

(3) Control Attributes

(4) File Parameter Instructions

MCP Communication

PUNCHED CARDS

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

a. Column 1 must contain an invalid character (80-column cards) or a question mark (96-column cards). An invalid character or question mark may not appear in any other column.

b. The remainder of the card may contain control instructions in free-field format; the MCP ignores information in the last eight columns.

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

d. The appearance of the "less than" (<) sign in a control message will cause the MCP to backspace its pointer one position for each < sign in memory while scanning the control instruction. This allows correction of mistakes without requiring that the entire message be re-entered. Even though this is intended mainly for messages entered via the keyboard, it will work with control instructions entered on punched cards as well. The "less than" sign may not be used for any other purpose.

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

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

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

Examples:

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

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

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

In the last example the pound sign (#) is not listed as a special character, so the identifier 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 invalid character is optional on continuation cards.

Example:

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

g. All control instructions are described on the following pages under headings which would indicate that each of them must consist of a separate card. This is not necessarily so; if the text of one control instruction is delimited by a space then this is considered the "logical end" of that control instruction. It may be followed by another control instruction on the same card as the example above indicates.

CONSOLE PRINTER

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

ZIP

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

GENERAL TERMS

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

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

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

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

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

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

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

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

unit–mnemonic	Device
CDx	96–Column Card Device
CRx	80–Column Card Reader
CPx	80–Column Card Punch
CSx	Magnetic Tape Cassette
DCx	Disk Cartridge
DISK	Head-per-Track Disk
DPx	Disk Pack
LPx	Line Printer
MTx	Magnetic Tape
PPx	Paper Tape Punch
PRx	Paper Tape Reader
SPO	Console Printer
SRx	MICR Reader Sorter

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

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

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

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

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

LIBRARY MAINTENANCE INSTRUCTIONS

CHANGE

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

The format of a CHANGE statement is:

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

The control word CHANGE may be abbreviated as CH.

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

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

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

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

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

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

The CHANGE statement may consist of additional cards where two or more "changes" may be made. For example:

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

Termination will occur when a semicolon (;) is detected.

ADD, LOAD

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

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

The format of the ADD and LOAD statements is:

```
[?]  { ADD  }  [TO disk-pack-identifier]  FROM library-tape-identifier  { =/=             }  ;
     { LOAD }                                                           { family-name/=  }
                                                                        { file-identifier }
```

The =/= causes every file on the tape to be considered.

Example:

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

DUMP, UNLOAD

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

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

The format of the DUMP and UNLOAD statements is:

```
[?]  { DUMP   }  [FROM disk-pack-identifier]  TO library-tape-identifier  { =/=             }  ;
     { UNLOAD }                                                           { family-name/=   }
                                                                          { file-identifier }
```

The =/= option indicates all files on the disk are to be DUMP/UNLOADED.

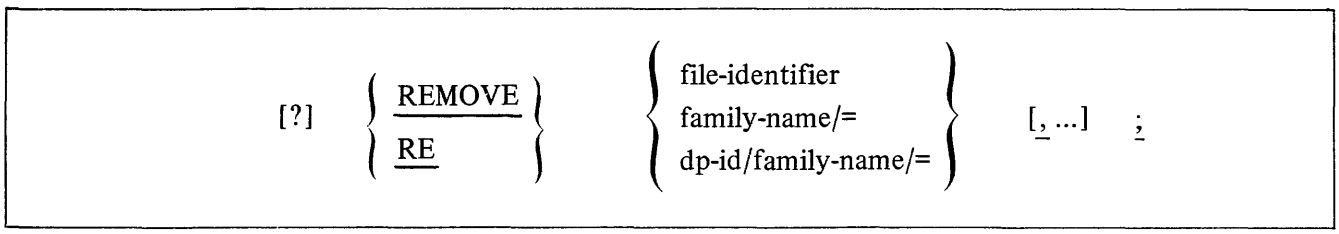
Example:

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

REMOVE

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

The format of the REMOVE statement is:



The control statement REMOVE may be abbreviated as RE.

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

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

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

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

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

Example:

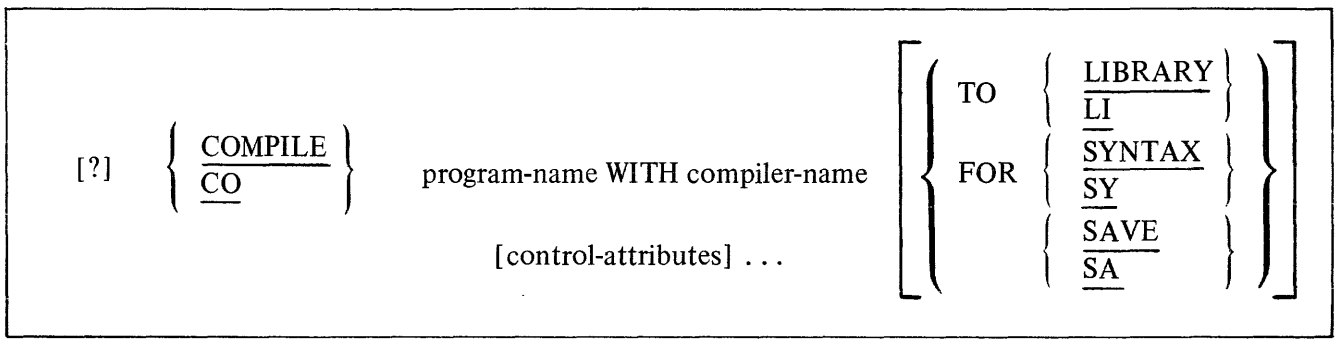
```
? REMOVE A/B
, X, Y,
Z;
```

CONTROL INSTRUCTIONS

COMPILE

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

The format of the COMPILE statement is:



The COMPILE statement may be abbreviated as CO.

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

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

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

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

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

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

DYNAMIC

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

The format of the DYNAMIC statement is:

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

The DYNAMIC control word may be abbreviated as DY.

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

EXECUTE

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

The format of the EXECUTE statement is:

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

The EXECUTE control word may be abbreviated as EX.

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

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

Example:

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

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

MODIFY

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

The format of a MODIFY statement is:

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

The MODIFY control statement may be abbreviated as MO.

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

Example:

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

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

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

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

CONTROL INSTRUCTION ATTRIBUTES

AFTER

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

The format of the AFTER statement is:

[?] { AFTER / AF } program-name

The AFTER control word may be abbreviated as AF.

Example:

EXECUTE ALPHA AFTER BETA or
EX ALPHA AF BETA

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

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

AFTER.NUMBER

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

The format of the AFTER.NUMBER statement is:

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

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

Example:

EXECUTE ALPHA AFTER.NUMBER 7 or
EX ALPHA AN 7

NOTE

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

THEN

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

The format of the THEN statement is:

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

The THEN control word may be abbreviated as TH.

Example:

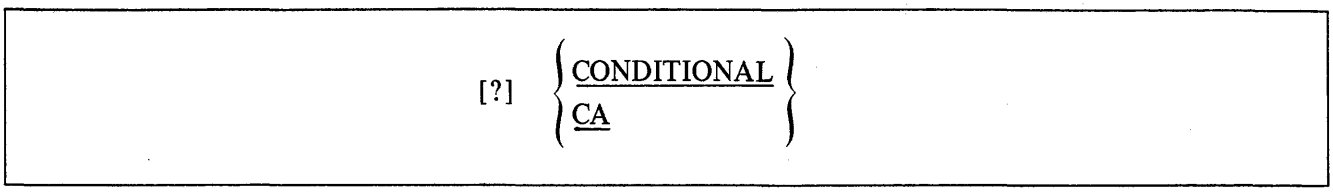
? EXECUTE ALPHA PRIORITY 14 MEMORY 20000 THEN COMPILE BETA COBOL SYNTAX

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

CONDITIONAL

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

The format of the CONDITIONAL statement is:



The CONDITIONAL control statement may be abbreviated as CA.

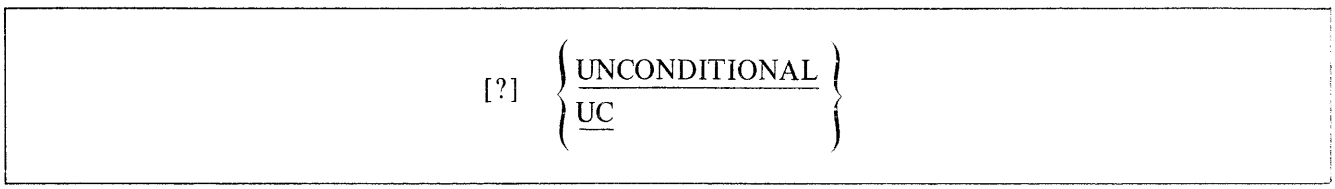
Example:

? EXECUTE A/B AFTER C/D CONDITIONAL or
? EX A/B AF C/D CA

UNCONDITIONAL

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

The format of the UNCONDITIONAL statement is:



The UNCONDITIONAL control statement may be abbreviated as UC.

Example:

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

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

CHARGE

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

The format of a CHARGE statement is:

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

The CHARGE control word may be abbreviated as CG.

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

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

DYNAMIC.SPACES

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

The format of the DYNAMIC.SPACES statement is:

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

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

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

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

For example:

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

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

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

or

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

FILE

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

The format of the FILE statement is:

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

The control word FILE may be abbreviated as FI.

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

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

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

```
? FILE internal-file-identifier  NAME  file-identifier
```

FILE ATTRIBUTES

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

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. This sets BACKUP.DISK and BACKUP. TAPE by implication.
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.

FILE ATTRIBUTE	FUNCTION
BCL	The recording mode of the file will be changed to BCL.
BINARY	The recording mode of the file will be changed to BINARY (80-column card and paper tape only).
BLOCKS.AREA[=]integer	Assign integer blocks (physical records) to an area.
BUFFERS[=]integer	The number of buffers assigned to the file will be altered to the value of the integer. The integer must be a positive number from 1 to 15.
COPY	The entire File Parameter Block except the internal file identifier of one file will be copied to the receiving file's File Parameter Block. The internal file-identifier will not be changed.

SYNTAX

COPY internal-file-identifier FROM { { # / IN / JOB.NUMBER } job-number (working copy) / { PN / PROGRAM.NAME } program-name (original copy) }

CYLINDER.BOUNDARY	Each area of a disk file will start at the beginning of a CYLINDER when the file is directed to a disk pack or disk cartridge.
DEFAULT	Override the disk allocation declared and use the file header block and record sizes. (Input disk and tape files only.)
DRIVE[=]integer	The file will be directed to the drive or EU specified by the integer. The drive must be a system disk. The integer must be a positive number from 0 to 15.
EBCDIC	The recording mode of the file will be changed to EBCDIC.
EU[=]integer	Same as DRIVE.
EVEN	The file will be changed to even parity.
FILE.TYPE[=]integer	The file's type may be changed at run time.

TYPE CODE	DESCRIPTION
0	Data
7	Interpreter
8	Code
9	Data

FILE ATTRIBUTE	FUNCTION
FORMS	The program will be suspended and the MCP will display a message for the operator to load special forms in the device (printer or punch) before the file is opened.
HARDWARE	A printer or punch file will be allowed to go to the hardware device assigned.
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.
INVALID.CHARACTERS [=] integer	The integer may contain a value of 0, 1, 2, or 3, and determines the course of action for invalid characters output to a train printer. 0 = Report all lines that contain invalid characters. The following console message will be output for each occurrance: FILE file-name IS PRINTING INVALID CHARACTERS ON LPx. 1 = Report all lines that contain invalid characters and stop the program at that point. 2 = Report once that the file is printing invalid characters. The following console message will be output: FILE file-name IS PRINTING INVALID CHARACTERS ON LPx. (one-time warning) 3 = Do not notify operator of invalid character output.
LABEL.TYPE[=] integer	The file will be processed as labeled (integer=0) or unlabeled (integer=1).
LOCK	The file will be LOCKED during file close or program termination time.
MAXIMUM.BLOCK.SIZE[=] integer	Fixed block size to be used for variable length records.
MULTI.PACK	The file will be considered a multi-pack file. (MPF)
NAME [=] file-identifier	The external file-identifier or dp-id will be changed to the value of file-identifier. If only the dp-id is to be changed the PACK.ID attribute may be used.

FILE ATTRIBUTE	FUNCTION
{ NO / NOT } file-attribute	When this option is used it will negate the file-attribute following the word NO or NOT. For example, a file assigned to go strictly to backup could be changed to go to the printer by entering a NO BACKUP file statement. The following is a list of file-attributes that the NO or NOT statement can negate. a. ALLOCATE.AT.OPEN b. BACKUP c. BACKUP.DISK d. BACKUP.TAPE e. CYLINDER.BOUNDARY f. DEFAULT g. FORMS h. HARDWARE i. INCREMENT.DRIVE j. INCREMENT.EU k. LOCK l. MULTI.PACK m. OPTIONAL n. VARIABLE o. WORK.FILE
ODD	The file will be changed to ODD parity.
OPTIONAL	Select optional file (COBOL only).
PACK.ID[=] disk-pack-id	Alter the pack-id.
PSEUDO	Makes file a pseudo type.
RANDOM	The file will be changed to a RANDOM access file.
RECORDS.BLOCK[=] integer	The number of logical records per block for a fixed record-length file.
RECORD.SIZE[=] integer	The number of bytes assigned for the logical record will be changed to the value of the integer.
REEL[=] integer	The value of the integer will determine the number of the first reel.
SERIAL	The file is to be processed sequentially.
SAVE[=] integer	A save factor representing the number of days a tape or disk file may be saved.
VARIABLE	The file will be processed using variable length records.
WORK.FILE	Assigns this file as a work file used internally.

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

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

FILE ATTRIBUTE ABBREVIATIONS

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

ADVERB	ADV
ALLOCATE.AT.OPEN	ALL
AREAS	ARE
ASCII	ASC
BACKUP	BAC
BACKUP.DISK	BDK
BACKUP.TAPE	BTP
BUFFERS	BUF
BLOCKS.AREA	B.A
BCL	BCL
BINARY	BIN
COPY	CPY
CARD.READER	CRD
CARD.PUNCH	CPC
CYLINDER.BOUNDARY	CYL
CASSETTE	CAS
DRIVE	DRI
DISK.CARTRIDGE	DCG
DISK	DSK
DISK.PACK	DPC
DISK.FILE	DFL
DEFAULT	DEF
DATA.RECORDER.80	DRC
EU	EU
EVEN	EVN
EBCDIC	EBC
FORMS	FMS
HARDWARE	HAR
INCREMENT.EU	INC
INCREMENT.DRIVE	INC
INVALID.CHARACTERS	INV
LOCK	LOC
LABEL.TYPE	LAB
MFCU	MFC
MAXIMUM.BLOCK.SIZE	MAX

MULTI.PACK	MUL
NAME	NAM
NO	NO
NOT	NOT
OPTIONAL	OPT
ODD	ODD
PACK.ID	PID
PAPER.TAPE.READER	PTR
PAPER.TAPE.PUNCH	PTP
READER.PUNCH.PRINTER	RPP
READER.SORTER	RSR
RECORD.SIZE	RSZ
RANDOM	RAN
REEL	REE
RECORDSBLOCK	R.B
REMOTE	REM
SERIAL	SER
SAVE	SAV
TAPE	TAP
TAPE.9	TP9
TAPE.7	TP7
TAPE.PE	TPE
UNIT.NAME	UNI
VARIABLE	VAR
WORK.FILE	WFL

FREEZE

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

The format of the FREEZE statement is:

```
[?] [OBJ] {FREEZE}
          {FR    }
```

The FREEZE control word may be abbreviated as FR.

HOLD

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

The format of the HOLD statement is:

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

The HOLD control word may be abbreviated as HO.

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

INTERPRETER

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

The format of the INTERPRETER statement is:

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

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

Examples:

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

INTRINSIC.NAME

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

The format of the INTRINSIC.NAME statement is:

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

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

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

Example:

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

                    or

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

INTRINSIC.DIRECTORY

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

The format of the INTRINSIC.DIRECTORY statement is:

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

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

Example:

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

MEMORY

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

The format of a MEMORY statement is:

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

The MEMORY control word may be abbreviated as ME.

The integer expresses the dynamic memory size in bits.

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

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

Examples:

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

                    or

? COMPILE    program-name    COBOL SYNTAX

? MEMORY = 50000
```

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

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

PRIORITY

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

The format of a PRIORITY statement is:

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

The PRIORITY control word may be abbreviated as PR.

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

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

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

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

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

SCHEDULE.PRIORITY

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

The format of the SCHEDULE.PRIORITY statement is:

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

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

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

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

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

Example:

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

NOTE

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

SWITCH

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

The format of the SWITCH statement is:

[?] [OBJ] { SWITCH / SW } switch-identifier-number [=] value

The SWITCH control word may be abbreviated as SW.

The switch-identifier-number must be a decimal digit from zero to nine (0-9) that references the switch or switches to be set. To determine what switches are available, the specific language manual for the program for which the switches are being set must be referenced.

The value is the value that the switch or switches will be assigned.

Example:

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

? SW0=5 SW1=3
```

UNFREEZE

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

The format of the UNFREEZE statement is:

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

The UNFREEZE control word may be abbreviated as UF.

VIRTUAL.DISK

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

The format of the VIRTUAL.DISK statement is:

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

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

Integer must be eight digits or less.

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

FILE PARAMETER INSTRUCTIONS

DATA

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

The format of the DATA control instruction is:

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

The control word DATA may be abbreviated as DA.

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

END

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

The format of the END statement is:

```
? END
```

The END control statement cannot be abbreviated.

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

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

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

KEYBOARD INPUT MESSAGES

General

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

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

Keyboard Entry Procedure:

a. Press INPUT REQUEST button.

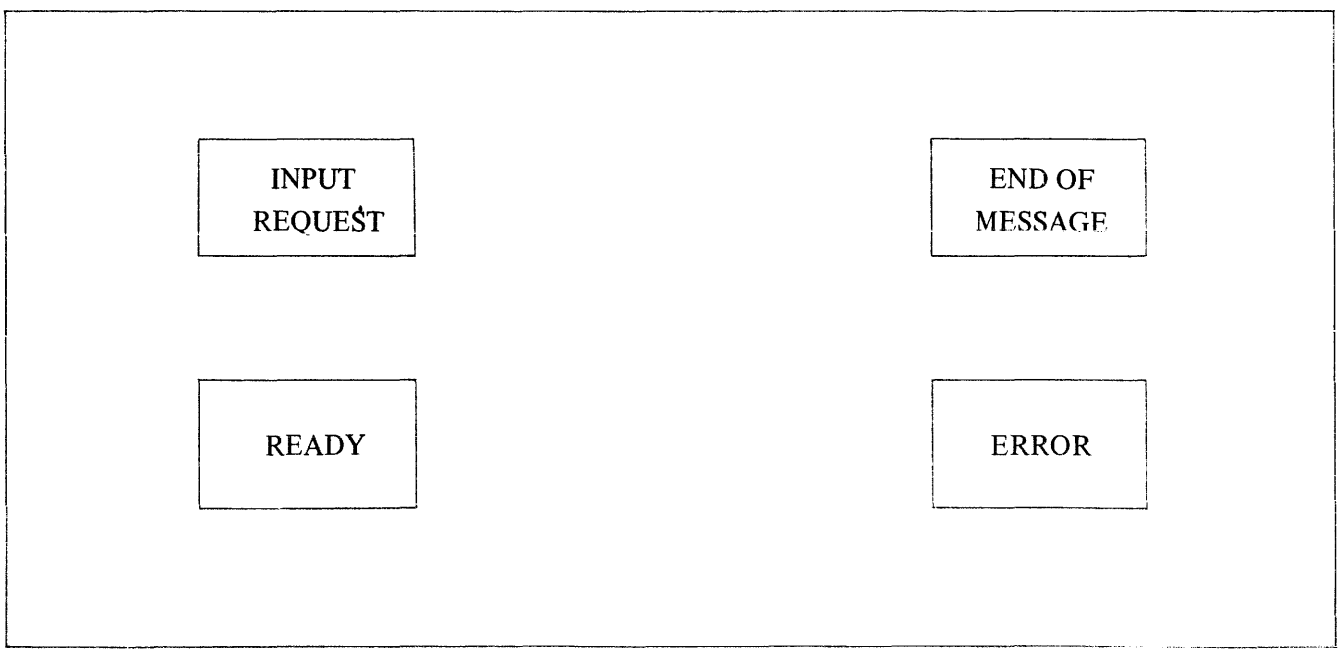
b. Wait for the READY indicator to light.

c. Type in message.

d. Depress END OF MESSAGE button (ETX) to terminate message.

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

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



Keyboard Input Messages

AX	(Response to ACCEPT)
BB	(Backup Blocks per Area)
BD	(Backup Designate)
BF	(Display Backup Files)
CD	(List Card Decks in Pseudo Readers)
CE	(Change to Entry System Software)
CL	(Clear Unit)
CM	(Change System Software)
CN	(Change to Non-Trace System Software)
CP	(Compute)
CQ	(Clear Queue)
CS	(Change to Standard System Software)
CT	(Change to Trace System Software)
DF	(Date of File)
DM	(Dump Memory and Continue)
DP	(Dump Memory and Discontinue)
DR DT	(Change MCP Date)
DS	(Discontinue Program)
ED	(Execute Pseudo Deck)
EM	(ELOG Message)
ET	(ELOG Transfer)
FM	(Response to Special Forms)
FN	(Display Internal File Name)
FR	(Final Reel of Unlabled Tape File)
FS	(Force from Schedule)
FT	(Change File Type)
GO	(Resume Stopped Program)
HS	(Hold in Waiting Schedule)
HW	(Hold in Waiting Schedule until Job EOJ)
IL	(Ignore Label)
KA	(Disk Analyzer)
KP KC	(Print Disk Sements)
LC	(Load Cassette)
LD	(Pseudo Load)
LG LN	(Transfer and Print Log)
LT	(List File Types)
MP	(List MPF Tables)
MR	(Close Output File with Purge)
MX	(Display MIX)
OF	(Optional File Response)
OK	(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)
PW	(Set Program Working Set-MCPI)
QC	(Quit Controller)
QF	(Query File)
QP	(Query Program)
RB RF	(Remove Backup Files)
RD	(Remove Psuedo Card Files)

RL	(Relabel User Pack)
RM	(Remove Duplicate Disk File)
RN	(Assign Pseudo Readers)
RO	(Reset Option)
RP	(Ready and Purge)
RS	(Remove Jobs from Schedule)
RT	(Remove MPF Table)
RY	(Ready Peripheral)
SD	(Assign Additional System Drives)
SL	(Set LOG)
SO	(Set Option)
SP	(Change Schedule Priority)
ST	(Suspend Processing)
SV	(Save Peripheral Units)
SW	(Set Switch)
TD	(Time and Date)
TI	(Time Interrogation)
TL	(Transfer LOG)
TO	(Display Options)
TR	(Time Change)
TS	(Test Switches)
UL	(Assign Unlabeled File)
WD	(Display MCP Date)
WM	(Display Current MCP and Interpreter)
WS	(Display Schedule)
WT	(Display MCP Time)
WW	(List Contents of NAME TABLE)
WY	(Program Status Interrogation)
XC	(Remove Disk Segments-Temporarily)
XD	(Remove Disk Segments-Permanently)

AX INPUT MESSAGE (Response to ACCEPT)

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

The format of the AX message is:

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

All responses are assumed to be alphanumeric format. The input message starts in the first position after the AX on the input line.

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.

The AX message has an unsolicited console feature in that the operator may enter as many AX message responses as needed for a given program prior to the actual ACCEPT. The AX messages must be entered in the order they will be used, since the queue is structured on a first-in, first-out basis.

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

BB

BB INPUT MESSAGE (Backup Blocks per Area)

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

The format of the BB message is:

BB integer

Each block is 900 bytes or 5 segments, and a backup file is always assigned 25 areas. There are 200 blocks or 1000 segments per area assigned at COLDSTART as the BB default value.

If the integer input via the BB message is less than 5, 200 will be assigned by default.

BD INPUT MESSAGE (Backup Designate)

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

The format of the BD message is:

BD { disk-pack-identifier / "blank" (system pack) }

BF INPUT MESSAGE (Display Backup Files)

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

The format of the BF message is:

```
BF [pack-identifier/] { =/=     }
                      { integer }
                      { PRT/=   }
                      { PRN/=   }
                      { 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 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 INPUT MESSAGE (List Card Decks in Pseudo Readers)

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

The CD message format is:

CD

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

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

CE INPUT MESSAGE (Change to Entry System Software)

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

The format of the CE message is:

```
CE
```

CL INPUT MESSAGE (Clear Unit)

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

The format of the CL message is:

CL unit-mnemonic

Example:

```
CL LPA
```

CM

CM INPUT MESSAGE (Change System Software)

The CM input message allows the operator to identify programs to the system for subsequent usage. The purpose of the CM message is to identify a file, exclusively on System Disk, that contains the program to be used for a designated function.

The format of the CM message is:

CM system-software-mnemonic program-identifier

The resultant action of the CM message will not take affect until the next Clear/Start.

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

Example:

CM MX MCP/XYZ

The CM message at the next Clear/Start makes the program MCP/XYZ the experimental MCP.

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

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

The format of the CN message is:

CN

CAUTION

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

CP INPUT MESSAGE (Compute)

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

The format of the CP message is:

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

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

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

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

The CP message will calculate an arithmetic expression strictly on a left-to-right basis. Therefore, quantities contained in parentheses or brackets are invalid. Spaces are not used as delimiters and are ignored.

The response is displayed in both decimal and hexadecimal formats.

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

Example:

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

response: CP: @0000F7@=247

CP @F@

CP: @00000F@=15

CQ INPUT MESSAGE (Clear Queue)

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

The CQ message format is:

CQ

CS INPUT MESSAGE (Change to Standard System Software)

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

The format of the CS message is:

```
CS
```

CT INPUT MESSAGE (Change to Trace System Software)

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

The format of the CT message is:

CT

CAUTION

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

DF

DF INPUT MESSAGE (Date of File)

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

The format of the DF message is:

```
DF  { file-identifier }
    { family-name/=   }
```

DM INPUT MESSAGE (Dump Memory and Continue)

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

The format of the DM message is:

mix-index DM

Processing automatically continues when the dump is finished.

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

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

Example:

```
2 DM
```

DP INPUT MESSAGE (Dump Memory and Discontinue)

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

The DP message format is:

mix-index DP

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

Example:

```
1 DP
```

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

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

The DR message format is

{DR / DT} mm/dd/yy

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

DS

DS INPUT MESSAGE (Discontinue Program)

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

The format of the DS message is:

```
mix-index DS
```

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

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

ED INPUT MESSAGE (Execute Pseudo Deck)

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

The format of the ED message is:

ED integer

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

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

EM

EM INPUT MESSAGE (ELOG Message)

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

The format of the EM message is:

EM input-message

ET INPUT MESSAGE (ELOG Transfer)

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

The format of the ET message is:

ET

FM INPUT MESSAGE (Response to Special Forms)

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

The format of the FM message is:

mix-index FM unit-mnemonic

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

The message

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

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

Example:

```
3 FM LPA
```

FN INPUT MESSAGE (Display Internal File Name)

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

The format of the FN input message is:

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

The MCP 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 Unlabeled Tape File)

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

The format of the FR message is:

```
mix-index FR
```

The FR message is a response to the message:

```
mix-index NO FILE
```

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

FS INPUT MESSAGE (Force from Schedule)

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

The format for the FS message:

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

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

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

NOTE

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

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

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

FT

FT INPUT MESSAGE (Change File Type)

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

The format of the FT message is:

```
FT  file-identifier  file-type
```

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

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

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

The LT message will list the file types.

GO INPUT MESSAGE (Resume Stopped Program)

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

The format for the GO message is:

mix-index GO

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

HS

HS INPUT MESSAGE (Hold in Waiting Schedule)

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

The format of the HS message is:

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

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

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

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

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

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

The format of the HW message is:

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

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

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

IL

IL INPUT MESSAGE (Ignore Label)

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

The format of the IL message is:

mix-index IL unit-mnemonic

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

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

NO FILE . . .

DUPLICATE INPUT FILE . . .

file-identifier NOT IN DISK DIRECTORY

It is assumed that the system operator knows that the file on the unit selected is the file needed regardless of the original file-identifier's location. If the unit-mnemonic specifies a disk drive, the directory on that drive will be searched for the required file-identifier.

NOTE

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

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 [disk-pack-identifier] { =/= | DSKAVL | family-name/file-identifier | file-identifier | family-name/= }

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

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

The DSKAVL will list the available areas on the disk.

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

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

Examples:

KA

KA DSKAVL

KA dp-id

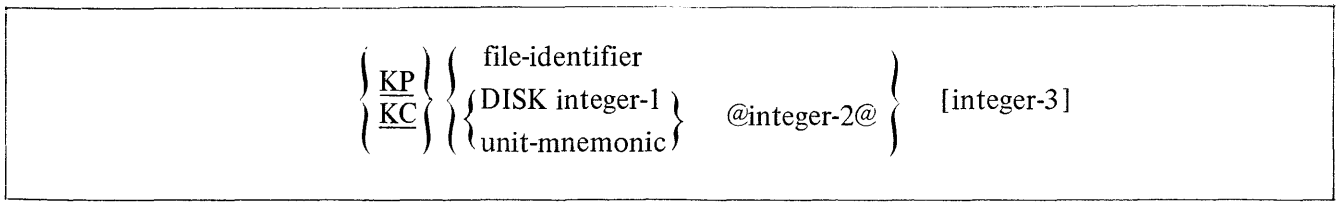
KA file-identifier

KC
KP

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

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

The format of the KC or KP message is:



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

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

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

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

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

Examples:

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

LC INPUT MESSAGE (Load Cassette

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

The format of the LC message is:

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

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

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

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.

Example:

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

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

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

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

The format of the LG, LN message is:

{ LG / LN }

LT

LT INPUT MESSAGE (List File Types)

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

The format of the LT message is:

LT

MP INPUT MESSAGE (List MPG Tables)

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

The format of the MP message is:

MP

MR

MR INPUT MESSAGE (Close Output File with Purge)

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

The format of the MR message is:

mix-index MR

MX INPUT MESSAGE (Display MIX)

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

The format of the MX message is:

```
MX
```

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

Example:

```
MX

    program-name = 1  PR:04

    program-name = 2  PR:05

END MX
```

OF

OF INPUT MESSAGE (Optional File Response)

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

The format of the OF message is:

```
mix-index OF
```

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

OK INPUT MESSAGE (Continue Processing)

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

The format of the OK message is:

```
mix-index OK
```

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

Examples:

job-specifier DUPLICATE INPUT FILES . . .

job-specifier DUPLICATE FILE ON DISK . . .

job-specifier NO DISK . . .

job-specifier NO MEMORY . . .

job-specifier FILE file-identifier NOT PRESENT

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

OL

OL INPUT MESSAGE (Display Peripheral Status)

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

The format of the OL message is:

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

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

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

The following responses are generated:

unit-mnemonic IN USE BY program-name file name

unit-mnemonic LABELED file-name

unit-mnemonic NOT READY

unit-mnemonic UNLABELED

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

NULL unit-type-code TABLE

OU INPUT MESSAGE (Specify Output Device)

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

The format of the OU message is:

mix-index OU unit-mnemonic

Example:

```
4 OU DPC
```

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

PB

PB INPUT MESSAGE (Print/Punch Backup)

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

The format of the PB message is:

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

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

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

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

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

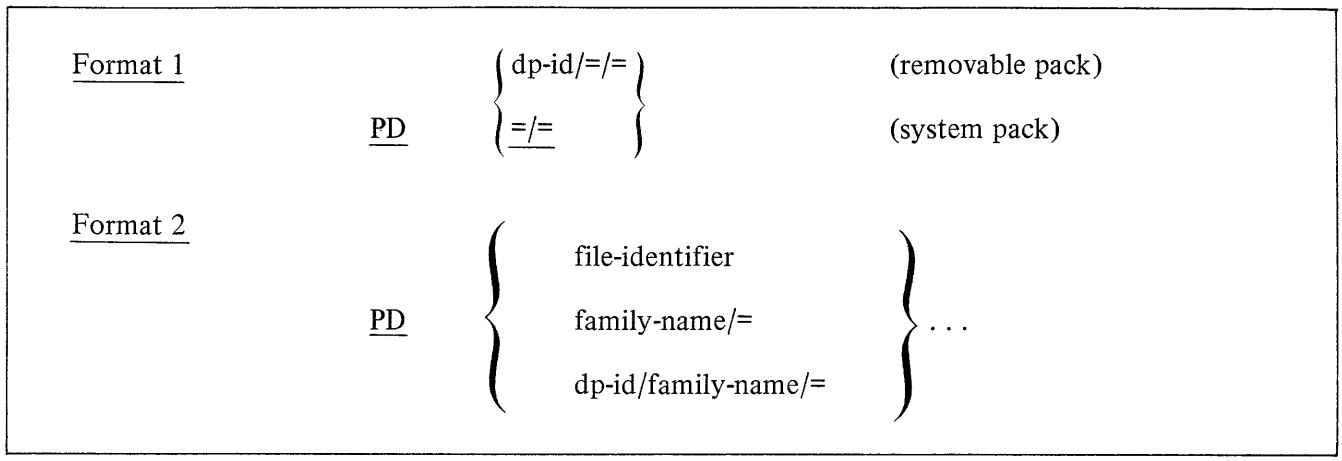
The integer-2 option is a counter to tell the MCP the number of copies of each file to be printed or punched for output. If this option is omitted, one (1) is assumed.

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

PD INPUT MESSAGE (Print Directory)

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

The PD message has two formats:



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

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

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

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

file-identifier NOT IN DIRECTORY

Examples:

Does a file named COBOLZ reside on the system pack?

request: PD COBOLZ

response: PD = COBOLZ (affirmative response)

What files reside on the system pack?

request: PD =/=

response: PD = file-identifier-1

PD = file-identifier-2

PD = . . .

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

request: PD MASTER/PAYROLL/QUARTERLY

response: PD = MASTER/PAYROLL/QUARTERLY

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

request: PD ALPHA, BETA, CHARLIE

response: PD = ALPHA

PD = BETA

CHARLIE NOT IN DIRECTORY

PG INPUT MESSAGE (Purge)

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

The format of the PG message is:

PG unit-mnemonic [serial-number]

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

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

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

The serial number is not used when purging a tape.

Example:

```
PG DPA 000456
```

PM

PM INPUT MESSAGE (Print Memory)

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

The format of the PM message is:

PM [integer [SAVE]]

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

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

The programs DUMP/ANALYZER and either MCPI/ANALYZER (MCPI) or MCPII/ANALYZER (MCPII) 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 INPUT MESSAGE (Power Off)

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

The format of the PO message is:

PO unit-mnemonic

A system pack may not be powered off.

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

unit-mnemonic HAS integer USERS

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

unit mnemonic MAY NOW BE POWERED DOWN

to be displayed.

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

PR

PR INPUT MESSAGE (Change Priority)

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

The format of the PR message is:

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

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

PS INPUT MESSAGE (PROD Schedule)

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

The format of the PS message is:

PS

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

PW

PW INPUT MESSAGE (Set Program Working Set - MCP I)

The PW input message is used to set the field PROG.WORKING.SET in the program's Program Parameter Block to the size in bytes of the memory space needed to hold the program's working set.

The format of the PW message is:

PW program-identifier { blank = | [=] size-in-bytes }

When the blank = entry is input, the current value of the Program Working Set will be used. When the size-in-bytes is specified, the Program Working Set will be set to the value indicated.

NOTE

If the Program Working Set is made too large for available memory, the MCP will ignore the value assigned. If set too small, the program will run inefficiently.

The PW message responds by returning the following message.

PW program-identifier = size-in-bytes

Therefore the Program Working Set may be interrogated for size by entering this message:

PW program-identifier =

Example:

PW A/B = (Test Size)

PW A/B = 5000 (Response)

PW A/B = 4500 (Set Size)

PW A/B = 4500 (Response)

QC INPUT MESSAGE (Quit Controller)

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

The format of the QC message is:

QC

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

NETWORK CONTROLLER ALREADY RUNNING DS OR DP

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

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

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

QF

QF INPUT MESSAGE (Query File)

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

The format of the QF message is:

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

Examples:

QF A/B CG

QF A/B FILE BACKUP

QP INPUT MESSAGE (Query Program)

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

The format of the QP message is:

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

Examples:

QP 14 PRIORITY

QP 15 CHARGE FREEZE

RB
RF

{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} [disk-pack-identifier/] { =/= / integer / PRT/= / PRN/= / PCH/= }

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

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

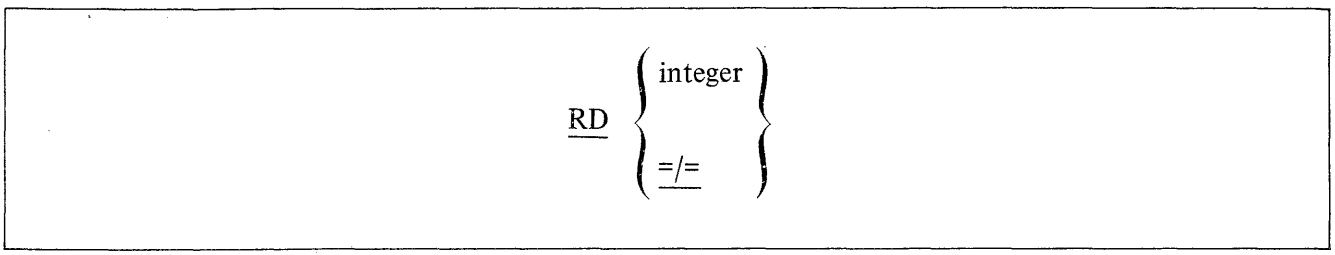
The =/= option will remove all backup files from disk.

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

RD INPUT MESSAGE (Remove Pseudo Card Files)

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

The format of the RD message is:



RL

RL INPUT MESSAGE (Relabel User Pack)

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

The format of the RL message is:

RL unit-mnemonic disk-pack-id { R / U }

RM INPUT MESSAGE (Remove Duplicate Disk File)

The RM input message allows the system operator to remove a disk file from the disk directory in response to a DUPLICATE FILE ON DISK message.

The format of the RM message is:

mix-index RM

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

Example:

```
1 RM
```

RN

RN INPUT MESSAGE (Assign Pseudo Readers)

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

The format of the RN message is:

RN integer

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

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

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

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

RO INPUT MESSAGE (Reset Option)

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

The format of the RO message is:

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

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

Example:

request: RO EOJ

response: EOJ = 0

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

The TO message may be entered to determine which options are set at any given time. The option indicator equals one when set and zero when reset. A complete list of the MCP options and their status will be displayed.

RP

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:

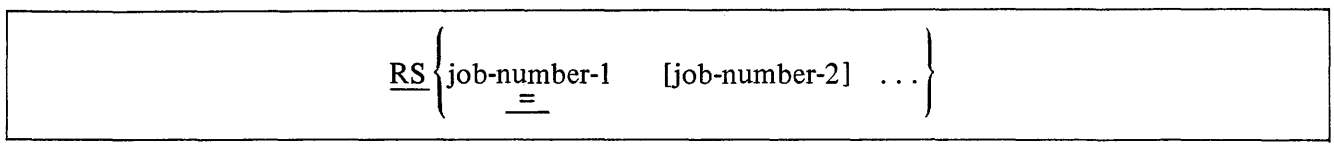
RP unit-mnemonic [unit-mnemonic] . . .

The RP message can be used for tape only.

RS INPUT MESSAGE (Remove Jobs from Schedule)

The RS input message will allow the system operator to remove a job from the schedule prior to its being entered in the MIX for execution.

The format of the RS message is:



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

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

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

The "=" option will remove all jobs from the schedule.

If the requested program(s) are not in the schedule, the MCP will notify the operator that an invalid request has been entered.

Example:

```
RS   33 ,  34 ,  35 ,  36

#33 RS-ED
#34 RS-ED
#35 RS-ED

36 NULL SCHEDULE          (job 36 not in schedule)
```

RT

RT INPUT MESSAGE (Remove MPF Table)

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

The format of the RT message is:

RT file-identifier

RY INPUT MESSAGE (Ready Peripheral)

The RY input message allows the system operator to ready a peripheral unit and make it available to the MCP.

The format of the RY message is:

```
RY unit-mnemonic-1 [unit-mnemonic-2] . . .
```

Any number of units may be made ready with one RY message.

When a removable disk cartridge or disk pack is placed on a system, the MCP must be notified of its presence with the RY message.

If the designated unit is not in use and is in the remote status, the RY message causes all exception flags maintained by the MCP for the specified unit to be reset. After the unit has been made ready, the MCP attempts to read a file label (input devices only).

Example:

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

SD

SD INPUT MESSAGE (Assign Additional System Drives)

The SD input message gives the system operator the ability to assign additional system drives for the MCP.

The format of the SD message is:

```
SD unit-mnemonic  serial-number
```

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

At COLDSTART, there is only one system drive, so additional drives may be added by the SD message. Once a system drive has been added to the system, it cannot be removed without performing a COLDSTART.

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

unit-mnemonic IS NOW A SYSTEM PACK–CLEAR START REQUIRED

SL INPUT MESSAGE (Set LOG)

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

The format of the SL message is:

```
SL integer-1 [integer-2]
```

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

The integer-2 entry is optional and is the maximum number of areas desired. It must be between 2 and 105, inclusive. Default is 25.

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

LOG NOW SET–CLEAR START REQUIRED

or

NO SPACE TO BUILD LOG

SL 0 will cause the LOG option to be reset.

SO

SO INPUT MESSAGE (Set Option)

The SO input message allows the system operator to set the options used to direct or control some of the MCP functions.

The format of the SO message is:

```
SO option-name-1  [option-name-2] . . .
```

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

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

The TO input message may be entered to determine which options are set at any given time. The option indicator equals one when set and zero when reset. A complete list of the MCP options and their status will be displayed.

SP INPUT MESSAGE (Change Schedule Priority)

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

The format of the SP message is:

SP job-number integer

The Schedule Priority is separate from the priority of the job when it is in the mix.

The job-number will identify the program in the schedule that is to be affected by the SP message.

The integer in the SP message specifies the new priority that will be assigned to the program. Priorities may range from zero through 14, where zero is the lowest priority and 14 is the highest priority.

To change the priority of a program in the schedule with a job-number of 33 to a priority of 7, the following SP message would be used.

```
SP  33  7
```

This program would be selected from the schedule ahead of the other programs with a lower priority.

The following message would be displayed in response to the above input message:

```
program-name  33  PR = 07
```

ST

ST INPUT MESSAGE (Suspend Processing)

The ST input message provides a means for the system operator to temporarily suspend the processing of a program in the MIX.

The format of the ST messgae is:

```
mix-index ST
```

The mix-index identifies the program to be suspended.

The MCP will not suspend the program until all I/O operations in progress for that program have been completed.

When the MCP suspends a program, it is rolled-out to disk and the memory it was using is returned to the MCP for reallocation.

A suspended program will retain the mix-index and peripherals assigned to it; the MCP will use this to identify the program when referenced by another keyboard input message.

To restart a program after it has been suspended, the GO message must be used. If for some reason all of the conditions necessary for the program to run are not met when the GO message is issued, the MCP will not restart the program.

Example:

```
3 ST
```

SV INPUT MESSAGE (Save Peripheral Units)

The SV message allows the system operator to make a peripheral unit inaccessible to the MCP until a Clear Start operation occurs, or an RY input message is used to ready the unit.

The format of the SV message is:

```
SV unit-mnemonic [unit-mnemonic] . . .
```

Any number of peripheral units may be saved with one SV input message.

When the SV message is entered and the unit is not in use, the specified unit is marked SAVED and "unit-mnemonic SAVED" is displayed by the MCP.

If the unit is in use, the MCP will respond with "unit-mnemonic TO BE SAVED" and will save the unit as soon as it is no longer being used.

Example:

```
SV LPA
```

SW

SW INPUT MESSAGE (Set Switch)

The SW input message allows the system operator to set programmatic switches.

The format of the SW message is:

```
mix-index SW switch-identifier-number [=] value
```

Programmatic switches may also be set at schedule time by using the SWITCH control statement attribute. Refer to Section 2, SWITCH control attribute.

The switch-identifier-number must be a decimal digit from zero to nine (0-9) that references the switch or switches to be set. To determine what switches are available, the specific language manual for the program for which the switches are being set must be referenced.

The value is the value that the switch or switches will be assigned.

Example:

```
5 SW1 = @4F@ SW8 = 6
```

TD INPUT MESSAGE (Time and Date)

The TD input message allows the system operator to request that the MCP type the current values of the time and date.

The format of the TD message is:

```
TD
```

The MCP displays the date and time in the following format:

DATE = mm/dd/yy TIME = hh:mm:ss.t

Where:

hh – hours
mm – minutes
ss – seconds
t – tenths of seconds

TI

TI INPUT MESSAGE (Time Interrogation)

The TI input message allows the system operator to interrogate the MCP as to the amount of processor time the program has used up to the time the interrogation was made.

The format of the TI message is:

```
mix-index TI
```

The mix-index identifies the program for which the interrogation was requested.

The time is given in hours, minutes, seconds, and tenths of seconds.

Example:

```
4TI
COBOL: A/B = 4    CPU TIME = 00:03:15.7
```

TL INPUT MESSAGE (Transfer LOG)

The TL input message allows the system operator to transfer the information in the SYSTEM/LOG to LOG/*#integer*. To print the LOG refer to the LG input message.

The format of the TL message 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 format of the TO 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 format of the TR message is:

```
TR integer
```

The time specified by the integer is designated according to a 24-hour clock, and must be four digits in length.

This message is not accepted by the MCP if the value of the integer is greater than 2400 hours.

Example:

Set the time in the MCP to 7:19 P.M.

TR 1919

TS

TS INPUT MESSAGE (Test Switches)

The TS input message allows the system operator to test the programmatic switches set by the SW console message or the SWITCH control statement attribute.

The format of the TS message is:

mix-index TS

The output of the TS message is in hexadecimal format.

Example:

4 TS

@0002ABC000@

UL INPUT MESSAGE (Assign Unlabeled File)

The UL message allows the system operator to designate the unit on which a particular unlabeled input file is located in response to a "FILE NOT PRESENT" message from the MCP.

The format of the UL message is:

```
mix-index UL unit-mnemonic [integer]
```

The UL message is used only if the unit designated is to be acted on as an unlabeled file. The MCP assumes the file on the designated unit is the file requested by the program that caused the "FILE NOT PRESENT" message.

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

If integer is used, the MCP spaces forward "integer" blocks prior to reading the first data block into the object program. Tape marks are read as blocks. This is done at the time the file OPEN is performed.

Example:

A program with a mix-index of 1 calling for an unlabeled input tape file could be assigned a tape on a unit with the unit-mnemonic of MTA with the following UL message:

```
1 UL MTA
```

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

```
1 UL MTA 3
```

WD

WD INPUT MESSAGE (Display MCP Date)

The WD input message permits the system operator to request the current date used by the MCP.

The format of the WD message is:

```
WD
```

WM INPUT MESSAGE (Display Current MCP and Interpreter)

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

The format of the WM message is:

```
WM
```

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

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

WS

WS INPUT MESSAGE (Display Schedule)

The WS input message allows the system operator to interrogate what program or programs are currently in the schedule and their status.

The format of the WS message is:

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

The job-number is assigned by the MCP as the program is entered into the schedule.

The MCP response to the WS message gives the program-name, schedule number, memory required in KB's, program priority, and the length of time the program has been in the schedule.

Example:

```
WS 4

ALPHA = 4 NEEDS 8 KB PR = 4 IN FOR 00:08:37.4
```

WT INPUT MESSAGE (Display MCP Time)

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

The format of the WT message is:

WT

WW

WW INPUT MESSAGE (List Contents of NAME TABLE)

The WW input message gives the operator the ability to list the different types of system software/ firmware in the NAME TABLE.

The format of the WW message is:

WW { = / system-software-mnemonic }

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

WY INPUT MESSAGE (Program Status Interrogation)

The WY message allows the system operator to check the current status of one program or all the programs in the MIX.

The format of the WY message is:

[mix-index] WY

The mix-index identifies the program in the MIX that is to be checked and its status displayed on the console printer. If the mix-index is omitted the MCP will display the program status of the entire MIX.

The MCP response to the WY message is:

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

Example:

```
1 WY
     ALPHA  =   01 WAITING OPERATOR ACTION

WY
     ALPHA  =   01 WAITING OPERATOR ACTION
     BETA   =   02 EXECUTING
     DELTA  =   03 AX–WAITING FOR KEYBOARD INPUT
     ECHO   =   04 IN COMMUNICATE QUEUE
```

XC

XC INPUT MESSAGE (Remove Disk Segments-Temporarily)

The XC input message allows the temporary removal of contiguous disk segments from the Working Available Table.

The format of the XC message is:

XC { DCx / DPx / DISKn } @beginning-segment-address@ number-of-segments

XC Characteristics

x - used with the DC (Disk Cartridge) and the DP (Disk Pack) and must be an alphabet character that references the DP or DC in the operation.

n - used with the DISK (Head-per-track) and must be a decimal integer less than 16 referencing the DISK EU in the operation.

@beginning-segment-address@ - is the beginning disk segment address of those segments to be removed expressed in hexadecimal format.

number-of-segments - is the number of contiguous segments to be removed.

Restrictions

The disk segments to be removed must be in the Working Available Table. Therefore, when the affected tracks contain files or other information, the tracks or segments must be made available before the MCP will accept the XC message. The following message will be output in this situation:

REQUESTED SEGMENTS NOT ENTIRELY AVAILABLE

A file residing on a Disk Cartridge may be contained on segments prior to the beginning address. It must also be made available before the XC message will be valid.

The removed disk segments will not be made available for use until the next Clear/Start.

If the segments are to be disabled permanently refer to the XD message.

XD INPUT MESSAGE (Remove Disk Segments-Permanently)

The XD input message allows the permanent removal of contiguous disk segments from the Working Available Table and the Master Available Table.

The format of the XD message is:

XD { DCx / DPx / DISKn } @beginning-segment-address@ number-of-segments

XD Characteristics

x - used with the DC (Disk Cartridge) and the DP (Disk Pack) and must be an alphabet character that references the DC or DP in the operation.

n - used with the DISK (Head-per-track) and must be a decimal integer less than 16 referencing the DISK in the operation.

@beginning-segment-address@ - is the beginning disk segment address of those segments to be removed expressed in hexadecimal format.

number-of-segments - is the number of contiguous segments to be removed.

The XD message removes the segments from both the Working Available Table and the Master Available Table. The removed segments are not placed in the Temporary Available Table.

The removed segments or tracks cannot be recovered without a disk reinitialization, and the data contained on those removed segments cannot be accessed and must be considered lost.

MCP OUTPUT MESSAGES

General

The MCP communicates to the system operator via the console printer. Messages can either be originated by the MCP for information and possible operator action, or they can originate from an executing program. In either case, the MCP has complete control over all messages.

All output messages are indented one space by the MCP, in order for the operator to easily distinguish them from input messages.

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

Syntax

The paragraphs below outline the syntax used in defining the MCP messages in this section.

Classification: MCP messages are listed in alphabetical order using the first word of the actual message as the key. The job-specifier portion and any "optional" type entries are not considered part of the key.

Job-Specifier: Job-Specifier is simply used to identify the job for which that message is intended. The format of the job-specifier is:

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

The compiler-name portion is only printed when the executing program is a compilation.

Terminal-reference: The phrase "terminal-reference" following any message indicates that a termination message will be printed. Any time this message is printed, the program must be DS-ED or DP-ED, except when the TERM option is set causing the program to be terminated automatically.

The format of the terminal-reference message is:

: P = nn, S = nn, D = nn
(@–@, @–@, @–@) DS or DP

or

: S = nn, D = nn
(@–@, @–@) DS or DP

NOTE

There are situations usually occurring when memory is approaching saturation that the program-identifiers will be omitted from console messages and referenced only by their mix-index number. This is done to conserve memory. For example, the following message:

3 NO MEMORY

might be used instead of

A/B/C = 3 NO MEMORY

MCP Messages

job-specifier–ABORTED

job-specifier–ACCEPT

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

ATTEMPTED TO WRITE OUT OF BOUNDS

unit-mnemonic ASSIGNED TO SYSTEM USE

unit-mnemonic AVAILABLE AS OUTPUT

BACKUP FILE nnnnnn NOT REMOVED–NOT ON DISK

BACKUP TAPE NOT FOUND–"RY" unit-mnemonic

BATCH COUNT COMMUNICATE ISSUED WHILE SORTER FLOWING terminal-reference

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

job-specifier–{ BOJ. / EOJ. / DS-ED. } = job-number PR = nn [integer SYNTAX ERRORS] TIME = hh:mm:ss.

job-specifier–CANNOT ACCEPT "[IL'UL'OF'FR'FM'OU'OK'RM'MR]"MESSAGE

CANNOT ACCEPT DATA STATEMENT FROM THE SPO

unit-mnemonic CANNOT BE OPENED OUTPUT FOR file-identifier

CANNOT CHANGE PACK-ID OR FAMILY NAMES WITH EQUALS . . . id's . . .

CANNOT FIND UNIT REQUESTED FOR FN

CANNOT READ LABEL ON unit-mnemonic

CANNOT READ THE LABEL ON unit-mnemonic

CANNOT REMOVE PACK.ID OR FAMILY NAMES WITH = -S file-identifier

CANNOT SAVE THIS DEVICE unit-mnemonic

file-identifier CHANGED TO new-file-identifier

CHAR OR BIT STRING IS INCOMPLETE . . . . input message . . . .

CLEAR/STARTB1700 MCPII MARK nnn.nn mm/dd/yy hh:mm

***CLEAR/START REQUIRED

CLEAR/START REQUIRED–SYSTEM/PRINTCHAIN MISSING

COMPILE program-name CTRL RCD ERR: . . . .

job-specifier–CONTROL STACK OVERFLOW terminal-reference

job-specifier–"CONVERT" ERROR terminal-reference

COULD NOT CHANGE THE MCP

job-specifier–CPU TIME = hh:mm:ss.t

CURRENT MCP IS identifier USING interpreter-id

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

DECK nnnn = 50 CHAR

DECK nnnn IN USE BY program-name

**DECK NUMBER nnnn NOT ON DISK

DEFAULT CHARGE NO. = nnnnnn

DISK ERROR ON OVLY { READ / WRITE } FROM { DISK ADDRESS@nnnn@ / MEMORY ADDRESS@nnnn@ }

job-specifier—DISK FILE DECLARED SIZE EXCEEDED ON file-identifier terminal-reference

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

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

job-specifier—DIVIDE BY ZERO terminal-reference

**DR PLEASE

job-specifier—DUPLICATE INPUT FILES file-identifier

END BF

END MX

END PD

job-specifier—ENDING DATA OVERLAY ADDRESS = nnnn, WHILE BR = nnnn
terminal-reference

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

unit-mnemonic ERROR/pack-id IS [RESTRICTED or INTERCHANGE] PACK

unit-mnemonic ERROR unit-id

job-specifier—EVALUATION OR PROGRAM PTR STACK OVERFLOW terminal-reference

EXECUTE program-name CTRL RCD ERR: . . .

job-specifier—EXPONENT OVERFLOW terminal-reference

job-specifier—EXPONENT UNDERFLOW terminal-reference

job-specifier—EXPRESSION OUT OF RANGE terminal-reference

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

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

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

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

FN = "internal-file-identifier"

FREE UP SOME DISK AND CLEAR/START

GOOD MORNING, TODAY IS name-of-day, hh:mm:ss.t $\left\{\begin{matrix}\text{AM}\\\text{PM}\end{matrix}\right\}$ JLN DT = yy/ddd

unit-mnemonic HAS nnnn USERS

unit-mnemonic HAS BEEN PURGED

job-specifier–unit-mnemonic HOPPER EMPTY

INVALID BIT CHARACTER– . . .

INVALID BIT SPECIFIER– . . .

INVALID CHAR COL nn

INVALID CHARACTER . . .

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

job-specifier–INVALID CASE terminal-reference

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

unit-mnemonic INVALID CONTROL CARD

INVALID DECK NUMBER . . .

INVALID ED MESSAGE DECK NUMBER

job-specifier–INVALID index terminal-reference

INVALID JOB NUMBER

INVALID MC-CHARGE OPTION ALREADY SET

INVALID MIX NUMBER

INVALID MNEMONIC . . .

job-specifier–INVALID LINK terminal-reference

job-specifier–INVALID OPERATOR terminal-reference

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

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

job-specifier–INVALID PARAMETER terminal-reference

INVALID PG

job-specifier–INVALID RETURN terminal-reference

INVALID SD–SERIAL NUMBER REQUIRED

INVALID SERIAL NUMBER

INVALID SL–LOG ALREADY SET

job-specifier–INVALID SUBSCRIPT terminal-reference

job-specifier–INVALID SUBSTRING terminal-reference

INVALID SYNTAX for {CHANGE / REMOVE} COMMA IS REQUIRED FOR MORE THAN ONE {CHANGE / REMOVE}

unit-mnemonic INVALID TYPE CODE . . .

INVALID unit-mnemonic

INVALID UNIT MNEMONIC FOR FN, MUST BEGIN WITH ALPHA

"IL" REQUIRES A PARAMETER

file-identifier IN USE

job-specifier–INSUFFICIENT MEMORY TO OPEN file-identifier

job-specifier IS EXECUTING

pack-id IS ALREADY A SYSTEM DRIVE

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

pack-id IS AN INTERCHANGE PACK

unit-mnemonic IS NOT A USER PACK

pack-id IS NOT INITIALIZED

job-specifier IS NOT STOPPED

pack-id IS {RESTRICTED / INTERCHANGE} PACK

job-specifier IS SUSPENDED

job-specifier–INTEGER OVERFLOW terminal-info

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

INV OPTION option-name

unit-mnemonic LABELED . . . . REEL nnnnnn

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

unit-mnemonic { LABELED . . . / UNLABELED } IN USE BY job-specifier . . .

file-identifier LOAD TERMINATED–DISK ESTIMATE ERROR

file-identifier LOADED

unit-mnemonic LOCK OUT

job-specifier LOCKED DISK FILE file-identifier

option-name LOCKED

unit-mnemonic LOCKED

LOG NOW SET–CLEAR/START REQUIRED

LOG OPTION NOT SET

LOG TRANSFER COMPLETE

pack-id MAY NOW BE POWERED DOWN

unit-mnemonic MEMORY ACCESS ERROR WAIT TILL UNIT IS RESET AND TRY AGAIN

job-specifier–unit-mnemonic MEMORY PARITY

MISSING PARENTHESIS . . .

unit-mnemonic MISSING PACK-ID

MCP RAN OUT OF WORK SPACE WHILE LOOKING FOR interpreter-id WANTED BY program-name = job-number

MODIFY program-name CTRL RCD ERR: . . .

NO SEGMENT DICTIONARY SPACE for program-name = job-number

job-specifier–NO SPACE AVAILABLE FOR [CODE or DATA] [PAGE nnnn] SEGMENT nnnn

NO SPACE AVAILABLE FOR interpreter-name SOUGHT BY program-name = job-number

NO SPACE FOR program-name = job-number

NO SPACE IN INTERPRETER DICTIONARY FOR interpreter-name SOUGHT BY program-name = job-number

**NO SYSTEM DISK FOR PSR DIRECTORY

**NO USER MEMORY FOR CD

file-identifier NOT A BACKUP FILE–REQUEST IGNORED

pack-id NOW A SYSTEM DRIVE–CLEAR/START REQUIRED

unit-mnemonic NOT AVAILABLE

NOT A DISK PACK–CANNOT RL

NOT A QUOTE-MARK . . .

file-identifier NOT CHANGED– { " < FILE-NAME > /=" NOT ALLOWED | BLACK OR ZERO FIRST NAME | file-identifier ALREADY ON DISK | NOT ON DISK | IN USE | RESTRICTED FILE }

NOT ENOUGH MEMORY FOR CM

job-specifier–NAME OR VALUE STACK OVERFLOW terminal-reference

job-specifier–NEEDS AN AX REPLY

program-name job-number NEEDS nnnnnnKB PR = nn hh:mm:ss.s

job-specifier–NO DISK AVAILABLE FOR DUMP

NO DISK SPACE TO BUILD LOG

job-specifier–NO MEMORY AVAILABLE FOR DUMP

NO MEMORY FOR KA

**NO MEMORY FOR PSEUDO READER

**NO MEMORY FOR PSR DATA DIRECTORY (PSR = Pseudo Reader)

NO OVLY DISK AVL FOR program-name = job-number AMT RQD: nnnn
SEGMENTS–RS–ED

NO PRINTER AVAILABLE

NO PRINTER AVAILABLE FOR KP

NO PROGRAMS RUNNING

job-specifier–NO PROVISION FOR I/O ERROR ON file-identifier terminal-reference

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

NO PSEUDO DECKS ON DISKS

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

file-identifier NOT IN DIRECTORY

file-identifier NOT IN DISK DIRECTORY

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

file-identifier NOT LOADED–IN USE BY SYSTEM

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

file-identifier NOT ON DISK

pack-id NOT ON LINE

unit-mnemonic NOT READY

NULL SCHEDULE

NULL . . . TABLE

NUMBER OF PSEUDO READERS CHANGED TO nnnnnn

unit-mnemonic OFF LINE

OUT OF MEMORY SPACE

job-specifier–OUTPUT UNIT NOT AVAILABLE FOR BACKUP

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

PM CANNOT FIND DUMPFILE/integer FOR DUMP/ANALYZER

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

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

job-specifier–unit-mnemonic PRINT CHECK

PRINTER NOT READY

job-specifier–PROGRAM ABORTED terminal-reference

job-specifier–PROGRAM IS NOT WAITING SPO INPUT–AX IGNORED

PSEUDO/nnnnnn NOT ON DISK

PSEUDO/nnnnnn NOT REMOVED–INUSE

job-specifier–unit-mnemonic PUNCH CHECK

unit-mnemonic = { PURGED LABEL / file-identifier [REEL nnnnnn] / UNLABELED }

unit-mnemonic READ CHECK

job-specifier–READ OUT OF BOUNDS terminal-reference

unit-mnemonic RELABELED pack-id { R / U }

job-specifier–REQUESTED A { CODE / DATA } SEGMENT OF LENGTH ZERO terminal-reference

job-specifier–REQUESTED A CORE SPACE NOT EQUAL TO THE SIZE I JUST COMPUTED AS HIS REQUIREMENT–RS–ED MY SIZE = nnnn HIS SIZE = nnnn

job-specifier–READ REQUESTED ON OUTPUT FILE file-identifier terminal-reference

program-name REQUESTED BY "FN" NOT IN DIRECTORY

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

{ RO / SO } REQUIRES THREE OR FOUR CHARACTERS

device-mnemonic REQUIRED FOR REEL nnnnnn file-identifier

message REQUIRES MIX NO.

job-number RS-ED

unit-mnemonic REWINDING

unit-mnemonic { SAVED / TO BE SAVED }

SD REQUIRES NULL MIX

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

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

nnnn SEGS REQ FOR SYSTEM DUMP FILE

SERIAL NUMBER REQUIRED

SPACE REQUIRED BEFORE " or @ . . .

job-specifier–STACK OVERFLOW terminal-reference

job-specifier–SUPERFLUOUS EXIT terminal-reference

SYSTEM/LOGOUT NOT IN DIRECTORY

job-specifier–TANK OVERFLOW terminal-reference

3 DISK SEGMENTS NEEDED FOR SYSTEM/PRINTCHAIN

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

THERE ARE NO RELEVANT BACKUP FILES–PB IGNORED

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

job-specifier–unit-mnemonic TIMEOUT @nnnnnn@

TOKEN TOO LONG–REQUEST IGNORED . . . .

job-specifier–TOO LONG IN USE ROUTINE

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

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

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

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

**TR PLEASE

job-specifier–UNDEFINED RUN TIME ERROR terminal-reference

job-specifier–UNEXPECTED POCKET SELECT terminal-reference

job-specifier–UNINITIALIZED DATA ITEM terminal-reference

unit-mnemonic UNIT PURGED

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

UNIT-MNEMONIC MUST START WITH ALPHA

unit-mnemonic UNLABELED

pack-id WRITE–LOCKOUT

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

job-specifier ZIPPED AN INVALID CONTROL CARD

SECTION 3
SYSTEM SOFTWARE

DISK CARTRIDGE INITIALIZER

General

A disk cartridge must be initialized before it can be used on the system. The purpose of disk initialization is threefold. One, it assigns addresses to all segments on the disk. Two, it checks to see what segments, if any, are unusable (cannot be read from or written to). Any segment found to have errors will cause the entire track in which it resides to be removed from the MASTER AVAILABLE TABLE. If any flaws occur in track ZERO or ONE the entire pack is considered faulty and cannot be used on the system. Three, skeleton table entries, the disk directory, and available tables, for example, are built and the label is written in segment zero.

Disk Initialization Instructions

The Disk Initializer program does not operate under the control of the MCP and must be loaded and executed through the cassette reader on the control panel.

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

Card Columns	Description
1	Drive Mnemonic Letter (A...P)
2	"V" = Verify; Blank = Initialize
3-8	Disk Cartridge Serial Number
10-19	Label
21	TYPE of Cartridge S = System U = Unrestricted R = Restricted I = Interchange
23	Julian Date (YYDDD)
29-42	Remarks (Owner's Name)

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

a. Place the DISK INITIALIZER cassette in the cassette reader in the control panel. The BOT light should be lit at this time.

b. Place the console printer on-line.

c. Place input cards in the card reader. One card for each cartridge to be initialized or verified followed by the ? END card.

d. Set the system MODE switch to the TAPE position and press the CLEAR, then START buttons. This loads the bootstrap loader from the cassette tape and halts the processor.

e. Set the system MODE switch to the RUN position and press START (DO NOT PRESS CLEAR). This will load and execute the initializer.

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

B 1700 DISK CARTRIDGE INITIALIZER – MARK level-number

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

Example:

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

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

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

Disk Verification

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

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

COLDSTART

General

The COLDSTART routine is used to load basic system software and firmware to disk. The routine is furnished on a cassette tape and is loaded via the control panel cassette reader.

The following actions are performed by COLDSTART:

a. Constructs and initializes the disk directory and available tables on the system disk if head-per-track.

b. Loads the MCP from magnetic tape to system disk.

c. Loads the SDL Interpreters for both the 1710 and 1720 series of computers from magnetic tape to the system disk.

d. Loads the CSM firmware for both the 1710 and 1720 series of computers from magnetic tape to system disk.

e. Loads the System Initializer from magnetic tape to system disk.

f. Loads SYSTEM/LOAD.DUMP and FILE/LOADER from magnetic tape to system disk.

g. Makes appropriate entries in the NAME TABLE for all system software and firmware loaded.

h. Constructs the COLDSTART VARIABLES on system disk.

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

NOTE

When a COLDSTART is performed on a system disk that was previously in operation, all the files entered in the disk directory are lost and must be reconstructed. This is due to the disk directory being initialized and cleared by the COLDSTART.

Procedure

The COLDSTART procedure is as follows:

a. Mount a "system" pack on drive 0, (if not a head-per-track system).

b. Set MODE switch to TAPE.

c. Place the COLDSTART cassette in the cassette reader. Cassette is automatically rewound.

d. Press CLEAR, then START.

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

f. Set MODE switch to RUN, press START.

g. Cassette will continue to read. If the system HALTS with @ 4 @ in the L register, the cassette has a hash total error and must be reloaded otherwise it should contain @AAAAAA@. When the cassette has finished loading, the STATE light will come on, and COLDSTART will begin execution.

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

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

The system disk created by COLDSTART is a single system pack configuration, and does not contain a LOG. Once the system is running under MCP control, the number of system drives may be increased using the SD message, and the LOG option set with the SL message.

CLEAR/START and MEMORY DUMP PROCEDURE

General

A Clear/Start is used by the system operator to restore the system to an operable state. A Clear/Start must be performed under any of the following conditions:

a. System Power-up.

b. an unscheduled halt.

c. an uninterruptible system software loop.

d. the system software/firmware is changed (via CM message).

A Clear/Start performs the following functions:

a. Terminates all programs being executed.

b. Empties the schedule.

c. Writes correct parity and zeros throughout memory.

d. Loads the MCP, SDL Interpreter, System Initializer and the Central Service Module (CSM) specified by the NAME TABLE entries selected.

e. Returns control to the MCP.

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

Clear/Start Procedure

a. Halt processor with the INTERRUPT switch.

b. Place Clear/Start cassette in cassette reader.

c. Press CLEAR.

d. Set MODE switch to TAPE position.

e. Press START (When tape stops, check the L register for all A's. At this point enter any temporary changes to be made in the T or X registers.)

f. Set MODE switch to RUN position.

g. Press START.

The same Clear/Start program is usable on any system and with either the MCP I or the MCP II.

Name Table

The NAME TABLE is built during COLDSTART and resides on disk. It identifies firmware and system software that can be used in the operational environment of the system.

The operator may select from NAME TABLE different environments for operation. However, not all systems will be able to use many of these programs since they are strictly for experimental system software development and system software debugging.

The main advantage of the NAME TABLE method of selecting an operating environment is the ability to at all times recover to the standard mode of operation.

A typical COLDSTART procedure will load and identify for the system the following:

a. A standard MCP

b. A SDL Interpreter for both the B 1710 and B 1720 series of computers

c. A CSM for both the B 1710 and B 1720 series of computers

d. A System Initializer

e. SYSTEM/LOAD.DUMP

f. FILE/LOADER

g. SYSTEM/MEM.DUMP

This is enough system software and firmware to begin operations on whatever hardware is available. A system pack may be moved from one system to another and started by merely performing a Clear/Start.

Operating Environments

The CM message is used to identify the function of various programs to the system for subsequent usage. See the CM input message for the syntax to be used.

The following list describes the function code or the system software mnemonic and its meaning.

NAME TABLE Entry Number	System Software Mnemonic (Function Code)	Meaning
0	N	Standard System Initializer
1	NE	Entry System Initializer
2	NX	Experimental System Initializer
3	G1	1710 Central Service Module
4	G2	1720 Central Service Module
6	GE	Entry Central Service Module
7	G1T	1710 Trace Central Service Module

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

The purpose of the CM input message is to identify a file on System Disk to be used for a designated function.

Example:

```
CM    MX      MCP/XYZ
```

The above example makes the file MCP/XYZ the experimental MCP and will be the program executed when an experimental MCP is called for.

Selecting Environments

With the appropriate files loaded and CM-ed, there are four general environments which can be selected as a basis for operation:

a. Standard MCP (MCP II)

b. Standard MCP with Trace

c. Entry MCP (MCP I)

d. Entry MCP with Trace

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

a. STANDARD vs. ENTRY

b. TRACE vs. NON-TRACE

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

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

A Clear/Start is required to effect any change. The choices become the new basis for operation. They remain in effect until they are changed explicitly, but they can be switched on a temporary basis during the Clear/Start procedure.

Temporary Environment Changes

Operations following a Clear/Start can be tailored to the needs of system programmers by setting the following values in the T register.

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

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

Bits	Description
7	Run with experimental CSM
8-11	When bit 1 is set, the following programs will be run.

Value	Identification
0	SX
1	SDD
2	SIO
3	SDL, using SL to load with interpreter

12-23	Must be left zeros

Another option that can be made during Clear/Start is the designation of the system disk. To override the usual Clear/Start selection, load the following values in the X register.

Bits	Value
16-19	Port
20-23	Channel

Memory Dump Procedure

The memory dump as well as other temporary changes may be accomplished during the Clear/Start procedure. Between steps (e) and (f) in the Clear/Start Procedure simply set the proper bits in the appropriate register and continue with the normal Clear/Start procedure.

The memory dump requires that bit 0 of the T Register be turned on at this time.

Firmware Detected Errors

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

L Register Value Bits 0-15	Program Identification
@ 0000 @	Central Service Module
@ 000F @	SYSTEM/INIT
@ 00F0 @	CLEAR/START
@ 0F00 @	MEM/DUMP

L Register Value Bits 16-23	Error Description
@ 01 @	No device on the designated I/O channel.
@ 02 @	I/O device on channel is not disk. (See T register.)
@ 03 @	Disk is not idle. (See T register for status.)
@ 04 @	Time-out while waiting for service request.
@ 05 @	Bad reference address. (X = good, Y = bad.)
@ 06 @	Bad status count after service request. (See T register.)
@ 07 @	Bad result status from I/O control. (See T register.)
@ 08 @	Seek time-out (timed by system software).

L Register Value Bits 16-23	Error Description
@ 09 @	Memory parity error in I/O descriptor.
@ 0A @	Memory parity error in I/O data.
@ 0B @	Time-out waiting for I/O operation to complete.
@ 0C @	Exception condition after 15 retries. (See T register.)
@ 0D @	Exception on test I/O operation.
@ 0E @	Designated port and channel is not disk.
@ 0F @	No disk on system.
@ 10 @	Designated port is invalid.
@ 11 @	Designated channel is invalid.
@ 12 @	Not enough memory for this program.
@ 13 @	Memory parity after CSM overlay.
@ 14 @	Parity error somewhere in memory.
@ 15 @	NAME TABLE entry (number in T register) is zero or blank.
@ 16 @	Memory dumpfile port not equal to 7.
@ 17 @	Memory dumpfile address equal zero.
@ 18 @	Disk address in INITIALIZER IPB equal zero.
@ 19 @	MCP type field in HINTS is zero.
@ 1A @	Invalid stand-alone program specified.
@ 1B @	Stand-alone SDL file not available.
@ 1C @	No console printer on system.

DISK FILE COPY

General

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

Cards are used as input for the DISK/COPY routine. Any number of files may be copied during one execution of DISK/COPY.

DISK/COPY Operating Instructions

The following figure represents the DISK/COPY control deck.

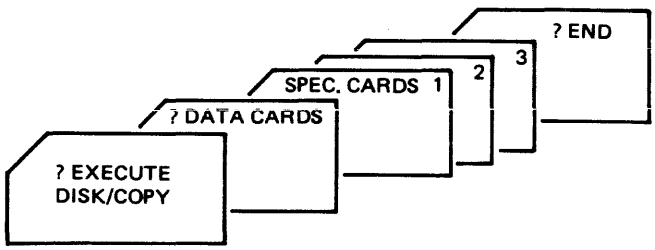


Figure 3-1. DISK/COPY Control Deck

Specification Cards

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

Specification cards are free-form. Each card must contain two disk file-identifiers with the first file-identifier being the file to be copied, and the second file-identifier being the new copy of the file.

The format for the file-identifiers is the same as used for MCP control cards. See the REMOVE control instruction for further syntax explanation.

If the file-identifier is to be retained when copying to another disk, the new file-identifier may specify only the name of the pack-id followed by a slash.

Examples:

a. To copy file AAA on a systems disk to another location on the systems disk with the name BBB:

AAA BBB

b. To copy a file AAA on a systems disk to another disk named NEWDISK and retain the file-identifier:

```
AAA     NEWDISK/AAA/
```

c. Since the file-identifier is not changed in example (b), the same result would be obtained by using the following specification card.

```
AAA     NEWDISK/
```

DMPALL

General

The program DMPALL has two separate functions: (1) printing the contents of files, and (2) reproducing data from one hardware device to another. Execution may be from either the console printer or card reader.

Printing

Printing files consist of the following:

a. Data may be card, magnetic tape, paper tape, or disk.

b. Any file can be read up to a 1000 bytes per logical record.

c. Contents can be printed in byte, digit, or combined form.

d. Printing may begin with a specified record number and terminate after a specified number of records are printed.

Reproducing

Reproducing files may be executed as follows:

a. A file may be reproduced from any card, magnetic tape, paper tape, or disk.

b. File-identifiers, record lengths, and blocking factors may be changed during the reproduction.

c. Reproducing may begin with a specified record number and terminate after a specified number of records.

Operating Instructions

CONSOLE PRINTER

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

```
DMPALL   =   mix-index   BOJ.

DMPALL = =   mix-index   ENTER SPECS.

DMPALL   =   mix-index   ACCEPT.
```

The operator replys to the ACCEPT message by entering an AX message containing the specifications needed to perform the DMPALL operation.

The directory of a LIBRARY tape created by the program SYSTEM/LOAD.DUMP can be either punched or printed using the following procedure:

```
mix-index AX PD [PUNCH] tape-identifier
```

When PUNCH is specified, the tape directory will be output to cards for use with the program SYSTEM/LOAD.DUMP. With PUNCH omitted, the default print option will list the directory.

CARDS

The DMPALL execute control deck has the following format:

? EXECUTE DMPALL FILE SPEC NAME specification-file-identifier;

? DATA specification-file-identifier

(specification cards)

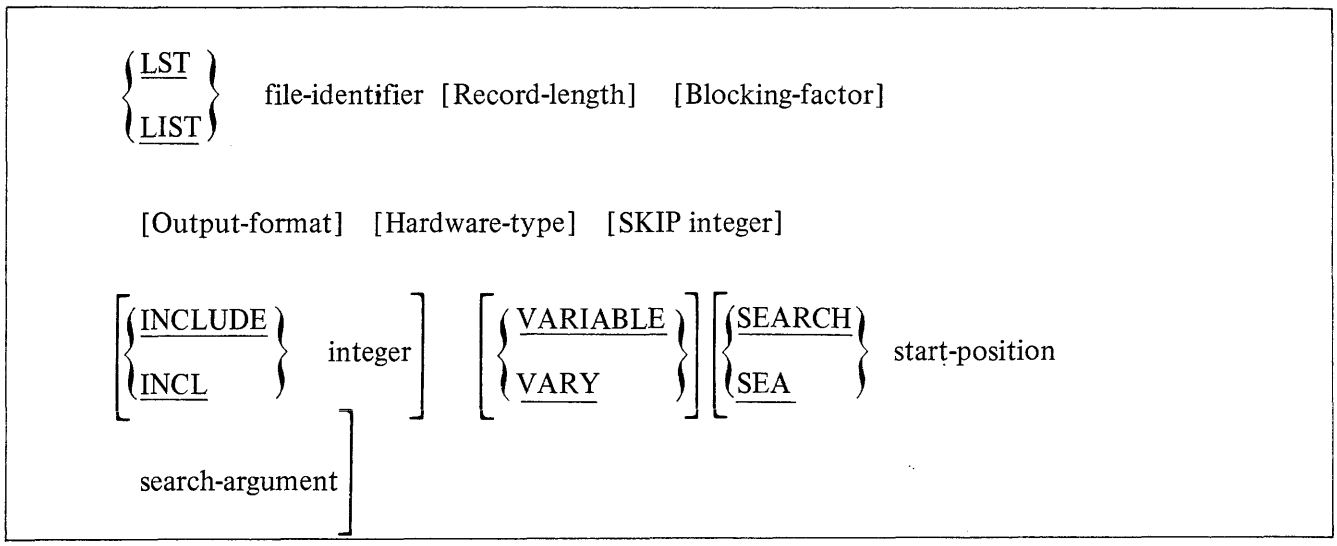
? END

A semicolon must terminate the specification string, after which comments may be entered. There may be more than one card in a specification card file.

All specification entries are free form in the first 72 columns of the card, and may be separated by either a space or a comma, or a combination thereof. The card file containing the specifications (one per card) is loaded to disk, and each specification is executed in turn from there.

Print Specifications

The specification string for printing a file is as follows:



The file-identifier entry must immediately follow the LIST or LST entry, and is required for all files. The format of the file-identifier entry is the same as used MCP control instructions; therefore may consist of from one to three separate identifiers separated by slashes. A file-identifier that is entirely numeric or which contains special characters must be surrounded by quotes.

The record-length in bytes must be the first numeric entry following the file-identifier. If omitted, a record-length of eighty is assumed. For disk files the record-length used will be that of the file when created.

The blocking-factor must be the second numeric entry following the file-identifier. If omitted, a blocking factor of one is assumed. For a disk file when both the record length and blocking factor entry are omitted, the blocking factor with which the file was created will be used.

The output-format entry may be specified as:

a. Alpha: A or ALFA.

b. Numeric: N, NUM, H, or HEX.

c. Alphanumeric: When entry is omitted.

The hardware-type entry may be one of the following:

a. Card files: CRD or CARD

b. Magnetic tape files: MTP or TAPE

c. Paper tape files: PPT or PAPER

d. Disk files: DSK, DISK, or the entry may be omitted.

e. 96-col. card files: C96 or CARD96

The SKIP integer entry may be entered to begin printing with a specified record as denoted by the integer.

The INCLUDE or INCL integer entry may be used to specify how many records should be included in the printout.

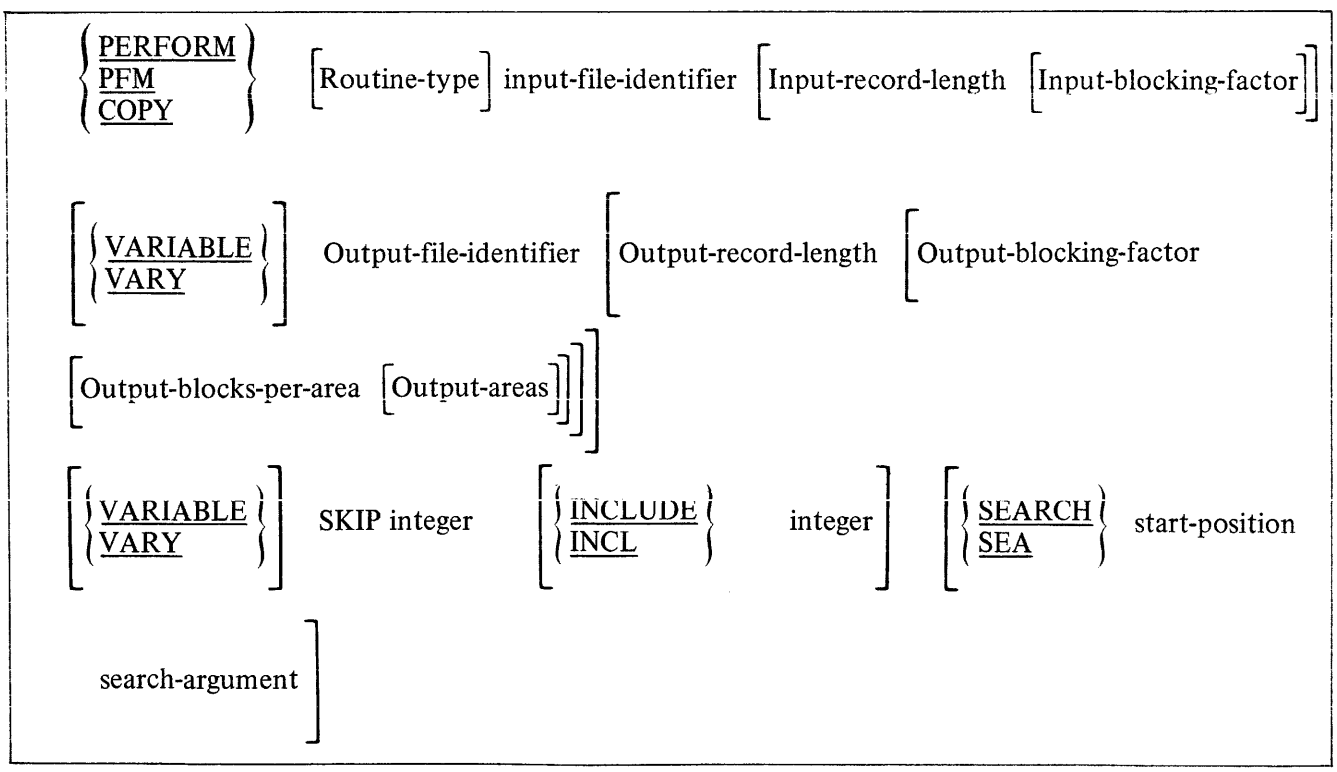
The VARIABLE or VARY entry may be used to specify tape or disk files having variable length records.

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

The printed output is headed with the file-identifier, record length, blocking factor, the current date, and the time. In addition a printout of a disk file will have the value of the End-of-File pointer in the heading. A running record count is printed in the left hand margin.

Reproducing Specifications

The reproduction string consists of the following specifications:



PERFORM, PFM, or COPY informs DMPALL that media conversion is desired.

The Routine-type entry may be either in the long-hand or short-hand form.

The long hand form utilizes the names of two of the following media:

a. Card files: CARD

b. Magnetic tape files: TAPE

c. Paper tape files: PAPER

d. Disk files: DISK or the entry may be omitted.

e. 96-col. card: CARD96.

f. Binary 80-col. card reproduction: BINBIN

The short hand form uses a combined abbreviation format.

OUTPUT DEVICES

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

Example:

To go from card to magnetic tape the short-hand form Routine-type would be CRDMTP. The long-hand form would be CARD TO TAPE with the TO being optional.

The format of input-file-identifier is the same as used in MCP control instructions.

The input-record-length must be the first numeric entry following the input-file-identifier in bytes. If omitted, a record length of eighty is assumed for all files except disk files which will use the record length of the file when created.

The input-blocking-factor must be the second numeric entry following the input-file-identifier. If omitted, a blocking factor of one is assumed. For a disk file where both the record length and blocking factor entries are omitted, the blocking-factor with which the file was created will be used.

The VARIABLE or VARY entry may be used after the input-file-identifier entries to indicate that the input file will have variable length records, but not variable length output.

The format of the output-file-identifier is the same as for the input-file-identifier.

The first numeric entry following the output-file-identifier must be the output-record-length in bytes. If omitted, a record length of eighty is assumed unless the input file and the output file are both disk files. Then the default output-record-length will be assumed to be the same as the input-record-length.

The output-blocking-factor must be the second numeric entry following the output-file-identifier. If omitted, a blocking-factor of one is assumed unless the input file and the output file are both disk files and the output-record-length entry was omitted. Then the default output-blocking-factor will be assumed to be the same as the input-blocking-factor.

The number of blocks.per.area must be the third numeric entry following the output-file-identifier. This entry is only applicable to disk files. If omitted, 100 blocks.per.area is assumed unless both the input file and the output file are disk files and the record-length, blocking-factor entries were omitted for both the input file and the output file. Then the number of blocks.per.area for the input file will be used for the output file as well.

The Output-areas is the number of areas set for the output file. Default is 25.

The VARIABLE or VARY entry may be used after the output identifier to indicate variable length input records with variable length records being produced.

The SKIP integer entry may be used to skip to a specified record prior to creating the output file.

The INCLUDE or INCL integer entry may be used to specify how many records should be included in the output file.

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

Examples:

a. Keyboard Console Input

EXECUTE DMPALL

DMPALL = mix-index BOJ.

DMPALL = mix-index ENTER SPECS.

DMPALL = mix-index ACCEPT.

A response of

LIST PACKA/PAYROLL/ A SKIP 50

causes a disk file located on the removable disk PACKA to be printed in alpha format beginning with the fiftieth record.

A response of

1AX COPY CRDDSK CARD SOURCE 80 2

causes a card file with the file-identifier of CARD to be written to a disk file, 80 character records, blocked 2, with a file-identifier of SOURCE.

A response of

```
1AX  COPY PROGRAM/B  CCC/PROGRAM/B
```

causes a disk file PROGRAM/B located on a system disk to be copied to the removable disk CCC with the file-identifier PROGRAM/B. The new copy on disk CCC will be an exact copy. Therefore, record length, blocking, number of areas, and area size will be the same as the original file.

b. Card Input

? EXECUTE DMPALL FILE SPEC NAME SPECCARDS will allow the operator to enter any number of specifications via a card reader. DMPALL will look for a card file with the file-identifier SPECCARDS. The specifications will be loaded to disk, and then executed one-at-a-time from there.

```
? EXECUTE DMPALL FILE SPEC NAME SPECCARDS;
? DATA SPECCARDS
  COPY CRDDSK XXX 80 1 DSKFIL 80 1
  LIST DSKFIL A
? DATA XXX
  (card data deck)
? END
```

The specifications will cause the card file XXX to be loaded to disk, then listed in alpha format.

FILE/LOADER

General

The purpose of FILE/LOADER is to load card decks to disk punched by the program FILE/PUNCHER.

The FILE/LOADER card deck consists of the standard EXECUTE control card, a dollar card, an asterisk card, the data cards, and the END card.

Dollar Card

The dollar card is output by FILE/PUNCHER and identifies the file to be loaded. The dollar card can also be modified by the operator to change the name of the file-identifier.

The format of the FILE/LOADER dollar card is:

$ file-identifier

The "$" must be in column one and the file-identifier being free-form from column 2 through 80.

Dollar Dollar Card ($$)

Files produced by the MIL compiler (Micro Implementation Language) must be loaded using the $$ card to distinquish them from card files output by FILE/PUNCHER. The asterisk (*) card must not be used when using the $$ card.

Below is the card format produced by the MIL compiler which takes six cards to fill a disk segment.

Column	Description
1-6	Load address (Relative)
7	Blank
8-9	Number of bits on card
10	Blank
11-70	Data in hexadecimal format (30 Bytes)
71-72	Blank
73-80	Card sequence number

The format of the FILE/LOADER dollar dollar card is:

$$ file-identifier

Asterisk Card

The asterisk card is used to input the values for the file which is being loaded to disk. This card is produced by FILE/PUNCHER and should not be changed prior to input. When the asterisk card is missing, the card file is assumed to be a code file. The asterisk card must not be used when the first card of the file is a dollar dollar ($$) card.

The format of the FILE/LOADER asterisk card is:

Column	Description	
1	"*" Asterisk Sign	
3	File Type 1 LOG 3 Control Deck 4 Backup Punch 5 Backup Print 6 Dump 7 Interpreter 8 Code 9 Data	
5-10	EOF pointer	Right Justified, Leading Zeros Optional
12-17	Record Size in bits	
19-20	Records.per.Block	
22-24	Areas	
26-31	Segments.per.Area	

NOTES

(1) If a code file is being loaded, the asterisk card is optional and default values are assumed.

(2) If a code or interpreter file is designated on the asterisk card, only the EOF pointer is used. All other fields are ignored. If the EOF pointer field is blank, 100 segments for the interpreter or 500 segments for the code will be used as default values.

(3) All code and interpreter files will be closed with CRUNCH which frees the area not being used for the file.

Example:

```
? EXECUTE FILE/LOADER DATA CARDS
$ file-identifier
* . . . (Optional)
  data deck
[$ file-identifier ]
[*  ————————      ]
[  data deck       ]
? END
```

RESPONSE: File-identifier LOADED (Displayed after each load)

Error Messages

MISSING "$" IN COLUMN ONE

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

MISSING file-identifier

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

SEQUENCE ERROR FOLLOWING nnnnnnn–file-identifier NOT LOADED

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

RECORD.SIZE SPECIFIED nnnn–file-identifier NOT LOADED

AREAS SPECIFIED = 0 – file-identifier NOT LOADED

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

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

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

INVALID FILE TYPE SPECIFIED–file-identifier NOT LOADED

BLOCK SIZE 56 – file-identifier NOT LOADED

EMPTY DECK–file-identifier NOT LOADED

There are no cards following the specification card(s).

"*" CARD INVALID–file-identifier NOT LOADED

An asterisk card following a dollar dollar card is invalid.

FILE/PUNCHER

General

The purpose of FILE/PUNCHER is to output disk files to cards in a hexadecimal format that is acceptable as input to FILE/LOADER. The dollar card and the asterisk card used by FILE/LOADER are also output when FILE/PUNCHER is executed.

The file-identifier is supplied to the program by an AX input message. For example:

EXECUTE FILE/PUNCHER

FILE/PUNCHER=mix-index ENTER FILE IDENTIFIER
FILE/PUNCHER=mix-index ACCEPT

mix-index AX file-identifier (free-form)

After punching the output file, the program will repeat the above messages and wait for another file-identifier to be entered. By responding with a blank file-identifier, the program will go to EOJ.

Below is the card format produced by FILE/PUNCHER which takes five cards to fill a disk segment.

Column	Description
1-72	Data in hexadecimal format (36 bytes)
73-80	Card sequence number

Error Messages

file-identifier NOT ON DISK

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

SORT

General

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

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

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

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

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

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

SORT Execution Deck

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

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

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

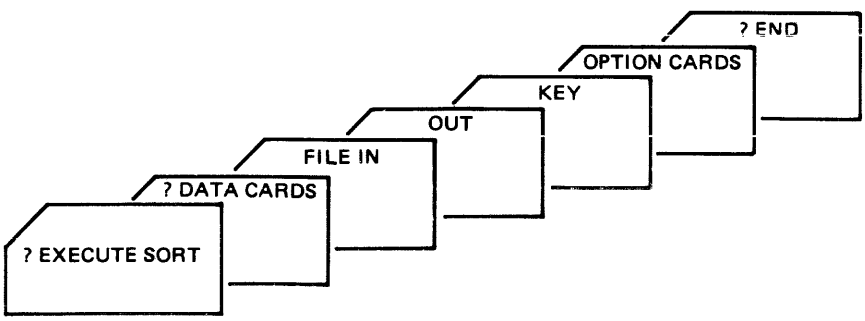


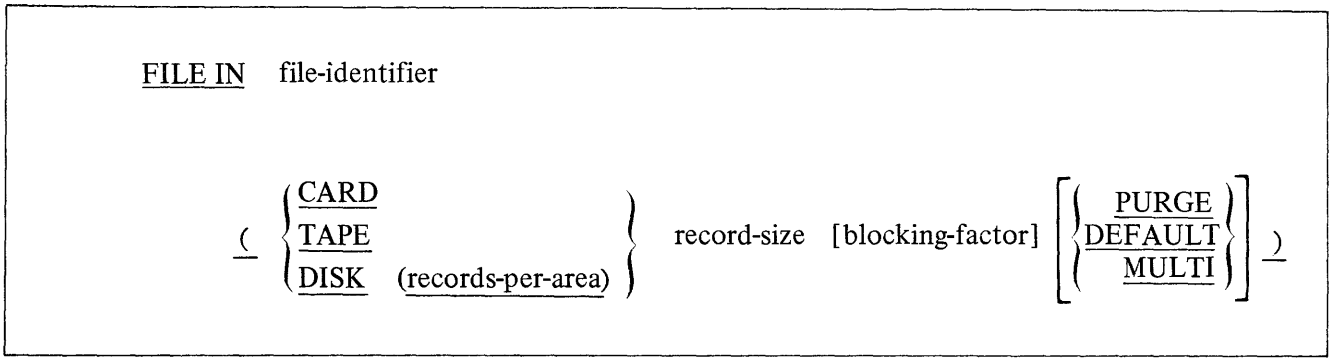
Figure 3-2. SORT Execution Deck

The FILE Statement

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

FILE IN

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



DP-ID

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

FILE-IDENTIFIERS

File-identifiers are standard.

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

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

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

CARD

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

TAPE

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

DISK

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

RECORDS-PER-AREA

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

RECORD-SIZE

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

BLOCKING-FACTOR

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

PURGE

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

DEFAULT

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

This option applies only to disk files.

MULTI

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

OUT

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

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

The OUT statement has the following format:

```
OUT  file-identifier

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

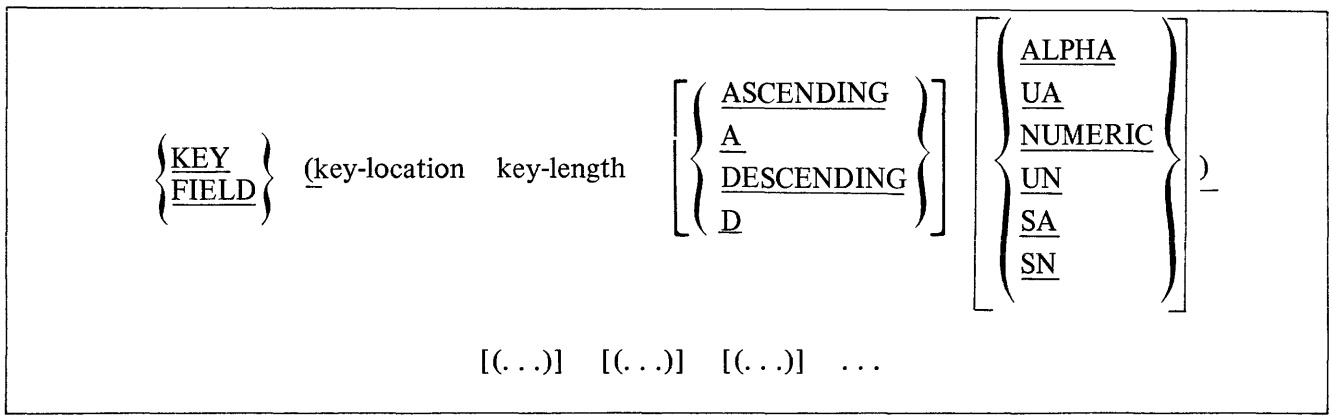
Note that in the above example parentheses serve as delimiters between parameters so that additional spaces are permitted but not required.

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

KEY-LOCATION

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

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

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

KEY-LENGTH

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

ASCENDING or A

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

DESCENDING or D

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

ALPHA or UA

ALPHA or UA (unsigned alpha) indicates that the data is alphanumeric, and the key-location of the field is counted in 8-bit units from the beginning of the record. ALPHA or UA need not be specified as they are the default when no data type is specified.

NUMERIC or UN

NUMERIC or UN (unsigned numeric) indicates that the data is 4-bit numeric, and the relative position of the field is counted in 4-bit units.

SA

SA (signed alpha) indicates that the data is alphanumeric and that some or all of the keys may contain a minus sign.

The key-location is specified as the most significant byte of the key.

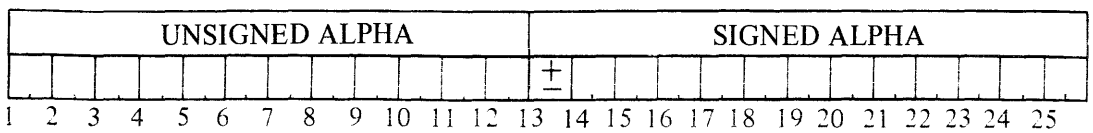
The minus sign is represented as a hexadecimal D.

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.

The minus sign is represented as a dexadecimal D.

Examples to illustrate several key descriptions follow:



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

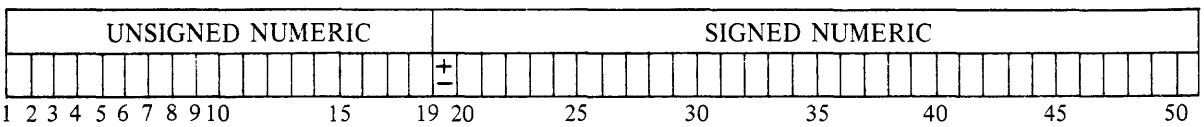
KEY (5 2) describes the field that starts with the fifth byte and is two bytes long. The data is type ALPHA and the output sequence is to be ascending order.

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

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

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

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



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

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

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

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

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

SORT Option Statements

The purpose of the sort option statements is to allow the user to optimize the sort and add commments 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

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

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

Example: MEMORY 15000

integer RECORDS

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

Example: 12500 RECORDS

TIMING

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

This option does not apply to the INPLACE sort.

BIAS

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

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

Example: BIAS 60%

The percent sign is optional and may be omitted.

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

TAGSORT

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

Input files can be of any type.

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

TAGSORT cannot be used with INPLACE sort.

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

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

INPLACE

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

SYNTAX

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

COMMENT

Any non-reserved word or character.

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

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

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

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

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

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

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

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

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

SORT Reserved Words

(
)
,
%
/

A
ALPHA
ASCENDING
ASSEMBLE

BIAS

CARD
CARDS
COMP
COMPILE
COMPLEMENT
CORE

D
DEFAULT
DESCENDING
DISK
DISTRIBUTE

E
EVEN
EXECUTE

FIELD
FIELDS
FILE

GENERATE

IDENT
IN
INPLACE

KEY
KEYS

MEMORY
MERGE

NOPRINT
NUMERIC

O
ODD
OUT

PACK
PAPER
PARTITION
PRINTER
PURGE

RECORDS
RESTART

SA
SAVEBLOCK
SB
SN
SYNTAX

TAGSEARCH
TAGSORT
TAPE
TIMING

UA
UN
USERBLOCK

V

WAIT

ZIP

COBOL CROSS REFERENCE UTILITY PROGRAM (COBOL/XREF)

General

The COBOL Cross Reference Utility Program accepts as input a syntactically correct COBOL source program, and depending on the option selected, outputs a cross reference listing, or a program source listing only, or both a cross reference and a program source listing.

Operating Instructions

The source program may reside on disk, magnetic tape, or punched cards. The figure below is an example of a COBOL/XREF execution card deck.

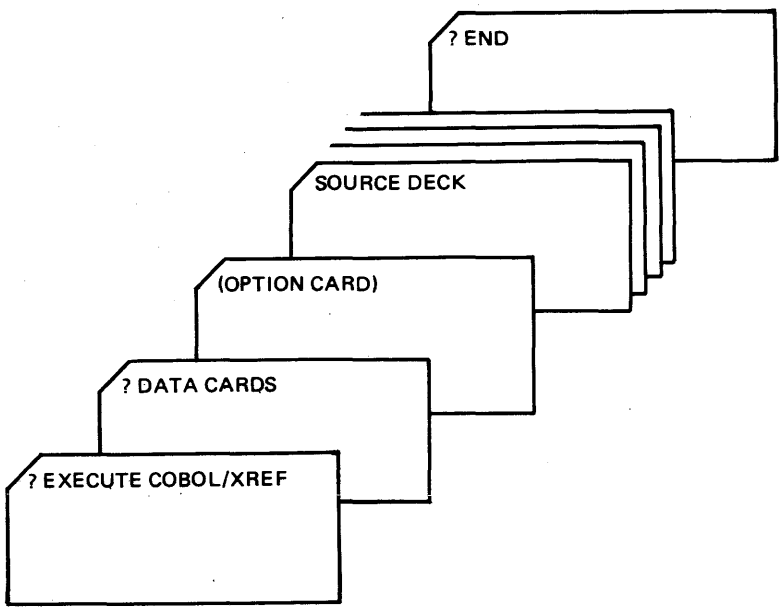
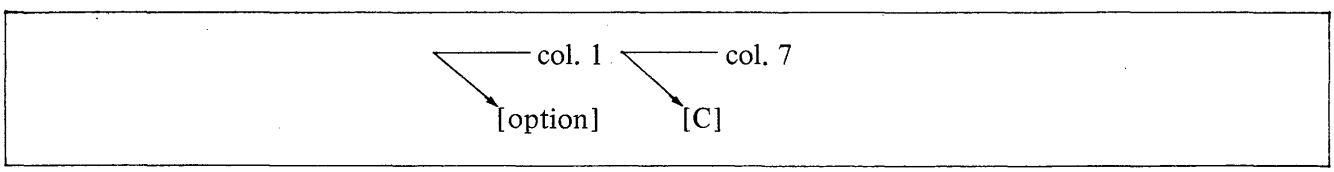


Figure 3-3. COBOL/XREF Execution Deck

Option Cards

The option entry specifies the location of the source medium and the output desired. Below is the format of the COBOL/XREF option card.



The option entry must begin in column 1.

The letter C denotes that the COBOL COPY verb is used within the source program. C, when used, must be in column 7. Requested library files must reside in the disk directory. The library sequence numbers within a source program are indicated by a L to the left of the sequence number.

The options and their descriptions are as follows:

CARD	The input source program is punched cards. Produces a cross reference listing. CARD is the default input. If the option entry is omitted the input is assumed to be cards.

CDLIST — The input source program is punched cards. Produces both a source program listing and a cross reference.

DISK — The input source program having a file identifier COBOLW/SOURCE resides on disk. Produces a cross reference listing.

DKLIST — The input source program having a file identifier of COBOLW/SOURCE resides on disk. Produces both a program source listing and a cross reference.

LISTCD — The input source program is punched cards. Produces a source program listing only.

LISTDK — The input source program is on disk. Produces a source program listing only.

LISTTP — The input source program having a file identifier of SOLT is on magnetic tape. Produces a source program listing only.

TAPE — The input source program having a file identifier of SOLT is on magnetic tape. Produces a cross reference listing.

TPLIST — The input source program having a file identifier of SOLT is on magnetic tape. Produces both a source program and cross reference listing.

Internal File Names

Internal File Identifiers	External File Identifiers	Description
CARDS	CARDS	Input Source Cards
TAPES	SOLT	Input Source Tape
DISKS	COBOLW/SOURCE*	Input Source Disk
CBXPRT	XREFER	Printer Output

*Refer to the COBOL Compiler Option NEW for the creation of this file.

Examples:

Card:

```
? EXECUTE COBOL/XREF
? DATA CARDS
CARD C
    (source deck)
? END
```

Disk: ? EXECUTE COBOL/XREF

? DATA CARDS

DISK

? END

NOTE

If more than one cross reference file may exist, the FILE statement must be used to make each file identifier unique.

Tape: ? EXECUTE COBOL/XREF

? DATA CARDS

TAPE

? END

LOG/CONVERSION

General

The LOG/CONVERSION program extracts information from the file LOG/#n and creates an output file NEW.LOG/#n in COBOL-RPG format. The output file NEW.LOG/#n is assigned the same eight digit number (n) as the input file.

Execution

The program LOG/CONVERSION cannot be executed until the file SYSTEM/LOG has been transferred to LOG/#n (Refer to the LG and TL Input Messages). The eight digit numerical identifier is assigned using an ACCEPT message.

NEW.LOG/#n

The file NEW.LOG/#n may contain three types of records:

1. Clear/Start
2. Program Parameter Blocks (PPBs)
3. File Parameter Blocks (FPBs)

A Clear/Start record is constructed for each Clear/Start performed on the system. Each program executed on the system will have a Program Parameter Block record followed by one or more File Parameter Blocks, depending on the number of files declared.

DISK/DUMP

General

The DISK/DUMP program is a segment-by-segment transfer from cartridge to cartridge or pack to pack. If a segment cannot be read or written, two retries will be attempted. If the retries are unsuccessful, the DISK/DUMP will not be allowed. The pack label is checked for validity but not copied.

Operating Instructions

The DISK/DUMP program does not operate under the control of the MCP. It is loaded and executed through the cassette reader on the control panel. Information will be supplied to DISK/DUMP through the console printer as to which drives to use. To terminate the DISK/DUMP program, enter a blank after the last dump has been made.

After the DISK/DUMP program has been loaded, the following messages will be output:

DISK DUMP MARK version-number

ENTER INPUT DRIVE 〈DC ? or DP ?〉

ENTER OUTPUT DRIVE 〈DC ? or DP ?〉

Then after each successful execution of a disk dump, the following set of messages are output:

DUMP COMPLETE FROM input-drive TO output-drive

ENTER INPUT DRIVE (DCx or DPx) OR BLANK TO TERMINATE

END DISK DUMP (if a blank is entered)

Example:

DISK DUMP MARK IV.1

ENTER INPUT DRIVE – 〈DC ? OR DP ?〉

DCC

ENTER OUTPUT DRIVE – 〈DC ? OR DP ?〉

DCB

DUMP COMPLETE FROM DCC TO DCB

ENTER INPUT DRIVE – 〈DC ? OR DP ?〉 OR BLANK TO TERMINATE

END DISK DUMP

Error Messages

1. DISK ERROR – RESULT STATUS IN "T" (Observe the T register to determine type of error. Press START for a retry.)
2. INVALID RESPONSE – TRY AGAIN
3. CANNOT COPY FROM PACK TO CARTRIDGE

4. CANNOT COPY FROM 200 TPI TO 100 TPI

5. COMPARISON ERROR ON COPIED DATA (Press START to restart.)

6. TEMPORARY TABLE FILLED–CANNOT COPY (input drive). (Clear/Start this pack and try again.)

7. INVALID DRIVE ENTRY (drive-entry)

8. VALID ENTRIES ARE A–X (an entry's high limit depends on the system. It can be from D . . . X.)

9. DISK NOT READY (input-or-output drive)

10. DISK NOT PRESENT (input-or-output drive). Reset the disk control and reload program.

11. WRITE LOCKOUT (output-drive)

12. PACK LABEL BAD (input-or-output-drive)

13. INVALID DUMP–REMOVED SECTORS ON (input-or-output-drive). Pack with bad sectors cannot be used with DISK/DUMP.

14. I/O ERROR (input-or-output-drive address address-of-error)

    Ten retries have been made on input parity errors.

    Two retries have been made on timeout errors.

    Ten retries, rewrites, and 10 more retries have been made on output parity errors.

REMOTE JOB ENTRY SYSTEM (RJE)

Introduction

The B 1700 Remote Job Entry System uses data communication techniques that allow an operator to enter through a B 1700 computer a job or program that is to be executed on a Central computer. Both the B 1710 and the B 1720 series of computers may be used as input. The central computer or "host" may be any Burroughs Medium or Large System. Following execution of the job by the central system, all output information for the console printer, line printer, and/or card punch is transmitted to the remote B 1700 system.

Figure 3-4 is an example of a typical RJE system. Although all central computer systems can accomodate more than one remote system, the maximum number depends on the central system restrictions.

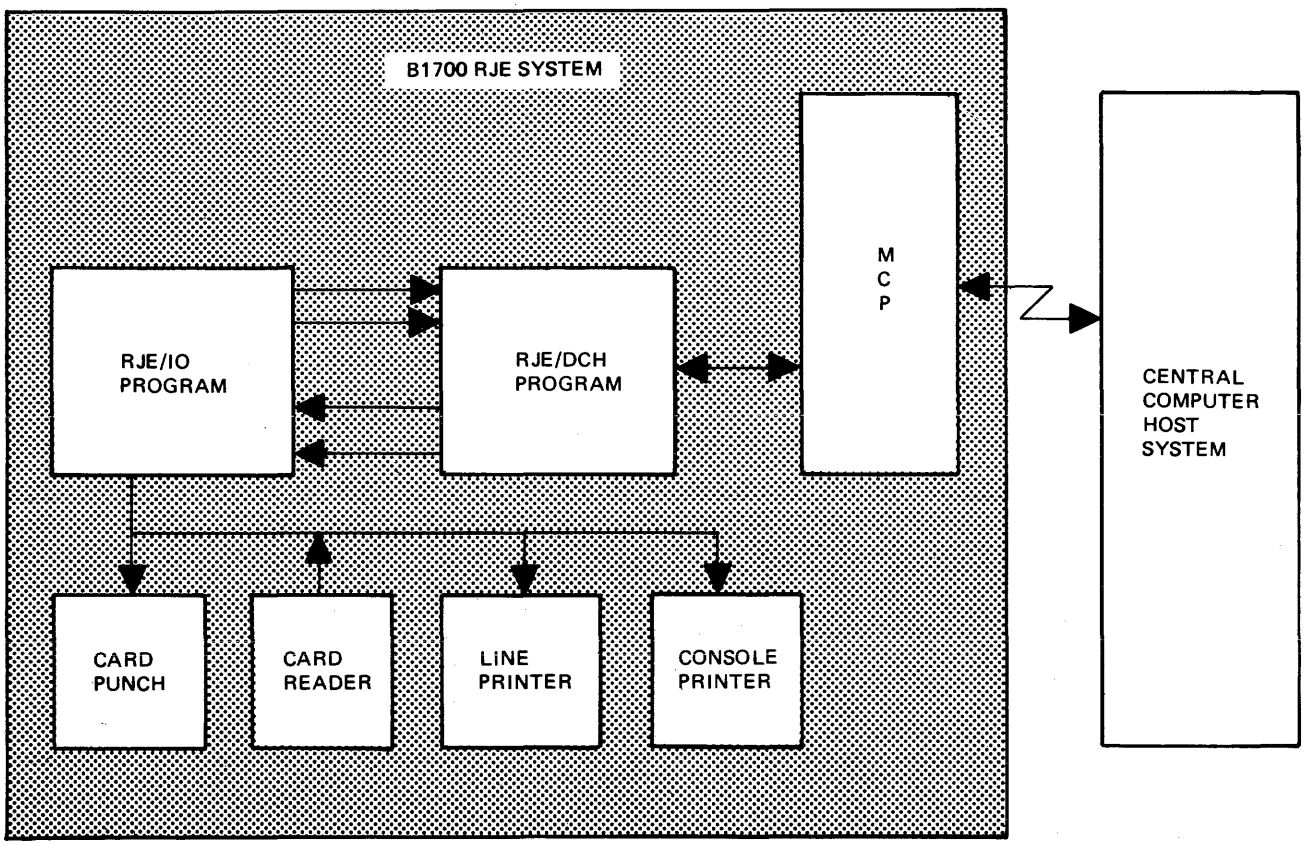


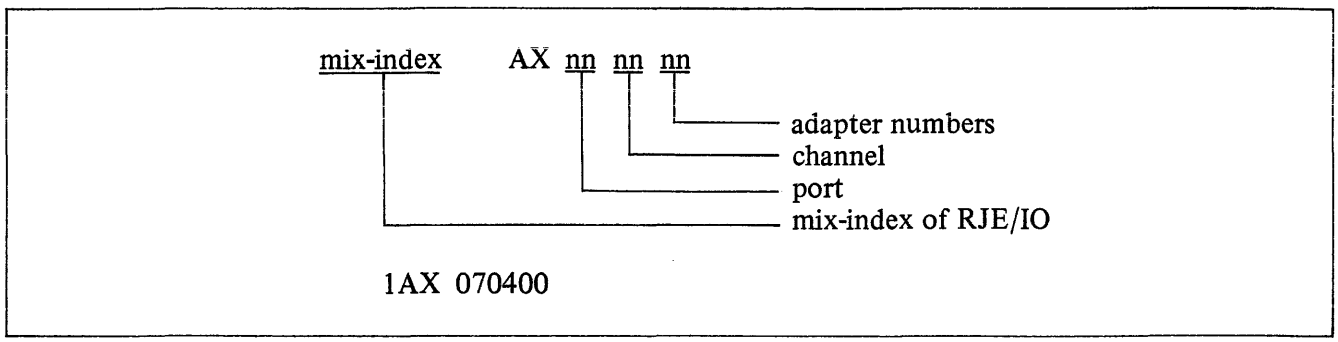
Figure 3-4. Remote Job Entry System

The Remote Job Entry system software consists of two programs: RJE/DCH and RJE/IO. The RJE/DCH program is a data communications handler that performs all transmitting and receiving of data to and from the central computer, and sends all messages received from the central computer to RJE/IO.

The RJE/IO program performs all input/output operations for the B 1700 RJE system. Also it sends all messages received from the central system to the appropriate output device.

Operating Instructions

The B 1700 Remote Job Entry system software resides on the systems disk, and is initiated by executing RJE/IO. The RJE/IO program then zips the program RJE/DCH which first displays the date and time of execution on the console printer, and then makes a request for the port, channel, and adapter numbers using an ACCEPT (AX) message. For example:



With the acceptance of this message, RJE/DCH will establish communication with the Central System. When switched lines are used, the B 1700 operator must manually dial-up the central system. Either one of the two following messages will be displayed on the console printer to indicate communication has been established between the remote and central system.

ON LINE – Indicates that the remote system detected the central system first

HOST ESTABLISHING – Indicates that the central system detected the remote system first

Remote Deck Control Cards

A special feature has been implemented in the MCP to distinguish those control cards for the remote (B 1700) system from the control cards for the host (Large or Medium Burroughs computer) since each require an invalid character to define a control card. This feature uses the control words STREAM and TERMINATE to allow programs to read control cards in RJE mode.

Example:

```
? STREAM RJE/CARDS

        remote deck(s)

? TERMINATE RJE/CARDS
```

All card images in the remote deck(s) will be transmitted to the central system. If the last card in the remote deck(s) is not the TERMINATE card, the CARD READER NOT READY message is output and waits for more card input.

RJE System Control Messages

Input messages for the remote jobs being processed by the central computer may be input by the B 1700 console printer using the following format:

mix-index AX (message)

The mix-index of the AX message refers to the mix index number of the RJE/IO program. The content of the message sent by the AX message is dependent on the central systems demands. In general, these messages include all central system input messages that allow the operator to check the status of, and exert control over, all programs and data files entered via the remote system.

Remote Control Message Entry

Local control messages or those being entered through the B 1700 and intended for B 1700 RJE system software are designated by the use of a period (.) immediately preceding the context of the message.

Example:

mix-index AX (.message)

Blanks (spaces) are allowed anywhere within the text of the message, but the first character must be a period (.).

There are eight local control messages. Any other messages entered using the period as the first character will result in the following message being displayed:

ERROR: INVALID OPERATOR IN . INSTRUCTION, RE-ENTER

Valid Local Messages

Following are the eight valid local messages.

.RE or .READ

This command directs the RJE/IO program to open a card file named RJE/CARDS and begin reading the remote deck that is to be transmitted to the central system. RJE/CARDS is closed after the entire remote deck has been read.

```
1 AX.READ
```

.CL or .CLOS

The .CL or .CLOS commands may be used to close all output files that were opened by RJE/IO. The output of the central system is a continuous stream of data, and if it is directed to a backup disk or tape, it is sometimes desirable to divide the data into a set of logical backup files. These commands are useful for that function.

```
1 AX.CLOS
```

.CLCP

The .CLCP command will close the card punch file.

```
1AX.CLCP
```

.CLLP

The .CLLP command will close the line printer.

```
1 AX.CLLP
```

.ST or .STOP

The .ST or .STOP command terminates the current RJE session. A message is sent from RJE/IO to RJE/DCH instructing it to cease all activity and to terminate itself. All queued messages are lost.

```
1AX.STOP
```

.WT or .WAIT

The .WT or .WAIT command instruct the RJE/DCH program to cease all line activity and wait to answer a call from the central system. The RJE/DCH issues a test command for a ringing phone. When control returns to RJE/DCH indicating the phone is ringing, RJE/DCH answers the phone and indicates so by the following message displayed on the console printer.

PHONE RINGING

A bell will ring at the remote system when the PHONE RINGING message is displayed. At that point, RJE/DCH attempts to re-establish the line connection with the central system.

```
1 AX.WAIT
```

.EST

The .EST command causes the RJE/DCH program to attempt to re-establish the line connection with the central system. The message ONLINE is displayed when the line connection is re-established.

The .EST command may be used in conjunction with the retry function of RJE. The error message RETRIES-UP is displayed by the RJE/DCH program when the current buffer being sent to the central system is not being received.

```
1 AX.EST
```

.LOG

The .LOG command outputs on the console printer, a summary of the line exceptions that have occurred during transmission. By analyzing the number displayed, the operator can obtain an indication of the quality of the line connection and the rate of error activity on the line. Each .LOG command entry resets the counters to zero. The format of the summary messages are below.

NAKS–SENT–BECAUSE–OF–PARITY–ERRORS = ⟨number⟩

NAKS–SENT–BECAUSE–OF–NO–BUFFERS = ⟨number⟩

TIMEOUTS–IN–READ–OPERATIONS = ⟨number⟩

TIMEOUTS–IN–WRITE–OPERATIONS = ⟨number⟩

OTHER–EXCEPTIONS–IN–READ–OPERATIONS = ⟨number⟩

OTHER–EXCEPTIONS–IN–WRITE–OPERATIONS = ⟨number⟩

Error Conditions

During a RJE session, certain error conditions may occur that may require operator intervention. The RJE/DCH program which outputs the error messages will try to recover from the error condition automatically. However, if it cannot do so, the system operator must discontinue RJE processing. The format of the error messages appears below. The result descriptor and operator code fields indicate the 24-bit hexadecimal representation of the particular result descriptor error and the operator in which the error occurred.

PEW1 @ result descriptor @/@ op-code @

The program was reinstated but the COMPLETE bit was not set in the result descriptor. This error message is displayed when the operation that resulted in the error was a WRITE-READ operation.

PER1 @ result descriptor @/@ op-code @

The program was reinstated but the COMPLETE bit was not set in the result descriptor. This error results from a .READ operation.

PER2 @ result descriptor @/@ op-code @

The program was reinstated but the COMPLETE bit was not set in the result descriptor. This error results from the READ of a WRITE-READ sequence. Both result descriptors and op-codes of the WRITE-READ will be displayed.

LOSS OF DATA SET READY

This error was caused during the last I/O sequence because the data set was down. The error is recoverable, but the data set should be checked.

LOSS OF CLEAR TO SEND

A loss-of-line occurred during the last attempt to transmit by the RJE/DCH program. The error is recoverable but the data set should be checked.

MEMORY PARITY ERROR

While attempting to send a message to the central system, a parity error occurred while fetching the message from the B 1700. This error is unrecoverable.

RETRIES-UP

This message indicates that the current buffer being sent to the central system is not being acknowledged. This may be the result of either the central system not acknowledging, thereby returning a NAK, or line problems including errors mentioned previously causing exceptions whenever the message is transmitted.

The RETRIES-UP message is displayed by the RJE/DCH program after fifty attempts of message transmission have failed. RJE/DCH continues to try to transmit the message until successful, or until RETRIES-UP is displayed again, or until an .EST message is received by RJE/IO. If it is the latter then the buffer is discarded and a new one obtained if possible.

SECTION 4
PROGRAM PRODUCTS

COMPILERS

Compilers generate executable code from a programmer's source statements. Each compiler has various options and operational techniques which affect its output. The following pages discuss each compiler and its individual operating procedures.

The COMPILE card, DATA card, and the Label equate (FILE) cards are standard for all compilers and are not discussed in detail for each compiler concerned. See the Control Instruction section for their particular usage and syntax.

REPORT PROGRAM GENERATOR

General

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

Compilation Card Deck

A program written in Burroughs RPG, called a source program, is accepted as input by the RPG compiler. The compiler has two major functions: (1) verify all syntax rules outlined in the RPG Program Manual, and (2) convert the source program language into COBOL S-Language which is then ready for execution.

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

Following is an example of an RPG compilation deck.

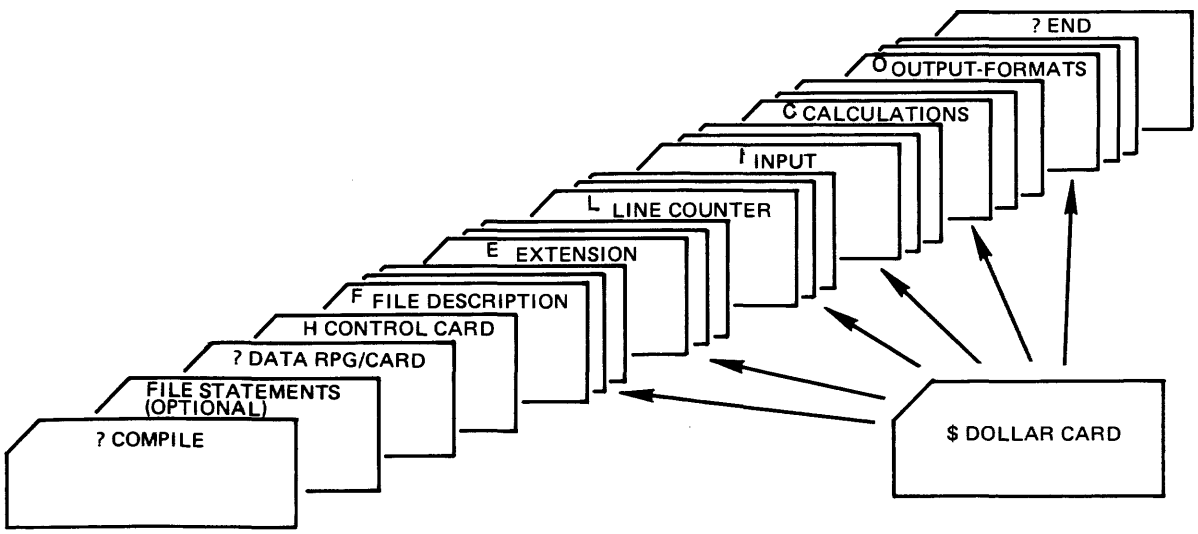


Figure 4-1. RPG Compilation Deck

Dollar Card Specifications

Dollar Card Specifications allow the RPG Compiler or Translator to accommodate various extensions to other manufacturers RPG and RPG II languages, which cannot be handled on the other specification forms. Dollar Cards also allow certain compiler-control options to be set or reset during compilation.

Dollar cards may appear anywhere within the source deck, as required. Only one option can be entered on a card and must be in the following format:

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

RPG Extensions

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

NOTE

None of the following operations may be "reset".

PACKID Specifies the pack name of a disk file. Similar to $ FAMILY and $ FILEID, default of blank dp-id name and the MCP will assume systems pack. This entry should be included to ensure correct handling of files by the MCP.

FAMILY Specifies the external family name (MFID) associated with the file. The VALUE field contains the name which is one to ten characters, left-justified.

FILEID Specifies the external file identification (FID) associated with the file. The VALUE field contains the name which is one to ten characters, left-justified.

AREAS Specifies the maximum number of areas to be allocated for the file (disk files only). The VALUE field contains an integral value, 1 to 40, right-justified, leading zeros optional. The default value assigned is 40, unless specified otherwise.

RPERA Specifies the maximum number of logical records that will be written in each disk area. The VALUE field contains an integral value, right-justified, leading zeros optional. The default value assigned is 500 unless specified otherwise.

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.

CLOSE Explicit close allows all input serial files to remain opened until End-of-Job. Default is the implicit close of files at End-of-File.

AAOPEN Is a file option used to set a bit in the MCP file parameter block and allocate all disk space areas at the beginning of the program.

ONEPAK Specifies that this particular file must be contained on one disk.

CYL Allocates file areas starting on an integral cylinder boundary.

DRIVE Allocates a physical drive to that particular file. VALUE field must be 0-15. Option may not be reset and is not related to PACKID.

REFORM Input and update disk files are assumed to have the 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.

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.

NONEPACK
File may be multipack.

Compiler-Directing Options

LIST Specifies that the compiler produce a single spaced output listing of the source statements with the error or warning messages. This option is set "on" by default. Resetting to "off" will not inhibit the errors or warning messages from printing.

LOGIC Specifies that the compiler produce a single-spaced listing of each source specification line followed immediately by an intermediate code used to generate COBOL-S code. The listing is produced after the NAMES listing (if the NAMES option is set), and does not include addresses or bit configurations, but only the opcodes and logical operands of the program.

MAP Specifies that the compiler produce a single-spaced listing detailing the program's memory utilization. The MAP listing is produced after the LOGIC listing (if the LOGIC option is set).

NAMES Specifies that the compiler is to produce a single-spaced listing of all assigned indicators file names, and field names. The attributes associated with each file and field are also listed. The NAMES listing is produced immediately after the normal source input listing.

RSIGN Indicates to the compiler, the location of the sign in numeric data items. When set, all signs are assumed to be right-justified; when reset, all signs are assumed to be left-justified. This option may be set and reset at different points in the Input and Output-Format Specifications, allowing different fields to have different sign positions. If the option is used, it will override the sign position specified in the Control Card Specifications.

SEG Orders the compiler to begin placing code in an overlayable segment identified by the integer in the VALUE field (right-justified, between 0 and 7 inclusive). Segmentation is an automatic function of the RPG compiler and optimized for its best uage. When the SEG option is used, automatic segmentation is not suppressed.

SUPR Specifies that the Compiler is to suppress all warning messages from the source program listing. (Error messages still print.)

XMAP Specifies that the compiler print a single-spaced listing of all the code generated, complete with actual bit configurations and addresses. Combined with the listing produced by the LOGIC option, complete information about the generated code of the program is available. The XMAP listing is produced after the MAP listing if the MAP option is set.

STACK Due to infrequent stack overflow conditions during program execution, the user may now change the stack size of the resultant program. This should only be used when a STACK overflow condition has occurred. The default stack size is 313 bits which will allow 8 entries in the stack. To increase the stack size add 39 bits, for each additional stack entry, to the default size of 313.

BAZBON This specifies that if an indicator is assigned to a field to test for ZERO or BLANK in the Input or Calculation Specifications and the same field is used in the Output Specifications with a BLANK AFTER designation, that indicator will be turned ON after the field is blanked during the output operations. Should a N (not) be specified in column 8 the indicator will be turned OFF, overriding the original RPG I or RPG II specifications.

ZBINIT This specifies that all ZERO BLANK indicators are initialized ON at Beginning-of-Job or if a N (not) is specified in column 8 they will be initialized OFF regardless of the specifications for RPG II or RPG I.

XREF The XREF option must be placed at the beginning of the RPG source program, prior to the first File Specification and prior to H card if present. This option allows the RPGXRF file to be created during compilation for use as input to the RPG/XREF program. At the completion of the compilation it is necessary to manually execute the RPG/XREF program in order to obtain the cross reference listing.

PARMAP Produces a single-spaced listing of the compiler-generated paragraph names, source statement numbers, and actual segment displacements of the emitted code. This listing may be used to relate to the LOGIC listing.

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.

XLATE Specifies termination of the compiler prior to generation of the code file.

XLIST Specifies a single-spaced listing of the COBOL source language will be produced during the execution of COFIRS.

Internal File Names

The RPG Compiler's internal file-identifiers and external file-identifiers for use in file statement are as follows:

Internal	External	Description
LINE	RPG/LIST	Source output listing to the line printer.
SOURCE	RPG/CARD	Input file from the card reader.
TABCRD	RPG/VECTOR	Input file for TABLES from the card reader.

RPG Internal File Names

RPG TO COBOL TRANSLATOR (COFIRS)

General

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

Execution of Translator

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

Example:

```
?   COMPILE program-name RPG LIBRARY

?   DATA RPG/CARD

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

?   END
```

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

The following is an example of the execute statement:

```
?   EXECUTE COFIRS
```

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

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

```
?   COMPILE program-name  COBOL LIBRARY

?   FILE SOURCE NAME RPGCOB;

?   DATA CARDS

     $ MERGE

?   END
```

COBOL COMPILER

General

The COBOL compiler is designed in accordance with the COBOL standard as specified by the American National Standards Institute (ANSI). The COBOL compiler can function with any system that runs under the control of the MCP.

The COBOL compiler in conjunction with the MCP allows for various types of actions during compilation which are explained in the following paragraphs.

Compilation Card Deck

Control of the COBOL source language input is derived from presenting the compilation card deck to the MCP. See figure 4-2.

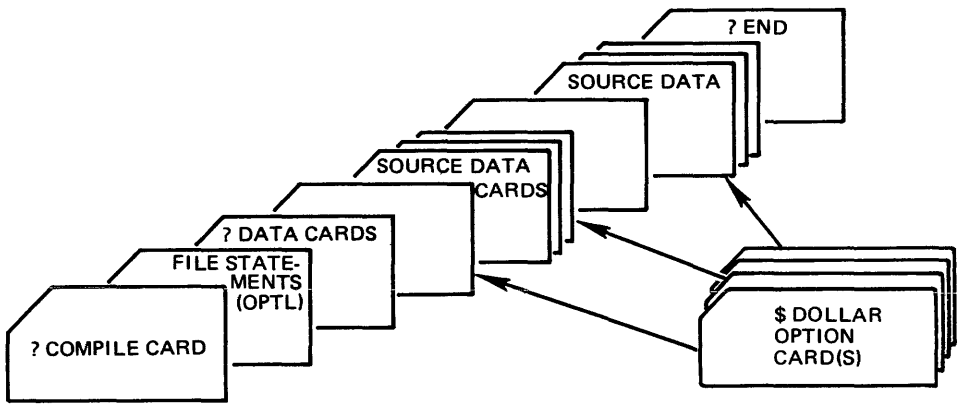


Figure 4-2. COBOL Compilation Deck

Dollar Option Card

The third card, excluding file statement cards, is the COBOL $ Option card. This card is used to notify the compiler which options are desired during a compilation. Without the $ Option Card, $ CARD LIST CHECK SINGLE will be assumed.

The $ Option card has the following characteristics:

a. A $ sign must appear in column 7.

b. There must be at least one space separating options on a card.

c. There may be more than one option per card.

d. The options may be in any order.

e. Any number of $ cards may be used and may appear anywhere in the source deck. The option will be set or reset from that point on.

f. Columns 1 - 6 are used for sequence numbers

The format of the $ Option card is as follows:

$ [NO] option-1 · · · [NO] option-n

OPTIONS

The options available for the COBOL compiler are listed below:

CARD — Input is from the source language cards or paper tape. This option is for documentation only.

LIST — Creates a single-spaced output listing of the source language input, with error and/or warning messages, where required.

LISTP — Lists the source images during the first compilation pass, and prints the error messages as they occur.

SINGLE — Causes the output listing to be printed in a single-spaced format.

DOUBLE — Causes the output listing to be printed in a double-spaced format.

CODE — List object code following each line of source code from the point of insertion.

MERGE — Primary input is from a source other than a card reader and may be merged with a patch deck in the card reader. It is assumed to be from a disk file, with a file-ID of COBOLW/SOURCE, by default.

If it is desired to change the input file-ID or change the input device from disk to tape, a LABEL EQUATION CARD must be used. The NEW option may be used with the MERGE option to create a new output source file plus changes.

NEW — Creates a NEW output source file with changes, if any, entered through the use of the MERGE option, but does not include compiler option cards which must be merged in from the card reader when compiling from disk or tape.

The output file will be created on disk by default with the file-ID of COBOLW/SOURCE.

If it is desired to change the output file-ID or change the output device from disk to tape, a LABEL EQUATION CARD must be used.

CHECK — This option will cause the compiler to check for sequence errors and print a warning message for each sequence error. The CHECK option is set on by default at the beginning of each compile, but may be terminated with the NO CHECK option.

SUPPRESS — Suppresses all warning messages except sequence error messages. The sequence error message can be suppressed with the NO CHECK option.

SPEC — If syntax ERRORS occur, this option negates the control and LIST option and causes only the syntax errors and associated source code to be printed. Otherwise the CONTROL and LIST options remain in effect.

"Non-numeric literal"
Is inserted in columns 73-80 of all following card images when creating a new source file and/or listing. This option can be turned off or changed by a subsequent control card with the area between the quote marks containing blank characters.

SEQ Starts resequencing, the output listing and the new source file if applicable, from the last sequence number read in and increments the sequence number by ten or by last increment presented in a previous $-option card. When resequencing starts at the beginning of the program source statements the sequence will start with 000010.

SEQ nnnnnn
Starts reweauencing the output listing and new source file if applicable from the sequence number specified by nnnnnn and increments the sequence numbers by ten.

SEQ +nnnnnn
Starts resequencing the output listing and new source file if applicable from the last sequence number read in and increments by the number specified by +nnnnnn. When resequencing starts at the beginning of the program source statements, the sequence will start with 000010.

SEQ nnnnnn +nnnnnn
Starts resequencing the output listing and new source file if applicable from the sequence number specified by nnnnnn and increments by the value of +nnnnnn.

NO SEQ Terminates the SEQ option and resumes using the sequence number in the source statement as it is read in.

CONTROL Prints the $-option control cards on the output listing. The LIST option must be on.

NO When the NO option precedes one of the above options, with the exception of MERGE which cannot be terminated, it will terminate the function of that option.

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.

ANSI When used, will inhibit the EXTENSION of AT END . . . ELSE, and during compilation will flag them as syntax errors.

STACK [integer]
Is used to increase the program stack by "integer" bits. The default size, when at least one PERFORM statement is used, is 1000 bits.

NOCOP When used will generate COP entries in the code instead of a COP table causing more memory to be utilized but faster program execution.

The NEW option does not have to be included when operating with a tape or disk source input, thus allowing temporary source language alterations without creating a new source output file.

The MERGE option without the NEW option allows a disk or tape input file to be referenced and to have external source images included from the card reader on the output listing and in the object program. A new output file will not be created.

Columns 1 - 6 of the Compiler Option Control card may be left blank when compiling from cards. A sequence number is required when compiling from tape or disk when the insertion of the $ option is requested within the source input.

Source Data Cards

The Source Data cards follow the $ Option control cards. These cards have two functions: (1) to update and create a newer version of a program, and (2) cause temporary changes to the tape or disk source program.

The following two paragraphs outline the Source Data Cards that are available to use with the COBOL Compiler:

a. VOID Patch Card. Punch the beginning sequence number in card columns 1-6 followed by a $ sign in column 7 with the word VOID starting in column 8, and terminate with the optional ending sequence number. This will delete the source statements beginning with the 6-digit sequence number through the ending 6-digit sequence number. For example:

    nnnnnn $VOID [nnnnnn]

    If the ending sequence number is omitted, only the source statement associated with the beginning sequence number will be deleted. For example:

    nnnnnn $VOID

b. CHANGE or Addition Patch Card. Punch the 6-digit sequence number in card columns 1-6 of the card that is to be changed or added, followed by the data to be input in their applicable columns. These cards must be arranged in the sequential order of the source program in order to be MERGED correctly into the program.

The COBOL Compiler has the capability of merging inputs from punched cards or paper tape, either of which may be merged with magnetic tape or disk.

The output listing will indicate any inserts and/or replacements when in the MERGE mode.

The following are examples of a COBOL compile deck.

Example 1:

```
?   COMPILE ALPHA WITH COBOL FOR SYNTAX
?   DATA CARDS
    $ CARD LIST DOUBLE
    . . . source program deck . . .
?   END
```

Example 2:

```
?   COMPILE ALPHA WITH COBOL SAVE
?   DATA CARDS
    $ CARD NO CHECK DOUBLE
    . . .source program deck. . .
?   END
```

Internal File Names

The COBOL compiler's internal file-identifiers and external file-identifiers for use in Label Equation are as follows:

Internal File-name	External File-ID	Description
CARDS	CARDS	Input file from the card reader. If $ MERGE is used, this file will be merged with the input file on disk or tape. The default input is from the card reader.
SOURCE	COBOLW/SOURCE	Input file from disk or tape when the MERGE option is used. The default input is from disk.
NEWSOURCE	COBOLW/SOURCE	Output file to disk or tape for a NEW source file when the NEW option is used. The default output is to disk.
LINE	LINE	Source output listing to the line printer.

COBOL Internal File Names

FORTRAN COMPILER

General

FORTRAN (FORmula TRANslation) was designed for writing programs concerned with scientific and engineering applications in mathematical-type statements. The FORTRAN compiler translates these statements into object code which can be executed by the B 1700.

B 1700 FORTRAN is designed to be compatible with FORTRAN IV, Level H, and to contain ANSI Standard FORTRAN as a subset.

Compilation Card Deck

Control of the FORTRAN source program is derived by presenting to the MCP the FORTRAN compilation card deck. See figure 4-3.

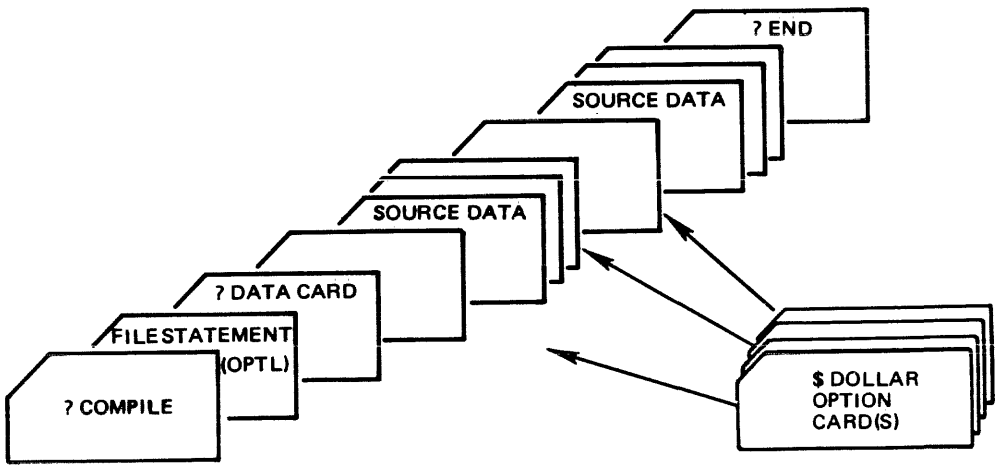


Figure 4-3. FORTRAN Compilation Deck

Dollar Option Card

The third card, excluding Label equation cards, and the standard COMPILE and DATA cards, is the FORTRAN compiler $ Option control card. This card is used to notify the compiler as to which options are required during the compilation. By omitting the $ Option card, the options "CARD LIST SINGLE" are assumed.

The format for the FORTRAN $ Option control card is:

$ [NO] option-1 · · · [NO] option-n

The FORTRAN $ Option control card has the following characteristics:

a. A $ sign may appear in column 1 or 2. When placed in column 2, the $ option card will be included in the new output source file if such a file is generated.

b. There must be at least one space between each item.

c. Options may be in any order.

d. Columns 73-80 are reserved for sequence numbering.

e. Any number of option cards may appear within the source deck.

OPTIONS

The options that are available for the FORTRAN compiler are as follows:

BIND	Causes the intermediate code files to be bound into an executable code file. This is a default option; if BINDING is not desired then NO BIND should be used.
CARD	Input is from source language cards. This is a default option.
CODE	Lists the object code for each source code line from the point of its insertion into the source deck.
DOUBLE	Causes output source listing to be double spaced.
DYNAMIC [integer]	This specifies the size in words to be assigned for an object program's dynamic memory. The compiler by default will assign the dynamic memory either of two ways: (1) if the data pages are less than 10, it assigns a size equal to the sum of the data pages, or (2) if the data pages exceed 10, then the size of the 10 largest data pages are used.
ERRORTRACE	Provides a FORTRAN level trace of subprogram and statement usage prior to the detection of a run-time error. The ERRORTRACE option must be placed before the first executable statement of the main program or any subprogram. Once set it may reset at any point by the NO ERRORTRACE option.
LIST	Creates a single spaced output listing of the source statements with error and/or warning messages. This is a default option.
MERGE	The MERGE option allows source input from disk or tape (disk by default, file-identifier SOURCE) to be merged with source statements from a card reader. The NEW option must be used with the MERGE option to create a new output source file. When the NEW option is not used, both the output listing and the object code file will reflect the merged statements but a new output source file will not be created.
NEW	Creates a new source output file having a file-identifier of NEWSOURCE. The new output file will include any changes made by the use of the MERGE option and any compiler option statements that have the dollar sign in column two.

NO — When used in conjunction with the following options, it will negate or put them in a reset condition. There must be a space between NO and the option.

BIND
CODE
DOUBLE
LIST
SEQERR
SAVEICM
ERRORTRACE
PROFILES
TRACEF

PAGE — Causes the output listing to eject at that point and start a new page.

PROFILES — Is an optimization aid that indicates to the user those areas of a program that can be optimized to improve program performance. At run time the following data will be output by using the PROFILES option:

a. Frequency of subprogram usage.

b. Time spent in each subprogram.

c. Use of individual statements within a subprogram.

d. Use of each statement during program execution.

The PROFILES option must be placed before the first executable statement of the main or subprogram. To reset the option use the $ NO PROFILES at any point within the program.

SAVEICM — Causes the intermediate code files for each syntax-error-free program part to be made a permanent disk file at the end of the compilation.

SEQ — Causes resequencing of the output listing and the new source file, if applicable, starting with the default number 00001000 and incrementing sequence numbers by 1000.

SEQ nnnnnnnn [nnnnnnnn]

Causes resequencing of the output listing and the new source file if applicable. SEQ is followed by either an eight digit number which is the starting sequence number, or two eight digit numbers with the first number being the starting sequence number and the second the resequencing increment value. The default resequence increment is 1000.

SEQERR — Causes a warning message to be printed for statements out of sequence.

SINGLE — Causes the output listing to be printed in single spaced format. This is a default option.

STACKSIZE [integer]

Specifies the size in words to be allocated for the object program Evaluation stack. Default size is 100; maximum size is 4096.

TRACEF — Causes a FORTRAN level trace to be printed for each FORTRAN statement executed in the program. This option may be inserted anywhere within the program. Once set, it remains set until reset by using NO TRACEF.

VOID	Causes the source input image corresponding to the sequence number of the VOID card to be deleted from the input disk file.
VOIDnnnnnnnn	Causes a series of source images to be deleted starting from the sequence number in the sequence number field (73-80) through and including the sequence number of the VOID option.

Internal File Names

The FORTRAN Compiler's internal file-identifiers and external file-identifiers for use in Label Equation are as follows:

Internal File-name (file-number)	External File-ID (Label)	Description
CARDS	CARDS	Input file from the card reader.
LINE	LINE	Source output listing to the line printer.
SOURCE	NEWSOURCE	When $ NEW is used the output file will go to disk or tape. The default output is to disk (80 character records, blocked 2).
SOURCE	SOURCE	Input file is from disk or tape when $ MERGE is used. The default input is from disk and assumed to be 80 character records, blocked 2.

FORTRAN Internal File Names

BASIC COMPILER

General

BASIC is a problem-oriented language designed for a wide range of applications and may be easily applied to business, commercial, engineering and scientific processing tasks. The BASIC language is designed for use by individuals who have little previous knowledge of computers, as well as individuals with considerable programming experience. A distinct advantage of BASIC is that its rules of form and grammar are quite easily learned.

B 1700 BASIC includes the capabilities of the original Dartmouth College BASIC plus extensions provided for compatibility with the General Electric MARK II® BASIC language.

The BASIC compiler, in conjunction with the Master Control Program, enables source programs to be compiled through the use of a card reader or a card device. Compilation of the BASIC source language input is achieved by presenting the compilation card deck to the MCP. Control cards included in the compilation deck are of two general types: (1) MCP control cards, and (2) compiler $ Option control cards. The structure of the BASIC compilation deck is discussed in the text that follows:

Compilation Card Deck

The entities comprising the structure of the BASIC compilation deck and the order of their occurrence are shown in figure 4-4 below.

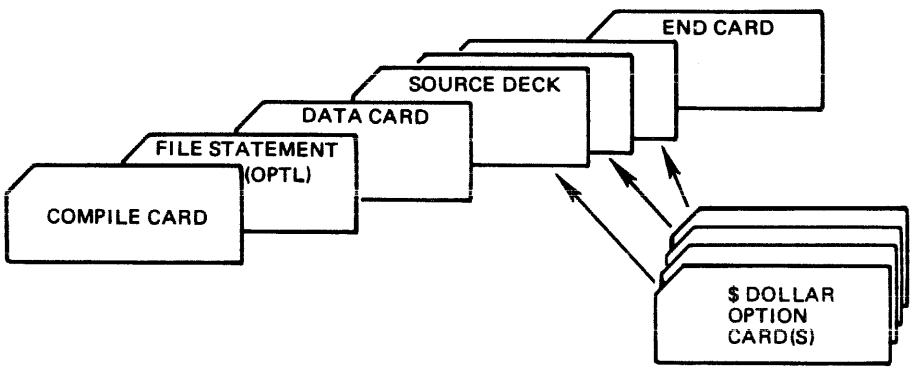


Figure 4-4. BASIC Compilation Deck

Dollar Option Card

The third card, excluding the optional Label Equation cards and the standard COMPILE and DATA cards, is the BASIC $ Option card. This card is used to notify the compiler which options are desired during a compilation. By omitting the $ Option card, the options "CARD LIST SINGLE" are assumed.

The $ Option cards for the BASIC compiler have the following characteristics:

a. All option cards are in a free-form format.

b. A line-number,which is required to be sequential within the program, cannot be greater than five digits and must precede the $ sign.

c. The $ sign may appear anytime after the line-number and before the first option.

d. All options listed on the card may appear in any order.

e. There must be at least one space between each option.

f. $ cards may be used anywhere within the source deck to either set or reset an option.

The format of the $ Option card is:

line-number $ [NO] option-1 · · · [NO] option-n

OPTIONS

The following options are available for the BASIC compiler.

CARD	Symbolic input is from source language cards. At the present time, this option is for documentation purposes only.
LIST	Creates a compilation output listing of the source language input, with error and/or warning messages, where required. LIST is a default option.
SINGLE	Causes the compilation output listing to be printed in a single-spaced format. SINGLE is a default option.
DOUBLE	Causes the compilation output listing to be printed in a double-spaced format.
CODE	Lists the object code generated for a source statement from the point of insertion into the source deck.
NO	Each of the above options may be preceded with NO. This enables the options to be set for selected program parts and then reset as desired. When an option is preceded by NO, there must be at least one space between the word NO and the option to be terminated.

Source Input Cards

The source program cards have the following characteristics:

a. Each card is taken as a different line and can contain only one statement. If the 96-column cards are used, the source statement must be contained in the first 80 columns.

b. There can be no continuation cards.

c. Each card between the ? DATA card and the ? END card must contain a line-number.

d. A line-number starts in column 1 and can be a length of 5 digits.

e. The first non-numeric character will terminate the line-number when less than 5 digits.

f. The line-number is used both as a statement label and sequence number.

g. Each statement is sequence checked by the BASIC compiler as it is read in.

h. Spaces or blanks have no significance within a source statement except for information contained in string constants. Spaces can be used to make a program more readable.

Intrinsic Files

The BASIC intrinsic files (identified by the name BAS.INTRIN/ nnnnnnnn) must be present on disk when a compiled BASIC program is executed; however, they are not needed when compiling the BASIC program. The intrinsic files contain input/output routines and intrinsic functions provided by the BASIC language. If the intrinsic files reside on a user pack the INTRINSIC.DIRECTORY control instruction must be used to identify the user pack, otherwise, the intrinsics are assumed to reside on the system pack.

Example:

```
? EXECUTE program-name
? INTRINSIC.DIRECTORY dp-identifier
? END
```

Sample Compilation Deck

In the following example, a BASIC program is to be compiled to LIBRARY and the object program, EXAMPLE/PROGRAM, is to be entered in the disk directory of a removable disk cartridge labeled BAS. In addition, the BASIC compiler resides on the removable disk, BAS. A $ card is enclosed to cause the compilation output listing to be printed in a double-spaced format. The options CARD and LIST being default options are not required, but are included on the $ card for documentation purposes only.

```
? COMPILE BAS/EXAMPLE/PROGRAM  BAS/BASIC/LIBRARY
? DATA CARDS
10 $ CARD LIST DOUBLE
20 INPUT X, Y, Z
30 PRINT "X="; X, "Y="; Y, "Z="; Z
40 END
? END
```

In the next example the compiled program EXAMPLE/PROGRAM is ready for execution. The compiled program as well as the BASIC intrinsic files and the BASIC interpreter reside on the removable disk pack labeled BAS. The card file labeled INPUT is required during execution of this program.

```
? EXECUTE BAS/EXAMPLE/PROGRAM

? INTRINSIC.DIRECTORY = BAS

? INTERPRETER = BAS/BASIC/INTERP2

? END

? DATA INPUT

    12, 32, 56

? END
```

Internal File Names

The BASIC Compiler's internal file-identifiers and external file-identifiers for use in Label Equation are as follows:

Internal File-name	External File-ID	Description
CARDS	CARDS	Input file from the card reader.
LINE	LINE	Source output listing to the line printer.

BASIC Internal File Names

UPL COMPILER

General

The User Programming Language (UPL) is a problem oriented language developed for writing B 1700 system software. The UPL Compiler is a single pass compilation that transforms the programmer's source statements into object code. Figure 4-5 illustrates the generation of an UPL object program.

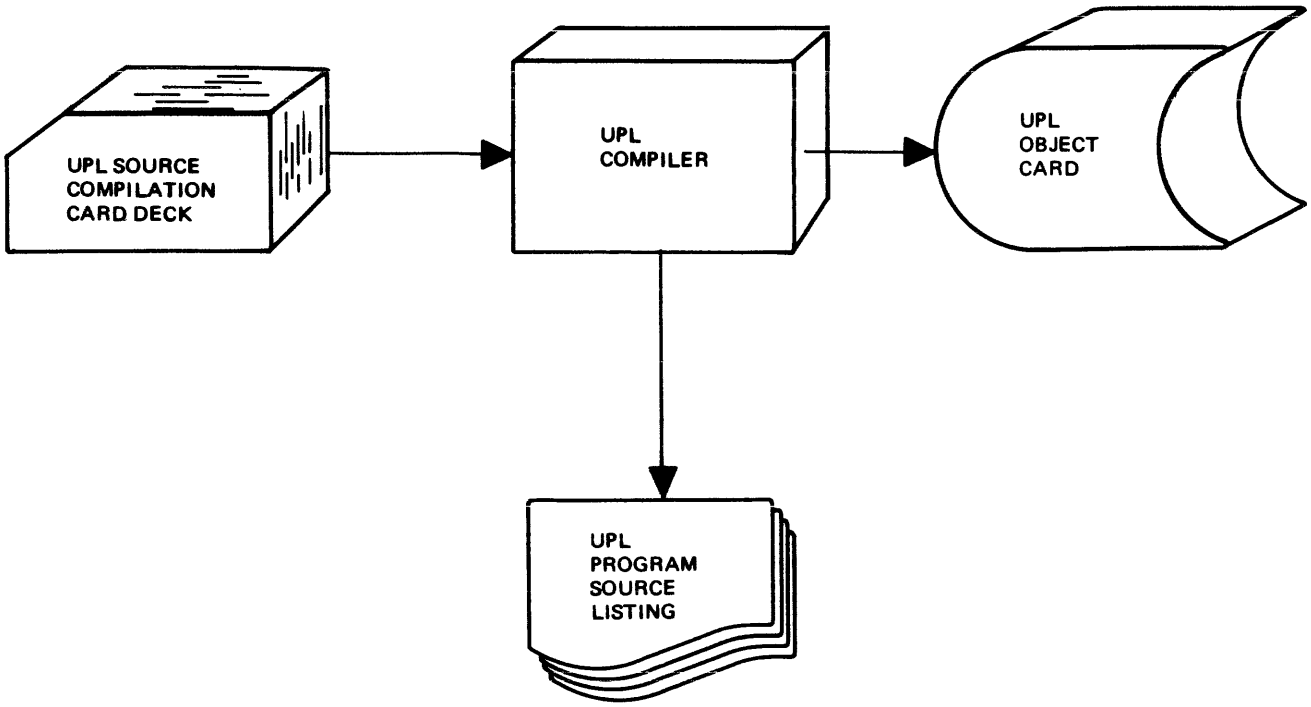


Figure 4-5. UPL Compilation Process

Compilation Card Deck

Figure 4-6 contains an example of a UPL Compilation deck.

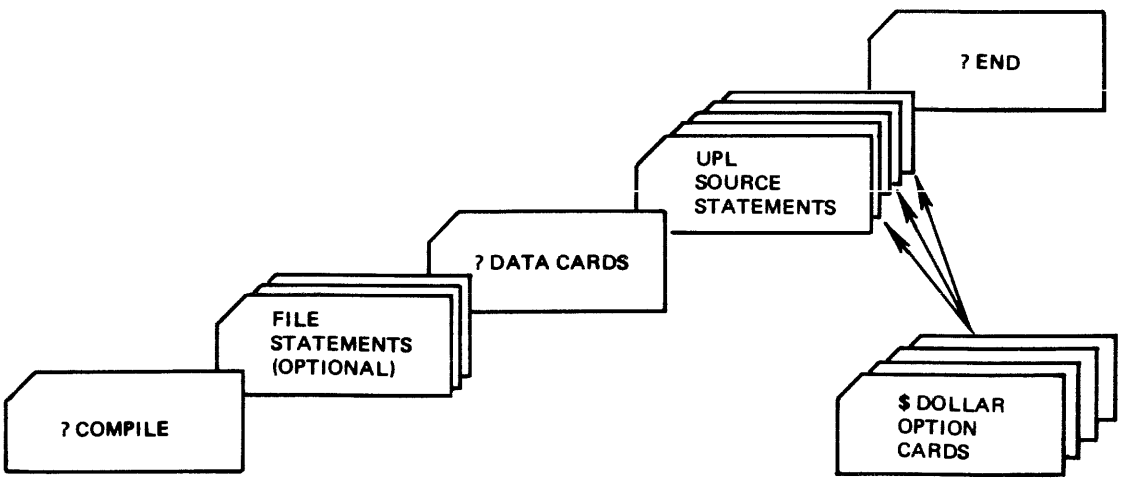


Figure 4-6. UPL Compilation Card Deck

Compiler Options

The UPL Compiler has certain options available through a Dollar card ($) that gives the operator the ability to override some of the standard compiler functions, alter stack sizes, suppress error and/or warning messages, and merge and create a new source file from an existing file. Dollar options are either in a set or reset condition. The UPL Compiler is preset with the following options: AMPERSAND, CHECK, LIST, and SINGLE.

The UPL Compiler option card has the following format:

```
$ [NO] option-1 · · · [NO] option-n
```

The UPL Compiler Dollar option card has the following characteristics:

a. Column one must contain a $ sign.

b. There must be at least one space between options on the same card.

c. Options may be set or reset in any order.

d. Columns 73-80 are reserved for sequential numbering.

e. There is no limit as to the number of options being set or reset during the compilation.

f. The option NO when appearing before any other option resets or negates that option.

The UPL Compiler options and their description are as follows:

AMPERSAND	Prints those ampersand cards that are examined.
CHECK	Checks the source input file for sequence errors.
CODE	Print the SDL object code generated for each source statement.
CONTROL	Prints all compiler option cards from that point. If the option control word CONTROL is required to be printed, the $ CONTROL (space) CONTROL format must be used.
CREATE.MASTER	This option must be the first card in the compilation deck and causes the compiler to perform the following functions: a. Dump information to the master information files. b. Create a new source file. c. Create a new code file.
CSSIZE integer	Assigns the number of entries in the Control stack represented by integer and overrides the compiler estimate.
DEBUG	Compiler debug only.
DETAIL	Causes the compiler to list the expansion of all define invocations.
DOUBLE	Double space listing.

DYNAMICSIZE integer	Assigns the value of the integer as the estimated memory size allocated for paged arrays. Integer is expressed in bits.
ESSIZE integer	Assigns the number of entries in the EVALUATION stack by integer and overrides the compiler estimate.
FORMAL.CHECK	The checking of the actual parameters passed to each procedure during execution against the TYPE and LENGTH specifications of their corresponding formal declarations. Also, the values returned from function procedures will be checked against the TYPE and LENGTH in the procedure head statement. Lack of correspondence is a run time error.
INTERPRETER file-identifier	Causes the program when executed to use the assigned Interpreter rather than the compiler default interpreter.
INTRINSIC file-identifier (family-name only)	Causes the program when executed to use those intrinsics with the assigned family-name rather than the compiler assigned family-name.
LIST	Prints the source input that was compiled. The NO option when invoked with LIST will reset the LISTALL option also.
LISTALL	Prints all source input regardless if conditionally excluded. The LISTALL option sets the LIST option, but NO LISTALL does not reset LIST.
MERGE	Indicates to the compiler that the source file is on tape or disk and there are cards to be merged during the current compilation.
NEW	Creates a new primary source file.
NO	The presence of the NO option immediately before any other option causes that option to be reset from that point on during the compilation.
NSSIZE integer	Assigns the number of entries expressed by integer to the Name stack thereby overriding the compiler's estimated size.
PAGE	Ejects page.
PPSIZE integer	Assigns the number of entries expressed by integer to the Program Pointer stack thereby overriding the compiler's estimated size.
SEQ beginning-sequence-number increment	Causes the output file to be resequenced beginning with the number used with SEQ.
SINGLE	Single space listing.
SIZE	Outputs at the end of the listing, the code segment names and their sizes.
SUPPRESS	Causes all warning messages to be suppressed. To suppress sequence error messages invoke the NO CHECK option.

VOID sequence number

Causes all records in the primary source file (as in the case of the MERGE) to be removed from the sequence number of the VOID card itself through the sequence number entered with the VOID option.

The VOID option has the following restrictions:

a. Must be the only compiler option on the card.

b. Cannot be preceded by the NO option.

c. Must contain a sequence number in columns 73-80.

VSSIZE integer

Assigns the number of entries expressed by integer to the size of the VALUE stack thereby overriding the compiler estimated size.

XMAP

Causes an extended SDL object code MAP file to be created showing the relative displacement of object code per source card sequence number, per Code Segment.

XREF
XREF.ONLY

The XREF options may be used in one of the two following modes:

a. A $XREF card at the beginning of the source deck will cause the compiler to build and XREF file, then ZIP SDL/XREF to sort and print the file at the end of the pre-pass. The compilation will continue.

b. A $XREF.ONLY card at the beginning of the source deck will cause the compilation to be terminated at the end of the pre-pass after the SDL/XREF program has been ZIPPED.

Internal File Names

The UPL Compilers internal and external file identifiers are as follows:

Internal	External	Description
CARDS	CARDS	Card source input file.
SOURCE	SOURCE	Primary source input file if MERGE option used.
NEWSOURCE	NEWSOURCE	Updated source output file if NEW option used.
LINE	LINE	Line printer file.

NDL COMPILER

General

The Network Definition Language (NDL) is a high level language for data communication and provides a means of generating a B 1700 Network Controller. The B 1700 NDL Compiler translates the input source code and outputs a NDL program listing, a Network Controller code file, and the Network Information File (NIF). Figure 4-7 below illustrates the NDL generation process.

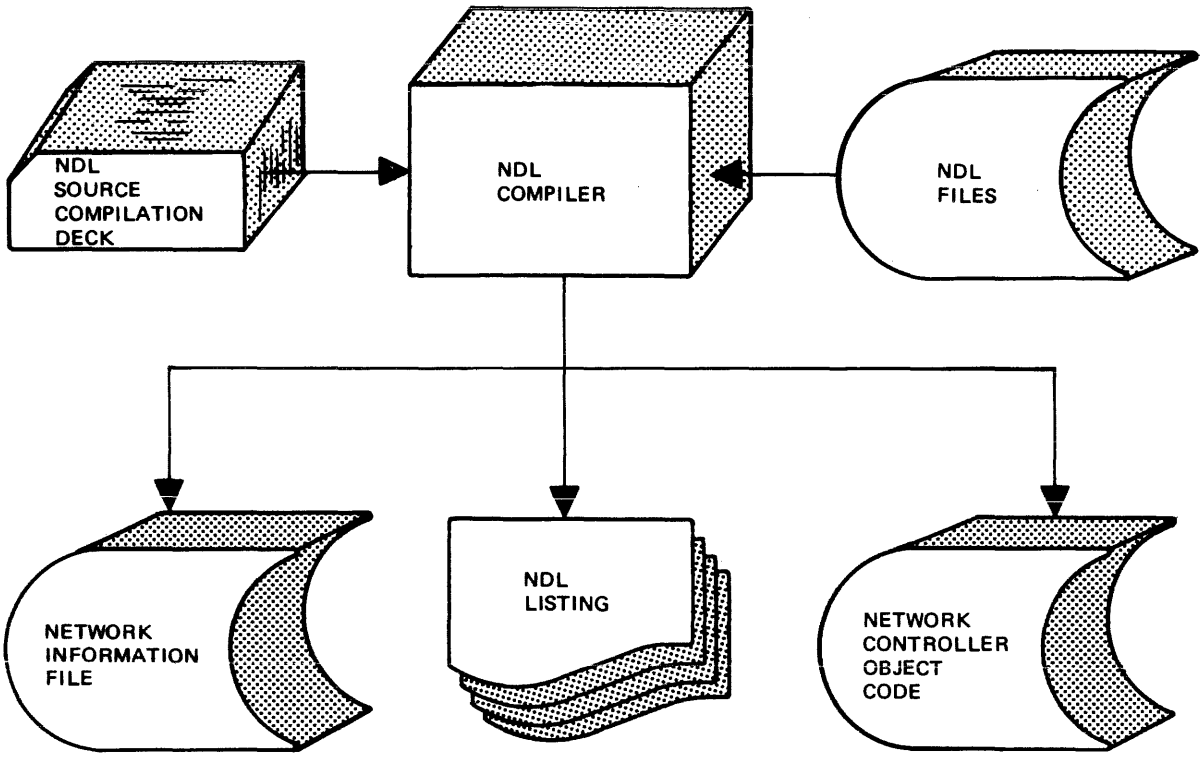


Figure 4-7. NDL Generation Process

Compilation Card Deck

Figure 4-8 contains an example of a NDL compilation card deck used to compile a NDL program.

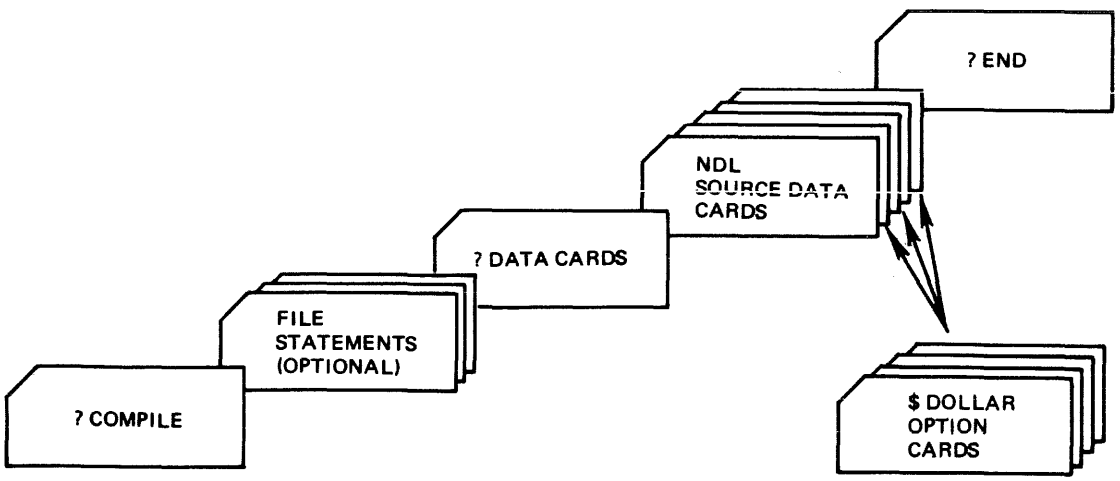


Figure 4-8. NDL Compilation Card Deck

Compiler Options

There are various options available that when invoked affect the compilation process. The options cover areas such as list format, error and warning message handling, maintenance, changing stack sizes, and merging source code.

An option is either in a set or reset condition. The NDL compiler is preset with the following options: LIST, CHECK, and DOUBLE. All other options must be invoked using the dollar option card at compile time. The NDL dollar option card has the following format:

```
$ [NO] option-1 . . . [NO] option-n
```

The dollar symbol ($) must be in column one with one or more spaces separating each option specified. With the one exception LIBRARY, there may be multiple options per card.

The available options and an explanation of their functions appear alphabetically as follows:

Option	Description
CHECK	This option causes the compiler to print warning messages for sequence errors in the source language input. A sequence error will occur when the sequence number of the last card is greater than or equal to the current sequence number.
CODE	The generated SDL code (S-operators) will be listed on the line-printer.
CONTROL	The dollar ($) option cards will be output on the object program listing.
CSSIZE integer	This option is used to alter the Control stack size to integer entries.
DOUBLE	Double space listing.
DYNAMICSIZE integer	Sets the Network Controller's dynamic memory size to integer bits.
ESSIZE integer	The Network Controller's Evaluation stack size may be set to integer entries.
FORGETERRORS	Directs the compiler to generate the object Network Controller despite syntax errors.
LIBRARY	The NDL source code specified by standard identifier is retrieved from the NDL source/standards and inserted in the user's program following the $ LIBRARY card. The LIBRARY option may not be included on a card containing other options. When the LIBRARY option is used to access standard REQUEST and CONTROL routines, the standard REQUESTS must precede the standard CONTROLS.
LIST	The source code will be listed.

LST	The source code will be listed.
MERGE	This option is used to merge the primary input with the secondary input.
NEW	A new source file will be created for use later as secondary input when this option is specified.
NIF	This option allows the creation of a new Network Controller in about half the time required for a total compilation. The old requests and line control code must remain unchanged.
NSSIZE integer	The Network Controller's Name stack size may be set to integer entries.
NO	Options may be reset by specifying $ NO followed by the name of the option to be reset. This allows options to be set and reset at the user's discretion. NO does not affect the VOID or LIBRARY options.
PPSSIZE integer	Sets the Network Controller Program Pointer stack size to integer entries.
SEQ	The source may be sequenced by supplying a beginning sequence number and an increment. The numbering will begin at SEQ BASE and will be incremented by SEQ INCRMT. A plus sign (+) is used to separate SEQ BASE and SEQ INCRMT which are both integers. $ SEQ SEQBASE + SEQ INCRMT If only $ SEQ is specified thereby omitting SEQ BASE and SEQ INCRMT, the numbering will start with 00000000 and increment by 100.
SGL	Single space listing.
SINGLE	Single space listing.
SUPPRESS	Prohibits the syntax warnings to be printed on the object program listing.
VSSIZE integer	The Network Controller Value stack size may be set to integer bits.
VOID	When VOID is used in conjunction with $ MERGE, it eliminates certain unwanted secondary source records from the new new source file being created. By specifying $ VOID, the secondary source record with the current sequence number is skipped by the compiler. $ VOID may also be followed by an eight character integer which instructs the compiler to skip all secondary source records beginning at the current sequence number and continuing until a secondary source record is read that has a sequence number higher than the eight character integer specified.

Internal File Names

The NDL's internal and external file identifiers are as follows:

Internal	External	Description
CARDS	CARDS	Input file from card reader.
LINE	LINE	Source output listing to line printer.
SOURCE	SOURCE	Input file from disk or tape when the MERGE option is invoked.
NEWSOURCE	NEWSOURCE	Output file to disk or tape when the new option is invoked. Default is to disk.
NIF	NDL/NIF	Network Information File
ADDRESS	NDL/ADDRESS	Network Controller Address File
MACRO	NDL/MACRO	Skeletal Network Controllers
LIBRARY	NDL/LIBRARY	Library

MIL COMPILER

General

The Micro Implementation Language (MIL) is a symbolic coding technique that makes available all the capabilities of the B 1700 processor. A MIL program contains a set of micro instructions that are directly executable upon the B 1700 hardware. MIL assumes interpretive or indirect processing of information contained in main memory.

Compilation Card Deck

Figure 4-9 contains an example of a MIL compilation deck.

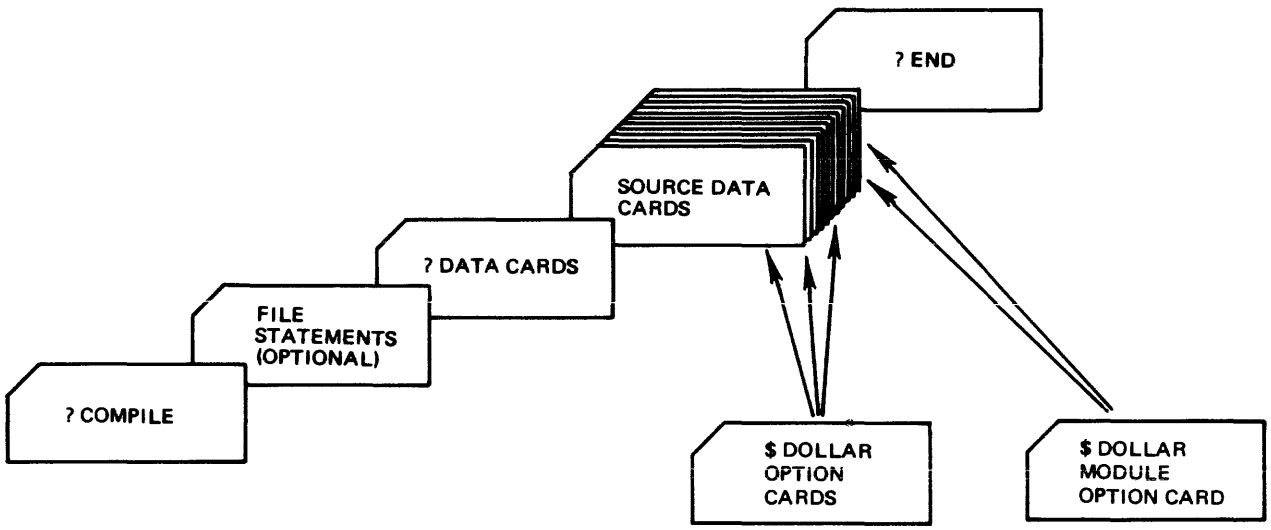


Figure 4-9. MIL Compilation Card Deck

Compiler Options

The $ Option Card is used to notify the MIL Compiler as to which options are required by the programmer during compilation.

The $ Option card for the MIL Compiler has the following format:

$ [NO] option-1 . . . [NO] option-n

The MIL $ Option Card has the following characteristics:

a. Column one must be a $ sign.

b. There must be at least one space between options.

c. Options may be in any order.

d. Columns 73-80 are reserved for sequence numbering.

e. Any number of $ Option Cards may appear anywhere within the source deck.

f. The optional word NO appearing before any option RESETS that option.

The MIL Compiler is preset with the following options: LIST, ALLCODE, SINGLE, AMPERSAND, and CHECK.

ALLCODE	Lists all codes generated by each MIL statement.
AMPERSAND	Prints all ampersand (&) cards.
CHECK	Checks for sequence errors.
DEBUG	Debugs compiler only.
DECK	Punches an object deck.
DOLLAR	Prints all dollar ($) cards.
DOUBLE	Double spaces listing.
EXPAND	Prints all statements within a macro invocation.
FORCE	Outputs all files regardless of syntax errors.
HEADINGS	Prints headings and titles on top of each page; does not affect line count.
LINES.PER.PAGE	Specifies the number of lines to be put on the page of a listing.
LIST	Lists all MIL source input that is compiled.
LISTALL	Lists all MIL source input regardless whether conditionally excluded.
MERGE	Merges the secondary source of input with the file SOURCE. When a duplicate sequence number exists the record from the card file will be used.
NEW	Creates a new source file.
NO	Resets option.
PAGE	Ejects page of listing at that point.
PAGE.NUMBERS	Numbers the pages of the listing and maintains a count.
PARAMETER.BLOCK	Used in conjunction with DECK and causes a parameter block to be punched with the deck. Used primarily with interpreters that are to be run with the MCP.
SEQ	Resequences and outputs NEWSOURCE file and listing.
SINGLE	Single spaces program listing.
SUBSET	Generates code for the B 1710 series processors.

SUPPRESS	Suppresses all warning messages except sequence error messages.
VOID	Voids those images from the secondary input file SOURCE which have sequence fields less than or equal to the terminating sequence field. If the terminating sequence field is missing, then the only image voided is the first one with the same sequence field as the VOID card.
XREF	Produces a listing of all user specified names and labels with each identifier associated with its sequence number for each declaration and invocation.
XREF.LABELS	Produces a cross reference of labels only.
XREF.NAMES	Produces a cross reference of user specified names only.

Module Option Dollar Card

The module option dollar card ($) is used to set or reset user defined toggles used in conjunction with IF statements in the conditional inclusion of source statements. It may be used anywhere within the source deck, and each module option dollar card affects only those user defined toggles which are referenced on that card. A user defined toggle can only be referenced by an IF statement when declared (set or reset) on a module option dollar card.

Example:

```
$ SET SYSTEM1, RESET SW2, SET SW4, SET SW5
```

Internal File Names

The MIL Compiler's internal and external file identifiers are as follows:

Internal	External	Description
CARDS	CARDS	Input file from the card reader.
LINE	LINE	Source output listing to the printer.
SOURCE	SOURCE	Input file from disk or tape when the MERGE option is invoked.
NEWSOURCE	NEWSOURCE	Output file to disk or tape when the NEW option is invoked. Default is to disk.

Object Code Deck Format

The DECK option causes the object code to be output to punched cards. The cards have the following format with all fields except the program identifier in hexadecimal format.

Card Columns	Description
1-6	24-bit control memory address.
8-9	8-bit count of the number of bits of data on this card.
11-70	Contains up to 240 bits of data, left justified.
72-80	Program identifier, used for documentation only.

Compiler Restrictions

a. The only source of input is the card reader, unless otherwise specified by the MERGE option. Once the MERGE option has been invoked, card only input is not possible.

b. When dollar cards ($) are not included in the compilation deck, the default options will prevail.

c. Options may be reset only by using the NO option. A space must separate NO and the option being reset.

d. Comments may appear on dollar cards only if preceded by either an asterisk (*) or a percent (%) sign.

e. Dollars cards are not included as part of the NEWSOURCE file when the option NEW is specified.

SDL COMPILER

General

The Software Development Language (SDL) was developed specifically for writing the system software for B 1700 systems. SDL is a high-level, procedure oriented language. All programs written in SDL source language must be processed by the SDL Compiler. The SDL Compiler transforms the source statements into S-Code to be interpreted by a set of micro-instructions called firmware.

Compilation Card Deck

Figure 4-10 contains an example of a SDL compilation card deck.

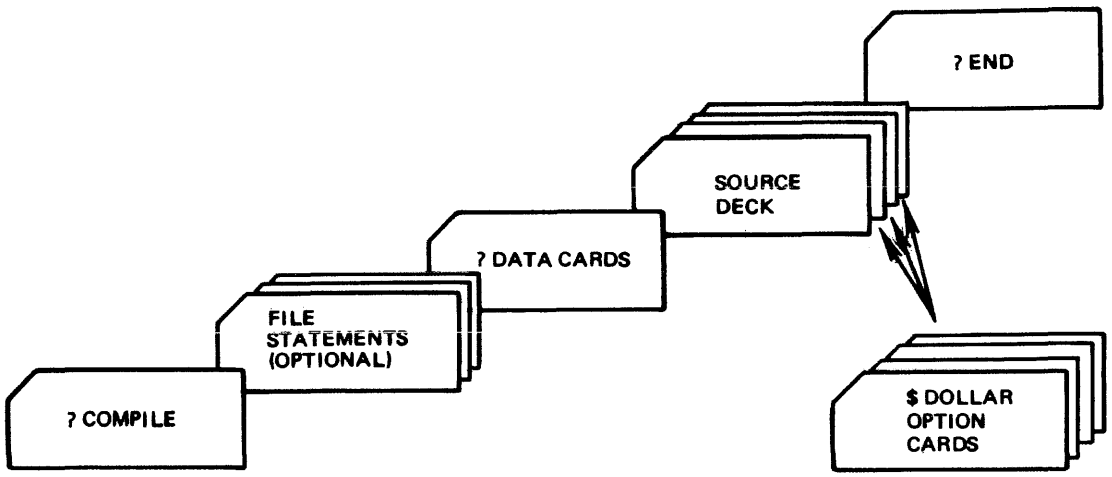


Figure 4-10. SDL Compilation Card Deck

Compiler Options

The SDL Compiler has certain options that are available to the operator or programmer that may be implemented at the time of compilation. These options are input by card along with the source deck and have the following format:

$ [NO] option-1 . . . [NO] option-n

The SDL Dollar ($) Options have the following characteristics:

a. Column one must contain a $ sign.

b. There must be at least one space between options.

c. Options may be in any order.

d. Columns 73-80 are reserved for sequence numbering.

e. Any number of options may appear anywhere within the source deck.

f. The option NO appearing before any other option resets or negates that option.

The following is a list of the SDL Compiler options and their definitions.

AMPERSAND	Prints those ampersand cards that are examined.
CHECK	Checks the source input file for sequence errors.
CODE	Prints the SDL object code generated for each source statement.
CONTROL	Prints all Compiler Dollar Option cards from that point. If the option word CONTROL is to be printed, $ CONTROL (space) CONTROL format must be used.
CREATE.MASTER	When used, this option must be the first card in the compilation deck and causes the compiler to perform the following functions: a. Dump information to the master information files. b. Create a new source file. c. Create a new code file.
CSSIZE integer	Assigns the number of entries in the Control stack represented by integer and overrides the compiler estimate.
DEBUG	Debugs compiler only.
DETAIL	Causes the compiler to list the expansion of all define invocations.
DOUBLE	Double spaces listing.
DYNAMICSIZE integer	Assigns the value of the integer as the estimated memory size allocated for paged arrays. Integer is expressed in bits.
ESSIZE integer	Assigns the number of entries in the Evaluation stack by integer and overrides the compiler estimate.
FORMAL.CHECK	Causes the checking of the actual parameters passed to each procedure during execution against the TYPE and LENGTH specifications of their corresponding formal declarations. Also, the values returned from function procedures will be checked against the type and length in the procedure head statement. Lack of correspondence is a run time error.
INTERPRETER file-identifier	Causes the program when executed to use the assigned interpreter rather than the compiler default interpreter.
INTRINSIC file-identifier (family-name only)	Causes the program when executed to use those Intrinsics with the assigned family-name rather than the compiler assigned family-name.

LIST	Prints the source input that was compiled. The NO option when invoked with LIST will reset the LISTALL option also.
LISTALL	Prints all source input regardless if conditionally excluded. The LISTALL option sets the LIST option on, but NO LISTALL does not reset LIST.
MERGE	Indicates to the compiler that the source file is on tape or disk and there are cards to be merged for the current compilation.
NEW	Creates a new primary source file.
NO	The presence of NO immediately before any other option negates that option.
NSSIZE integer	Assigns the number of entries expressed by integer to the Name stack thereby overriding the compiler estimated size.
PAGE	Ejects page.
PPSIZE integer	Assigns the number of entries expressed by integer to the Program Pointer stack thereby overriding the compiler estimated size.
SEQ beginning-sequence-number increment	Causes the output file to be resequenced beginning with the number used with SEQ.
SINGLE	Single spaces listing.
SIZE	Outputs at the end of the listing the code segment names and their sizes.
SUPPRESS	Causes all warning messages to be suppressed. To suppress sequence error message invoke the NO CHECK option.
VOID sequence number	Causes all records in the primary source file (as in the case of the MERGE) to be removed from the sequence number of the the VOID card itself through the sequence number entered with the VOID option. The VOID option has the following restrictions: a. Must be the only compiler option on the card. b. Cannot be preceded by the NO option. c. Must contain a sequence number in columns 73-80.
VSSIZE integer	Assigns the number of entries expressed by integer to the size of the Value stack thereby overriding the compiler estimated size.
XMAP	Causes an extended SDL object code MAP file to be created showing the relative displacement of object code per source card sequence number, per Code Segment.

Internal File Names

The SDL Compiler's internal and external file identifiers are as follows:

Internal	External	Description
CARDS	CARDS	Card source input file.
SOURCE	SOURCE	Primary input source file if MERGE option used.
NEWSOURCE	NEWSOURCE	Updated source output file if NEW option used.
LINE	LINE	Line printer file.

SDL Recompilation

The recompilation of an SDL program creates a Master Information File.

The create master must take place once and then may be followed by successive recompilations. Both the create master and the recompilation may be performed at the same time. In addition it is possible to perform successive regular compilations without invoking the recompilation facility.

CREATING MASTER INFORMATION FILES

In order to create Master Information Files, the first card of the compilation source file must be a $ CREATE.MASTER option card. This option causes the SDL Compiler to perform the following functions:

a. Save information needed for the recompilation into master files.

b. Create a new source file about which the information is to be used.

c. Use the Master Information Files and the new source file to create a new output code file.

The following files contain the information to be saved and used in the recompilation process.

NEWSOURCE

NEW.INFO.FILE

NEW.BLOCK.ADDRESS.FILE

NEW.SECONDARY.FILE

NEW.FPB.FILE

The following information is contained in the master information files: the input source images, Lexic Level one procedure boundaries for both the source file and object file, Lexic Level zero symbol tables, a record of all code addresses that have been emitted, the object code from which code addresses that have been emitted, the object code from which code addresses that have been emitted, the object code from which code addresses have been excised, the File Parameter Blocks, and SCRATCHPADS. (Refer to B 1700 Master Control Reference Manual, dated June, 1974, and B 1700 System Reference Manual, dated December, 1973.)

CREATE MASTER RESTRICTIONS

The create master operation has the following restrictions:

a. $ CREATE.MASTER must be the first card of the compilation source card file.

NOTE

This is to include any ampersand cards; they sould be sequenced.

b. $ NEW is not needed for this operation.

c. $ SEQ should be used if any input source images do not contain sequence numbers.

RECOMPILING

Recompilation is performed on a Lexic Level one procedural basis. That is, the outermost procedure containing a recompilation source card is the procedure which is recompiled. The code that is produced by the recompilation will be merged into, and in some cases replace some of the information created during the create master process.

The recompilation is invoked by including as the first card of the recompilation source deck a $ RECOMPILE.

The $ RECOMPILE causes the compiler to use the recompilation source deck (usually referred to as "patches") and the master information files to locate the Lexic Level one procedures and generate the same information for them as was generated for the entire program in the create master operation. This information is then combined, procedure by procedure, with the Master Information Files to produce the final form of the program that is turned into a new code file.

RECOMPILATION RESTRICTIONS

The recompilation process has the following restrictions:

a. The $RECOMPILE must be the first card of the recompilation source deck (patch deck).

b. The recompilation source deck may contain dollar cards, and ampersand (SET and RESET) cards, followed by the patch cards.

c. Lexic Level zero code cannot be patched. This includes all global data, Lexic Level one procedure headings, and the main program.

d. Neither $ SEQ or $ MERGE options may be invoked while using $ RECOMPILE process.

e. The source file that is input during the recompilation must be on disk in order that it may be accessed randomly.

CREATE MASTER AND RECOMPILE OPERATION PERFORMED TOGETHER

Both the create master and the recompilation process may be performed at the same time. Simply adhere to the rules for each separate operation and use $ RECOMPILE CREATE.MASTER as the first card of the source deck. It should be noted, however, that this procedure updates some of the information in the file MASTER.INFO.FILE. Therefore, the file must be saved because if any subsequent recompilations are desired, they must be performed against the saved master file.

GENERAL INFORMATION

1. The only information which may be listed during a recompilation is that which is being recompiled.

2. Both the source file used with $ CREATE.MASTER and the file created by $ CREATE.MASTER may be on tape, but the new source file must be placed on disk prior to any recompilations.

3. Because of the disk space required for recompilation, it is advantageous to keep source files on tape until needed.

4. The source image file created by the create master process contains no information other than the source images. Therefore it may be used in a regular compilation.

SDL COMPILATION DECK EXAMPLES

Compile and Create Master

```
? COMPILE SA SDL LIBRARY
? FILE SOURCE NAME SA0206/SOURCE TAPE;
? FILE NEWSOURCE NAME SA0410/SOURCE TAPE;
? FILE NEW.INFO.FILE NAME SA0410/INFO;
? FILE NEW.BLOCK.ADDRESS.FILE NAME SA0410/BLOCK.ADDRESS;
? FILE NEW.SECONDARY.FILE NAME SA0410/SECONDARY;
? FILE NEW.FPB.FILE NAME SA0410/FPB;
? DATA CARDS
$ CREATE.MASTER
$ MERGE LIST SINGLE SIZE SEQ
[PATCH CARDS]
[99999999 CARD]
? END
```

Recompile

```
? COMPILE SA SDL LIBRARY
? FILE SOURCE NAME SA0410/SOURCE;
? FILE MASTER.INFO.FILE NAME SA0410/INFO;
? FILE MASTER.BLOCK.ADDRESS.FILE NAME SA0410/BLOCK.ADDRESS;
? FILE MASTER.SECONDARY.FILE NAME SA0410/SECONDARY;
? FILE MASTER.FPB.FILE NAME SA0410/FPB;
? DATA CARDS
$ RECOMPILE
$ LIST SINGLE SIZE
$ VSSIZE 10000 NSSIZE 100
[PATCH CARDS]
[99999999 CARD]
? END
```

```
? COMPILE SA SDL LIBRARY
? FILE SOURCE NAME SA0410/SOURCE;
? FILE MASTER.INFO.FILE NAME SA0410/INFO;
? FILE MASTER.BLOCK.ADDRESS.FILE NAME SA0410/BLOCK.ADDRESS;
? FILE MASTER.SECONDARY.FILE NAME SA0410/SECONDARY;
? FILE MASTER.FPB.FILE NAME SA0410/FPB;
? FILE NEWSOURCE NAME SA0411/SOURCE TAPE;
? FILE NEW.INFO.FILE NAME SA 0411/INFO;
? FILE NEW.BLOCK.ADDRESS.FILE NAME SA0411/BLOCK.ADDRESS;
? FILE NEW.SECONDARY.FILE NAME SA0411/SECONDARY;
? FILE NEW.FPB.FILE NAME SA0411/FPB;
? DATA CARDS
$ RECOMPILE CREATE.MASTER
$ VSSIZE 10000 NSSIZE 100
$ LIST SINGLE SIZE
[PATCH CARDS]
[99999999 CARD]
? END
```

cut along dotted line

BURROUGHS CORPORATION
DATA PROCESSING PUBLICATIONS
REMARKS FORM

TITLE: B 1700 SYSTEMS
SYSTEM SOFTWARE
OPERATIONAL GUIDE

FORM: 1068731
DATE: 3-75

CHECK TYPE OF SUGGESTION:

☐ ADDITION ☐ DELETION ☐ REVISION ☐ ERROR

GENERAL COMMENTS AND/OR SUGGESTIONS FOR IMPROVEMENT OF PUBLICATION:

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

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

1068731 3-75 Printed in U.S. America