RM/COS™

USER GUIDE

RYAN-McFARLAND CORPORATION

609 Deep Valley Drive
Rolling Hills Estates
California, 90274

MAN903 PUBLICATION HISTORY

Original Issue (Release 2.2) ..... November 1, 1982
Revision A (Release 2.3) ..... May 9, 1983
Revision B (Release 2.4) ..... January 23, 1984

PROPRIETARY RIGHTS NOTICE

RM/COS is a proprietary product of

Ryan-McFarland Corporation
609 Deep Valley Drive
Rolling Hills Estates
California 90274

TABLE OF CONTENTS

Chapter 3
JOB DESCRIPTION LANGUAGE

Chapter 4
SCREEN EDITOR

Chapter 5
LINE EDITOR

Chapter 6
DATA COMMUNICATIONS

Chapter 7
SYSTEM STATUS CODES

Appendix A
NUMERIC LISTING OF RM/COS SYSTEM STATUS CODES

Appendix B
COBOL SUBROUTINE LIBRARY PACKAGE

Appendix C
MEMORY REQUIREMENTS

Appendix D
FILE SIZE CALCULATIONS

Appendix E
VENDOR SUPPLIED BATCH STREAMS

Appendix F
BATCH JDL SYNTAX SUMMARY

Appendix G
GLOSSARY

Appendix H
COBOL RUNTIME DEBUG

Appendix I
STATION CHARACTERISTICS

Appendix J
CHARACTER SET CONVERSIONS

PREFACE

RM/COS is an operating system designed for the business environment. RM/COS provides both multiuser and multitasking capability. The operating system and runtime support are geared specifically for RM/COBOL application programs. Application program development is supported with the inclusion of two text editors and a COBOL compiler in the system. In addition, RM/COS software provides the typical commercial program support and utilities available on most minicomputer systems, such as record level locking, file data integrity across power failures, automatic data file compression, sort/merge, and 3780 communications.

This manual describes the following:

- RM/COS system concepts with which an applications designer should be familiar,
- Job Description Language (JDL) commands with which the operator communicates with RM/COS,
- the text editors,
- the data communications systems,
- RM/COS system configuration, and
- detailed descriptions of RM/COS error messages.

ADDITIONAL MANUALS

The following manuals provide additional information needed by an RM/COS user:

Operator Guide - describes details of how an operator installs and interacts with RM/COS on a specific computer model. There are several versions of the Operator Guide, one for each computer model supported.

Terminal Guide - provides specific details on various terminals supported by RM/COS.

COBOL Language Manual - describes the syntax and semantics of the COBOL language supported by RM/COS.

CHAPTER 1

RM/COS SYSTEM CONCEPTS

INTRODUCTION

RM/COS incorporates a ANSI COBOL compatible file subsystem in a reliable, efficient, multi-user operating system. In addition, data integrity and data security required and expected by business applications have been designed into RM/COS. Special features are available to allow the design of turnkey end user systems.

This chapter describes the concepts of the various subsystems of the RM/COS operating system. The glossary which appears in Appendix G is also a useful introduction to the concepts of the RM/COS system.

FILE SUBSYSTEM

The RM/COS File Subsystem is that part of the operating system which is responsible for managing mass storage (disk) files and non-mass storage devices and presenting a device independent file interface to the application program.

Device Names

Each device connected to the computer is given a unique four character device name. The first two characters are letters which identify the type of device and the last two characters are numerals which specify to which device the name applies. Device names are bound to specific hardware devices by the system configuration process described in Chapter 2. The following are the devices in a typical RM/COS system configuration, and the device names assigned to them:

Disk Drive 1	DS01
Disk Drive 2	DS02
Terminal (station)	ST01
Line Printer	LP01

Some additional devices in possible RM/COS configurations might have the following devices names:

Disk Drive 3	DS03
Disk Drive 4	DS04
Magnetic Tape Unit 1	MT01
2nd Terminal	ST02
3rd Terminal	ST03
2nd Line Printer	LP02

The device name DUMY refers to a "dummy" device: a sink for data. Data written to DUMY goes nowhere; a read from DUMY returns an end of file status. DUMY is not included in the system configuration process like "real" devices.

In addition, the names ME and MYLP may be used in place of device names. JDL commands which expect a device name as the response to a prompt also accept the value ME to indicate the device name of the terminal (e.g., ST01) for the partition in which the process is running. Similarly, the name MYLP refers to a printer connected to the terminal indicated by the name ME.

Disk Volumes, Disk Files and Pathnames

RM/COS disk volumes may be divided among several separate files. Files are grouped into sets called directories,

which are themselves files. Directories are hierarchically organized into a tree structure having as its root the volume directory. Figure 2-1 illustrates an example of an RM/COS directory structure. Each file on a disk has a unique designation, called a pathname, which is made up of the volume name followed by zero or more directory names and then followed by the file name with all names separated by periods.

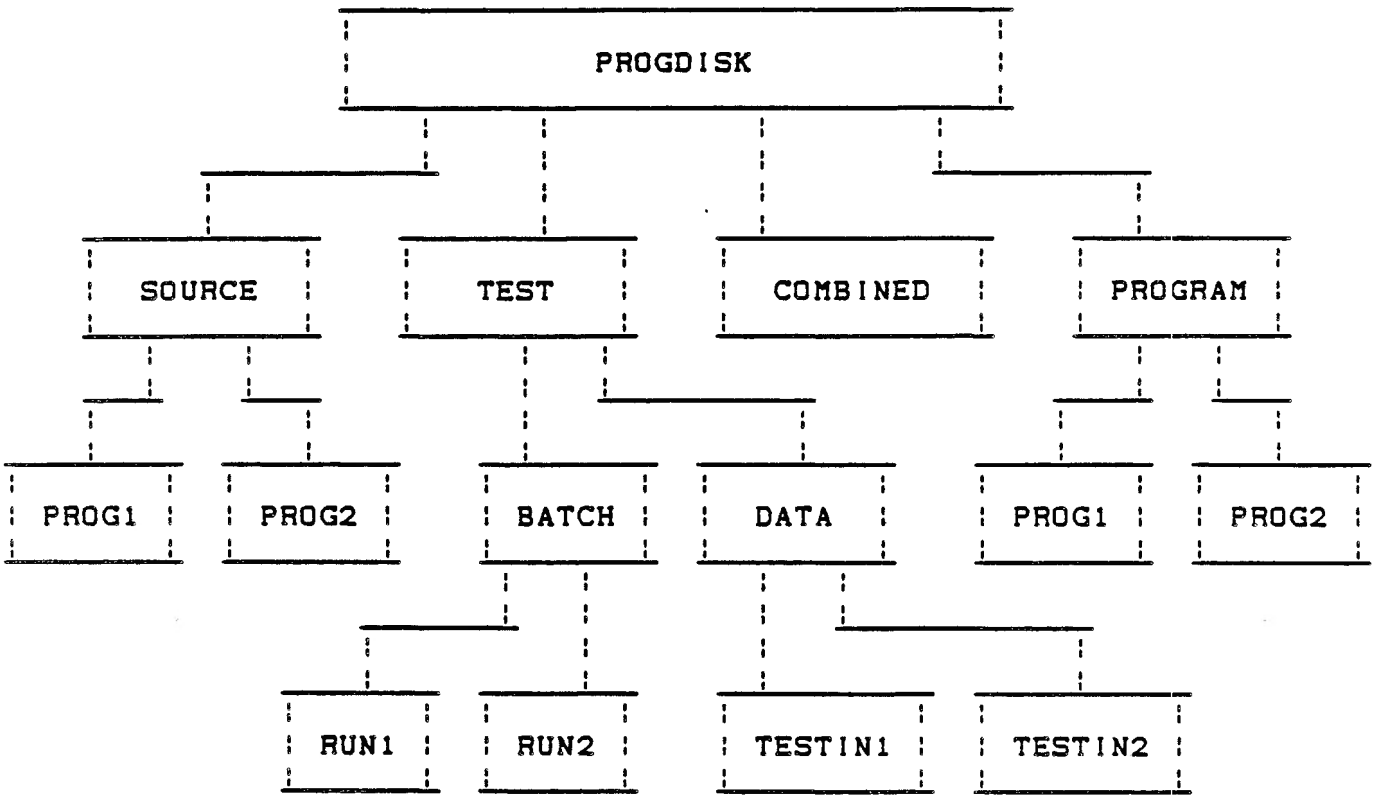


Figure 2-1
RM/COS Directory Structure Example

For example, in Figure 2-1 the following are the pathnames for all directory files:

```
PROGDISK
PROGDISK.SOURCE
PROGDISK.PROGRAM
PROGDISK.TEST
PROGDISK.TEST.BATCH
PROGDISK.TEST.DATA
```

The following are the pathnames for all files which are not directories:

```
PROGDISK.SOURCE.PROG1
PROGDISK.SOURCE.PROG2
PROGDISK.PROGRAM.PROG1
PROGDISK.PROGRAM.PROG2
PROGDISK.TEST.BATCH.RUN1
PROGDISK.TEST.BATCH.RUN2
PROGDISK.TEST.DATA.TESTIN1
PROGDISK.TEST.DATA.TESTIN2
PROGDISK.COMBINED
```

Note that within a directory, all file names must be unique; however, as seen in the above example, two directories may contain the same file name without ambiguity. The volume name is specified for a disk cartridge using the INITIALIZE command. File names for directory files and nondirectory files are specified by the DIRECTORY and CREATE commands, respectively.

A disk drive device name may be substituted for the volume name in a pathname, indicating to RM/COS that the volume directory of the disk volume loaded in that drive is to be used. For example, if the disk volume PROGDISK is loaded in DS02, then the pathnames PROGDISK.COMBINE and DS02.COMBINE are equivalent. Similarly, the volume directory name may be omitted from the pathname (i.e., the pathname begins with a period), indicating to RM/COS that the volume directory of the system disk volume is to be used.

NOTE

> Although the disk drive device name may be substituted for the volume directory name, such practice is not recommended because it defeats a verification that the correct disk volume is actually being referenced.

The tree structure of directories encourages the designer to group files with related content into a common directory; figure 2-1 illustrates such grouping, with source files grouped in the directory PROGDISK.SOURCE, program files grouped in the directory PROGDISK.PROGRAM, etc. Such grouping may provide some operational convenience on RM/COS (e.g. copying entire directory structures using FCOPY), depending upon the application.

Logical Names

COBOL applications programs and some Job Description Language commands require logical names to refer to devices or files. Logical names are bound to actual devices or

files by using the ASSIGN, CREATE, CONNECT, or TAPE-ASSIGN commands; the binding is severed by using the RELEASE command.

File Characteristics

Each file on an RM/COS disk volume has certain conceptual characteristics which are used to control the means by which individual logical records are processed and to control the accessibility of files to various applications and users. These characteristics are:

- logical record length,
- block size,
- number of blocking buffers,
- initial disk allocation,
- secondary disk allocation,
- organization,
- type,
- privilege level,
- delete protection, and
- write protection.

The logical record length specifies the maximum number of characters that may be contained in a single logical record of the file. This length does not necessarily correspond to any physical record size.

The block size specifies the number of characters which are to be contained in a logical block. A logical block is the unit of data transferred to or from a disk device in one operation. A logical block may contain one or more logical records and, except for indexed files, may contain only a portion of a logical record. Logical blocks begin on physical sector boundaries. Thus, block size is typically a multiple of sector size to minimize disk storage waste.

The number of blocking buffers specifies the amount of a user partition which is to be used, while a file is open, as buffer storage for logical blocks. Within limits, file IO will use multiple blocking buffers to enhance IO time at the expense of the additional memory used.

Initial disk allocation and secondary disk allocation, described in detail below, specify the amount of physical storage area needed to contain the data for the file.

RM/COS supports the three types of file organization used in COBOL programs: sequential, relative and indexed. In addition, RM/COS supports three types of file organization for system data: directory, program, and multipartite direct secondary (MDS). The multivolume sequential disk (MVD or MVSD) file organization is intended for use with the

FILE-BACKUP and FILE-RESTORE commands, but COBOL programs may access such files in the same manner as sequential files.

A file can be of one or more of six types. The spool type, which applies to sequential organization files only, should be specified for files that are assigned to PRINT in the File Control entry of a COBOL program.

The work type is specified for files to be written in the deferred write mode. In deferred write mode the records are not actually written to the disk until the memory space occupied by the records in memory is needed or until the file is closed. When work type is not specified, the file is written in the immediate write mode. Each record is written to the disk when the WRITE statement in the program is executed, and in the event of a power failure, the RECLOSE command can be used to recover all records in the file.

The auto type is specified for files which are open and updated over long periods of time and require a high degree of data integrity while minimizing the necessity of data recovery in the event of a power failure or other fault. This type causes the system to simulate an open and close operation around any file modification operation, minimizing the time the file is in a state requiring data recovery. The auto and work types are mutually exclusive.

The scratch type is specified for a temporary file. Scratch type files are automatically deleted when they are released from a logical name, and they are always treated as work files.

The compressed type file occupies less disk space because consecutive repeated characters are encoded as one or two characters before it is written, and restored to the record after it is read. The saving in disk space is greater for records that contain groups of two or more consecutive blanks or binary zeroes (NULs) and three or more consecutive characters other than blank or or binary zero (NUL). Files that consist of records with no such groups of repetitive characters should not be compressed. Relative organization files may not be of type compressed.

NOTE

The REWRITE statement cannot be used on a sequential organization file of compressed type.

The single type is used for files which must be allocated in a single contiguous group of Allocatable Disk Units (see Disk Allocation below). Normally only MDS organization files must have this type.

RM/COS files are assigned privilege levels when they are created. A privilege level is an integer value in the range of 0 through 65535. However, the CREATE command does not allow a user to specify a privilege level higher than his own privilege level (assigned to the user ID along with a passcode). A user cannot access a file that has a higher privilege level than his own. A CHANGE command can modify a file privilege level to any value no higher than the user privilege level.

Files may be protected from being deleted or from being modified. A write-protected file may be neither deleted nor modified; a delete-protected file may not be deleted. Directories may be delete protected. Delete protection may be applied when a file or directory is created. A CHANGE command can apply or remove either delete or write protection.

Several of these characteristics are analogous to concepts and clauses of the COBOL language. The table below summarizes the COBOL clauses of features which apply:

Characteristic	COBOL
logical record length	Implicit from record descriptions
block size	BLOCK CONTAINS clause
number of block buffers	RESERVE AREAS clause
organization	ORGANIZATION clause
spool type	WRITE with ADVANCING clause

Disk Allocation

As stated previously, RM/COS disk volumes may be divided among several separate files. This requires that the available storage on a disk volume be managed by RM/COS and allocated to files as needed.

RM/COS divides a disk volume into units called Allocatable Disk Units (ADU). Each ADU is an integral multiple of physical disk sectors. The number of disk sectors in an ADU is dependent on the capacity of the disk volume. ADUs are numbered from zero. ADU number zero starts at cylinder zero, head zero, sector five (the sixth sector on the disk) on most disks. On disks which reserve the first disk track for booting, ADU number zero starts at the second sector of the second track. There may be no more than 8192 ADUs per disk volume. The ADU size in sectors is calculated as follows:

```
sectors-per-ADU =
        max (ceiling (750/bytes-per-sector),
                     ceiling (total-sectors/8192))
```

(Formula 2-1)

When a file is created, two disk allocation parameters are provided describing the initial amount and, if the file size may exceed the initial size, the secondary, or incremental, amount of disk storage required for the file. These parameters, which are stated as the number of records, are converted to the number of ADUs required. This conversion is made using several worst-case assumptions; a total description of the conversion computations may be found in Appendix D. RM/COS then allocates the number of ADUs required for the initial amount of disk storage by first attempting to find a group of contiguous ADUs large enough to satisfy the requirement and then, if no contiguous space is large enough, by finding the smallest number of discontiguous groups which will satisfy the requirement. At this point the creation of the file is complete.

As data are added to a file, the amount of storage in the current allocation may be exhausted. If the secondary allocation creation parameter was nonzero, then RM/COS will attempt to allocate additional disk storage sufficient to hold the added data in increments of the secondary amount. First, if the ADU with a number one higher than the last ADU allocated to the file is unallocated, then that ADU and as many contiguous ADUs as required to satisfy the secondary allocation requirement will be allocated to the file. If this fails to allocate enough disk storage, then a search will be made for ADUs identical to the search described for the initial amount.

A file may have no more than 47 discontiguous groups of ADUs allocated to it, nor may a file be larger than the disk storage available on a disk volume. The first restriction may have meaning to an applications designer if several files are allocated on the same disk volume and allowed to grow in parallel with small secondary allocations; if this situation occurs, the designer should attempt to correctly size the initial allocation parameter when creating the files. The second restriction is lifted for an MVD organization file, which acts as a sequential file which spans several removable disk volumes.

A file may have no more than 65,535 blocks. Thus the total amount of data contained in a file is constrained by the product of the block size times the maximum number of blocks (i.e., the number of blocks which can be contained on a disk volume or 65,535, whichever is less).

USER MEMORY PARTITIONS

The available user memory under RM/COS is divided into segments called partitions. Every process must execute in a partition, and each partition can support only one process at a time. All logical name and synonym assignments are local to the partition in which they are made. The size of each partition, the location of each partition in physical memory, and the priority of each partition is defined by System Configuration (see Chapter 2), but may be changed with the PARTITION command. The memory comprising a partition is dedicated to that partition even when the partition is idle. A design goal of RM/COS is that all memory allocation after IPL be within the partition of the process needing the memory, preventing one user's excessive demands for memory from interfering with the correct operation of other users' applications.

Partition Types

Common to all RM/COS implementations are three types of partitions: terminal, nonterminal, and shared file. During System Configuration, there is associated with each terminal device a terminal partition. A terminal partition becomes active when a user logs-in at a terminal device, and becomes idle when the user QUITs. All but one of the remaining partitions are nonterminal partitions, and may be activated by a BATCH command. The shared file partition is not available to any process, but instead is available for the memory structures which represent disk files that are not assigned exclusively to a single user, i.e., that are assigned Read Only, Shared, or Exclusive Write.

An additional partition type is an unused partition. An unused partition is one assigned no memory during System Configuration and cannot later be assigned any memory with a PARTITION command.

Partition Numbering

Partitions are identified by numbers. The number of partitions is defined by System Configuration.

Each terminal device defined by System Configuration has a partition whose number matches the last two digits of the terminal's device name. Terminal ST01 is assigned partition 1; terminal ST10 is assigned partition 10. The number of the highest numbered terminal determines how many partitions are terminal partitions; if the highest numbered terminal is ST02, the first two partitions are terminal partitions.

All remaining partitions are nonterminal partitions, except the last which is the shared file partition. Nonterminal partitions are numbered consecutively, starting at 101. The shared file partition is always numbered 999.

All JDL commands which expect a partition number as input accept a partition number of 0 to indicate the partition in which the process is executing. All commands except MESSAGE and STATUS treat a null input as identical to a value of zero.

Partition Priority

A partition's priority is an integer from 1 to 65535, inclusive, which determines what fraction of the available CPU time a process executing in the partition receives. A priority of 1 maximizes the CPU time available to a process; a higher number reduces the CPU time available to a process. If two processes are contending for the CPU, one has a priority of 1 and the second has a priority of 2, the process with a priority of 1 will receive twice as much time as the other process. In a more complicated example, if 4 processes are contending for the CPU with priorities of 1, 1, 2, and 3, the processes will receive 6/17ths, 6/17ths, 3/17ths, and 2/17ths, respectively of the available CPU time. If only one process can use the CPU because all other processes are awaiting I/O requests, it will receive all of the CPU time regardless of its priority.

Memory Allocation within a Partition

As JDL commands and applications programs are executed by a user, memory within that user's partition is allocated and deallocated on an as needed basis. RM/COS strives to prevent fragmentation of the user's memory by grouping long-term allocations, such as file structures and program directories, at one end of the partition memory while allocating short-term allocations, such as JDL command processor memory and blocking buffers, at the other end of the partition memory. In general, this is a successful mechanism with no other considerations. If, however, the applications designer is trying to use very large programs or otherwise is trying to allocate virtually all of a partition's memory, certain steps should be taken. Primarily, the designer should be cautious to assign or create those files which will be associated with the partition for the longest time first, release the assignment of files as soon as possible, and have as few files open simultaneously as possible. Similarly, the designer should cancel called programs whenever the called program is no longer needed.

The applications designer should refer to Appendix C for detailed information regarding RM/COS memory requirements.

JOB DESCRIPTION LANGUAGE

RM/COS provides a flexible means of communicating with the operating system called Job Description Language (JDL). Job Description Language features include:

* An interactive mode of entering commands
* A reasonable type of abbreviation
* Passcode log-in
* A batch mode of entering commands
* Interactive entry of selected command parameters in batch mode
* User selected command prompts for interactively entered parameters in batch mode
* Conditional execution of batch mode commands
* Synonym substitution
* Automatic translation of lower-case input to upper-case

Interactive Mode

The interactive mode allows entry of JDL commands at the keyboard of any display terminal in the system. Entry of commands and parameters on the system line and display of error messages on the error line leaves the remaining 23 lines of the terminal screen available for display of output directed to the terminal. The system line consists of the leftmost columns of row 24 (the bottom line) of the terminal screen. The error line consists of the rightmost columns of the same row.

The requirement of a passcode for log-in provides a measure of security against unauthorized use of the system. The privilege level of a user denies access to files or JDL commands that require a higher privilege level.

When entering commands, the operator may enter an abbreviation of the command name rather than the full word. Any abbreviation formed by omitting letters from the end of the word is valid if enough letters remain to uniquely identify the word being abbreviated. For example, when YES and NO are the valid responses, the letter Y uniquely identifies YES, and is all that need be entered. YE also uniquely identifies YES, and is a valid abbreviation. Similarly, B uniquely identifies command name BATCH, because no other command name begins with B. However, CO does not uniquely identify COBOL, since CONTINUE also begins with CO. One letter may be entered for some command names, two letters for many, and three or more letters are required for a few. If enough letters are not entered, the system displays an error message and the operator must reenter the command with additional letters. Sometimes a three or four letter abbreviation is more easily remembered than a shorter one, and JDL accepts the longer abbreviation.

Several commands have parameters consisting of one or more of a list of options. The same abbreviation rule applies to these options. Any abbreviation that uniquely identifies an option within the set is valid. For example, the following select COBOL compiler options:

```
DEBUG     NOOBJ     OBJLIST   XREF
NOLIST    NONE      PROCX
```

Single letters D, P, O and X uniquely identify options DEBUG, OBJLIST, PROCX, and XREF. However NOL, NOO, and NON are the minimum valid abbreviations for NOLIST, NOOBJ, and NONE respectively.

Interrupt Mode

It may be necessary to enter JDL commands while a program is executing at a terminal. JDL may be activated, interrupting execution of the program. This is the interrupt mode of JDL, and not all commands are valid. Those commands that may be entered in both the interrupt mode and the normal mode are designated as interrupt mode commands in the command descriptions. The interrupt mode commands perform the functions needed during an interruption of the program.

The command prompt is preceded by an asterisk (*) as a signal to the user that he has entered the interrupted mode.

System Keys

Some keys on the keyboard perform special functions in the RM/COS system. Special attention should be given to these keys and their various functions whenever they are mentioned. Note too that the key function(s) may vary with the mode of operation in effect at the time the key is entered. (See your Terminal Guide for the terminal keys which correspond to the Command, Acknowledge, Interrupt, and Log-in Keys).

The Command Key is a dual mode key. The Command Key is used to return to the command prompt after parameter entry has begun while in the terminal or interrupt mode. At all other times, the Command Key is treated as any other function key and is available via COBOL ACCEPT statements with the ON EXCEPTION phrase. The Command Key does not interrupt a batch stream, a JDL processor, or an executing COBOL program.

The Acknowledge Key is used to inform the system that the operator has seen a system error message displayed on the error line. This key is also used to allow the system to continue the display of information which fills more than one screen.

The Interrupt Key must be used to interrupt a batch stream, a JDL processor, or an executing COBOL program in a terminal partition. Since the Interrupt Key requires the simultaneous depression of two different keys, it is unlikely that an operator will accidentally enter the interrupt mode.

The Log-in Key is used to initiate the log-in process at an inactive terminal. On most keyboards, the Log-in Key is the same as the Return Key. After pressing the Log-in Key, the user must enter a valid user ID in response to the "Please log in" prompt, unless automatic log-in has been selected in the System Definition File. The System Definition File also

allows the system to be configured such that log-in is automatically initiated at the system console terminal (ST01) following IPL, in which case the terminal prompts for a user ID or automatically logs-in without the necessity of pressing the Log-in Key.

Batch Mode

The batch mode of the Job Description Language (JDL) increases the flexibility of the RM/COS System. A series of frequently used commands may be written on a file (a batch stream) and executed with a single BATCH command.

The JDL commands used in the batch mode are essentially the same commands used in the interactive mode. There are some differences, however:

* The command syntax differs, as appropriate for stored commands.
* Several commands apply differently in the batch mode.
* Four additional commands are provided to control execution in batch streams.

The most significant feature of batch mode JDL is conditional execution. This allows commands to be executed on error conditions, on conditions set by commands in the batch stream, or on conditions set by the user.

A batch stream may be written to interactively request parameters and to supply appropriate default parameters. The prompts may be specific to the program being executed, replacing the general prompts of the JDL command.

Synonym Capabilities

RM/COS supports synonym substitution, whereby one string of characters (usually short) can be used in place of another string of characters (usually longer).

Refer to Chapter 2, User Definition File Edit for the required steps to enable synonym substitution. If a synonym table exists, possible synonym substitution will be attempted.

The synonym substitution algorithm will automatically attempt to replace strings of file pathname type parameters up to the first terminator (blank or nonalphanumeric character except for dollar ($) or hyphen (-)). Synonym evaluation of "@.FILE" is valid and will yield the value

".FILE". For all other parameter types synonym substitution requires a leading @ character. If no match is found, the synonym will be used as the actual value. The caret character (^) may be used to define the end of a substring for synonym substitution.

The resulting value of a synonym substitution is processed as if it were part of the actual parameter value.

The @ character is always removed from a parameter item including quoted strings even if no synonym substitution occurs. In other words, the @ character is recognized as a synonym substitution request, not as the literal character @, therefore it is not possible to define a synonym value or a quoted string that contains an @ character.

Synonym substitution, if necessary, is performed using existing definitions. For example if synonym A has the value AX, the following statement:

```
SYN,SYN=A,VALUE=@A^@A
```

would result in A now having the value AXAX.

Nested substitution (multiple @) is supported up to four levels. Synonyms are evaluated "innermost" first. For example, if the following synonyms were defined

Synonym	Value
X	Y
Y	Z
Z	THE END

then the statement

```
MESSAGE, TEXT="@@@X", STATION=0
```

would result in THE END being displayed on the terminal. First, @X would be replaced by Y. Then @Y would be replaced by Z. Finally, @Z would be replaced by THE END.

System Synonyms

Several system synonyms will automatically be defined for all partitions with a system synonym table area. The synonym $PART will have a three digit value equal to the partition number in which this synonym table resides. The synonym $TYPE will have the value T if the partition is a terminal partition and will have the value N if the partition is a nonterminal partition.

The SYNONYM JDL command will not allow modification of any system synonyms.

System synonyms are not saved during log-off.

An example of the use of system synonyms would be to create a file whose name was a function of the partition executing the batch stream containing the command. Thus a single generic batch stream could be used from many partitions. The following is an example of such a command:

```
/CREATE.LOGICAL=NEWFILE.NAME=.FILE@$PART.
        ALLOCATION=100.TYPE=COMP
```

Nonterminal Partitions

Batch streams started in a nonterminal partition will have a copy of the current synonym table of the invoking partition, if one exists, transferred to the nonterminal partition. The system synonym $PART will be updated to the number of the nonterminal partition, and the synonym $TYPE will have the value N. Synonyms defined in nonterminal partitions are not saved.

Examples

Valid	Invalid
FILE@$PART	X@^$PART^XY
T$@$PART^XYZ	will evaluate to X$PART^XY
"CUSTOM MESSAGE @NAME"	which will generate an error
@@X^@Y	

Expert Mode

For those users familiar with RM/COS, an expert mode is available. Expert mode allows specification of parameter values in conjunction with the entering of the JDL command. The syntax and semantics for expert mode are the same as for batch mode command executing from a terminal partition except that the leading slash (/) character is not allowed. Only those required parameters that have no defaults must be entered and all unentered required parameters will be automatically prompted.

If a JDL command is immediately followed by a period (.) then only the required parameters, if any, are prompted. This feature is quite useful for commands that have many optional or defaulted parameters that are normally not changed. Such a command is BATCH which normally prompts for the name of the batch stream as well as three additional optional parameters which are usually defaulted. By entering BATCH. at the terminal, only the batch stream name is requested and the optional parameters are not displayed.

A shorthand technique can be created by a combination of synonyms and expert mode. For example, the command

```
SYNONYM,SYNONYM=B,VALUE="BATCH,NAME="
```

establishes a synonym that is a portion of an expert mode command. If the user wishes to initiate a batch stream, the following command

```
@B<file name>
```

will cause the batch stream in <file name> to be started. Thus,

```
@B.EDIT
```

would start the EDITOR, or

```
@B.COBOL
```

would start the COBOL compiler, or

```
@B^DEMO.PAYROLL
```

would start the batch stream DEMO.PAYROLL (assuming that volume DEMO had been previously loaded). Notice that the caret (^) is required to indicate that the synonym to be replaced was B not BDEMO.

Parameter Passing

Use of the C$SETS and C$MAPS routines in conjunction with a synonym table is a mechanism for passing small amounts of data between different COBOL programs, instead of using a file for the same purpose.

LOGGING-IN

In order to control the availability of RM/COS services and data an operator must log-in prior to any use of RM/COS. By using the log-in procedure described below, the operator is identified with regard to the amount of data and types of commands and processes available. The requirement for log-in may be defeated by the application designer (see Chapter 2, System Configuration, System Function Selection).

After loading the system as described in your Operator Guide, perform the following steps at a terminal to log-in and activate the JDL processor. The log-in process may be terminated during steps 1, 2 or 3 by pressing the Command Key or the Interrupt Key.

1. Press the Log-in Key (the Return Key) to activate the log-in JDL processor. If the terminal type of your terminal was not selected during system configuration, enter a valid terminal type in response to the prompt:

   Select terminal type:

   The list of available terminal types is in your Operator Guide.

2. Enter a valid user ID in response to the prompt:

   RM/COS VERSION 2.4.nn Please log in:

   Note that the log-in JDL processor may already be active, and the above prompt displayed, because the system may be configured to activate log-in at the system console terminal (see Chapter 2). In this case, the user need not press the Log-in Key and may simply enter a valid user ID in response to the prompt.

3. Enter the passcode assigned to the user ID in response to the prompt:

   Passcode:

   To enhance security, the system does not display the passcode as it is typed.

4. The JDL processor then either displays the command prompt or begins processing the batch command file associated with the user ID entered.

COOPERATING USER PROCESSES

RM/COS has two provisions to allow two or more user processes (i.e. programs), each running in its own partition, to act as cooperating user processes. These provisions are record locking and user event signalling. These features may be used by the applications designer separately or in combination as required.

Record Locking

A file may be shared among cooperating user processes by designating an access type other than Exclusive All when assigning the file with the ASSIGN JDL command. RM/COS prohibits concurrent modifying operations on a shared file (e.g. two WRITEs, DELETEs, a WRITE and a DELETE, etc.) while allowing concurrent nonmodifying operations (namely, two or more READs). Thus, the integrity of the file structure is assured while that file is being shared.

Record locking is used to prevent simultaneous updates of an individual record. Without record locking two processes could act as follows:

> Program A reads record X. Program B then reads record X, modifies the contents of the record, and rewrites the record. Program A then modifies the contents of the record and rewrites it.

If this were to occur, any changes in record X made by Program B would be lost. On RM/COS, however, Program A would lock record X at the time of the read, thus gaining exclusive access to that record until Program A issues a subsequent IO operation for the same file (in this example, a rewrite). Only then could Process B gain access to record X. For further information about record locking, refer to the COBOL Language Manual sections on the READ verb.

User Event Signalling

User event signalling may be used by cooperating processes to communicate the occurrence of an event of mutual interest to those processes. In the RM/COS environment, user event signalling may be combined with shared files and record locking to provide event-driven processing; that is, a program modifying a file may notify other programs that a modification has occurred. User event signalling is supported by the COBOL subroutine library package (see Appendix B).

CHAPTER 2

SYSTEM CONFIGURATION

INTRODUCTION

System configuration is a process of tailoring the factory delivered system to the requirements of a specific installation. System configuration allows:

(1) definition of the number and size of partitions;
(2) specification of devices;
(3) the selection of a subset of the standard JDL commands and the privilege level of each command;
(4) definition of the log-in initial display; and
(5) definition of the valid user IDs and their associated user characteristics (passcode, privilege level, initial batch pathname, system synonym availability, user synonym table size, and synonym file pathname).

The system configuration process is performed by the use of utilites provided as a standard part of the system. The remainder of this chapter describes how the utilities are used to configure the system. Frequent reference must be made to the Operator Guide since system configuration is intimately related to the hardware.

The system, once configured, is not bound to the configuration but may be further refined or adjusted to suit current conditions by means of various JDL commands (e.g. PARTITION modifies partitions and VARY disables or enables devices).

WARNING

Certain system files cannot be deleted or renamed. They may be modified only as described in the following section. The names of these files are

.SYSDEFIL*
.JDLDEFIL
.NEWSFILE
.USRDEFIL

---

* Or the name of the System Definition File last specified by use of the INSTALL-SYSTEM JDL command.

SYSTEM CONFIGURATION PROCEDURE

The following steps are required to perform a RM/COS System Configuration:

1. Perform IPL procedure with standard system disk provided by the factory.

   NOTE

   A back-up copy of the system disk should exist before initiating the system configuration procedure.

2. Log-in at a terminal by entering the Log-in Key (the Return Key on most keyboards; see Terminal Guide) and perform normal system initialization.

3. Invoke the Screen Editor by use of batch stream .SEDIT. In response to the "Name to edit" prompt, select the name of the System Definition File (.SYSDEFIL or the name last set by use of INSTALL-SYSTEM).

4. Use screen editor commands to modify the file as required. Refer to Chapter 4 for description of the screen editor commands.

5. Optionally repeat steps 1-4 with the JDL Definition File (.JDLDEFIL) enabling/disabling a subset of JDL commands and setting the associated privilege levels.

6. Optionally repeat steps 1-4 with the News File (.NEWSFILE) to define the initial log-in display.

7. After completing Step 6 repeat the IPL procedure to obtain the new configuration.

8. If user definitions are to be changed, use the BATCH command to initiate the batch stream .MODUSER as described later in this chapter. User definition changes become effective immediately, not at the next IPL.

Note the following points:

1. No file names should be altered; these should be entered exactly as stated above except for the System Definition File which may have a different name than .SYSDEFIL as established by the command INSTALL-SYSTEM (see Chapter 3).

2. The new system configuration does not take effect until after the next IPL.

SYSTEM DEFINITION FILE

The System Definition File is a standard 80 byte logical record sequential source file. The factory supplied name for this file is .SYSDEFIL, but the name may be changed by use of the INSTALL-SYSTEM command. The file has four sections:

System Function Selection Section (S record);
Device Specification Section (C and U records);
Partition Specification Section (P records); and

The records must be in the order S, (C, U...)..., P and D, though some sections may be omitted. Records with an asterisk in column one are treated as comments; any other character in column one is an error and terminates processing of the file.

All numeric fields in these records are variable in length, either decimal or hexadecimal. A hexadecimal number has a greater than sign (>) as the first nonblank character. The numeric value begins with the first nonblank character and ends with the first blank or comma following the nonblank characters. A field of all blanks is treated as 0. Illegal characters cause an error.

System Function Selection Section

The System Function Selection Section is used to enable automatic log-in, enable automatic log-in initiation, and specify the command prompt. This record must be the initial record in the file. The format of the record in the System Function Selection Section is as follows:

Col 1	S
Col 4	if this field is: A = automatically log-in the system manager when the Log-in Key is pressed without requiring any user input. any other character = require user to enter a valid user identification and passcode to log-in to the system.
Col 5	if this field is: L = automatically initiate log-in at the console terminal following IPL without requiring the user to press the Log-in Key. The console terminal is the lowest numbered terminal in the system, usually ST01. any other character = require user to press the Log-in Key to initiate log-in.
Col 8-9	Command prompt characters. If this field is blank, the command prompt defaults to [ ].
Col 12-15	Year. Only used on systems which have a battery-backed clock which does not retain the year.
Col 18-21	Keyboard type-ahead buffer size, in decimal. If invalid or blank, the default (five characters) is used.
Col 24-27	Event log buffer size. The size of the allocated event log buffer. If nonblank it should be at least 256.

If this record is omitted, the default is to require user log-in with a valid user ID, require pressing the Log-in Key to initiate log-in, set command prompt characters to [ ], allocate a five character type-ahead buffer for each terminal, and disabled event logging.

Device Specification Section

The Device Specification Section defines all devices available to the user. The standard system disk defines the devices expected in a typical system. This configuration may be reduced or expanded to include additional disks, terminals, printers, magnetic tape units, or communications devices. The Operator Guide gives additional details about the device specifications allowed on particular hardware models supported by RM/COS.

The Device Specification Section consists of controller specification (C) records and unit specification (U) records. The section must be ordered such that each C record is immediately followed by all of the U records of units associated with the controller specified by the C record.

The format of a controller specification record in the Device Specification Section is as follows:

Col 1	C
Col 3	board-type (see Operator Guide)
Col 5+	optional device-address and other controller parameters (see Operator Guide)

The format of a unit specification record in the Device Specification Section is as follows:

Col 1	U
Col 3-6	Device name (e.g. ST02)
Col 12+	controller/controller-number/unit-number and other unit parameters (see Operator Guide)

C and U records must follow the S record, if present, and precede the P records. U records must immediately follow the C record for the controller with which the units are associated.

The second field of the U record gives the name of the device being defined, which must be unique. The first two characters of the device name indicate the device type and are limited to the following:

DS	Disk
LP	Line printer
MT	Magnetic tape
ST	Station
BL	2780 or 3780 Bisynchronous communications link
UL	User controlled communications link

The two suffix characters of the device name must be decimal digits. The only restriction placed on these digits is that the device suffix of station devices identifies which partition is used by that station.

Partition Definition Section

The Partition Definition Section consists of one or more records in the following format:

Col 1	P
Col 3-9	Partition size in bytes
Col 11-17	Partition priority

The number of P records may be changed from the number originally provided in the System Definition File. The number of P records determines the number of partitions.

The highest numbered terminal device determines how many of the P records define terminal partitions. For example, if ST02 is the highest numbered terminal, then the first two P records define terminal partitions, even if no ST01 device is specified. The remaining P records except the last P record define nonterminal partitions. The last P record defines the shared file partition.

A partition size of zero specifies an unused partition. An unused partition is treated as nonexistent except for the purpose of partition numbering. It may only be given memory by changing the System Definition File and rebooting (reIPLing) the system.

JDL DEFINITION FILE

The JDL Definition File (.JDLDEFIL) is a standard 80 byte logical record sequential source file that is used to define characteristics of JDL commands for the system. Throughout the file any record that does not begin with the character J in column one is treated as a comment.

The JDL Command Definition File contains of one of the following records for each JDL command:

Col 1 J

Col 5-24 Command name

Col 30 T = This command is allowed in terminal mode. The commands CHAIN, LOOP, REPEAT, and REPOINT must not be allowed in terminal mode.

Col 31 B = This command is allowed in batch mode. The commands CONTINUE, HALT, and KTASK are not allowed in batch mode because they require a partition number, but this can be modified without jeopardizing system integrity.

Col 33 M = This command is allowed in interrupt mode if it is allowed in terminal mode. Commands which build permanent memory structures, modify the disk, or obtain files or directories with other than Read Only access must not be allowed in interrupt mode. The forbidden commands include ASSIGN, CHANGE, CONNECT, CREATE, DELETE, DIRECTORY, EXECUTE, FCOPY, FILE-BACKUP, FILE-RESTORE, FILE-VALIDATE, FTS, INSTALL-SYSTEM, MODIFY, PATCH, PRINT, QUIT, RECEIVE, RECLOSE, RELEASE, RENAME, REPLACE, SCRATCH, SEND, TAPE-ASSIGN, and VCOPY. Other commands may be allowed in interrupt mode if desired by the system manager.

Col 34 U = This command is allowed before the time has been initialized. As provided by the factory, the commands BATCH, CONTINUE, EXIT, LOOP, QUIT, REPEAT, SETCOND, and TIME are allowed. These commands enable the batch stream .TIME to be executed and continued if interrupted.

Col 35-39 Minimum privilege level in decimal required to use the command.

The user should not delete any records in the JDL Command Definition Section. To remove a JDL command entirely, blank both columns 30 and 31. To create a turnkey system such that all user action occurs through batch streams, blank columns 30 and 33 in all commands except CONTINUE and QUIT. Note that if the CONTINUE command is disabled in interrupt mode and the user accidently enters the Interrupt Key, the user's terminal will be unusable until the system is rebooted.

NEWS FILE

The News File (.NEWSFILE) is a standard 80 byte logical record sequential source file that contains the information to be displayed upon the terminal when the Log-in Key is entered.

If no initial display is desired, delete or blank all records in the file. The file must exist and cannot be deleted.

Only the first 23 records of the file will be displayed.

USER DEFINITION FILE PROCEDURE

For each user of the RM/COS System there exists one logical record in the User Definition File (.USRDEFIL). This record contains the name of the user, the passcode assigned to the user and other user-dependent information. The user record information in the User Definition File profiles a valid user with his assigned attributes for the system. These characteristics form a user definition data base that is used by the system to validate: (a) initial system log-in, (b) privilege level of JDL commands requested, and (c) user file accesses.

User Definition File manipulation is performed interactively after System Configuration time and completion of IPL. The file remains unaltered through system configuration modifications and system loadings. Only upon an explicit execution of the interactive program and the subsequent modifications are the User Definition File records altered. User Definition File changes become effective at the next log-in of the corresponding user.

INITIAL USER DEFINITION FILE

The RM/COS system is shipped from Ryan-McFarland Corporation with only one user record defined. This is the user record of the system manager and permits a terminal activation and log-in with the predefined passcode and privilege level entries in the sole user record. The system manager may then execute the User Definition File manipulation program and enter the full roster of users and user characteristics to be recognized by the system. The paramount considerations in assigning user privilege levels are: (a) all common user privilege levels should be less than the system manager privilege level and (b) the system manager privilege level should be unique. Upon completion of each user file manipulation procedure the system manager may elect to alter his own user definition record options. The system manager privilege level must remain equal to or higher than that assigned to the user definition file itself if the user definition file is to be modified ever again.

The initial user definition record has the following values:

```
User ID = SYSTEM MANAGER
Passcode = RM/COS
Privilege Level = 65535
Initial Batch Name = .TIME
System Synonym? = NO
Users Synonym Table Size = 0
Synonym File Name = <blank>
```

USER DEFINITION FILE EDIT

Modification of the User Definition File is accomplished through execution of a factory supplied batch stream. The procedure is interactive: commands, parameter value responses and an optional listing of the user definition file are activated by user entries in a terminal partition. The fields of any user record (password, privilege level, etc.) may be altered to reflect the desired user profile and new users may be added with assigned passwords and privilege levels. Inactive user IDs may be deleted.

The following steps define the edit procedure:

(1) Follow the procedure described in Appendix E, Execute Modify User Identifications.

(2) The prompt

USER DEFINITION FILE EDIT COMMAND:

is then displayed. The user may then enter one of five commands:

ADD
CHANGE
DELETE
LIST
QUIT

Each command may be entered in truncated form. For example L, LI, LIS and LIST are all valid variants of the command LIST.

(3) Entry of the QUIT command terminates an edit session.

ADD User Definition

Format:

```
ENTER USER ID TO BE ADDED:
                PASSCODE:
              PRIVILEGE: 0
      INITIAL BATCH NAME:
         SYSTEM SYNONYM?: NO
  USER SYNONYM TABLE SIZE: 0
       SYNONYM FILE NAME:
```

Where:

ENTER USER ID TO BE ADDED
: ID to be assigned to the new user. A valid value is an alphanumeric character string up to thirty characters in length including embedded blanks. If no value is entered, the command prompt line is redisplayed.

PASSCODE
: Passcode to be assigned to the new user. A valid value is an alphanumeric character string up to eight characters in length, including embedded blanks. The first blank encountered terminates the passcode. A null input indicates that no passcode will be required.

PRIVILEGE
: Privilege level to be assigned to the new user. A valid value is a number not greater than 65535.

INITIAL BATCH NAME
: Pathname of a batch stream to be automatically invoked at the completion of successful user log-in. If the field is omitted or the first character is a blank, the system will enter the JDL interactive mode at the completion of successful user log-in. The pathname entered is checked for correct syntax, however no checks are made to ensure that the pathname entered actually exists. If an invalid pathname is entered, any attempt to log-in under that user ID will result in an error message that the batch stream is undefined, and the user will be logged-in without an initial batch stream. The batch stream should start with the following JDL command to ensure that the system time is initialized:

```
T/BATCH, NAME=.TIME
```

except in the case where the initial batch name is .TIME (a vendor supplied batch stream for ensuring

that the time is initialized).

SYSTEM SYNONYM?
Either the value YES or NO. The response of YES indicates that the user wishes space to be allocated for system synonyms. If system synonyms are selected they will automatically be defined when the user logs-in.

USER SYNONYM TABLE SIZE
Amount of memory to be allocated to the user portion of the synonym table. The space required by each synonym value pair is:

length of synonym in bytes
+ length of value in bytes + 2

One additional byte is used to indicate the end of definitions within the table.

If the specified synonym table size is not large enough to hold all of the synonyms in the file, the user is given an error message indicating the size that would have been required to hold all of the synonyms, and the user is logged-in with as many synonyms as fit in the synonym table. No synonyms will be written when the user logs-off even if a synonym file had been specified. User response should be to use MODUSER to change the synonym table size specification, QUIT, and log-in again.

SYNONYM FILE NAME
Pathname of a file to be used to support permanent synonyms. If no file name is specified, synonyms will not be saved when the user logs-off. The pathname entered is checked for correct syntax; however no checks are made to ensure that the pathname entered actually exists. If an invalid pathname is entered, an error message will be generated at log-in time. The file specified must be a sequential, 78 byte logical record length file. At log-in time, if the file exists, is not undefined, and matches the logical characteristics of a synonym file, the contents of the file are used to initialize the synonym table. The file is rewritten with the current synonym table contents whenever a terminal partition is logged-off. No file specification is allowed for the master user.

The up arrow and down arrow keys may be used to position backward or forward through the prompt fields without changing their values.

CHANGE User Definition

Format:

```
ENTER USER ID TO BE CHANGED:
                    PASSCODE:
                  PRIVILEGE:
          INITIAL BATCH NAME:
             SYSTEM SYNONYM?:
     USER SYNONYM TABLE SIZE:
           SYNONYM FILE NAME:
```

Where:

ENTER USER ID TO BE CHANGED
: Identification of the user to be changed.

PASSCODE
: New passcode to be assigned to the user identified by the previous parameter.

PRIVILEGE
: New privilege level to be assigned to the user identified by the first parameter.

INITIAL BATCH NAME
: New pathname for the initial batch stream to be automatically invoked at the completion of a successful user log-in.

SYSTEM SYNONYM?
: New selection for system synonyms.

USER SYNONYM TABLE SIZE
: New synonym table size.

SYNONYM FILE NAME
: New pathname for the file to be used to support permanent synonyms.

If the user ID value is not matched by a user ID in the user definitiion file an error message is generated. Otherwise, the current profile for the user ID is displayed.

The up arrow and down arrow keys may be used to position backward or forward through the prompt fields without changing their values.

DELETE User Definition

Format:

```
ENTER USER ID TO BE DELETED:
```

Where:

```
ENTER USER ID TO BE DELETED
     ID of the user to be deleted.
```

If the value entered identifies the master user (system manager) the following error is generated: DELETE OF MASTER USER NOT PERMITTED. If the value entered is not matched by a user ID in the user definition file the following error is generated: USER ID NOT FOUND.

LIST User Definition

When the user enters LIST, the command is processed without further prompting. The current user definitions are listed to the device selected in response to the original list name prompt. If the listing is to a line printer, care should be taken to keep the printed listing secure since all user IDs and passcodes are included in the listing.

If for whatever reason the opening of either the listing file or the user definition file is unsuccessful, the command is aborted. If the error is caused by an attempted OPENing of the listing file, the error message

COMMAND ABORTED. LISTING FILE STATUS= XX

is displayed on the terminal screen, where XX is the COBOL I/O error status code.

QUIT User Definition

When the user enters QUIT, the command is processed without further prompting. The process terminates and control is returned to the JDL command mode.

CHAPTER 3

JOB DESCRIPTION LANGUAGE

Introduction

This chapter describes Job Description Language (JDL), the means by which an operator communicates with the RM/COS operating system. JDL is comprised of commands which are entered by the operator interactively or are read from a sequential file (also called batch streams).

General information regarding the use of JDL commands is at the beginning of this chapter, followed by a description of each available JDL command, in alphabetical order by command name. The syntax of each command in this chapter is for interactive entry. With experience, one should be able to formulate the batch stream syntax for any command from the interactive syntax; however, a summary of batch stream syntax appears in Appendix F.

ENTERING JDL COMMANDS

JDL commands are entered interactively on the system line in response to the following prompt:

```
[ ]
```

Enter the command name or an abbreviation of the command name. Press the Return Key. The JDL processor responds by displaying the first parameter prompt, or by executing the command if there are no parameters. The parameters required for each prompt are shown in the command descriptions. Enter the parameter and press the Return Key. The Return Key causes all characters currently displayed to be transmitted while the Erase Right Key transmits only those characters up to the cursor position. If a default value was changed and the number of characters entered is less than the default, the Erase Right Key should be used instead of the Return Key. If the parameter is optional or has a system defined default value then the user may select these by simply pressing the Return Key. During command prompting the Send Key acts identically to the Return Key; pressing either the Command Key or the Interrupt Key will stop parameter prompting and the command prompt ([ ]) will appear. Further information on the editing functions of terminal keys may be found in Appendix I; the actual keycap names of the various keys on each terminal may be found in the RM/COS Terminal Guide. When the Return Key is pressed after entering the last parameter, the JDL processor performs the command. When the command is complete the command prompt is again displayed.

For convenience, the JDL processor usually translates all lower-case letters to the corresponding upper-case letters. Strings enclosed in quotation marks are the only exception.

The following conventions apply to the parameter definitions in the command descriptions at the end of this chapter:

* Items in capital letters are reserved words. They may be entered as shown, or abbreviated as described in Chapter 2.

* Items in lower case letters are types of parameters to be supplied by the operator.

* Items enclosed in square brackets ([ ]) are optional.

* Items separated by a vertical line (|) indicate that one of the items or its abbreviation must be selected.

* An ellipsis (...) following an item indicates that

the item may be repeated one or more times.

* Items underlined are provided as system defaults.

* The names enclosed in angle brackets (< >) on the header line with the command name indicate valid modes for the command: terminal, interrupt, batch.

The types of parameters used in the command descriptions are:

string — A string of characters that may include any character except quotation mark ("), blank, right bracket (]), right parenthesis, semicolon (;), or comma (,). Any character other than quotation mark or semicolon may be included if the string is enclosed in quotation marks. Strings not enclosed in quotation marks are translated to upper case.

Examples: RMC248
"Error in BATCH stream"

acnm — A file pathname or device name. Note that this type of parameter is subject to automatic synonym substitution.

Examples: BLUE.BATCH.LOOP
DS01

int — An integer in the range of 0 through 65535, or a hexadecimal integer in the range of 0 through FFFF. (The SIZE parameter of the PARTITION command and the SECTOR parameter on the SDUMP and SMODIFY commands may be larger than 65535 or FFFF.) Hexadecimal integers are entered with a greater than sign (>) preceding the first digit. Leading zeroes are ignored.

Examples: 4537
>4F2

hex — A hexadecimal integer in the range 0 through FFFF. The greater than sign (>) preceding the first digit is optional. Leading zeroes are ignored.

Examples: 1E7
>4F2
01C3

name — A string of characters beginning with a letter (A-Z) and containing only letters, digits, the hyphen (-), and the dollar sign ($).

Example: VOL345

lname — A string of characters used as a logical name. A logical name consists of no more than eight characters, begins with a letter (A-Z), and contains only letters, digits, the hyphen (-), and the dollar sign ($).

Examples: FILE1
INPUT
IN-FILE

When the JDL processor detects an error in a parameter, or during execution of the command, the processor displays an error message in the error line. The user must positively acknowledge the error by pressing the Acknowledge Key. The JDL processor repeats the command prompt, and the user reenters the command name, and all parameters.

On occasion an error message will appear on the system error line that is not caused by an error in user command entry. This message is:

4001 FAIL LOAD

The memory requirements vary with each command processor supporting the individual JDL commands. The system loads the JDL processor for a particular command into a memory area assigned to a terminal when the command is entered at that terminal. This memory area is called a partition and is defined when the system is generated. When the 4001 error code is displayed, there is not enough memory space available in the partition for the requested JDL command processor. More memory must be assigned to the partition with a PARTITION command before the command that terminated in error can execute.

At any time during entry of parameters for a command, pressing the Command Key terminates the command. This returns control to the JDL processor, which then displays the command prompt. If the Interrupt Key is pressed while the command is executing, the command is suspended. An EXIT command must be entered to terminate the command. A CONTINUE command allows the interrupted command to continue.

COMMAND LISTINGS

Many JDL commands generate listings, either as the purpose or a side effect of their actions. These commands are:

DELETE	MAP-SYNONYMS
FCOPY	PRINT
FDUMP	QUIT
FILE-BACKUP	RECLOSE
FILE-RESTORE	RELEASE
FILE-VALIDATE	SDUMP
FMODIFY	SMODIFY
LIST	STATUS
MAP-FLAWS	TIME
MAP-KEYS	VCOPY
MAP-PROGRAMS	

The destination of the listings from these commands depends on several factors. In interrupt mode the listings always go to the terminal. In noninterrupt mode, if the logical name LO exists, it identifies the destination. If logical name LO does not exist, the listing goes to the terminal. For example, if a printed listing of one of these commands is needed, the following JDL command should precede the command that generates a listing.

```
[ ] ASSIGN
LOGICAL NAME: LO
NAME: LP01
ACCESS: EA
BUFFERS:
```

Every command that generates a listing will then send its output to the printer.

When output to the printer is no longer needed, the following command should be entered:

```
[ ] RELEASE
LOGICAL NAME: LO
```

When the output of a command is directed to a terminal, the command pauses after the first screen of information is displayed. The cursor is positioned in the bottom right hand corner of the screen to wait for operator action. Each succeeding screen is displayed after the Acknowledge Key has been pressed.

BATCH STREAMS

This section will describe how to write batch streams. The JDL command syntax for batch streams will be defined. Several techniques for utilizing the features of batch streams will be discussed.

Command Syntax

The syntax of batch mode JDL commands is as follows:

```
[letter].../[<options>] command [,keyword[(prompt)]=
            parameter] [,keyword[(prompt)]=parameter]...
```

where:

letter	is one or more printable ASCII characters or a blank. The characters ! @ # " and ; and lower-case letters are not allowed as condition codes.
options	is one or more of the following, separated by commas: NE, NL, NDM, or XL.
command	is a JDL command name, or a valid abbreviation.
keyword	is one of the prompts listed for the command, or a valid abbreviation.
prompt	is a text string to be used instead of the keyword when prompting the operator for the value of the parameter.
parameter	is a parameter value of the specified type, or a list of parameter values of that type. If enclosed in parentheses, the parameter value is displayed to the operator as a default value, and the operator may modify the value before the batch stream continues execution.

The letter, or letters, preceding the slash (/) specify conditional execution for the command. The command is conditionally executed depending on a match between one of the letters preceding the slash and the current condition code value of the partition. Details of the conditional execution letters are described in the conditional execution section.

The optional list preceding the command may be used to suppress listings and/or error messages being displayed to the terminal, or to cause the listing file to be opened

EXTEND rather than OUTPUT. The value NL indicates to suppress the listing output of those JDL processors that normally generate listings (see Command Listings above). If the NL option is selected, no JDL listing will be generated. The value XL indicates to those JDL processors that generate listings to open the listing file with an OPEN EXTEND, causing the listing to be appended to any data already in the file. The value NE indicates to suppress displaying of an error message to the error line if this JDL processor has an error. The error message is still written to the batch listing file if a listing file is specified and the condition code is still set to the letter Z. The value NDM indicates to suppress informative disk messages; i.e., "UNLOAD DSnn" and "INSERT DISK".

It is not necessary to enter keywords for optional parameters when the parameter is not required, or when the default parameter is selected.

Several commands have parameters that may consist of several values or reserved words. In the batch mode, lists of parameters must be enclosed in square brackets ([ ]) or angle brackets (< >); items within the list must be separated by commas. Single item lists in batch streams need not be enclosed in this manner.

A JDL command may be continued by stopping after a comma or an equal sign that is not included in a quoted string and continuing on the next record. Comments may be included in a record by entering a semicolon (;) between the last character of the command and the first character of the comment. When the first nonblank character of a record is a semicolon, the entire record is treated as a comment.

Syntax errors within a batch stream will terminate the batch stream without executing any additional commands.

JDL Listing Output

If the logical name LO is not assigned, JDL processor output is written to the listing file specified in the BATCH command. In a terminal partition, when logical name LO is not assigned and no listing file is specified, the listing is displayed on the screen of the initiating terminal unless the NL option was specified. In a nonterminal partition, when logical name LO is not assigned and no listing file is specified, the listing lines are discarded.

Conditional Execution

The condition code of a partition controls conditional execution of the JDL commands in a batch stream executing in that partition. Initially, the condition code is a blank. The condition code may be set by a SETCOND command in the

batch stream or executed by the operator. When a batch stream executes a MESSAGE command to a specific terminal and requests a reply, the reply is the new condition code for the partition. When an error occurs during execution of a JDL command processor, the condition code is set to the letter Z. When the command executes without error, the condition code is not altered.

A command in a batch stream may have one or more condition letters preceding the slash. The command does not execute unless the condition code is equal to one of the condition letters of the command. If the condition code is a blank, the command will be executed only if no (nonblank) condition letters precede the slash.

If the first nonblank character is the pound sign (#) then the command will be executed only if the condition code is not one of the condition letters preceding the slash. A pound sign (#) with no intervening nonblank characters before the slash means to execute this command only if the condition code is not a blank.

A batch stream may contain optional commands, controlled by a condition code set before the batch stream is executed or by use of the C$SCC subroutine. The commands for each option are preceded by the condition letter defined for the option. Commands applicable to more than one option must have the condition letter for every option to which they apply. Similarly, the operator may be requested to specify or change the condition code with a MESSAGE command specifying a reply.

Unconditional Execution

A command in a batch stream will be unconditionally executed if the first nonblank character in the record is an exclamation (!).

BATCH STREAM TECHNIQUES

This section will discuss various techniques that may be used within batch streams.

Performance Considerations

Most JDL commands are not reloaded from disk if they are already in memory. Therefore, like JDL commands should be grouped wherever possible. For example if three ASSIGN and two CREATE JDL commands are required in a batch stream, the ASSIGN commands should be grouped and not have the CREATE commands interspersed.

Control of Logical Names

To minimize memory fragmentation, each batch stream should release any logical name assignments that it makes. The release operation should be made using an unconditional execution JDL command to ensure execution even if errors occured. For example if the logical names SOURCE, OBJECT and LIST were assigned in a batch stream, the following statements should appear at the end of the batch stream:

```
!/RELEASE, LOGICAL NAME = <SOURCE, OBJECT, LIST>
```

Condition Codes

All JDL commands set the condition code to the letter Z if any execution time errors occur. By using the special condition code letter Z it is possible to detect certain errors and prompt the user for corrective action or to select between a set of options automatically.

For example, a batch stream is needed that among other things must assign a listing output file that may or may not exist. By use of the Z condition code it is possible to either make the assignment or create the file if it does not exist. The following is an example of such a sequence:

```
   /<NE>ASSIGN, LOGICAL NAME=LIST,
            NAME=.LISTING.OUTFILE,
            ACCESS=EA
Z  /CREATE, LOGICAL NAME=LIST,
            NAME=.LISTING.OUTFILE,
            ALLOCATION=1, SECONDARY=1,
            TYPE=<WORK,COMPRESSED,SPOOL>
Z  /SETCOND
```

This example assumes that the directory LISTING exists and that the CREATE will not fail. If detection of a possible CREATE failure is necessary the sequence should be modified as follows:

```
   /<NE>ASSIGN, LOGICAL NAME=LIST,
            NAME=.LISTING.OUTFILE,
            ACCESS=EA
Z  /SETCOND, VALUE=C
C  /CREATE, LOGICAL NAME=LIST,
            NAME=.LISTING.OUTFILE,
            ALLOCATION=1, SECONDARY=1,
            TYPE=<WORK,COMPRESSED,SPOOL>
Z  /EXIT
C  /SETCOND
```

In this case the condition code is changed to a character other than a Z before the CREATE. If a CREATE error occurs the batch stream is terminated by the EXIT command. If an error does not occur, the condition code is unchanged i.e., still a C, and the SETCOND command will then be executed to set the condition code to a blank for the remainder of the batch stream.

User Prompting

When a batch stream is executed in a terminal partition, parameters may be entered interactively at the terminal. This may also be done for a batch stream executing in a nonterminal partition, so long as no UNCOUPLE command has been used to disassociate the nonterminal partition from its parent terminal partition. When a required parameter is not specified in the batch stream command, the prompt for that parameter is displayed, and the operator must enter the parameter, or accept the default value.

For example, the LOAD command has two parameters, both required. The following command could be entered in a terminal partition batch stream:

```
/LOAD, DEVICE=DS02
```

When this command is executed, the following prompt will be displayed:

```
VOLUME:
```

A user default, appropriate for an application, may be supplied in a terminal partition batch stream. The default value is specified as if it were a parameter, except that it is enclosed in parentheses, as follows:

```
/LOAD,DEVICE=(DS02)
```

When this command is processed, the default is displayed and the operator may use the default or enter another value. The resulting prompts follow:

```
DEVICE: DS02
VOLUME:
```

Often an application related prompt is more appropriate for a requested parameter. An application related prompt is entered, in parentheses, immediately following the parameter keyword. A default value or a pair of parentheses must be entered following the equal sign. The ASSIGN command may be entered as follows, in the batch stream:

```
/ASSIGN, LOGICAL NAME=LO, NAME(PRINTER)=()
```

The following display results when the command is processed:

```
PRINTER:
```

A user default could be supplied, by entering it within the parentheses to the right of the equal sign, as follows:

```
/ASSIGN, LOGICAL NAME=LO, NAME(PRINTER)=(LP01)
```

The display for this command is as follows:

```
PRINTER: LP01
```

In order for any characters other than letters (A-Z) and numbers (0-9) to appear in an application-related prompt, the prompt string must be enclosed in quotation marks (") and may not itself contain quotation marks or semicolons. The preceding example could be as follows:

```
/ASSIGN, LOGICAL NAME=LO,
  NAME("PRINTER DEVICE NAME")=(LP01)
```

The display for this command is as follows:

```
PRINTER DEVICE NAME: LP01
```

A batch stream that contains commands that do not specify all required parameters, that specify user defaults or that specify application related prompts is not allowed in a nonterminal partition if an UNCOUPLE command has been used to disassociate the nonterminal partition from its parent terminal partition. Otherwise, user prompting is allowed. Prompts are preceded by the number of the nonterminal partition.

Looping

Use of the LOOP and REPEAT JDL commands allows more sophisticated batch streams including allowing operator intervention in error cases. For example, a batch stream may prompt the operator to specify the destination for the listing output of a report. If the required device or file is currently being used, the following batch stream would allow the operator to specify an alternative destination:

```
    /LOOP
Y   /SETCOND
    /ASSIGN, LOGICAL NAME=LIST,
             NAME("LISTING DESTINATION")=(LP01),
             ACCESS=EA
Z   /MESSAGE,
     TEXT="NOT AVAILABLE, RETRY?(Y OR N)",
     STATION=0, REPLY=YES
Y   /REPEAT
#Y  /EXIT
```

Another example of the use of LOOP and REPEAT is the case where the batch stream will automatically load a secondary volume if it is not loaded, or do nothing if it is already loaded or instruct the operator to insert the correct volume if it is not present. A sequence of JDL commands to perform this operation follows:

```
;   ATTEMPT TO ASSIGN TO KNOWN FILE
    /<NE>ASSIGN,LOGICAL=TEMP,
             NAME=DEMO.PROGRAM.MENU,ACCESS=RO
Z   /<NE>UNLOAD,VOL=DS02      ; UNLOAD POSSIBLE LOADED VOLUME
Z   /SETCOND, VALUE=Q
Q   /MESSAGE,TEXT="PLEASE INSERT VOLUME 'DEMO' IN DRIVE DS02",
             STATION=0
Q   /LOOP
Q   /<NE>LOAD, VOL=DEMO,DEVICE=DS02
Q   /SETCOND                  ; SUCCESSFUL LOAD OPERATION
Z   /MESSAGE,TEXT="INCORRECT VOLUME, PLEASE CORRECT",STATION=0
Z   /SETCOND, VALUE=Q
Q   /REPEAT
    /RELEASE, LOGICAL=TEMP    ; RELEASE POSSIBLE LOGICAL NAME
```

Synonyms

Use of synonyms in batch stream provides additional flexibility. Perhaps the only drawback of the use of synonyms is that in order to use a batch stream that contains synonyms, the user must have synonym capability enabled. A simple example will now be given that combines many of the techniques previously discussed including the use of synonyms. If the example of either assigning to or creating a file depending on the file existance is combined with the case wherein the operator specifies the file, a need for synonyms arises. Such a sequence follows:

```
   /LOOP
R  /SETCOND
   /SYN, SYNONYM=PATH,
         VALUE("Enter Pathname")=()
   /<NE>ASSIGN, LOGICAL NAME=SOURCE, NAME=PATH,
         ACCESS=EA
   /SETCOND, VALUE=S          ; SUCCESSFUL ASSIGN
Z  /MESSAGE,
      TEXT="Invalid Pathname. Retry (R), Create (C), Quit (Q)",
               REPLY=YES, STATION=0
R  /REPEAT
#CS /SETCOND, VALUE=Q         ; QUIT

C  /LOOP
R  /SYN, SYNONYM=PATH,
         VALUE("Enter Pathname")=()
R  /SETCOND, VALUE=C
C  /<NE>CREATE, LOGICAL NAME=SOURCE, NAME=PATH,
            ALLOCATION=1, SECONDARY=1, TYPE=<W,C>
C  /SETCOND, VALUE=S          ; SUCCESSFUL CREATE
Z  /MESSAGE, TEXT="Invalid Pathname. Retry? (R) or Quit (Q)",
            REPLY=YES, STATION=0
R  /REPEAT
!  /SYNONYM, SYNONYM=PATH, VALUE=""
#S /EXIT

S  /SETCOND
```

Testing

Directing the batch listing output to the terminal by selecting the LIST JDL=YES and LIST NAME=ME option in the BATCH command is an effective way to test and debug batch streams.

ASSIGN <Terminal, Batch>

Function:

The ASSIGN command assigns a logical name to a disk file or a character oriented device with one of four types of access:

RO -- Read Only
SH -- Shared
EW -- Exclusive Write
EA -- Exclusive All

Read Only access applies to files and allows only read operations. Shared access allows any operation by any program, except that when one program has the file opened output no other program may open the file with an open mode other than input. Exclusive Write access allows only the program in the partition that assigns the logical name to write to the file; other programs can read the file. Exclusive All access allows any operations, but only the program in the partition that assigns the logical name can access the file. If the Shared File Partition has a size of zero, two partitions can never assign the same file unless the access is Read Only.

Disk files with program organization may be assigned with only Exclusive All or Read Only access. If the file is assigned Exclusive All, new programs may be written to the file, but no programs on the file may be executed. If the file is assigned Read Only, programs may be executed but the file cannot be modified. If an access other than Exclusive All is requested, the program file will be assigned Read Only regardless of the access requested.

Terminals may be assigned only from their own terminal partition. The access requested when assigning a terminal is ignored; terminals are always assigned with Shared access.

Printer devices behave slightly differently depending on whether the device is assigned with Shared access. If a printer device is assigned with Exclusive Write or Exclusive All access, then when the logical name is released the paper in the printer will be forced to top of form. If Shared access is requested, then the printer is not accessed when the logical name is released.

Logical names are assigned to tape devices with the TAPE-ASSIGN command, not with the ASSIGN command. Logical names are assigned to communication link devices with the CONNECT command, not with the ASSIGN command. See the CONNECT command description and Chapter 6, Data Communications, for additional information on data

communications.

Format:

```
LOGICAL NAME: lname
NAME: acnm
ACCESS: [ RO | SH | EW | EA ]
BUFFERS: [ int ]
```

Where:

LOGICAL NAME
: Logical name which is to be assigned to a device or file.

NAME
: Device name of a device or the pathname of a file to which the logical name is to be assigned.

ACCESS
: One of four access types described above. The default for program organization disk files is RO, the default for all other disk files is SH, and the default for line printers is EA.

BUFFERS
: Number of blocking buffers to be allocated for I/O to a disk file. The program opening the file can override this parameter by specifying the number of buffers in the RESERVE integer AREAS clause in the COBOL program. If no RESERVE integer AREAS clause is specified and no number is entered the system allocates two buffers, except that only one buffer is allocated for a sequential or relative file opened INPUT. The system ensures that at least two buffers are allocated for indexed files opened other than INPUT. The buffers are allocated when the file is opened and released when it is closed. If a device name is assigned to a logical name, this parameter is ignored.

If the privilege level of the user is less than that of the file the logical name is not assigned.

Examples:

```
[ ] A
LOGICAL NAME: OUT
NAME: RED.PAY.MASTER
ACCESS: SH
BUFFERS: 2
```

The command assigns logical name OUT to file RED.PAY.MASTER. The access is shared and two blocking buffers are to be

allocated when the file is opened unless specified differently by the program.

```
[ ] AS
LOGICAL NAME: LO
NAME: LPO1
ACCESS: EW
BUFFERS:
```

The command assigns logical name LO to LPO1 for exclusive write access.

```
[ ] ASSIGN
LOGICAL NAME: LO
NAME: MYLP
ACCESS: EA
BUFFERS:
```

The command assigns logical name LO to the printer device connected to the associated with the executing partition. The assigned access is exclusive all. This command is valid only in a terminal partition and only if a "pass-through" printer was configured.

BATCH <Terminal, Batch, Interrupt>

Function:

A batch stream is a sequential file of JDL commands in the batch mode format. The commands in a batch stream perform the functions necessary to execute a program, or several programs in sequence. A batch stream may interactively request parameters for the batch stream commands. The prompts for these parameters may be the prompts the operator would see if the command were entered at the terminal, or the batch stream commands may issue prompts tailored to the program being executed.

Ryan-McFarland Corporation supplies several batch streams, described in Appendix E, to execute system utilities. The supplier of other programs for a customized system often supplies a batch stream to execute each program. These batch streams are convenient to use because only the BATCH command need be entered. All of the other commands required are in the batch stream.

The BATCH command may be used within a batch stream to call another batch stream (called nesting) or to initiate concurrent batch processing in another (idle) partition. Batch streams may be nested up to four levels deep. The CHAIN command is used to transfer batch processing control to another batch stream without nesting or to change nesting levels without returning through the calling batch streams.

Format:

```
NAME: acnm
LIST JDL: [ YES | NO ]
LIST NAME: [ acnm ]
PARTITION: [ int ]
```

Where:

NAME
Pathname of the batch stream to be executed.

LIST JDL
When YES is entered, the commands in the batch stream are listed in the batch listing file. When NO is entered the commands are not listed, but error messages and other listing output are listed in the batch listing file (see the Command Listings section for details about other listing output). When no value is entered this parameter's value is equivalent to NO if the BATCH command is initiating batch processing in a partition; otherwise it indicates no change to the previous value for this parameter.

When JDL commands are listed in the batch listing file, each command is preceded by is a number, a space, and a character followed by a colon (:). The number refers to the batch nesting level for this batch stream. The character is the condition code in effect when the JDL processor read the command (note that blank is a valid character). Any error messages resulting from execution of the command are written in the listing file immediately following the command, in addition to being sent to the initiating terminal.

LIST NAME
Pathname to which the commands of the batch stream, error messages, and other listing output are listed.

PARTITION
Number of the partition in which the batch stream executes. When no number is entered, the partition in which the command is entered is used. In the interrupted mode the number of an idle nonterminal partition must be entered.

Both the batch stream and the listing file must be sequential files with 80 byte logical record lengths. When a batch stream is initiated in a nonterminal partition, the error messages for the batch stream are displayed on the error line of the initiating terminal. The partition number in which the batch stream is executing is displayed with the error message.

The LIST JDL and LIST NAME parameters interact to produce a variety of listing options. The result of these parameter specifications also depends upon whether the BATCH command is initiating batch processing in a partition or is nested in a batch stream. The following table summarizes the combinations of these conditions.

LIST JDL Parameter	LIST NAME Specified	Initiating Batch	Result
YES	yes	yes	List commands, error messages, and listing output to the specified pathname
YES	yes	no	Same as above
YES	no	yes	Illegal
YES	no	no	List commands, error messages, and listing output to the current pathname if one exists; otherwise no listing is generated
NO	yes	yes	List only error messages and listing output to the specified pathname
NO	yes	no	Same as above
NO	no	yes	No listing is generated
NO	no	no	List only error messages and listing output to the current pathname if one exists; otherwise no listing is generated
null	yes	yes	List only error messages and listing output to the specified pathname
null	yes	no	List commands, if currently listing commands, error messages and listing output to the specified pathname
null	no	yes	No listing is generated
null	no	no	List commands, if currently listing commands, error messages and listing output to the current pathname if one exists; otherwise no listing is generated

When a BATCH command is entered from a terminal partition, the condition code is automatically set in the destination partition. The condition code is set to the character T if the time of day has not been initialized; otherwise it is set to the character blank. It is recommended that initial batch streams contain the following record preceding all other records in the file:

```
T/BATCH, NAME = .TIME
```

The batch stream named .TIME is supplied with the system (see Appendix E). It will cause the operator to be prompted for the date and time automatically if the time has not been initialized.

Examples:

```
[ ] B
NAME: .BATCH.EDITOR
LIST JDL:
LIST NAME:
PARTITION:
```

The command executes batch stream .BATCH.EDITOR in the user's terminal partition.

```
[ ] BA
NAME: ACREC.BATCH.RUN1
LIST JDL: YES
LIST NAME: .COPY.SPOOL
PARTITION: 103
```

The command executes batch stream ACREC.BATCH.RUN1 in partition 103. The commands are listed in file .COPY.SPOOL.

```
[ ] B.
NAME: .SEDIT
```

The command executes the Screen Editor batch stream .SEDIT in the user's terminal partition. Notice that the form B. only prompts for the required parameter NAME.

CHAIN <Batch>

Function:

The CHAIN command is similar to the BATCH command but allows specification of the number of levels (including zero) to return through the batch stack. The user may control the level to which batch streams are nested by combination of BATCH and CHAIN commands. For example, a CHAIN command with a return count of zero starts a new batch stream at the same level as the initiating batch streams, i.e., no unnesting takes place.

Format:

```
NAME = acnm,
LIST JDL = [ YES ! NO ],
LIST NAME = [ acnm ],
RETURN COUNT = int ! 0
```

Where:

NAME
Pathname of the batch stream to be executed.

LIST JDL and LIST NAME arguments for the CHAIN command are treated exactly the same as the corresponding arguments of the BATCH command. When the RETURN COUNT parameter is nonzero and LIST JDL or LIST NAME is not specified, the "current" specification assumed for that parameter is the one in effect for the level of batch processing to which the CHAIN command returns.

RETURN COUNT
Integer specifying the number of levels to return before starting the new batch stream. A count of zero means to stay at the current level; a count of one means to return one level, etc. The currently active batch file is closed at each level before the new batch file is opened. Batch listing files defined for exited batch streams are closed. Therefore a CHAIN may specify the same batch stream that contained the CHAIN command. If a RETURN COUNT greater than the current nesting level is specified, the nesting count is set to the highest level, all batch stream files are closed and the CHAIN command is processed as if the exactly correct value had been specified.

An error within a CHAIN command is fatal and terminates the batch stream.

CHANGE <Terminal, Batch>

Function:

File attributes and privilege levels may be altered with the CHANGE command. Privilege levels operate here as elsewhere: access is permitted only if the user's privilege level is not less than the file privilege level and alterations in file privilege levels can only be effected up to the user's level. Manipulation of file attributes are permitted upon successful access. Use the CHANGE command to apply or remove either write or delete protection.

Format:

```
NAME: acnm
PRIVILEGE: [ int ]
DELETE PROTECTION: [ YES ! NO ]
WRITE PROTECTION: [ YES ! NO ]
WORK FILE: [ YES ! NO ]
AUTO FILE: [ YES ! NO ]
EXPANDABLE: [ YES ! NO ]
```

Where:

NAME
: Pathname of the file to have attributes changed.

PRIVILEGE
: New privilege level for the file. When no response is entered, the privilege level is not altered.

DELETE PROTECTION
: The delete protection for the file. A response of YES applies delete protection. A response of NO removes delete protection if the file is not write protected. An attempt to remove delete protection without removing write protection on a write protected file is ignored. When no response is entered, delete protection is not altered.

WRITE PROTECTION
: Write protection for the file. A response of YES applies write and delete protection. A response of NO removes write protection. When no response is entered, write protection is not altered.

WORK FILE
Specifies whether a file is a WORK file. When no response is entered for either WORK FILE or AUTO FILE, the type of the file is not altered. If YES is specified for WORK FILE, the file will be a WORK file regardless of the specification for the AUTO FILE parameter.

AUTO FILE
Specifies whether a file is an AUTO file. A response of YES causes the file to be made an AUTO file, and removes the WORK attribute. Program files and MDS files should not be made AUTO files.

EXPANDABLE
Specifies whether a file or directory is expandable. If a file is changed from nonexpandable to expandable the secondary allocation amount is set to the current file allocation. A response of YES allows expansion. A response of NO prohibits future expansion. When no response is entered, no change is made. If the file was created with the SINGLE file type attribute, this parameter is ignored.

Examples:

```
[ ] CHAN
NAME: GREEN.SOURCE.JOHN.PROGA
PRIVILEGE:
DELETE PROTECTION: YES
WRITE PROTECTION: YES
WORK FILE:
AUTO FILE:
EXPANDABLE:
```

The command applies both delete and write protection to file GREEN.SOURCE.JOHN.PROGA. Note that all other attributes of PROGA remain unaltered.

```
[ ] CHAN
NAME: VOL1.WHSE.INV
PRIVILEGE: 1000
DELETE PROTECTION: NO
WRITE PROTECTION:
WORK FILE:
AUTO FILE:
EXPANDABLE:
```

The command changes the privilege level of and removes delete protection from file VOL1.WHSE.INV.

```
[ ] CHANGE
NAME: GREEN.SOURCE.JOHN.PROGA
PRIVILEGE:
DELETE PROTECTION: NO
WRITE PROTECTION: NO
WORK FILE:
AUTO FILE:
EXPANDABLE: YES
```

The command removes both types of protection from file GREEN.SOURCE.JOHN.PROGA and allows the file to be expanded if required.

COBOL <Terminal, Batch>

NOTE

A batch JDL file with the pathname .COBOL has been supplied to invoke the COBOL compiler. See Appendix E for a description of this file.

Function:

RM/COS supports compilation of COBOL programs with this command. The compiler requires a large amount of memory for compiler tables, with larger source programs requiring a proportionally larger partition. Therefore, the user should take care to obtain sufficient memory in the partition before invoking the compiler.

The compiler uses the following logical names:

SI	Source code input file
LO	Listing file
BO	Object code output file
EM	Error message text file

Assign these logical names to the appropriate files. Then enter the COBOL command. The BO file must be assigned to a program file or DUMY unless the NOOBJ option is specified; otherwise a 9413 (Invalid Open) error will be generated. The error message text file is provided as part of the operating system with the pathname .CBLERRTX.

Format:

```
OPTIONS: [ ANSI ][ ,DEBUG ][ ,NOLIST ][ ,NONE][ ,NOOBJ ]
         [ ,OBJLIST ][ ,PROCX ][ ,RESEQUENCE ][ ,XREF ]

COPY BLOCK SIZE: int : 512
```

Where:

OPTIONS
Specifies compilation options to be in effect for the current invocation of the compiler. The available options are:

ANSI - Allow all four forms of the SIGN clause (LEADING, LEADING SEPARATE, TRAILING, and TRAILING SEPARATE) and make the default operational sign position trailing combined. If this option is omitted, allow only the TRAILING SEPARATE form of the SIGN clause and make the default operational sign position trailing separate.

DEBUG - Compile all lines having the character D in column seven. These lines are treated as equivalent to comment lines if the DEBUG option is not specified.

NOLIST - Compile with no source listing and no data allocation map. Only source lines in error and the program summary statistics are listed. Logical name LO must be assigned even when this option is specified.

NONE - No options are selected. Useful for batch mode processing. Specification of NONE along with other options has no effect and the other options listed are effective.

NOOBJ - Compile with no object output. Logical name BO need not be assigned when this option is specified.

OBJLIST - Print an object listing.

PROCX - Print a procedure-name cross reference listing (paragraph-names and section-names only). This option is redundant and ignored if the XREF option is also specified.

RESEQUENCE - Print the source or copy file relative line number in the sequence area of the source record listing. This option does not change the source or copy files, but provides a file relative line number on the program listing for reference when editing.

XREF - Print a full cross reference listing (alphabet-names, condition-names, data-names, file-names, index-names, mnemonic-names, paragraph-names, and section-names)

COPY BLOCK SIZE

The largest block size of a library text file referenced by a COPY statement using an RM/COS pathname for a text-name. COPY files may be nested to five level, and if nesting is used, this parameter is the largest sum of the block size plus buffer overhead plus file assignment overhead in any nesting of COPY statements (see Appendix C for details and examples of calculating the value for this parameter). A value of zero may be specified if no COPY statements are in the source program to be compiled. A value of zero will allow the compiler more dynamic table space in a nonterminal partition. A default value of 512 is used if no value is specified.

The COBOL command executes the COBOL compiler using the requested options. The condition code will be set to the value Z if there were compilation errors (not warnings).

NOTE

The PROCX and XREF options use portions of the memory space used to maintain compile time tables and will reduce the size of a program that may be compiled within a fixed partition size. The PROCX option uses considerably less space than the XREF option.

COPY Statement Text-Names

COPY statements in COBOL source programs reference library text to be copied by use of user-defined words called text-names. Under RM/COS a text-name identifies the particular text file to be copied by the following rules:

(1) If the text-name is one to eight characters in length, it is first treated as if it were an RM/COS logical file name. If the logical file name is assigned to a file, that file is identified as containing the library text to be copied.

(2) If the text-name is longer than eight characters in length or, if one to eight characters in length but is not a logical file name assigned to a file, it is treated as if it were an RM/COS pathname. An internally generated logical file name is assigned to the file indicated by the pathname and that file is identified as containing the library text to be copied. The assignment is released when the end of file is encountered while copying text.

If a user synonym table exists, synonym resolution is attempted prior to assigning a logical file name to the pathname as follows:

(a) If the first character of the text-name is a period, synonym resolution is not attempted. The text-name must be the pathname of a library text file existing on the system disk volume.

(b) If the first name in the pathname (i.e., the one to eight character name preceding the first period or the end of the text-name) matches a synonym, that name is replaced by the synonym value and the logical file name is assigned to the resulting pathname.

(c) If the first (or only) name in the pathname does not match any synonym and the synonym COPYLIB has a value, the logical file name is assigned to the pathname composed of the value of the synonym COPYLIB, a period, and the value of the text-name (i.e., the text-name is appended to the value of the synonym COPYLIB).

The recommended method of specifying text-names is to define the synonym COPYLIB with the pathname of the directory which contains the file names of library text and to use those file names as text-names (apply rule 2c). Using this method, the form of a text-name would be:

<filename>

The library text would be copied from @COPYLIB.<filename>.

If the library text files are not all contained within one directory, then synonyms other than COPYLIB should be defined with the pathnames of those other directories and text-names of the form

<synonym>.<filename>

should be used (apply rule 2b). Files may also be copied by use of the text-name form

<synonym>

where the value of the synonym is the complete pathname.

It is not necessary to use synonyms in text-names which are RM/COS pathnames, but the use of synonyms is highly recommended to avoid having to edit source programs whenever the disk directory structure is modified.

Rule 2c allows the compilation of source programs written for earlier versions of RM/COS, which required all text-names to be logical file names, without having to assign the logical file names or changing the source program. The synonym COPYLIB must be defined and the text-names must correspond to the file names in the directory indicated by the value of COPYLIB. Rule 2b may be used if the files are in several different directories; define each text-name as a separate synonym with the value of the complete pathname. Rule 1 allows such programs to be compiled by the same methods used before pathnames were allowed.

The above rules for text-names are applied in the order indicated. Thus a mixture of techniques may be used within a single source program. If a file is copied more than once into a source program, assigning a logical file name to that file for use as a text-name (apply rule 1) will result in faster compilation since the assignment process need be executed only once. In most other cases it is better is to use pathnames for text-names. The assignment process consumes memory in the user's partition for each file assigned (see Appendix C). Use of pathnames for text-names allows the compiler to reserve space for assigning only one library text file at a time, freeing more memory for compiler table space.

Example 1:

```
[ ] COB
OPTIONS: D,X
COPY BLOCK SIZE:
```

The command compiles the statements in the file assigned to logical name SI, including those statements marked for debug. The compiler writes the object code to the file assigned to logical name BO. The compiler writes the program listing and a cross reference listing to the file or device assigned to logical name LO.

Example 2:

```
[ ] COBOL
OPTIONS: NOOBJ
COPY BLOCK SIZE:
```

The command compiles the statements in the file assigned to logical name SI, and writes the program listing to the file or device assigned to logical name LO. Compiling with no object code output is useful when a listing is required for debugging purposes.

COMBINE <Terminal, Batch>

Function:

The COMBINE command allows the concatenation of several program files into one new file, thus allowing easier handling of a large set of programs and reducing the amount of memory structures necessary for assigning a large number of program files.

Format:

```
NEW PROGRAM FILE: lname
EXCLUDE PROGRAMS: [ name [ ,name ]... ]
```

Where:

NEW PROGRAM FILE
: Logical name of an assigned program file into which the combined contents of all other assigned program files are to be written. The new program file must be assigned with Exclusive All access. All the program files to be combined must be assigned with Read Only access.

EXCLUDE PROGRAMS
: A list of program names which are not to be included in the NEW PROGRAM FILE.

The contents of all program files assigned Read Only (i.e. the old program files) are copied to the new program file in the order in which the old program files were assigned. Thus if file A contains programs X, Y and Z, file B contains programs M, N and P, and file C contains programs X, R and M, and the files are assigned in the order A, B, C, then the result of a COMBINE which excludes program N is a new program file containing the programs X, Y, Z, M, P and R. Duplicate program names are eliminated by copying only the first occurrence of a program. In this example, programs X and M on file C are not copied.

A MAP-PROGRAMS command may be used just prior to the COMBINE command in order to verify which programs will be combined into the new program file.

The new program file logical name must be released and reassigned with Read Only access in order to EXECUTE or CALL programs in the new combined program file or to MAP-PROGRAM the new combined program file.

The following sequence would perform the previously described operation:

```
/ASSIGN, LOGICAL NAME = P1, NAME = .A, ACCESS = RO
/ASSIGN, LOGICAL NAME = P2, NAME = .B, ACCESS = RO
/ASSIGN, LOGICAL NAME = P3, NAME = .C, ACCESS = RO
/ASSIGN, LOGICAL NAME = NEW, NAME = .NEWPROG, ACCESS = EA
/COMBINE, NEW PROGRAM FILE = NEW, EXCLUDE PROGRAMS = N
/RELEASE
```

CONNECT <Terminal, Batch>

Function:

CONNECT establishes a logical communication link and associates a logical name with the link. Once established, the link may then be used by other communication JDL processors. The communication link established by the CONNECT command is disconnected by the RELEASE JDL command. The link may also be disconnected as a result of certain link conditions (e.g., telephone line going "on hook") and link errors (e.g., expiration of disconnect timer). In the case of an unsolicited disconnection (not a RELEASE disconnect), a RELEASE and reCONNECT are required to reestablish a 2780/3780 communication link.

Format:

```
LOGICAL NAME: lname
NAME: name
TYPE: [ USER  |   3780   |   2780  ]
            [ , EBCDIC | , ASCII ]
                      [ , LRC   | , CRC  ]
                      [ , EVEN  | , ODD  | , NONE ]
TIMEOUT: [ int ]
```

Where:

LOGICAL NAME
: Logical name which is to be assigned to the logical link subsequent to link establishment.

NAME
: Device name of a communication link device (i.e., BISYNC link or "BL" type device and user link or "UL" type device) to be used to access the physical link.

TYPE
: An indication of the type of logical link to be established. The available options are:

  USER - If a user type link ("UL" device) is to be CONNECTed, USER may be specified; and if not specified, USER is assumed.

  3780 - If a BISYNC type link ("BL" device) is to be CONNECTed, either 2780 or 3780 may be specified. If neither is specified, 3780 is assumed.

2780 - If a BISYNC type link ("BL" device) is to be CONNLCTed, either 2780 or 3780 may be specified. The 2780 option selects a restricted subset of the 3780 protocol and is recommended only when communicating with a device that does not support the 3780 protocol.

EBCDIC - If a BISYNC type link ("BL" device) is to be CONNECTed, either EBCDIC or ASCII may be specified. If neither is specified, EBCDIC is assumed. Selecting the EBCDIC option causes EBCDIC link control characters to be used, and indicates that the data characters sent and received in coded mode are in EBCDIC. During coded mode transmissions characters are translated from ASCII to EBCDIC, and during coded mode reception characters are translated from EBCDIC to ASCII.

ASCII - If a BISYNC type link ("BL" device) is to be CONNECTed, either EBCDIC or ASCII may be specified. Selecting the ASCII option causes ASCII link control characters to be used.

LRC - If a BISYNC type link ("BL" device) is to be CONNECTed and ASCII is specified, either LRC or CRC may be specified. If neither is specified, LRC is assumed. The LRC option causes the block check on coded mode transmissions to be a single character exclusive-or of the text characters in the block.

CRC - If a BISYNC type link ("BL" device) is to be CONNECTed and ASCII is specified, either LRC or CRC may be specified. The CRC option causes the block check on coded mode transmissions to be a two character cyclic redundancy check using the standard communications polynomial CRC-16.

EVEN - If a BISYNC type link ("BL" device) is to be CONNECTed and ASCII is specified, either EVEN, ODD, or NONE may be specified. If no option is selected, EVEN is assumed on asynchronous links. The EVEN option causes all link control characters and coded mode data characters to be sent and checked with even parity.

ODD - If a BISYNC type link ("BL" device) is to be CONNECTed and ASCII is specified, either EVEN, ODD, or NONE may be specified. If no option is selected, ODD is assumed on synchronous links. The ODD option causes all link control characters and coded mode data characters to be sent and checked with odd parity.

NONE - If a BISYNC type link ("BL" device) is to be CONNECTed and ASCII is specified, either EVEN, ODD, or NONE may be specified. The NONE option causes all link control characters and coded mode data characters to be sent and checked with space parity.

TIMEOUT
The number of seconds that the CONNECT command is allowed to wait for a BISYNC link to be established. If no value is specified, the value given to the disconnect timeout by the unit specification record in the System Definition File is assumed. A value of zero causes the command to wait indefinitely (over 3 weeks). The Interrupt Key may be used to stop execution of the CONNECT command; an EXIT command will abort the connect operation and leave the link in a disconnected state.

See the SEND, RECEIVE, and FTS command descriptions, the C$COMM subroutine description and Chapter 6, Data Communications, for additional information on the use of data communicatons.

Examples:

```
[ ] CONNECT
LOGICAL NAME: LINK
NAME: BL01
TYPE:
```

The command connects link device BL01 to logical name "LINK." The logical link will be maintained using the 3780 protocol.

```
[ ] CONN
LOGICAL NAME: L
NAME: BL02
TYPE: 2780
```

The command connects link device BL02 to logical name "L". The logical link will be maintained using the 2780 protocol.

```
[ ] CONNECT
LOGICAL NAME: DATALINK
NAME: UL01
TYPE:
```

The command connects link device UL01 to logical name "DATALINK". The logical link will be maintained by the programmer using the C$COMM COBOL callable subroutine.

CONTINUE <Terminal, Interrupt>

Function:

The CONTINUE command is used following execution of a HALT command when the nonterminal process is to continue execution. It is also used to restart execution of a program in the user's terminal partition which was interrupted by activating JDL with the Interrupt Key.

Format:

```
PARTITION: [ int ]
```

Where:

PARTITION
Partition in which a stopped program is to continue execution. If no partition number is entered or a partition number of zero is entered, the user's terminal partition is restarted.

Examples:

```
[ ] CONT
PARTITION: 101
```

The command reactivates the program in partition 101.

```
[ ] CONT.
```

The command reactivates a terminal partition process interrupted by the Interrupt Key.

CREATE <Terminal, Batch>

Function:

The CREATE command places the file name in the directory specified by the pathname and performs file management functions required for defining a new file. The user specifies the file organization and type, and the record size and block size for the file. The user also assigns the privilege level, and may apply delete protection. The CREATE command allocates space on the volume for the file as requested by the user. The command may also assign a logical name to the file.

Format:

```
LOGICAL NAME: [ lname ]
NAME: [ acnm ]
RECORD SIZE: int | 80
BLOCK SIZE: [ int ]
BUFFERS: [ int ]
ALLOCATION: int
SECONDARY: int | 0
ORGANIZATION: SEQ | REL | INX | PGM | MDS | MVD
PRIVILEGE: [ int ]
DELETE PROTECTION: YES | NO
TYPE: [ SCRATCH ] [ ,SPOOL ] [ ,COMPRESSED ]
      [ ,WORK ] [ ,AUTO ] [ ,SINGLE ]
```

Where:

LOGICAL NAME
: Logical name which is to be assigned to the file after the file is created. If no name is specified, a file is created but not assigned. If a logical name is assigned, the assignment is made with Exclusive All access.

NAME
: Pathname of the file to be created. If no name is specified, the system disk is assumed. A disk or volume name is only valid for scratch type files (See TYPE parameter). If the ORGANIZATION is specified as MVD, the pathname must consist of a volume or device name (possibly null), a period, and the file name. An MVD file must be a direct descendent of a volume directory.

RECORD SIZE

Number of characters specified for the record in the File Description entry of the COBOL program. Files opened OUTPUT from a COBOL program will override this specification with the actual logical record size of the file as defined by the COBOL program. For an MDS file, the value of this parameter is ignored and the record size is set equal to the block size.

BLOCK SIZE

Number of characters specified for the block in the File Description entry of the COBOL program. The disk sector size (e.g. 128, 256 or 512) is used when no size is entered. It is recommended that this value always be a multiple of the disk sector size (see Appendix D, File Size Calculations) to minimize wasted disk space. Relative organization file performance will improve if the block size is an exact multiple of the quantity record size plus two when only one buffer is allocated. For an MDS file, the block size is forced to a multiple of the disk sector size which is equal to or greater than the value specified for this parameter.

BUFFERS

Number of blocking buffers to be allocated for I/O to the file if a logical name parameter is also specified. Blocking buffers have a size equal to the block size specified in the BLOCK SIZE parameter plus overhead (see Appendix C, Memory Requirements). The program opening the file can override this parameter by specifying the number of buffers in the RESERVE integer AREAS clause in the COBOL program. If no RESERVE integer AREAS clause is specified and no number is entered the system allocates two buffers, except that only one buffer is allocated for a sequential or relative file opened INPUT. The system ensures that at least two buffers are allocated for indexed files not opened INPUT. The buffers are allocated when the file is opened and released when it is closed.

Unlike other parameters to the CREATE command, the BUFFERS option is not preserved on disk; it applies to the file only until the file is released, and is ignored if no logical name is specified.

ALLOCATION
Number of records to be allocated to the file initially. This value, combined with the record size and block size of the file and the sector size of the disk, is converted to a number of ADUs (see Appendix D, File Size Calculations).

SECONDARY
Number of additional records to be allocated when the initial allocation has been filled. This value, combined with the record size and block size of the file and the sector size of the disk, is converted to a number of ADUs. A response of zero creates a file that is not expandable. A nonzero value is ignored if SINGLE is specified as one of the TYPE parameters.

ORGANIZATION
Specifies the organization of the file as follows:

SEQ	Sequential
REL	Relative
INX	Indexed
PGM	Program
MDS	Multipartite Direct Secondary
MVD	Multivolume Sequential Disk

PRIVILEGE
Privilege level of the file. The maximum privilege level that is valid is the user's own privilege level. If no value is entered the user's privilege level applies.

DELETE PROTECTION
If the response is YES, the file is delete protected. The protection must be removed with a CHANGE command before the file may be deleted. If the response is NO, the file is unprotected.

TYPE
One or more type attributes of the file. The possible type attributes are WORK, AUTO, COMPRESSED, SPOOL, SCRATCH, and SINGLE.

WORK attribute means that on output the logical records are not written immediately to disk, but are deferred until the block buffer is full, the buffer is needed for an input, or the file is closed. Output to work files will be faster than to nonwork files, but some data may be lost if the system is stopped before the file is closed. Files may be converted from work to nonwork and vice versa. It is suggested the WORK attribute be used unless data integrity is important. Note

that the WORK attribute is unnecessary for program files, because these files are always treated as work files.

AUTO attribute indicates that the file will be open and updated over long periods of time and requires a high degree of data integrity while minimizing the necessity of data recovery in the event of a power failure or other fault. Output to AUTO files causes the system to simulate an open and close operation around the file modification operation, minimizing the time the file is in a state requiring data recovery; output operations on AUTO files require more time due to the extra physical disk operations involved. The AUTO and WORK attributes are mutually exclusive. AUTO should not be used with program or MDS files.

COMPRESSED attribute means that two or more consecutive blanks or binary zeroes are replaced by a single byte and three or more consecutive repeated characters are replaced by two bytes when the logical record is output. Logical records are decompressed to their original form when input. Relative and program files cannot have the compressed attribute selected and the REWRITE statement cannot be used with compressed sequential files. It is recommended that the COMPRESSED attribute be used with sequential and indexed files to minimize disk storage and reduce transfer time.

SPOOL attribute indicates that the file is to contain data to be printed and all carriage control information generated by the COBOL application with such statements as BEFORE or AFTER ADVANCING is preserved.

SCRATCH attribute indicates that the file is a temporary file. Scratch files are treated as work files. The only pathname allowed for scratch files is a disk device or volume name, since scratchs afilesk areicnotorenterede inamthe sdisk directory. A scratch file exists until the logical name associated with it is releaseed. Scratch MVD files are prohibited.

SINGLE attribute indicates that the file must be allocated as a single extent on the disk. The SINGLE attribute is used when a file must be contiguous, as is the case with certain system files (see INSTALL command). A file with this attribute cannot be expandable, so a nonzero SECONDARY allocation is ignored.

When no type attribute is entered, the file is not a temporary file; it makes no provision for carriage control information; blanks in the file are not compressed; the file is written in the immediate write mode; and the file may consist of more than one extent.

Relative, indexed, and non-WORK non-AUTO MVD organization files are initialized by writing binary zeros in the allocated disk space when the file is opened output and whenever the allocation is expanded.

Examples:

```
[ ] CR
LOGICAL NAME: INP
NAME: GREEN.SOURCE.JOHN.PROGA
RECORD SIZE: 80
BLOCK SIZE: 240
BUFFERS:
ALLOCATION: 500
SECONDARY: 100
ORGANIZATION: SEQ
PRIVILEGE: 1000
DELETE PROTECTION: NO
TYPE: C,W
```

The command creates file GREEN.SOURCE.JOHN.PROGA, a sequential file of 80 character records and 240 character blocks. Initially, 500 records are allocated, and secondary allocations of 100 records are requested when needed. The file is a work file with compressed records and a privilege level of 1000. It is not delete protected, and is assigned to logical name INP. Two blocking buffers are allocated unless specified differently by the program that uses the file.

```
[ ] CREATE
LOGICAL NAME:
NAME: .OBJECT.REPORT
RECORD SIZE 80
BLOCK SIZE: 512
BUFFERS:
ALLOCATION: 1
SECONDARY: 1
ORGANIZATION: PGM
PRIVILEGE:
DELETE PROTECTION: NO
TYPE:
```

The command creates a program file .OBJECT.REPORT in preparation for a COBOL compilation. Notice that it is not necessary to specify the record size because the file is opened output by the COBOL compiler thus redefining the record size. The block size of 512 was choosen to decrease the time to load the object program at program initiation or when it is CALLed.

```
[ ] CREATE
LOGICAL NAME: INV
NAME: VOL1.WHSE.INV
RECORD SIZE: 128
BLOCK SIZE:
BUFFERS:
ALLOCATION: 300
SECONDARY: 0
ORGANIZATION: R
PRIVILEGE:
DELETE PROTECTION: YES
TYPE: WORK
```

The command creates file VOL1.WHSE.INV, a relative record file of 128 character records. The block size is equal to the disk volume sector size. The only allocation is 300 records. The file is a work file with the privilege level equal to the user. It is delete protected and is assigned to logical name INV. Two blocking buffers are allocated unless specified differently by the program that uses the file.

```
[ ] CRE
LOGICAL NAME: TEMP
NAME: GREEN.SCR
RECORD SIZE: 80
BLOCK SIZE:
BUFFERS: 1
ALLOCATION: 50
SECONDARY: 50
ORGANIZATION: SEQ
PRIVILEGE: 200
DELETE PROTECTION: NO
TYPE:
```

The command creates file GREEN.SCR, a sequential file of 80 character records. The block size is equal to the disk volume sector size. Initially, 50 records are allocated, and secondary allocations of 50 records are requested if needed. The file has a privilege level of 200. It is not delete protected, and is assigned to logical name TEMP. One blocking buffer is to be allocated unless specified differently by the program that uses it.

DELETE <Terminal, Batch>

Function:

When a file or directory is no longer needed it may be deleted using the DELETE command. The DELETE command deletes a file or a directory. When the pathname entered for a DELETE command is that of a file, only that file is deleted. When a directory pathname is entered, the files in the directory are deleted along with the directory while displaying a list of the deleted files. Any directories within the deleted directory, and the files they contain, are also deleted. No file having the specified directory in its pathname (at that level) remains following execution of the command. The DELETE command must be used with care when a directory is deleted.

Format:

```
NAME: acnm
DIRECTORY: YES ! NO
```

Where:

NAME
Pathname of the file or directory to be deleted.

DIRECTORY
When YES is entered, the JDL processor allows deletion of a directory. When NO is entered, the processor verifies that the pathname is a file pathname. If not, an error is displayed.

When the pathname specified is that of a single file, if the file is delete or write protected, currently assigned, or if the privilege level is higher than that of the current user, then the file is not deleted and an error message is displayed.

When the pathname specified is that of a directory, the DELETE processor generates a listing of each file found in the directory or its subdirectories. If a file found in the directory structure being deleted is delete or write protected, currently assigned, or has a higher privilege level than the current user, then the file is not deleted and an informative message is written to the listing file. If a file in a subdirectory or directory cannot be deleted, then the directory or subdirectory cannot be deleted, and an informative message is written. If after attempting to delete all files and subdirectories, the specified directory cannot be deleted, an informative message is written and an error message is displayed. If a file which should have been deleted was not deleted, remove delete protection for the file with a CHANGE command, release any logical name

assignment with a RELEASE command, and reenter the DELETE command.

The Interrupt Key may be used to stop a delete command. An EXIT command will abort the delete sequence leaving all remaining files and directories unaffected.

Examples:

```
[ ] DE
NAME: GREEN.SOURCE.JOHN.PROGA
DIRECTORY: NO
```

The command deletes file GREEN.SOURCE.JOHN.PROGA.

```
[ ] DELE
NAME: GREEN.SOURCE
DIRECTORY: Y
```

The command deletes directory SOURCE on volume GREEN. This deletes all files included in that directory, all files included in all subdirectories included in that directory, and the directory itself; that is, all files and directories accessed through directory GREEN.SOURCE. Any file or directory whose pathname begins with GREEN.SOURCE is deleted.

Example Listing 1:

```
.MASTER.SOURCE.FILE
.MASTER.SOURCE.FILE2
   NOT DELETED. CODE=5603 DEL PROT
.MASTER.SOURCE
   NOT DELETED. CODE=5605 NOT EMPTY
.MASTER.FILE
   NOT DELETED. CODE=5603 DEL PROT
.MASTER.FILE2
.MASTER
   NOT DELETED. CODE=5605 NOT EMPTY
```

Example Listing 2:

```
.MASTER.SOURCE.FILE2
.MASTER.SOURCE
.MASTER.FILE
.MASTER
```

DIRECTORY <Terminal, Batch>

Function:

The DIRECTORY command creates a directory level in the structure of a volume. Only the volume directory is created when the volume is initialized. All other directories in a volume structure must be created with DIRECTORY commands. The directory in which a new directory is listed must have been created prior to the creation of the new directory. For example, if a new volume has just been initialized, and the volume is to contain source code and listings for programs written by several programmers, the first file created on the volume might have the pathname GREEN.SOURCE.JOHN.PROGA. Directory GREEN.SOURCE would be created first, then directory GREEN.SOURCE.JOHN. The file GREEN.SOURCE.JOHN.PROGA could then be created, and the source code could be written.

Format:

```
NAME: acnm
EST NUMBER OF FILES: int ; 5
EXPANDABLE: YES ; NO
DELETE PROTECTION: YES ; NO
```

Where:

NAME
pathname of the directory to be created.

EST NUMBER OF FILES
Estimated number of files and directories needed in the directory to be created. The actual allocation is calculated using the algorithm described in Appendix D.

EXPANDABLE
YES allows the directory to be expanded to contain more files and directories than the specified number. NO limits the size of the directory to no more than the initial allocation (not necessarily the EST NUMBER OF FILES specified).

DELETE PROTECTION
YES applies delete protection to the directory. NO creates the directory without delete protection. Delete protection on the directory does not prevent deletion of files in the directory; it merely prevents deletion of the directory node itself.

All directories in the pathname except the last directory must have been previously created. An error message is displayed and the command is not executed if a required directory does not exist, or if the desired directory already exists.

Example:

```
[] DI
NAME: GREEN.SOURCE
EST NUMBER OF FILES: 5
EXPANDABLE: YES
DELETE PROTECTION: NO
```

The command creates directory SOURCE on volume GREEN. If the disk is a double density floppy, the original size of the directory is actually five entries, each of which may be the name of a directory or file. The directory expands, however, when a DIRECTORY or CREATE command attempts to add a sixth directory or file name to directory SOURCE. The directory is not delete protected.

EDITOR <Terminal, Batch>

NOTE

A batch JDL file with the pathname .EDIT has been supplied to invoke the line editor. See Appendix E for a description of this file.

Function:

The EDITOR command is used to invoke the line editor described in Chapter 5.

Format:

This command has no prompts.

Example:

[ ] EDITOR

The line editor is activated.

EXECUTE <Terminal, Batch>

Function:

The EXECUTE command loads a program into the partition in which the command is entered and executes the program. The DEBUG option may be specified for a COBOL program.

When a COBOL program contains CALL statements, the program file that contains the main program may also contain the called programs. Other program files that contain called programs must be assigned to logical names. Program files are searched in the order of assignment to locate the program modules required.

Format:

```
PROGRAM NAME: name
OPTIONS: [ DEBUG | NONE ]
```

Where:

PROGRAM NAME
: Name specified in the PROGRAM-ID clause of the IDENTIFICATION DIVISION of the program to be loaded and executed. The program name is found by searching all program files the user has assigned with Read Only access, in the order in which the files were assigned. This is the same order in which the logical file names appear in the MAP-PROGRAMS command.

NOTE

The logical name used to assign a program file is not used by EXECUTE. The ASSIGN step is required to generate internal tables defining all programs contained within a given program file. The program-name from the PROGRAM-ID paragraph is used to invoke the execution of a program in the same way that it is used to CALL a program.

OPTIONS
: The word DEBUG applies to a COBOL program and specifies execution in the debug mode (see Appendix H). When the word is not entered execution is in the normal mode. The word NONE is useful for batch mode processing.

The specified program is loaded and executed in the partition in which the EXECUTE command is executing. An EXECUTE command in a batch stream executing in a nonterminal partition may execute a COBOL program in the nonterminal partition, but the program may not use ACCEPT or DISPLAY statements and the DEBUG option may not be used.

Examples:

```
[ ] EXE
PROGRAM NAME: ACCR
OPTIONS: D
```

The command loads the program ACCR and executes it in the debug mode.

```
[ ] EXEC
PROGRAM NAME: JRNL
OPTIONS:
```

The command loads program JRNL into the terminal partition and executes the program.

EXIT <Terminal, Batch, Interrupt>

Function:

As an interactive JDL command, the EXIT command applies only in the interrupt mode. It requests termination of whatever interactive mode process is executing in the user's terminal partition.

In a batch stream, the EXIT command terminates the batch stream. The batch stream automatically terminates at end of file; an EXIT command is only required when a termination prior to the end of file is required. If the batch stream containing the EXIT command was invoked by a BATCH command in a batch stream, the initiating batch stream is resumed.

Format:

This command has no prompts.

Example:

```
[ ] EXI
```

FCOPY <Terminal, Batch>

Function:

A file or directory may be copied to another file or directory by entering an FCOPY command. Also, a file may be copied to or from a character oriented device. The FCOPY command generates a listing which includes the parameters specified and, for each file copied, the name of the file and either the number of records copied or the error code of the error which prevented the copy.

Format:

```
SOURCE: acnm
DESTINATION: acnm
```

Where:

SOURCE
Pathname of a file or directory to be copied, or a device name.

DESTINATION
Pathname of a file or directory to which the copy is to be written, or a device name.

In order to use the FCOPY command, the destination file or files in the destination directory must already exist. When the files in a destination directory do not already exist, and it is desired that the destination files be created with characteristics identical to the source files, then the directory can be copied with the FILE-BACKUP and FILE-RESTORE commands. If it is desired to make a physical copy of an entire disk volume onto an identical disk, the VCOPY command can be used.

The FCOPY command copies a file by reading a record from the source file and writing the record to the destination file, until every record from the source file has been processed. When a relative or indexed file is copied, the records are read in ascending order by relative key or prime record key, respectively. The FCOPY command extends the size of the destination file as needed to contain the file being copied; however, it does not reduce the allocation when more than the required number of ADUs have been allocated. Because files are copied by reading and writing each record individually, the source and destination files need not have the same file organizations, record sizes, block sizes, or other characteristics, with minor exceptions. A program file may be copied only to another program file; a directory may be copied only to a device or another directory, although the destination directory may have a different level in the destination pathname than the level of the

source directory. If the destination file has a record size less than that of the source file, records longer than the destination record size are truncated to the destination record size.

When the source file is a directory, the destination must be a directory or a device. When the destination is also a directory, the FCOPY command copies each file in the source directory to the file with the same name in the destination directory. If no file exists in the destination directory with the same name, the file is not copied, and an error message is written to the listing file. Each subdirectory in the source directory is copied to the subdirectory of the same name in the destination directory by individually copying each file in the subdirectory. When the destination is a device, the FCOPY command copies each file in the source directory to the device. When the destination is a printer, the paper is forced to top of page before each file.

When the destination file is an indexed file, either the file must already be defined, or the source file must also be an indexed file. An indexed file is defined when a program first successfully executes an OPEN statement on the file, or when a KEY JDL command is entered for the file. If the destination file is not already defined and the source file is an indexed file, the FCOPY command will make the key specification of the destination file match that of the source file.

The FCOPY command copies from or to a nondisk device as if it were a sequential file when a device name is entered as a source or destination. Specifying a printer as a destination is the most useful case of a device as an FCOPY parameter. Entering a disk device name copies the volume directory; that is, the entire disk contents that is contained under the volume directory. System image and JDL processor files may be copied by FCOPY.

Examples:

```
[ ] FCO
SOURCE: GREEN.SOURCE.JOHN.PROGA
DESTINATION: BLUE.SOURCE.PROGA
```

The command copies a file from volume GREEN to volume BLUE.

```
[ ] FC
SOURCE: GREEN.SOURCE.JOHN
DESTINATION: GREEN.COBOL
```

The command copies directory GREEN.SOURCE.JOHN to directory GREEN.COBOL.

```
[ ] FCOPY
SOURCE: GREEN.LIST.PROGA
DESTINATION: LP01
```

This command prints the contents of file GREEN.LIST.PROGA on printer LP01. The characters in the records would be printed and lines would be single spaced but any embedded carriage control characters would also be printed.

Example listing:

```
SOURCE PATHNAME:       SINGLE.PROGRAM

DESTINATION PATHNAME: .PROGRAM

DIRECTORY (0): .PROGRAM
           .C$SCRDU  COPIED   (5 RECORDS)
           .PRINTS   COPIED   (7 RECORDS)
*****      .C$SCRDM  ASSIGNING DESTINATION   ERROR 5003 BAD PATH
           .CHARU    COPIED   (5 RECORDS)
*****      .CHARM    ASSIGNING DESTINATION   ERROR 5003 BAD PATH
*****      .TESTCHAR INVALID DESTINATION FILE ORGANIZATION   ERROR 6221
END OF DIRECTORY: .PROGRAM

     3 ERRORS
```

FDUMP <Terminal, Batch, Interrupt>

Function:

The FDUMP command displays the contents of a logical block of a file, a member of an MDS file, or a File Description Entry (FDE) of a directory file.

Format:

```
NAME: acnm
MEMBER-BLOCK: string ! int
OFFSET: [ hex ]
COUNT: [ hex ]
```

Where:

NAME
: Pathname of the file from which data is to be displayed.

MEMBER-BLOCK
: Either the member name of a member in an MDS file, the file name of a FDE in a directory file, or the block number of a logical block in a file which is to be displayed. A logical block may be specified as a decimal or hexadecimal (leading ">") integer. Block numbers begin with zero and must be less than the number of blocks which currently exist in the file.

OFFSET
: The byte offset from the beginning of the member, FDE, or logical block at which the display of data should begin. The offset must be even. If omitted, a value of zero is assumed.

COUNT
: The number of bytes of data to be displayed. If omitted, the count defaults to the size of the logical block, member, or FDE referenced by the MEMBER-BLOCK parameter.

The FDUMP command is a listing command which displays the contents of a logical block in a file, the contents of a member in an MDS file, or an FDE in a directory file. To print the contents, assign logical name LO to a printer device name.

The heading of the listing shows the pathname and member name or block number parameters entered with the command.

On detail lines of the listing, the first four digits are the hexadecimal byte offset of the leftmost data byte displayed on the line. The next eight groups of four digits each are the hexadecimal representations of sixteen bytes of data. Each of these eight groups is suffixed by an apostrophe (') to designate a relocatable address, or by a space to designate absolute data. The rightmost sixteen characters are the ASCII representations of the data bytes with nonprintable characters replaced by periods.

Examples:

```
[ ] FD
NAME: ACCOUNTS.RECVABLE.MWQINC
MEMBER-BLOCK:  2
OFFSET:
COUNT: 40
```

The command displays the contents of the first 64 (>40) bytes of the third block in the file ACCOUNTS.RECVABLE.MWQINC.

```
[ ] FDUMP
NAME: .JDLFILE
MEMBER-BLOCK: EXECUTE
OFFSET: 100
COUNT: 20
```

The command displays the contents of 32 (>20) bytes at offset 256 (>100) of the member named EXECUTE in .JDLFILE.

FILE-BACKUP <Terminal, Batch>

Function:

The FILE-BACKUP command produces a specially formatted sequential file which contains the data from a single file or all files and subdirectories in a directory. The specially formatted file is used as the input file for the FILE-RESTORE and FILE-VALIDATE JDL commands. In addition to archiving data, the backup and restore operations can be used to copy directories from one disk to another.

Format:

```
BACKUP FILE: lname ! acnm
SOURCE NAME: [ acnm ]
CUTOFF DATE: [ string ! TODAY ]
RECORD SIZE: [ int ]
BLOCK SIZE:  [ int ]
OPTIONS: [ AFTER ! BEFORE ] [ ,VALIDATE ] [ ,APPEND ]
         [ ,PREMOUNT ] [ ,NONE ]
```

Where:

BACKUP FILE
: The logical name or access name of the file upon which the backup data will be written. If this parameter is the correct format for a logical name and the logical file name exists in the partition, then the file associated with the logical name is used. Otherwise, this parameter is used as a pathname.

SOURCE NAME
: Pathname of a file or directory to be backed up. If a directory is specified, all files in the directory or its subdirectories will be included, unless the file is excluded by CUTOFF DATE or the backup process cannot obtain access to the file. FILE-BACKUP is denied access to files assigned to other partitions unless the file is assigned with Read Only access. FILE-BACKUP may read files assigned in the current partition regardless of access mode, skipping only the backup file itself, the JDL command listing file assigned to logical name LO, and the current batch listing file. If no parameter is specified for the SOURCE NAME, the volume name of the system disk is assumed.

CUTOFF DATE
: Used in conjunction with the AFTER and BEFORE options to determine file inclusion based on update time, i.e., the time the file was last modified. A date may be entered as a six digit

integer in the form YYMMDD or as a string in the form YY/MM/DD. The keyword TODAY may be entered to specify the current date. If no parameter is entered, all files are included.

RECORD SIZE

The record size to be used for the backup sequential file. If no value or a zero is entered and the backup file is a disk file, then the record size specified when the disk file was created will be used. The record size of the backup file must be at least 256. If the APPEND option is specified, the same record size as that used to first write the file must be specified.

BLOCK SIZE

The block size to be used for the backup sequential file. If no value or a zero is entered and the backup file is a disk file, then the block size specified when the disk file was created will be used. This parameter is primarily useful for tape backups, where there may be no default block size, and a large block size may be required for efficient usage of the tape. If the APPEND option is specified, the same block size as that used to first write the file must be specified.

OPTIONS

One or more file backup options. The allowed options are AFTER, BEFORE, VALIDATE, APPEND, PREMOUNT and NONE.

If AFTER or BEFORE is entered, the CUTOFF DATE specified will be used to determine which files are included in the backup. If AFTER is entered, only files modified on or after the CUTOFF DATE will be backed up. If BEFORE is entered, only files modified before the CUTOFF DATE will be backed up. If neither BEFORE nor AFTER is entered, AFTER is assumed.

If VALIDATE is specified, then after the backup operation is complete the backup file will be read to confirm that all data records are present and that all redundancy checks are correct. The validate operation is not a verify operation; the data in the backup file is not compared against the original disk files.

If APPEND is entered, the FILE-BACKUP command will extend the specified BACKUP FILE by adding another backup segment. The segment index assigned will be displayed in the listing generated by the validate operation, except when the VALIDATE

option is omitted.

If PREMOUNT is entered and the BACKUP FILE parameter specifies a tape device name, then the system will not display the mount message for the initial tape volume, but will instead assume that the volume is already mounted in the drive. After the tape has been written and validated, the unload message normally displayed by the system will also be suppressed. If the backup file spans tape volumes, the system will revert to the normal procedure of displaying a mount message for each volume and an unload message after processing is complete. This option facilitates backup batch streams that will run in the middle of the night, where the operator mounts the tape volume in the drive before the day shift ends.

FILE-BACKUP generates a listing of the JDL command prompts, the files backed up, any errors encountered, and a summary. The summary indicates whether the backup and validate aborted, completed with errors or completed successfully (no errors).

Examples:

```
[ ] FILE-BACKUP
BACKUP FILE: .GREEN.BACKUP
SOURCE NAME: .GREEN
CUTOFF DATE:
RECORD SIZE: 256
BLOCK SIZE:
OPTIONS: VALIDATE
```

The command backs up the entire .GREEN directory to the<--* sequential file .GREEN.BACKUP. Note that even though be skipped by FILE-BACKUP since it is the backup file. The record size is specified to be 256 characters, overriding the size specified when the backup file was created. This backup file can be used subsequently to restore all or part of the .GREEN directory to its state at the time of the backup.

```
[ ] FILE-B
BACKUP FILE: MT01
SOURCE NAME: WIN
CUTOFF DATE: 83/02/05
RECORD SIZE: 500
BLOCK SIZE:  4096
OPTIONS: VALIDATE, AFTER
```

The command requests that a tape be mounted in magnetic tape unit MT01, and backs up all of the files on the disk volume "WIN" (including operating system files, if any) that have

been updated on or after February 5, 1983. The backup file is written as the first file on the tape, with fixed length records of 500 characters each in blocks of at most 4096 characters. Incremental backup operations, i.e., backing up files modified after a specified date, may be used to avoid backing up entire disks or directories on a frequent basis. Note that the system need not be quiescent for this backup operation to complete; however, other users should not have files which reside on WIN assigned with access of EA, EW or SH, because the backup operation will be denied access to such files.

The following batch stream backs up the disk volume "WIN" to one or more floppy diskettes. As many diskettes as will be required must already be initialized with the INITIALIZE command, and the first diskette, "WINBACK", must already be LOADed.

```
/ASSIGN, LOGICAL NAME=LO, NAME=LP01
/CREATE, LOGICAL NAME=BACKUP, NAME=WINBACK.BACKUP,
         RECORD SIZE=1000, BLOCK SIZE=4096,
         ALLOCATION=100, SECONDARY=100,
         ORGANIZATION=MVD, TYPE=<WORK,COMPRESSED>
/FILE-BACKUP, BACKUP FILE=BACKUP, SOURCE NAME=WIN
/RELEASE,LOGICAL NAME=<LO, BACKUP>
```

The FILE-BACKUP command writes to the logical name BACKUP. When the first diskette is full, the system requests the operator insert another diskette in the disk drive, and a file named .BACKUP is created on the second diskette. This sequence continues until the disk volume WIN is completely processed. The operator is requested to mount the first diskette, WINBACK, and the FILE-BACKUP command validates the entire backup file, requesting that the second and subsequent diskettes be mounted. After the backup file is validated, the operator is again requested to mount the first diskette, WINBACK, and the FILE-BACKUP operation is then complete.

The following batch stream moves all the files in the directory .PROJECTX from WIN1 to WIN2:

```
/ASSIGN, LOGICAL NAME=LO, NAME=LP01
/CREATE, LOGICAL NAME=BACKUP, RECORD SIZE=510,
         ALLOCATION=100, SECONDARY=100,
         TYPE=<SCRATCH, COMPRESSED>
/FILE-BACKUP,  BACKUP FILE=BACKUP,
         SOURCE NAME=WIN1.PROJECTX
/FILE-RESTORE, BACKUP FILE=BACKUP,
         DESTINATION NAME=WIN2.PROJECTX
/RELEASE,LOGICAL NAME=<LO, BACKUP>
```

The directory WIN1.PROJECTX is first backed up to a scratch, compressed, sequential disk file on the system disk with the logical name BACKUP. This backup file is then used to restore all the files to the directory WIN2.PROJECTX.

Example listing:

```
  RM/COS  VERSION 2.4.01     SUNDAY, MAY 8, 1983  14:39:59

BACKUP LOGICAL FILE NAME: BACKUP
SOURCE NAME: WIN.DEV
CUTOFF DATE:
RECORD SIZE: 500
BLOCK SIZE:  4096
OPTIONS:     VALIDATE

LVL  FILE     ORG      LAST UPDATE          CREATION          # BACKUP
                                                              RECORDS

 1 DEV        DIR   83/04/28 15:26:00  83/04/27 19:03:22        1
 2 SYSDEFIL   SEQ   83/04/29 13:52:57  83/04/29 13:52:05        6
 2 SYSFILE    MDS   83/04/28 14:56:49  83/02/20 14:32:33      224
 2 JDLFILE    MDS   83/05/02 14:11:58  83/04/28 15:26:00      731
 2 JDLDEFIL   SEQ   83/04/04 19:05:48  83/02/06 16:54:05        6
 2 C$SUBS     PGM   83/04/04 17:40:49  83/02/06 16:54:36       28
 2 C$SUBTCH   SEQ   83/04/04 17:26:32  83/03/02 14:22:48       19
 2 CBLERRTX   REL   83/04/04 19:04:57  83/02/06 16:55:39       22
 2 S$EDITOR   PGM   83/04/04 19:05:09  83/02/06 16:56:05       51
 2 S$MODUSR   PGM   83/02/06 16:56:37  83/02/06 16:56:29       12

Begin Validation

BACKUP INDEX: 1
SOURCE NAME: WIN.DEV
TIME OF BACKUP: 83/05/08 14:39:59
BACKUP CUTOFF:
BACKUP OPTIONS: VALIDATE

 WIN.DEV
  WIN.DEV.SYSDEFIL
  WIN.DEV.SYSFILE
  WIN.DEV.JDLFILE
  WIN.DEV.JDLDEFIL
  WIN.DEV.C$SUBS
  WIN.DEV.C$SUBTCH
  WIN.DEV.CBLERRTX
  WIN.DEV.S$EDITOR
  WIN.DEV.S$MODUSR

End of file encountered

Backup completed successfully
Validate completed successfully
```

FILE-RESTORE <Terminal, Batch>

Function:

The FILE-RESTORE command recreates a file or files using as input the specially formatted sequential file produced by FILE-BACKUP. The backup and restore operations can be used to copy directories from one disk to another.

Format:

```
BACKUP FILE: lname ! acnm
BACKUP NAME: [ acnm ]
DESTINATION NAME: [ acnm ]
OPTIONS: [ ADD ] [ ,REPLACE ] [ ,TRUNCATE ] [ ,PREMOUNT ]
RECORD SIZE: [ int ]
BLOCK SIZE: [ int ]
INDEX: int ! 1
```

Where:

BACKUP FILE
The logical name or access name of the file from which the backup data will be read. If this parameter is the correct format for a logical name and the logical file name exists in the partition, then the file associated with the logical name is used. Otherwise, this parameter is used as a pathname.

BACKUP NAME
The complete or partial pathname of a directory or file to be restored from the backup file. The leading edges of this pathname must match those of the access name specified as the SOURCE NAME parameter of the FILE-BACKUP command which created the backup file, except that the volume name may be omitted. If no BACKUP NAME is specified, the value given the SOURCE NAME parameter of the FILE-BACKUP command which created the backup file will be assumed. Each file found in the backup file with a pathname which includes the BACKUP NAME will be restored. The pathname used to restore the file is generated by replacing the leading edges of the file pathname which match the BACKUP NAME with the DESTINATION NAME.

DESTINATION NAME
The access name describing the position in a disk structure to add or replace directories and/or files as contained in the backup file and described by the BACKUP NAME parameter. If no DESTINATION NAME is specified, the volume name of the current system disk is assumed. If the last

edge of the DESTINATION NAME is not present in the destination disk structure, it will be created to correspond to the file organization and characteristics of the item referenced by BACKUP NAME.

OPTIONS

One or more file restore options. The allowed options are ADD, REPLACE, TRUNCATE and PREMOUNT.

If the ADD option is omitted, when the pathname generated for a file from the backup file does not already exist, that file from the backup file is skipped and an informative message is written to the listing. When the pathname generated for a directory from the backup file does not already exists, FILE-RESTORE creates a directory with the generated pathname regardless of whether the ADD option is specified or omitted.

If the REPLACE option is omitted, when the pathname generated for a file from the backup file already exists, that file from the backup file is skipped and an informative message is written to the listing.

The TRUNCATE options affects only the restoring of files with organizations of SEQ, REL, PGM, INX or MDS. Normally when a file is restored, at least as much disk space is allocated to the new file as was allocated to the original file that was backed up. If the TRUNCATE option is specified, when a file with an organization of SEQ, REL, PGM, INX or MDS is restored, only enough space is allocated to contain the records currently in the file. For an INX organization file, this option is effective only if the file was created or opened OUTPUT on a release 2.4 or later system.

If PREMOUNT is entered and the BACKUP FILE parameter specifies a tape device name, then the system will not display the mount message for the initial tape volume, but will instead assume that the volume is already mounted in the drive. After the tape has been read, the unload message normally displayed by the system will also be suppressed. If the backup file spans tape volumes, the system will display a mount message for the second and subsequent volumes and an unload message after processing is complete. This option facilitates restore batch streams that will run in the middle of the night, where the operator mounts the tape volume in the drive before the day shift ends.

RECORD SIZE
The record size of the data contained in the backup file. This value must be identical to that specified to the FILE-BACKUP command which wrote the backup file. If the backup file is a disk file, this parameter may be omitted and the record size of the backup file will be used.

BLOCK SIZE
The block size used for the backup sequential file. This value must be identical to that specified to the FILE-BACKUP command which wrote the backup file. If the backup file is a disk file, this parameter may be omitted and the block size of the backup file will be used.

INDEX
The segment number of the backup file to be used for the restoration process. This index corresponds to the index assigned by the FILE-BACKUP command, and displayed in the listings generated by the FILE-BACKUP and FILE-VALIDATE commands. For a backup file which has not had multiple segments appended, this parameter should have a value of one.

All files included in the BACKUP NAME description will be restored unless the file already exists and cannot be deleted.

FILE-RESTORE generates a listing of the JDL command prompts, the options on the FILE-BACKUP, the files restored, any errors encountered, and a summary. The summary indicates whether the restore aborted, completed with errors or completed successfully (no errors).

Examples:

```
[ ] FILE-RESTORE
BACKUP FILE: .GREEN.BACKUP
BACKUP NAME:
DESTINATION NAME: .GREEN
OPTIONS: ADD,REPLACE,TRUNCATE
RECORD SIZE:
BLOCK SIZE:
INDEX: 1
```

The command restores the entire .GREEN directory from the sequential disk file .GREEN.BACKUP. Note that even though .GREEN.BACKUP is contained in the .GREEN directory, it was skipped by FILE-BACKUP since it is the backup file. After processing of this JDL command, the .GREEN directory contains all the files present at the time the FILE-BACKUP was performed, and in addition contains all files created

after the backup.

```
[ ] FILE-RESTORE
BACKUP FILE: .GREEN.BACKUP
BACKUP NAME: .GREEN.SOURCE.BLUE
DESTINATION NAME: .GREEN.SOURCE.OLDBLUE
OPTIONS: ADD
RECORD SIZE:
BLOCK SIZE:
INDEX: 1
```

The command creates, with the access name of .GREEN.SOURCE.OLDBLUE, and restores the contents of the file which had the pathname of .GREEN.SOURCE.BLUE at the time the FILE-BACKUP operation was performed.

```
[ ] FILE-R
BACKUP FILE: MT01
BACKUP NAME: WINI
DESTINATION NAME: WINI
OPTIONS: ADD
RECORD SIZE: 500
BLOCK SIZE: 4096
INDEX: 1
```

The command requests that a backup tape be mounted in magnetic tape unit MT01 and restores all of the files and directories backed up in the first file on the tape which do not already exist on the disk volume "WINI". All files which already exist on the disk volume remain unchanged. A record size of 500 and a block size of 4096 are specified to match the record size and block size specified when the backup file was written to tape.

```
[ ] FILE-R
BACKUP FILE: WINBACK.BACKUP
BACKUP NAME: WIN
DESTINATION NAME: WIN
OPTIONS: ADD,REPLACE,TRUNCATE
RECORD SIZE:
BLOCK SIZE:
INDEX: 1
```

The command restores the disk volume WIN from the MVD organization file WINBACK.BACKUP which resides on one or more floppy diskettes. After reading the data on the first diskette, the system requests the operator insert the second and subsequent diskettes. After all of the diskettes are read, the system requests that the operator insert the first diskette, WINBACK, and the restore operation is then complete.

Example listing:

```
  RM/COS  VERSION 2.4.01     SUNDAY, MAY 8, 1983  14:44:02

BACKUP LOGICAL FILE NAME: BACKUP
BACKUP NAME: WIN
DESTINATION NAME: .BILL
OPTIONS: ADD,REPLACE,TRUNCATE
RECORD SIZE: 500
BLOCK SIZE:  4096
INDEX: 1

BACKUP INDEX: 1
SOURCE NAME: WIN.DEV
TIME OF BACKUP: 83/05/08 14:39:59
BACKUP CUTOFF:
BACKUP OPTIONS: VALIDATE

 .BILL.DEV
  .BILL.DEV.SYSDEFIL
  .BILL.DEV.SYSFILE
  .BILL.DEV.JDLFILE
  .BILL.DEV.JDLDEFIL
  .BILL.DEV.C$SUBS
  .BILL.DEV.C$SUBTCH
  .BILL.DEV.CBLERRTX
  .BILL.DEV.S$EDITOR
  .BILL.DEV.S$MODUSR

Restore terminated by end of file

Restore completed successfully
```

FILE-VALIDATE <Terminal, Batch>

Function:

The FILE-VALIDATE command validates a sequential backup file produced by FILE-BACKUP. The backup file is read and checked for proper structure and data integrity. In addition, a listing is produced which shows the structure and contents of the backup file. The backup file is NOT verified against the original disk files used to create the backup. All segments of an appended backup file are validated and listed by this JDL command.

Format:

```
BACKUP FILE: lname ¦ acnm
RECORD SIZE: [ int ]
BLOCK SIZE:  [ int ]
OPTIONS: [ PREMOUNT ]
```

Where:

BACKUP FILE
: The logical name or access name of the file from which the backup data will be read. If this parameter is the correct format for a logical name and the logical file name exists in the partition, then the file associated with the logical file is used. Otherwise, this parameter is used as a pathname.

RECORD SIZE
: The record size of the data contained in the backup file. This value must be identical to that specified to the FILE-BACKUP command which wrote the backup file. If the backup file is a disk file, this parameter may be omitted and the record size of the backup file will be used.

BLOCK SIZE
: The block size used for the backup sequential file. This value must be identical to that specified to the FILE-BACKUP command which wrote the backup file. If the backup file is a disk file, this parameter may be omitted and the block size of the backup file will be used.

OPTIONS
: One or more file validate options. The only option allowed is PREMOUNT.

  If PREMOUNT is entered and the BACKUP FILE parameter specifies a tape device name, then the system will not display the mount message for the

initial tape volume, but will instead assume that the volume is already mounted in the drive. After the tape has been read, the unload message normally displayed by the system will also be suppressed. If the backup file spans tape volumes, the system will display a mount message for the second and subsequent volumes and an unload message after processing is complete. This option facilitates validation batch streams that will run in the middle of the night, where the operator mounts the tape volume in the drive before the day shift ends.

FILE-VALIDATE generates a listing of the JDL command prompts, the options on the FILE-BACKUP, the files restored, any errors encountered, and a summary. The summary indicates whether the validate aborted, completed with errors or completed successfully (no errors).

Examples:

```
[ ] FILE-VALIDATE
BACKUP FILE: .GREEN.BACKUP
RECORD SIZE:
BLOCK SIZE:
OPTIONS:
```

The command validates the backup file on .GREEN.BACKUP and produces a listing of its contents.

```
[ ] FILE-V
BACKUP FILE: MT01
RECORD SIZE: 500
BLOCK SIZE: 4096
OPTIONS:
```

The command requests that a backup tape be mounted in magnetic tape unit MT01 and reads and validates the backup information on the first file of the tape. A record size of 500 and block size of 4096 are specified, in order to match the record size and block size specified when the backup file was written.

Example listing:

```
  RM/COS  VERSION 2.4.01     SUNDAY, MAY 8, 1983  17:23:35

BACKUP LOGICAL FILE NAME: BACKUP
RECORD SIZE: 500
BLOCK SIZE:  4096
OPTIONS:

BACKUP INDEX: 1
SOURCE NAME: WIN.DEV
```

```
TIME OF BACKUP: 83/05/08 14:39:59
BACKUP CUTOFF:
BACKUP OPTIONS: VALIDATE

 WIN.DEV
  WIN.DEV.SYSDEFIL
  WIN.DEV.SYSFILE
  WIN.DEV.JDLFILE
  WIN.DEV.JDLDEFIL
  WIN.DEV.C$SUBS
  WIN.DEV.C$SUBTCH
  WIN.DEV.CBLERRTX
  WIN.DEV.S$EDITOR
  WIN.DEV.S$MODUSR

BACKUP INDEX: 2
SOURCE NAME: WIN.REL
TIME OF BACKUP: 83/05/08 15:01:10
BACKUP CUTOFF:
BACKUP OPTIONS: VALIDATE

 WIN.REL
  WIN.REL.SYSDEFIL
  WIN.REL.SYSFILE
  WIN.REL.JDLFILE

End of file encountered

Validate completed successfully
```

FLAG-PROGRAM <Terminal, Batch>

Function:

The FLAG-PROGRAM command sets and removes certain attributes which control the execution of COBOL programs. The following attributes are currently defined:

PERFORM	Allow nonstandard PERFORM invocation. This suppresses detection of the 4404 Illegal Procedure Nesting error (see disclaimer below).
DIVIDE	Force all divides to be done in decimal mode, thus eliminating the truncation of fractional digits when a COMP-1 data item is divided by a COMP-1 data item within an arithmetic expression.

Format:

```
NAME: acnm
PROGRAM: name
ON-FLAG: [ PERFORM ] [, DIVIDE ]
OFF-FLAG: [ PERFORM ] [, DIVIDE ]
CODE: [ int ]
```

Where:

NAME
: Pathname of the program file containing the program to be flagged.

PROGRAM
: The name of the program to be flagged.

ON-FLAG
: One or more of the flagged attributes described above.

OFF-FLAG
: One or more of the flagged attributes described above.

CODE
: An integer in the range 1-255.

At least one of the parameters ON-FLAG, OFF-FLAG and CODE must be nonnull. The same attribute may not be entered for both ON-FLAG and OFF-FLAG.

NOTE

Specification of the PERFORM attribute for the ON-FLAG parameter voids any warranty that may be in effect for RM/COS. Problems reported that are traced to the use of this attribute will be billed at current time and material rates.

Example:

```
[ ] FLAG
NAME: GREEN.PROGRAM.PAYROLL
PROGRAM: PRSUB02
ON-FLAG: PER
OFF-FLAG:
CODE:
```

The command flags the program PRSUB02 on file GREEN.PROGRAM.PAYROLL so that nonstandard PERFORM invocation is allowed.

Example:

```
[ ] FLAG
NAME: RMCOS.PROGRAM.ACNTRECV
PROGRAM: ACNTREC5
ON-FLAG: DIV
OFF-FLAG: PER
CODE:
```

The command flags the program ACNTREC5 on file RMCOS.PROGRAM.ACNTRECV so that all divides will be done in decimal mode and clears the flag that allows nonstandard PERFORM invocation.

FLAW-ADU <Terminal, Batch>

Function:

The FLAW-ADU command marks disk ADUs as flawed ("bad"). RM/COS will not allocate flawed ADUs to any file.

Format:

```
DEVICE: name
ADUS: int [ ,int ]...
```

Where:

DEVICE
: Volume name of the volume to be flawed or device name of the disk drive in which in the volume is presently LOADed.

ADUS
: One or more integers, each of which represents the number of an ADU to be flawed.

The disk volume on which the ADUs are to be flawed must have been INITIALIZEd by RM/COS version 2.4 or later. Volumes INITIALIZEd by earlier versions of RM/COS record bad ADU information in different manner.

ADUs may also be flawed with the FLAW-TRACK command. A list of the flaws may be obtained with the MAP-FLAWS command.

Example:

```
[] FLAW-ADU
DEVICE: GREEN
ADUS: 517
```

The command marks ADU number 517 on disk volume GREEN as flawed.

Example:

```
[] FLAW-A
DEVICE: DS03
ADUS: 123,777,778
```

The command flaws ADUs 123, 777, and 778 on the disk volume in disk drive DS01.

FLAW-TRACK <Terminal, Batch>

Function:

The FLAW-TRACK command marks disk ADUs as flawed ("bad"). All the ADUs covering a single disk track are flawed. RM/COS will not allocate flawed ADUs to any file.

Format:

```
DEVICE: name
CYLINDER: int
HEAD: int
```

Where:

DEVICE
: Volume name of the volume to be flawed or device name of the disk drive in which in the volume is presently LOADed.

CYLINDER
: Number of the disk cylinder on which the bad track is located.

HEAD
: Number of the disk head under which the bad track is located.

The physical address of a location on a disk is indicated by a three numbers: the cylinder number and head number, which identify a track, and a number which indicates the position on that track. FLAW-TRACK accepts a cylinder and head to identify an unusable track. All ADUs which are on that track are marked bad. Note that one or two of the ADUs so marked may be partially on the bad track and partially on an adjacent track.

The disk volume on which the ADUs are to be flawed must have been INITIALIZEd by RM/COS version 2.4 or later. Volumes INITIALIZEd by earlier versions of RM/COS record bad ADU information in different manner.

ADUs may also be flawed with the FLAW-ADU command. A list of the flaws may be obtained with the MAP-FLAWS command.

Example:

```
[ ] FLAW-TRACK
DEVICE:  GREEN
CYLINDER: 10
HEAD: 0
```

The command marks all ADUs on the disk track at cylinder 10, head 0 on disk volume GREEN as flawed.

Example:

```
[ ] FLAW-T
DEVICE:  DS03
CYLINDER: >87
HEAD: 5
```

The command flaws the ADUs on the track at cylinder 87 hexadecimal (135 decimal), head 5 on the disk volume in drive DS01.

FMODIFY <Terminal, Batch>

Function:

The FMODIFY command is used to make corrections to files, including the System Image File and JDL Image File. Modifications may be applied to a logical block of a file, to a member of an MDS file, or to a File Description Entry (FDE) of a directory file.

Format:

```
NAME: acnm
MEMBER-BLOCK: string ! int
OFFSET: hex
VERIFY: [ hex['] [ ,hex['] ]... ]
PATCH:  hex['] [ ,hex['] ]...
CHECKSUM: [ hex ]
```

Where:

NAME
: Pathname of the file to which the modification is to be applied.

MEMBER-BLOCK
: Either the block number of the logical block to be modified, the member name of a member in an MDS file to be modified, or the file name associated with an FDE in a directory file to be modified. A block number may be specified as a decimal or hexadecimal (leading ">") integer. Block numbers begin with zero and must be less than the number of blocks currently existing in the file.

  When the memory resident part of the system is modified in the System Image File, RM/COS must be reloaded into memory to effect the modification. See your Operator Guide for system load procedures.

OFFSET
: Byte offset in hexadecimal of the first word to be modified within the specified logical block, member, or FDE. This value must be even since only 16 bit words are modified. All logical blocks, members, and FDEs are addressed relative to zero. Note that the greater than symbol (>) is not allowed to precede the offset value.

VERIFY
One or more hexadecimal integers, each of which represents a word of data. The data is compared with the current words of the logical block or member beginning at the specified byte offset. Each word of data entered for members in the JDL Image File may be suffixed by an apostrophe (') to designate that data word as being relocatable. The resident system image is not relocatable. When no data is entered, no verification is performed. Note that the greater than symbol (>) is not allowed to precede verification values.

PATCH
One or more hexadecimal integers, each of which represents a word of data, which are to replace the current words of the logical block, member, or FDE beginning at the specified byte offset. Each value entered for members in the JDL Image File may be suffixed by an apostrophe (') to designate that value as being relocatable. The resident system image is not relocatable. Note that the greater than symbol (>) is not allowed to precede patch values.

CHECKSUM
Hexadecimal integer which is the 16 bit arithmetic sum of all the values entered in response to the PATCH prompt plus one for each relocation suffix. This value is compared to the value computed for the PATCH values. When no checksum value is entered, no checksum verification is performed.

The FMODIFY command writes the data supplied by the operator after optionally verifying the checksum and the previous contents. When the supplied checksum does not match the computed checksum, or when any of the verification data, including relocation attributes, do not match the corresponding contents in the logical block, member, or FDE an error message is displayed and the patch data are not written.

The FMODIFY command lists the verification data, if any, the current data, the checksum of the patch data, and the patch data.

The heading of the listing shows the pathname and member name or block number parameters entered with the command.

On the detail lines of the listing, the first four digits are the hexadecimal byte offset of the leftmost data byte displayed on the line. The next eight groups of four digits each are the hexadecimal representations of up to sixteen bytes of data. Each of these eight groups is suffixed by an

apostrophe (') to designate a relocatable address, or by a space to designate absolute data. The rightmost characters, enclosed by greater than and less than symbols (> <) are the ASCII representations of the data bytes with nonprintable characters replaced by periods.

Note that patches are applied to the appropriate disk image; members or logical blocks currently loaded into memory are not affected when the disk is patched. Since some JDL processes are not reloaded from disk for consecutive invocations, patches made on disk to JDL processors currently in memory will not necessarily be effective for all users of the patched process until an intervening JDL process is invoked or the system is rebooted (reIPLed). In the case of the resident system, RM/COS must be rebooted to effect the patch. In any case, it is recommended that the system be idle while patches are being applied.

Examples:

```
[ ] FM
NAME: .SYSFILE
MEMBER-BLOCK: SYSTEM
OFFSET: 80FE
VERIFY: 1309
PATCH: 1000
CHECKSUM:
```

The command verifies that the word at byte 80FE of the resident system module on the system disk contains the specified contents. The command then replaces that word on the disk with the specified new value and writes the listing.

```
[ ] FMODIFY
NAME: .JDLFILE
MEMBER-BLOCK: FCOPY
OFFSET: 408
VERIFY: C068,12,D811,44A'
PATCH:  C069,10,D811,45A'
CHECKSUM: 9CE5
```

The command computes the checksum for the specified patch data and compares it to the specified checksum. It then verifies that the four words at byte 408 of the JDL command FCOPY in the MDS file named .JDLFILE on the system disk contains the specified contents and that the last word is flagged as being relocatable. The command replaces those words on the disk with the specified data, flags the fourth word as being relocatable and the remaining words as being absolute, and writes the listing.

Example listing 1:

```
FILE: .JDLFILE   MEMBER: FCOPY

VERIFICATION DATA
0100:  1000                                          >..<

CURRENT DATA
0100:  1000                                          >..<

PATCH DATA     CHECKSUM=005A
0100:  005A                                          >.Z<
```

This listing shows that the verification and current data matched, so the patch was applied to the member.

Example listing 2:

```
FILE: RED.JDLFILE   BLOCK: 5

CURRENT DATA
0100:  005A                                          >.Z<

PATCH DATA     CHECKSUM=005A
0100:  005A                                          >.Z<

PATCH ALREADY APPLIED
```

This listing shows that the current data match the patch data, thus no changes are made to the logical block and no errors are indicated.

FTS <Terminal, Batch>

Function:

The FTS (File Transfer Server) command sends or receives a file via a logical link depending on the command used at the remote end of the link. If the SEND command is used at the remote end, then FTS acts the same as a RECEIVE command. If the RECEIVE command is used at the remote end, then FTS acts the same as a SEND command. The name of the file to be sent or received must be specified by the REMOTE NAME parameter of the corresponding RECEIVE or SEND command at the remote end.

Format:

```
LINK LOGICAL NAME: lname
MODE: [ CODED | IMAGE ]
```

Where:

LINK LOGICAL NAME
: Logical name of the communications link to be used for the file transfer operation. See the CONNECT command description for additional information regarding establishment of the communication link.

MODE
: The transmission mode of the data to be sent. See the SEND command for additional information about the transmission mode. If MODE is specified in the FTS command and in a RECEIVE command at the remote end, the two modes must be the same. Normally, only one end should specify the transmission mode when two RM/COS systems are communicating.

After sending or receiving one file, the FTS command processor terminates normally. If multiple files are to be sent or received, a JDL batch stream may contain an FTS command within a loop. This design allows conditional actions to occur by use of JDL, without the necessity of disconnecting the link via a RELEASE at the remote end.

See the CONNECT, SEND, and RECEIVE command descriptions and Chapter 6, Data Communications, for additional information on the use of data communications.

Examples:

```
[ ] FTS
LINK LOGICAL NAME: LINK3780
MODE: IMAGE
```

The command activates the file transfer server process on the link CONNECTed to LINK3780. If a RECEIVE command is executed on the remote end of the link, the file transfer server will send the file in IMAGE mode. If a SEND command is executed on the remote end of the link, the file transfer server will receive the file according to the mode in which it is sent.

```
[ ] FTS
LINK LOGICAL NAME: LINK1
MODE:
```

The command activates the file transfer server. The remote end of the link may SEND or RECEIVE a file, specifying both the file pathname and the transmission mode.

HALT <Terminal, Interrupt>

Function:

The HALT command interrupts execution of a COBOL program or a JDL command processor in a nonterminal partition. (Execution in a terminal partition may be interrupted by pressing the Interrupt Key at the terminal, activating JDL in the interrupt mode as soon as any outstanding I/O operations are complete.)

Format:

```
PARTITION: int
```

Where:

PARTITION
Number of a nonterminal partition.

The HALT command stops the execution of the program in the specified partition as soon as any outstanding I/O operation is complete. The program remains in the partition, and may be restarted at the point at which it was stopped by entering a CONTINUE command. The Interrupt Key can be used to stop execution of the HALT command; the EXIT command can then terminate the HALT command.

Example:

```
[ ] H
PARTITION: 103
```

The command stops the execution of the program executing in partition 103.

INITIALIZE <Terminal, Batch>

Function:

The INITIALIZE command initializes a disk volume to the format required by RM/COS file management. When a disk is reinitialized, the files on the disk cannot be accessed once the INITIALIZE command begins. Use the INITIALIZE command only on new disks or on previously used disks that do not contain current data.

Format:

```
VOLUME: name
DEVICE: name
BAD ADU NUMBERS: [ int [, int]... ]
EST NUMBER OF VCATALOG FILES: int | 5
EXPANDABLE: YES | NO
FORMAT TYPE: [ SINGLE | DOUBLE | MIXED | QUAD ]
BYTES PER SECTOR: [ int ]
INTERLEAVE FACTOR: [ int ]
```

Where:

VOLUME
: Volume name for the disk. The volume name specified cannot be a name that matches a device name. The names that are disallowed are all names comprised of four characters in which the first two characters match a device prefix (DS, ST, LP, etc.) and the last two characters are decimal digits, regardless of whether such a device is configured in the system being used. The volume name ME is also disallowed.

DEVICE
: Device name of the disk drive on which the disk to be initialized is mounted or will be mounted. If the disk drive does not contain a loaded volume (i.e, a successfully LOADed RM/COS format disk or the system disk), the INITIALIZE processor assumes that the proper disk is already mounted and begins initializing it immediately after the last command parameter is entered.

  If an RM/COS format disk volume is currently LOADed in the disk drive, the INITIALIZE processor will issue a message requesting that another volume be mounted and then waits for acknowledgement. Before acknowledging the message, the user must mount the proper disk volume in place of the loaded disk volume. After the initializing completes, the processor issues a message requesting the remounting of the loaded

disk volume and again waits for acknowledgement of the message. The INITIALIZE processor requires that the same volume as was previously loaded be remounted; the processor also ensures that no open files exist for the loaded volume and dedicates the disk drive to prevent files from being opened while initializing the other volume. Each of the messages are acknowledged by pressing the Acknowledge Key after exchanging the volumes in the drive.

A disk volume may be initialized on the system disk drive (assuming a removable system disk) when only one partition is active.

BAD ADU NUMBERS

List of integers indicating the known defective Allocatable Disk Units (ADUs) on the disk volume to be initialized. Defective areas on disk volumes are known from the manufacturer's surface analysis and user experience with the disk volume. Unusable ADUs on the index track (track 0) make the entire disk unusable.

The ADU number for a defective area of a disk volume may be calculated as follows:

```
ADU =  floor ((cylinder * heads + head)
             * sectors-per-track + sector
                                 - ADU-0-sector
                               / sectors-per-ADU
```

All of the above numbers are zero-origin; the first sector on the disk is at cylinder zero, head zero, sector zero. ADU-0-sector is the logical sector number of the first sector of ADU number zero. On MIXED format disks, this is (sectors-per-track + 1). On all other disks, this is five.

EST NUMBER OF VCATALOG FILES

An integer indicating the initial number of records to be allocated to the volume directory of the disk volume to be initialized. The actual allocation in ADUs is determined by using the algorithm described in Appendix D.

EXPANDABLE

YES allows volume directory to be expanded to contain more files and directories than the specified number. NO limits the size of the directory to the initial actual allocation (not necessarily the EST NUMBER OF VCATALOG FILES).

FORMAT TYPE

This parameter is system dependent: see Operator Guide, Chapter 3. If the FORMAT TYPE parameter value is omitted the system will use a default value based upon the type of disk. If a value is specified that is not compatible with the type of disk, an error message will be displayed.

BYTES PER SECTOR

This parameter is system dependent; see Operator Guide, Chapter 3. BYTES PER SECTOR is an integer specifying the sector size. If the value of FORMAT TYPE is MIXED, this value is the number of bytes on those sectors which do not reside on track 0. If omitted, a default value will be used which is dependent on the disk type and the value of the FORMAT TYPE parameter.

INTERLEAVE FACTOR

This parameter is system dependent; see Operator Guide, Chapter 3.

The command processor initializes the disk, writes the volume name in the disk label, and writes the numbers of bad ADUs in the bad ADU map on the disk. The INITIALIZE command should not be used to initialize a disk that contains any current data, because data on the disk is erased during initialization.

The INITIALIZE command does not cause the volume to be loaded. In order to load the newly initialized disk, the LOAD command must be entered.

Example:

```
[ ] INI
VOLUME: RED
DEVICE: DS02
BAD ADU NUMBERS: 100,101
EST NUMBER OF VCATALOG FILES: 5
EXPANDABLE: YES
FORMAT TYPE:
BYTES PER SECTOR:
INTERLEAVE FACTOR:
```

The command initializes the disk in drive DS02 as volume RED. ADUs 100 and 101 are marked as bad ADUs in the bad ADU map. The volume catalog may expand.

INSTALL-SYSTEM <Terminal, Batch>

Function:

The INSTALL-SYSTEM command establishes a new operating system on a disk volume. If a system already exists on the volume, it is first removed before installing the new system (see the REMOVE-SYSTEM command description). The volume need not be loaded in order to install the new system. The new system may be installed on the current system disk volume, in which case the operator should IPL following the installation in order to load the new system into memory.

CAUTION

It is best to have a working backup of the system disk before installing a new operating system on the system disk volume. Rather than installing the new system on the system disk volume, make a copy of the system disk volume and install the new system on the copy. Then test the new system disk by using it to IPL the system.

Format:

```
VOLUME: name
DEVICE: name
SYSTEM FILE NAME: acnm ; .SYSFILE
JDL FILE NAME: acnm ; .JDLFILE
SYSDEFIL FILE NAME: acnm ; .SYSDEFIL
SYSTEM NAME: name ; SYSTEM
BOOT NAME: name ; BOOT
```

Where:

VOLUME
: The volume name of the disk volume on which the new system is to be installed. The volume need not be loaded. The volume may be the current system disk volume when only one partition is active.

DEVICE
: The device name of the disk drive on which the volume is mounted or will be mounted. The system disk drive may be used to install a system on the current system disk volume or on another volume when only one partition is active.

SYSTEM FILE NAME
: The pathname of the System Image File; the pathname should begin with a period, not a device name or volume name, since the file is assumed to be on the volume specified by the VOLUME and DEVICE parameters. This is the MDS file which contains the system and boot images. The file must already exist on the volume. The System Image File must consist of a single extent on the disk (i.e., must be contiguously allocated); the SINGLE file type attribute (see the CREATE command description) may be used to force contiguous allocation.

JDL FILE NAME
: The pathname of the JDL Image File; the pathname should begin with a period, not a device name or volume name, since the file is assumed to be on the volume specified by the VOLUME and DEVICE parameters. This is the MDS file which contains the JDL processor images. The file must already exist on the volume. The JDL Image File must consist of a single extent on the disk (i.e., must be contiguously allocated); the SINGLE file type attribute (see the CREATE command description) may be used to force contiguous allocation.

SYSDEFIL FILE NAME
: The pathname of the sequential file which is to be the System Definition File; the pathname should begin with a period, not a device name or volume name, since the file is assumed to be on the volume specified by the VOLUME and DEVICE parameters. See Chapter 2, System Configuration, and your Operator Guide for further information about the System Definition File.

SYSTEM NAME
: The member name of the system image in the System Image File.

BOOT NAME
: This parameter is system dependent; see Operator Guide, Chapter 3. BOOT NAME is the member name of the boot image in the System Image File.

The action of installing a system on a disk volume involves declaring the pathnames of files and the member names of members required at IPL time. The declaration serves the purpose of locating the specific files and members for IPL. Also, the files are marked as system files to prevent their corruption by incorrect use of JDL commands which manipulate files.

The INSTALL-SYSTEM command need not change an entire system on the volume. For example, it is possible to change only the System Image File pathname by specifying the same names as the existing system on the volume for all prompts except the SYSTEM FILE NAME prompt.

Following the completion of an INSTALL-SYSTEM command, any files which were previously part of the system and are no longer part of the system are still on the disk volume. To recover the space allocated to those files, the operator must delete them; the files are still delete protected and so prior to deleting the files, the operator must use the CHANGE command to remove delete protection.

Examples:

```
[ ] INST
VOLUME:  RMCOSSYS
DEVICE: DS01
SYSTEM FILE NAME: .SYSFILE
JDL FILE NAME: .JDLFILE
SYSDEFIL FILE NAME: .SYSDEFIL
SYSTEM NAME: SYSTEM
BOOT NAME: BOOT
```

The command removes the existing system, if any, from volume RMCOSSYS in disk drive DS01 and installs the new system. All file pathnames and image member names are the default names. Note that if the system is being installed on the current system disk, the operator should IPL following completion of the INSTALL-SYSTEM command.

```
[ ] INSTALL-SYSTEM
VOLUME: SYSVOL
DEVICE: DS02
SYSTEM FILE NAME: .SYSFILES.SYSIMAGE
JDL FILE NAME: .SYSFILES.JDLIMAGE
SYSDEFIL FILE NAME: .SYSFILES.SYSCONFG
SYSTEM NAME: SYSV22
BOOT NAME: BOOTV22
```

The command removes the existing system, if any, from volume SYSVOL in disk drive DS02 and installs the new system. The operator has organized System Image File, JDL Image File, and the System Definition File within the directory .SYSFILES with different file names than the default values. The system and boot image member names have also been changed.

KEY <Terminal, Batch>

Function:

The KEY command describes the key parameters of an undefined indexed file. The keys of an indexed file must be described before the FCOPY command can be used to copy a sequential organization to an indexed organization file.

The parameters entered to the prompts of the KEY command must match the values expected by any COBOL program using the file. Note that the KEY command is not necessary if a file will be used first by a COBOL program with an OPEN OUTPUT or OPEN I-O statement.

Format:

```
LOGICAL NAME: lname
STARTING COLUMN:  int | 1 [ ,int ]...
KEY LENGTH: int [ ,int ]...
DUPLICATES ALLOWED: NO [ ,YES | NO ]...
```

Where:

LOGICAL NAME
: Logical name of an indexed file whose keys need to be defined. If the file is not an undefined file, i.e., it has been opened before, then an error message is generated if the key descriptions do not match those already on the file.

STARTING COLUMN
: One or more integers defining the first column of the fields for the prime and alternate record keys. The column of the prime key must be listed first and the columns of the alternate keys must be entered in ascending column value order, regardless of the order of the RECORD KEY clauses in the COBOL program. The maximum number of alternate keys is 14.

KEY LENGTH
: One or more values of the length of each key. There must be the same number of length values specified as starting column values. When more than one key length is entered, the lengths must be in the same order as the starting columns (i.e., a one to one correspondence between starting column and key length exists by order of specification).

DUPLICATES ALLOWED
One or more YES or NO values indicating whether duplicate key values are allowed for each key. Duplicate prime record keys are illegal. There must be the same number of YES or NO values specified as starting column values. When more than one key is entered, the duplicate key indicators must be in the same order as the starting columns (i.e., a one to one correspondence between starting column and duplicates allowed exists by order of specification).

Examples:

```
[ ] KEY
LOGICAL NAME: SIMPLE
STARTING COLUMN: 1
KEY LENGTH: 5
DUPLICATES ALLOWED: NO
```

This command defines a single key, the prime record key, for the file SIMPLE. This key begins in column 1 of the record, and is 5 characters wide.

```
[ ] KEY
LOGICAL NAME: MASTER
STARTING COLUMN: 5,1,9
KEY LENGTH: 6,5,2
DUPLICATES ALLOWED: NO,N,YE
```

This command defines the prime and two alternate record keys of the file MASTER. The prime record key begins in column 5 and is 6 characters wide. The first record key begins in column 1 and is 5 characters wide, overlapping the first character of the prime record key. The second alternate key begins in column 9 and is 2 characters wide. This key is totally contained in the prime record key, and does allow duplicate values to occur.

KILL-PARTITION <Terminal, Interrupt>

Function:

The KILL-PARTITION command forces termination of the program in a nonterminal partition or a terminal partition other than the one in which the KILL-PARTITION command is executed. To force termination of a program in your own terminal partition, press the Interrupt Key on the terminal, activating JDL in the interrupt mode and then enter an EXIT command.

Format:

```
PARTITION: int
```

Where:

PARTITION
Number of the partition in which the program to be terminated is executing.

The KILL-PARTITION command waits for the designated partition to reach an interruptable point, and then terminates the partition. The Interrupt Key can be used to stop execution of the KILL-PARTITION command; the EXIT command can then terminate the KILL-PARTITION command, allowing execution in the designated partition to continue.

The KILL-PARTITION command is identical to the KTASK command.

Example:

```
[ ] KILL
PARTITION: 5
```

The command forces termination of the currently executing program in terminal partition 5.

KPRINTER <Terminal, Batch, Interrupt>

Function:

The KPRINTER command terminates the current output on a printer, and terminates execution of any JDL command active to the device. Printing will resume after the logical name associated with the printer is closed and the printer is reopened. this command is useful to stop printing of an undesired report.

Format:

DEVICE: name | LP01

Where:

DEVICE
Device name of the printer whose current output is to be terminated.

Example:

```
[] KP
DEVICE: LP01
```

The command terminates the printing of the file currently being printed on LP01.

KTASK <Terminal, Interrupt>

Function:

The KTASK command forces termination of the program in a nonterminal partition or a terminal partition other than the one in which the KTASK command is executed. To force termination of a program in your own terminal partition, press the Interrupt Key on the terminal, activating JDL in the interrupt mode and then enter an EXIT command.

Format:

```
PARTITION: int
```

Where:

PARTITION
: Number of the partition in which the program to be terminated is executing.

The KTASK command waits for the designated partition to reach an interruptable point, and then terminates the partition. The Interrupt Key can be used to stop execution of the KTASK command; the EXIT command can then terminate the KTASK command, allowing execution in the designated partition to continue.

Example:

```
[] KT
PARTITION: 103
```

The command forces termination of the currently executing program in nonterminal partition 103.

LIST <Terminal, Batch, Interrupt>

Function:

The LIST command provides a list of the contents of a directory or volume, and shows the organization, type, and attributes of each directory and file contained within the structure. A LIST command may be also entered with a file pathname; the command then lists the organization, type, and attributes of that file only. A LIST command to a specific file is not allowed if the file has a logical name assigned. A LIST command of a volume or a directory allows assigned files.

Format:

```
NAME: [ acnm ]
LEVELS TO DISPLAY: [ int ]
DETAIL INFORMATION: YES | NO
```

Where:

NAME
: Pathname of a directory or file to be listed. The entire contents of the volume are listed when the pathname consists of a volume name only. If no name is specified, the system disk is assumed.

LEVELS TO DISPLAY
: The number of levels of files to be displayed. If no number is specified, all files below the starting pathname are specified. The number is only meaningful if the pathname specified by NAME is a directory or volume name.

DETAIL INFORMATION
: YES requests that the list show additional information about each file. NO limits the list to one line of basic information per file. Most of this information is primarily of interest to the system programmer maintaining the system.

For each file, the list shows:

The level of the file in the volume structure:

Level 0 is the volume directory, $$$$VDIR.

Level 1 includes files and directories listed in $$$$VDIR.

Level 2 includes files and directories listed in a level 1 directory.

Level 3 includes files and directories listed in a level 2 directory.

etc.

The level of a file is displayed when it is not the same as that of the file listed above it, or when it is a directory name.

The file or directory name.

The file organization:

VCT -- Volume directory
DIR -- Directory
SEQ -- Sequential
REL -- Relative
INX -- Indexed
PGM -- Program
MDS -- Multipartite direct secondary
MVD -- Multivolume sequential disk

The privilege level of the file.

The block size for the file.

The record size for the file.

The number of ADUs allocated to the file.

The type:

S -- Spool
C -- Blank compressed
W -- Work
A -- Auto

The protection in effect for the file:

D -- Delete protected
W -- Write protected
S -- System file

File status flags:

U -- File cataloged but undefined
R -- File needs to be RECLOSEd. Either the file was opened OUTPUT or I-O and needs to be closed, or a RENAME command on the file was begun but never completed.

Time of the last update.

Time that the file was created.

If DETAIL INFORMATION was requested, then for each file, the list shows:

The sector number of the sector containing the file's FDE (File Description Entry).

The number of discontiguous extents.

The secondary disk allocation (increment), in ADUs.

The disk structure level of the file.

The number of records in the file. This number will be zero if the file was written with a version of RM/COS prior to 2.3.

For each allocation extent the list shows:

The ADU number of the beginning of the extent.

The number of ADUs in the extent.

The sector number of the beginning of the extent.

The number of sectors in the extent.

If a directory was specified, the command processor writes a line that shows the total of the number of ADUs for the files that were listed.

After listing each file, additional information is displayed if an entire volume was listed:

The number of cataloged ADUs. This is the number of ADUs allocated to each file that was listed, plus the number of ADUs containing the index track and any ADUs marked bad when the volume was initialized.

The number of scratch ADUs. This is the difference between the number of ADUs allocated on the volume and the number of cataloged ADUs. This number represents disk space allocated to current users' scratch files, or space that was allocated but not deallocated because a secondary allocation failed or power was lost. On a multiuser system, this number is decreased if a file was created or extended after the command processor read the allocation map and before the file was listed; the number is increased if a file was deleted after the command processor read the allocation map. The number of scratch ADUs may appear negative if a file was created or extended, or if two files are allocated the same disk space (called dual allocation). The RECLOSE command may be used to recover disk space that was allocated but not deallocated, and to determine if any

files are dually allocated.

The number of ADUs available. This is the number of ADUs on the volume, less the number of cataloged and scratch ADUs.

Examples:

```
[ ] LI
NAME: GREEN
LEVELS TO DISPLAY:
DETAIL INFORMATION:
```

The command lists the files and directories on volume GREEN.

```
[ ] LI
NAME: GREEN
LEVELS TO DISPLAY: 1
DETAIL INFORMATION:
```

The command lists the top-level files and directories on volume GREEN.

```
[ ] LIST
NAME: GREEN.SOURCE.JOHN.PROGA
LEVELS TO DISPLAY:
DETAIL INFORMATION:
```

The command lists file GREEN.SOURCE.JOHN.PROGA

```
[ ] LIS
NAME: GREEN.SOURCE
LEVELS TO DISPLAY:
DETAIL INFORMATION:
```

The command lists the files and directories in directory GREEN.SOURCE, and the files and directories accessible through that directory.

```
[ ] LI.
```

The command lists the files and directories on the system disk.

LOAD <Terminal, Batch, Interrupt>

Function:

RM/COS disks that contain files processed by the file management portion of the system are called volumes. When a volume is physically mounted in a disk drive, the system does not recognize the volume until a LOAD command is executed. The LOAD command makes the files on the disk available to programs executing on the computer.

Format:

```
VOLUME: name
DEVICE: name
```

Where:

VOLUME
Volume name of the volume to be LOADed.

DEVICE
Device name of the disk drive in which the disk being LOADed is mounted.

The LOAD command reads the volume name on the disk, and displays an error message which includes the correct volume name if it does not match the one entered for the VOLUME parameter.

Duplicate VOLUME names cannot be loaded at the same time. If it is necessary to have two volumes with the same name accessible at the same time, the RENAME command may be used to temporarily change the name of one of the volumes so that the duplicate volume can be loaded.

Example:

```
[ ] LOA
VOLUME: GREEN
DEVICE: DS02
```

The command loads volume GREEN in drive DS02.

LOGOUT <Terminal>

Function:

When the user enters LOGOUT following the command prompt, the JDL processor clears the screen and the terminal is logged-out (i.e., becomes inactive). The command also releases all logical names assigned at the terminal. If any logical names are assigned to a tape file set that was modified, the listing generated by RELEASE is written to LO or the screen, depending on whether LO is released before or after the file set. When a batch stream initiated by the terminal (and not UNCOUPLEd) has not completed the system displays an error message, and the terminal is not logged-out.

Note that the LOGOUT command is identical to QUIT command except that LOGOUT is restricted to terminal mode.

Example:

```
[ ] LOGO
```

LOOP <Batch>

Function:

The LOOP command identifies the next command as the beginning of a set of commands that may be executed several times. The command performs no other function. The set of commands beginning with a LOOP command may end with a REPEAT command. Loops may not be nested, although a loop may be iterated by several REPEAT commands.

Format:

This command has no prompts.

MAP-FLAWS <Terminal, Batch, Interrupt>

Function:

The MAP-FLAWS command lists the locations of all flawed ("bad") ADUs on the specified disk volume.

Format:

```
DEVICE: name
```

Where:

DEVICE
: Volume name of the volume to be mapped or device name of the disk drive in which in the volume is presently LOADed.

For each known flaw on the specified disk volume the list shows the ADUs marked bad and the phsyical disk addresses corresponding to the ADUs. ADUs are in decimal. A physical address is given as a cylinder, head, and sector number, in decimal. The first sector on a disk is at cyliner zero, head zero, sector zero.

Examples:

```
[] MAP-FL
DEVICE: DS01
```

This command maps the flawed ADUs on the volume in disk drive DS01.

```
[] MAP-FLAWS
DEVICE: WIN
```

This command maps the flawed ADUs on the disk volume named WIN, as follows:

```
Volume:  WIN                Disk Flaw Map

  ADU Range          Cylinder/Head/Sector
00517 - 00517    0010/00/014 - 0010/00/015
```

MAP-KEYS <Terminal, Batch, Interrupt>

Function:

The MAP-KEYS command provides a list of the attributes of an indexed file or all indexed files assigned in a partition.

Format:

```
LOGICAL NAME: [ lname ]
DETAIL INFORMATION: YES : NO
```

Where:

LOGICAL NAME
: Logical name assigned to an indexed file for which key information is desired. If no logical name is entered in response to the LOGICAL NAME prompt, all indexed files assigned by the partition will be mapped.

DETAIL INFORMATION
: Indicates whether or not the file should be read to display the information described below about the data and index structure.

For each defined file the list shows:

- Record size
- Block size
- Record format (i.e. compressed/uncompressed)
- Number of keys.

For each key within a file the list shows:

- Starting column position
- Key length
- Key characteristics

If the response to the DETAIL INFORMATION prompt is YES, the following information is also displayed:

- Number of records
- Average number of records per data block
- Number of full and partial data blocks
- Number of index node blocks
- Number of empty blocks

For each key, the detail information also displays:

Number of levels in the index tree
Maximum number of keys in an index node block
Average number of keys in an index node block
For each tree level, the number of index node blocks that contain a given number of keys

Examples:

```
[ ] MAP-K
LOGICAL NAME: MASTER
DETAIL INFORMATION: YES
```

This command maps the indexed file assigned the logical file name MASTER.

```
[ ] MAP-KEYS
LOGICAL NAME:
DETAIL INFORMATION: NO
```

This command maps all indexed files assigned in the partition.

MAP-PROGRAMS <Terminal, Batch, Interrupt>

Function:

The MAP-PROGRAMS command lists the program files currently assigned within the partition and all program names contained within each file. The files are listed in the order that they were assigned. Program files assigned with Exclusive All access are not listed.

Format:

This command has no prompts.

For each file the list shows:

Logical name assigned to the file
Number of programs contained in the file

For each program within a file the list shows:

Program name
Length of the procedure area
Length of the data area
Total length
Count of arguments listed in the USING PHRASE of the PROCEDURE DIVISION
Date and time of compilation

Examples:

[ ] MAP-P

[ ] MAP-PROGRAMS

MAP-SYNONYMS <Terminal, Batch, Interrupt>

Function:

The MAP-SYNONYMS command lists the contents of the synonym table, its current size, and the space remaining.

Format:

This command has no prompts.

Examples:

[] MAP-S

[] MAP-SYNONYMS

MESSAGE <Terminal, Batch, Interrupt>

Function:

The MESSAGE command displays a message on the system line of a specified terminal, or of all terminals. When the message is displayed on a single terminal, a reply may be requested from that terminal. The reply is a single character that becomes the condition code character for the partition which executed the MESSAGE command.

Format:

```
TEXT: string
STATION: [ int ]
REPLY: YES ! NO
```

Where:

TEXT

String containing at most 55 characters if REPLY=NO, or at most 52 characters if REPLY=YES.

STATION

Number of the terminal on which the string is to be displayed. When no number is entered, the string is displayed on all other logged-in terminals. A station number of 0 displays the string on the terminal which initiated the batch stream containing the command. If the initiating terminal partition has been UNCOUPLEd from the partition executing the MESSAGE command, an error (7101) is generated.

REPLY

When YES is entered, the receiving terminal may respond with a single character reply. This reply sets the condition code value for the sending partition (the one executing the MESSAGE command). If no character is entered, the condition code is set to a blank. The response of YES is valid only when a single terminal is specified in response to the STATION prompt.

When NO is entered no response is required but either the Return Key or the Acknowledge Key must be pressed to clear the message.

The text is displayed on the system line of the specified terminal. The identifier of the sending terminal precedes the text, separated from it by a colon (:). When a reply is requested, it is indicated by a colon (:) following the message text.

A "broadcast" message is displayed one station at a time starting at the first defined station and proceding to the last (highest numbered) station skipping inactive stations and the sending station. As soon as the message is displayed on one station, it is displayed on the next station in the sequence, without waiting for the operator to acknowledge it. When the message is displayed at a station, it must be acknowledged by pressing the Return Key or the Acknowledge Key before any other terminal I/O operation (such as JDL input) can proceed.

In either the specific station or broadcast mode the message will be displayed immediately at the receiving station unless the receiving station is waiting for the user to acknowledge either an error message or a message from a prior MESSAGE command. In either of these cases, the message waits until the current message has been acknowledged before displaying the new message.

The sending station will not return to the JDL input mode until the message has been displayed (not acknowledged). The Interrupt Key may be used to interrupt the message process and perform any valid interrupt mode command. If the CONTINUE command is entered, the message process will continue to wait for the opportunity to display the message. If the EXIT command is entered, the message process will be aborted.

Examples:

```
[ ] ME
TEXT: "THE LINE PRINTER IS NOT AVAILABLE"
STATION:
REPLY: NO
```

The command displays the message at all logged-in stations other than the sending terminal.

```
[ ] MESS
TEXT: "YOUR REPORT IS NOW BEING PRINTED"
STATION: 3
REPLY: NO
```

The command displays the message on station 3.

PARTITION <Terminal, Batch, Interrupt>

Function:

The PARTITION command changes the size or priority (or both) of a partition. Increasing the size of a partition is required when:

a) The command processor for a JDL command that has been entered is too large for the space remaining in the partition;

b) A program that is being loaded into the partition is too large for the space remaining in the partition; or

c) File memory structures built by ASSIGN or CREATE require more memory than the space remaining in the partition.

Reducing the size of a partition to zero makes the partition inactive. If the partition is a terminal partition, the terminal also becomes inactive. When the size of a partition is reduced, the system will not reduce the size below a minimum value except when a size of zero is specified.

The size of the shared file partition must be increased when the number of files or devices assigned for shared access increases the requirements for shared file data structures beyond the current size of the partition. Additional memory for this partition is taken from the memory area between the shared file partition and the adjacent lower numbered partition.

The priority of a partition is an integer that is used by the scheduler in scheduling execution of a program in the partition. The scheduler allows a time slice to each active partition in turn, in accordance with the priority of the partition. The number of time slices that the scheduler skips the partition (between the time slices during which the program in the partition executes) is one less than the priority. Thus, the lower the priority number, the larger the program's share of processing time.

Format:

```
PARTITION: [ int ]
SIZE: [ int ]
PRIORITY: [ int ]
```

Where:

PARTITION
Number of the partition to be changed. When zero or no number is entered, the partition in which the command is executed applies.

SIZE
Number of bytes of memory allocated to the partition. If a number is not entered, the size of the partition is not changed.

PRIORITY
Priority of the partition. If a number is not entered, the priority is not changed.

Changing the size of a partition affects other partitions. For terminal and nonterminal partitions other than the shared file partition, additional memory requirements are supplied by using memory allocated to adjacent higher numbered idle partitions. These partitions must have sufficient memory for the expansion, or an error message is displayed and the size of the partition is unaltered. When the size of a partition is reduced, only the next higher numbered partition must be idle.

The shared file partition attempts to expand using the unused memory between the last nonterminal partition and the shared file partition. (This amount is indicated by the available memory given in the STATUS command output.)

Except for the user's partition, a partition must be idle to have its size changed.

Changing the priority of a partition affects the execution of the program executing in the partition. Entering a number lower than the current priority gives the program a larger share of the total processing time. Increasing the number reduces the program's share of processing time.

Examples:

```
[] PAR
PARTITION:
SIZE: 20000
PRIORITY: 1
```

The command changes the size and priority of the partition of the user's terminal. The requested size, 20,000 bytes, is relatively large. The priority, 1, is the highest priority.

```
[ ] PART
PARTITION: 104
SIZE: 10000
PRIORITY:
```

The command changes the size of partition 104 to 10,000 bytes. This size is adequate for most JDL commands and for small and medium sized programs.

PRINT <Terminal, Batch>

Function:

The PRINT command prints one or more files in accordance with the carriage control information contained in the file. When the pathname of a file is entered for a PRINT command, that file is printed. When the pathname of a directory is entered for a PRINT command, all files of type spool in the directory and in the structure defined by the directory are printed.

Format:

```
NAME: [ acnm ]
DELETE AFTER PRINT: YES ! NO
```

Where:

NAME
: Pathname of a file or directory. If no name is specified, the system disk is assumed.

DELETE AFTER PRINT
: When YES is entered, the file is deleted after it has been printed unless it is delete protected. When NO is entered, the file is not deleted.

When the pathname is a file pathname, the PRINT command processor prints the contents of the file. If it is a spool file, the carriage control information controls the print operation. If not a spool file, the contents are printed on single spaced lines.

When the pathname is a directory pathname, the spool files in the directory are printed, using the carriage control information. Files in the directory that are not spool files are not printed. If the DELETE AFTER PRINT option was selected, each file in the directory is deleted after it is printed. The directory is not deleted.

The PRINT command uses logical name LO to access the printer. If logical name LO is not assigned, the file contents are displayed to the terminal.

Examples:

```
[] PR
NAME: GREEN.REPORT
DELETE AFTER PRINT: YES
```

The command prints all spool files in directory GREEN.REPORT and deletes the files after they are printed.

```
[ ] PRI
NAME: GREEN.SOURCE.JOHN.PROGA
DELETE AFTER PRINT: NO
```

The command prints the contents of file GREEN.SOURCE.JOHN.PROGA even if it is not type spool. The file is not deleted.

QUIT <Terminal, Batch>

Function:

Terminal Mode

When the user enters QUIT following the command prompt, the JDL processor clears the screen and the terminal is logged-off (i.e., becomes inactive). The command also releases all logical names assigned at the terminal. If any logical names are assigned to a tape file set that was modified, the listing generated by RELEASE is written to LO or the screen, depending on whether LO is released before or after the file set. When a batch stream initiated by the terminal (and not UNCOUPLEd) has not completed the system displays an error message, and the terminal is not logged-off.

Example:

```
[ ] Q
```

Batch Mode

In batch streams, the QUIT command allows a parameter:

LOGOFF = YES ! NO

The QUIT command terminates the batch stream and all higher level batch streams. If there are no higher level batch streams, the EXIT and QUIT commands operate identically. If the QUIT command is in a nonterminal batch stream, the partition is made idle.

In a terminal partition, if the LOGOFF option was omitted, or not selected, interactive mode processing is resumed. If YES was selected for the LOGOFF option, the user is logged off as if the QUIT command were entered by the user.

RECEIVE <Terminal, Batch>

Function:

The RECEIVE command receives a file via a logical link replacing the contents of a local file with the data received. A local file of the pathname indicated by LOCAL NAME in the RECEIVE command or by REMOTE NAME in the corresponding SEND command must exist at the time the RECEIVE command is processed.

Format:

```
LINK LOGICAL NAME: lname
LOCAL NAME: [ lname | acnm ]
REMOTE NAME: [ lname | acnm [, lname | acnm ] ... ]
MODE: [ CODED | IMAGE ]
```

Where:

LINK LOGICAL NAME
Logical name of the communication link to be used for the file transfer operation. See the CONNECT command description for additional information regarding establishment of the communication link.

LOCAL NAME
Logical name or access name of the file whose contents are to be replaced by the contents of the file received. If this parameter is the correct format for a logical name and the logical file name exists in the partition, then the file associated with the logical name is used. Otherwise, this parameter is used as an access name. The LOCAL NAME may be omitted if and only if the remote end specifies a REMOTE NAME in the corresponding SEND command. If the remote end used an FTS command or is not an RM/COS system, then the LOCAL NAME must be specified in the RECEIVE command.

REMOTE NAME
The logical name(s) and/or access name(s) of the file or files at the remote end of the link which are to be concatenated in the order indicated and sent across the link. If the file specification is the correct format for a logical name and the logical file name exists in the SENDing or FTSing job's partition, then the file associated with the logical name is used. Otherwise, this parameter is used as an access name. If more than one file is specified, none of the files may be program or MDS organization. The REMOTE NAME may be omitted if and only if the remote end specifies a LOCAL

NAME in the SEND command or is not an RM/COS system. If the remote end used an FTS command or a SEND command without a LOCAL NAME then the REMOTE NAME must be specified in the RECEIVE command. If the remote end of the link is not an RM/COS system, then the REMOTE NAME should be omitted in the RECEIVE command.

MODE

The transmission mode of the data to be sent. In the RECEIVE command, the MODE is ignored unless a REMOTE NAME is also provided. If the remote end of the link used an FTS or SEND command which specifies MODE, then the RECEIVE command must omit MODE or specify the same mode. Normally, only one end should specify the transmission mode when two RM/COS systems are communicating. See the SEND command description for additional information on the transmission mode.

When receiving into a program file, the sending file must also be a program file or a relative file that contains a copy of a program file. An MDS file, however, may be transmitted and need not be received into an MDS file; an MDS file may be received into a sequential file and later copied to an MDS file.

See the SEND, CONNECT, and FTS command descriptions and Chapter 6, Data Communications, for additional information on the use of data communications.

Examples:

```
[ ] RECE
LINK LOGICAL NAME: DATALINK
LOCAL NAME: LP01
REMOTE NAME:
MODE:
```

The command receives a file over the link CONNECTed to DATALINK and prints it on the line printer.

```
[ ] RECEIVE
LINK LOGICAL NAME: LINK3780
LOCAL NAME: .PROGS.INVOICE
REMOTE NAME: .PROGS.INVOICE
MODE: IMAGE
```

The command receives a file from a remote processor using the link CONNECTed to LINK3780. The remote processor is an RM/COS system which has executed an FTS or SEND command without a LOCAL NAME. The file received replaces the contents of file .PROGS.INVOICE. The RECEIVE command requests that the file of the same name at the remote end of

the link be sent in image mode. If the local file is a program file, then the remote file must also be a program file.

```
[ ] RECEI
LINK LOGICAL NAME: LINK
LOCAL NAME: .CMDFILE
REMOTE NAME:
MODE:
```

The command receives a file from the remote system using the link CONNECTed to LINK. The file RECEIVEd replaces the contents of file .CMDFILE. Note that if the data RECEIVEd consists of JDL commands, then file .CMDFILE may be processed as a batch stream of commands. This capability allows the RM/COS user to "program" a cooperating remote RM/COS system via the SEND/RECEIVE file transfer mechanism.

RECLOSE <Terminal, Batch>

Function:

The RECLOSE command restores the integrity of disk structures which were not completely modified because power was lost or a disk volume was removed improperly. It is designed for the following situations:

1. A file was opened for I-O or OUTPUT and never closed. The RECLOSE command restores the structure of the file, except for a sequential WORK file, to make all records written to disk accessible.

2. A RENAME command was moving a file from one directory to another. The RECLOSE command ensures that the file appears in only one directory.

3. ADUs are allocated but are not part of any file. This may be a result of SCRATCH files never being released, a secondary allocation not completing successfully, or a DELETE command not completing the deallocation of disk space. The RECLOSE command rebuilds the allocation map to match the allocated files.

4. An ADU is allocated to more than one file. RECLOSE enumerates the ADUs in this condition, and to which files each ADU belongs.

Format:

NAME: [acnm]

Where:

NAME

Pathname of a file, directory, or disk volume. If a file name is specified, the file must be in a state which requires reclosing. A file without the AUTO attribute requires reclosing if it was open I-O or OUTPUT when the system was stopped; for a file with the AUTO attribute, a modification operation must have been in process when the system was stopped. If a directory is specified, all files which require reclosing in the directory are processed. If a disk volume is specified, RECLOSE performs actions 2 through 4 above in addition to processing all files which require reclosing; no user may have files assigned to a volume being reclosed. If no name is specified, the system volume is assumed.

If a non-WORK file requires reclosing, RECLOSE recovers all records except possibly the record actually being modified when the system stopped. For sequential and relative files, this recovery consists of locating the last record in the file. For indexed files, the index structure of each key is rebuilt from the contents of the data records. The records are added to the index in the order they are found instead of the order they were originally written. This may cause the order of records with the same alternate key value to change. Building new index structures may cause an indexed file to require more disk space than previously allocated. Therefore it may not be possible to recover an indexed file which occupies an entire disk volume.

If a WORK file requires reclosing, RECLOSE recovers whatever records it can locate on the disk. For sequential files, no recovery is possible. For relative files, records lying entirely within a block written to disk are correct; records which span blocks may be partially modified. For indexed files, records which were being rewritten may appear twice; one of the instances of the record will be deleted.

The RECLOSE command generates a listing which identifies the file, directory, or volume being processed, and the actions performed. The messages, in the order they may be written, and their meanings, are as follows:

RENAME pathname
: A RENAME command was in progress on the specified pathname when power was lost. The file will be made accessible with the pathname given.

DELETE pathname
: A RENAME command was in progress on the specified pathname when power was lost. The file is available under a pathname specified in a preceding RENAME message. The pathname on the DELETE message is removed.

ADU nnnnn IS ALLOCATED MORE THAN ONCE.
: The specified ADU is allocated to more than one file, to both a file and the system image, or is allocated and was marked bad when the volume was initialized. The Dual Allocation List will indicate which files are affected. Dual allocation is normally a result of the operator replacing a disk volume with another disk volume without entering UNLOAD and LOAD commands. Ryan-McFarland Corporation should be notified of any instances of dual allocation that cannot be attributed to operator error.

NEW ALLOCATION MAP CREATED. nnnnn ADUS RECOVERED.
: This message states how many ADUs were previously

allocated but did not belong to any file. These ADUs are now available for use. This number may be negative if dual allocation previously occurred and one of the files involved was deleted without the allocation map being rebuilt immediately afterwards.

ADU nnnnn pathname
The specified ADU, identified in a previous message as being allocated more than once, belongs to the specified pathname. The records of the questionable file should be copied to a new file, the old pathname deleted, and the RECLOSE command entered again to build a valid bit map. If the data in the file was indeed destroyed, this copy procedure may hang when the specified ADU is reached. In this case, the old pathname must be deleted, the RECLOSE command entered again to build a valid bit map, and the data in the file must be recreated.

DUAL ALLOCATION LIST COMPLETE.
This identifies the end of the list of files whose contents may be invalid as a result of dual allocation.

RECLOSE pathname
The specified file required reclosing. The data records are recovered, as described above.

Examples:

```
[ ] RECLOSE
NAME: SINGLE
```

The entire disk volume named SINGLE is recovered.

```
[ ] RECL.
```

The entire system disk volume is recovered.

```
[ ] RECL
NAME: GREEN.SOURCE.JOHN.PROGA
```

The single file GREEN.SOURCE.JOHN.PROGA is recovered.

Example listing:

```
RECLOSE OF: SINGLE   VOLUME NAME: SINGLE
NEW ALLOCATION MAP CREATED.       0 ADUS RECOVERED.
RECLOSE   SINGLE.TEMP
```

RELEASE <Terminal, Batch>

Function:

The RELEASE command disassociates the device or file that is assigned to a logical name from that logical name. A RELEASE command must be executed for a previously assigned logical name before reassigning that logical name to a different device or file.

Format:

LOGICAL NAME: [lname [,lname]...]

Where:

LOGICAL NAME
List of logical names to be released. Note that RELEASE merely disassociates the file or device from the logical name assigned to it. The file itself continues to exist, unless it is a disk file of type SCRATCH (see the CREATE command description).

When no logical name is entered, the command processor releases all logical names assigned in the partition.

An attempt to RELEASE a logical name that does not exist does not generate an error message if the RELEASE command is being executed within a batch stream.

The RELEASE of a logical name associated with a printer device causes the printer to advance to top of form if the forms are not so positioned, unless the printer device was assigned with Shared access.

The RELEASE of a logical name associated with a logical link causes the link to be disconnected.

The RELEASE of the last logical name associated with a tape file set will cause any tape volume of the file set that is loaded to be unloaded. If any file of the file set was written or modified, the RELEASE command writes to the listing file sufficient information for the volumes of the file set to be correctly specified to a later TAPE-ASSIGN command. If logical name LO was also specified on the LOGICAL NAME parameter, or if all logical names are being released, then the RELEASE command will write the listing information to the screen if LO was released before the tape file set; otherwise, the listing information is written to LO, and the logical name LO is not released. (Examples of the listing output may be found in the description of the TAPE-ASSIGN command.)

Examples:

```
[ ] REL
LOGICAL NAME: OUT
```

The command releases the assignment of the device or file associated with logical name OUT.

```
[ ] RELEASE   or  [ ] REL.
LOGICAL NAME:
```

The command releases all logical name assignments for the terminal.

REMOVE-SYSTEM <Terminal, Batch>

Function:

The REMOVE-SYSTEM command removes the operating system from a disk volume. The volume may no longer be used as a system disk volume. The system files are not deleted, but are marked as nonsystem files and nonwrite protected. After the REMOVE-SYSTEM command is used, the CHANGE command may be used to remove delete protection and then the DELETE command may be used to delete the files.

Format:

```
VOLUME: name
DEVICE: name
```

Where:

VOLUME NAME
The volume name of the disk volume from which the existing system is to be removed. The volume need not be loaded. The volume may not be the current system disk volume.

DEVICE:
The device name of the disk drive on which the volume is mounted or will be mounted. The system disk drive may be used to remove a system from a volume other than the current system disk volume.

Example:

```
[ ] REMOVE-SYS
VOLUME: RMCOSSYS
DEVICE: DS01
```

The command removes the system from the disk volume RMCOSSYS. The volume need not be loaded or mounted at the time this command is executed. If the volume is not in drive DS01, a message will be displayed requesting the mounting of the volume. After the operator mounts the volume, the system removal from the volume may be continued by pressing the Acknowledge Key. After the system removal process completes, the operator will be prompted to remount the volume previously loaded in DS01, if any; after remounting the loaded volume, the operator again presses the Acknowledge Key.

RENAME <Terminal, Batch>

Function:

The RENAME command is used to change the name of a file, directory or volume. The new file or directory must be unique in the directory in which it is contained.

Format:

```
OLD: acnm
NEW: acnm
```

Where:

OLD
: Pathname of a file or directory to be renamed, the name of a currently loaded volume to be renamed, or the device name of a disk drive containing a volume to be renamed. Since no verification takes place on unloaded volumes, only loaded volumes should be renamed whenever possible.

NEW
: New pathname for the file or directory, or the new name for a volume. A new volume name must not be the same as any currently loaded volume.

A directory name at any level may be changed. If the new pathname is that of an existing file or directory, no change occurs and an error message is displayed.

Example:

```
[ ] REN
OLD: GREEN.SOURCE.JOHN.PROGA
NEW: GREEN.SOURCE.JOHN.ACCR
```

The command changes the file name portion of pathname GREEN.SOURCE.JOHN.PROGA to ACCR.

```
[ ] RENAM
OLD: GREEN.SOURCE.JOHN
NEW: GREEN.SOURCE.COBOL
```

The command changes directory pathname GREEN.SOURCE.JOHN to GREEN.SOURCE.COBOL. The pathname for the file in the first example is now GREEN.SOURCE.COBOL.ACCR.

```
[ ] RENAME
OLD: GREEN.PROGB
NEW: GREEN.SOURCE.JOHN.PROGB
```

The command changes the pathname GREEN.PROGB to GREEN.SOURCE.JOHN.PROGB, effectively removing PROGB from volume directory GREEN and placing it in level 2 directory JOHN. Directories SOURCE and JOHN must have previously been created.

```
[] REN
OLD: DS02
NEW: NEWSYS
```

The command changes the volume in disk drive DS02 to have a volume name of NEWSYS.

REPEAT <Batch>

Function:

The REPEAT command terminates a set of commands that may be executed several times. The REPEAT command causes the JDL processor to go back to the command following the most recently executed LOOP command preceding the REPEAT command and execute commands in sequence. To exit from the loop, the condition code must be set to a character that will prevent execution of the REPEAT command. That is, the loop is repeated as long as the executing condition of the REPEAT command is met.

Format:

This command has no prompts.

REPLACE <Terminal, Batch>

Function:

The REPLACE command replaces an assigned cataloged disk file with a scratch file. The scratch file is cataloged under the name of the assigned cataloged file with the same privilege level and delete protection. After the command executes, the scratch file has the logical name that was assigned to the cataloged file. The previous version of the cataloged file is deleted from the disk. The System Image File and the JDL Image File cannot be replaced. An example of the use of this command is found in the batch stream .SEDIT in Appendix E.

Format:

```
SCRATCH NAME: lname
LOGICAL NAME: lname
```

Where:

SCRATCH NAME
Logical name of the scratch file. This file must not be an undefined file.

LOGICAL NAME
Logical name which is assigned to a cataloged disk file with Exclusive All access. This file must reside on the same disk volume as the scratch file, and must not be write protected. If this name is assigned to one of the system files, then the scratch file must have the same organization, record size, and block size as the system file.

REPOINT <Batch>

Function:

The REPOINT command provides a mechanism to control the contents of the batch listing file. The REPOINT command causes the current batch listing file to be repositioned to the beginning. The REPOINT command itself will not be included in the newly repositioned listing file. This command is useful if very long and complicated batch streams are generated, but a large listing file is not desirable.

Format:

This command has no prompts.

SCRATCH <Terminal, Batch>

Function:

The SCRATCH command creates a scratch disk file on the same disk volume and with the same default attributes as those of an already assigned file. The user may selectively override attributes of the existing file with those of his own choosing. This command is used in conjunction with the REPLACE command to edit files without the danger of a system failure while the file is being written (see the batch stream .SEDIT in Appendix E).

Format:

```
LOGICAL NAME: lname
TEMPLATE NAME: lname
RECORD SIZE: [ int ]
BLOCK SIZE: [ int ]
BUFFERS: [ int ]
ALLOCATION: [ int ]
SECONDARY: [ int ]
ORGANIZATION: [ SEQ | REL | INX | PGM | MDS ]
TYPE: [   WORK | NOT-WORK ]   [ , AUTO | NOT-AUTO ]
      [ , COMPRESSED | NOT-COMPRESSED ]
      [ , SPOOL | NOT-SPOOL ] [ , SINGLE | NOT-SINGLE ]
```

Where:

LOGICAL NAME
: Logical name which is to be assigned to the new scratch file after the file is created.

TEMPLATE NAME
: Logical name of the file whose attributes are to be copied.

RECORD SIZE
: Number of characters specified for the record in the File Description entry of the COBOL program. If no record size is entered, the record length of the template file is used. Files opened output from a COBOL program will override this specification with the actual logical record size of the file as defined by the COBOL program. For an MDS file, the value of this parameter is ignored and the record size is set equal to the block size.

BLOCK SIZE

Number of characters specified for the block in the File Description entry of the COBOL program. If no block size is entered, the block size of the template file is used. For an MDS file, the block size is forced to a multiple of the disk sector size which is equal to or greater than the value of this parameter or the template file block size.

BUFFERS

Number of blocking buffers to be allocated for I/O to the file. Blocking buffers have a size equal to the block size specified in the previous parameter plus overhead (see Appendix C, Memory Requirements). The program opening the file can override this parameter by specifying the number of buffers in the RESERVE integer AREAS clause in the COBOL program. If no number is entered, the value entered when the template file was assigned is used. If no RESERVE integer AREAS clause is specified and no number is entered for the scratch file nor for the template file, the system allocates two buffers, except that only one buffer is allocated for a sequential or relative file opened INPUT. The system ensures that at least two buffers are allocated for indexed files not opened INPUT. The buffers are allocated when the file is opened and released when it is closed.

ALLOCATION

Number of records to be allocated to the file initially. This value, combined with the record size and block size of the file and the sector size of the disk, is converted to a number of ADUs (see Appendix D). If no number is entered, the file is allocated as many ADUs as are allocated to the template file.

SECONDARY

Number of additional records to be allocated when the initial allocation has been filled. This value, combined with the record size and block size of the file and the sector size of the disk, is converted to a number of ADUs. A response of zero creates a file that is not expandable. If no number is entered, the secondary allocation amount is the same number of ADUs as that of the template file. A nonzero value is ignored if SINGLE is specified or implied as one of the file type attributes.

ORGANIZATION

Specifies the organization of the file as follows:

SEQ	Sequential
REL	Relative
INX	Indexed
PGM	Program
MDS	Multipartite Direct Secondary

When no organization is entered, the file organization of the template file is used.

TYPE

One or more type attributes to be overridden. If no parameters are entered, the file has the same attributes as that of the template file, except that it is a SCRATCH file. The allowed responses are WORK, NOT-WORK, COMPRESSED, NOT-COMPRESSED, SPOOL, NOT-SPOOL, SINGLE, NOT-SINGLE, AUTO, and NOT-AUTO.

The WORK attribute means that on output the logical records are not written immediately to disk, but are deferred until the block buffer is full, the buffer is needed for an input, or the file is closed. Output to work files will be faster than to nonwork files, but some data may be lost if the system is stopped before the file is closed.

The AUTO attribute indicates that the file will be open and updated over long periods of time and requires a high degree of data integrity while minimizing the necessity of data recovery in the event of a power failure or other fault. Output to AUTO files causes the system to simulate an open and close operation around the file modification operation, minimizing the time the file is in a state requiring data recovery; output operations on AUTO files require more time due to the extra physical disk operations involved.

The AUTO and WORK attributes are mutually exclusive. The AUTO and WORK attributes cannot affect data integrity because scratch files are discarded during the disk reclose process; instead these attributes are only meaningful if the scratch file being defined later becomes a cataloged disk file by the use of the REPLACE command. If the AUTO attribute is specified for a scratch file, output operations to the file will require more time as described above. The presence or absence of the WORK attribute has no affect on a scratch file. The recommended

strategy for the specification of these attributes is to specify NOT-AUTO on the SCRATCH command; after the scratch file becomes a cataloged disk file by use of the REPLACE command, the CHANGE command may be used to give the file the desired attributes.

COMPRESSED attribute means that two or more consecutive blanks or binary zeroes are replaced by a single byte and three or more consecutive repeated characters are replaced by two bytes when the logical record is output. Compressed logical records are decompressed to their original form when input. Relative, program and MDS files cannot have the compressed attribute selected and this attribute, if specified or implied from the template file, is ignored for those organizations. The REWRITE statement cannot be used with compressed sequential files. It is recommended that the COMPRESSED attribute be used with sequential and indexed files to minimize disk storage and reduce transfer time.

SPOOL attribute indicates that the file is to contain data to be printed and all carriage control information generated by the COBOL application with such statements as BEFORE or AFTER ADVANCING is preserved. Only sequential files may be SPOOL type files and this attribute, if specified or implied from the template file, is ignored for other file organizations.

SINGLE attribute indicates that the file must be allocated as a single extent on the disk. The SINGLE attribute is used when a file must be contiguous, as is the case with certain system files (see INSTALL command). A file with this attribute cannot be expandable, so a nonzero SECONDARY allocation is ignored.

Examples:

```
[ ] SCR.
LOGICAL NAME: OUTPUT
TEMPLATE NAME: INPUT
```

The command creates a scratch file with the same characteristics as the file assigned by logical name INPUT. The logical name OUTPUT is assigned to the created scratch file. If logical name OUTPUT is released, the scratch file will be deleted. The REPLACE command may be used to preserve the file.

```
[ ] SCRATCH
LOGICAL NAME: NEWFILE
TEMPLATE NAME: OLDFILE
RECORD SIZE:
BLOCK SIZE:
BUFFERS:  4
ALLOCATION: 500
SECONDARY: 50
ORGANIZATION: REL
TYPE: NOT-WORK
```

The command creates a scratch file assigned by the logical name NEWFILE with the same characteristics as the file assigned by the logical name OLDFILE, except that the organization is forced to be relative, the primary disk allocation is forced to 500 records, the secondary disk allocation is forced to 50 records, and if the REPLACE command is used to convert it to a permanent file, it will be a nonwork file (i.e., immediate write). If the template file is COMPRESSED or SPOOL, these attributes will be ignored since they are not valid for a relative file. Four block buffers will be allocated when NEWFILE is opened.

SDUMP <Terminal, Batch, Interrupt>

Function:

The SDUMP command is used to display the contents of a sector of the disk.

Format:

```
DEVICE: name ! DSO1
SECTOR: int
NUMBER: int ! 1
```

Where:

DEVICE
Device name of the disk drive in which the disk to be displayed is mounted or volume name is disk has been loaded.

SECTOR
Number of the sector to be displayed, specified as a decimal or hexadecimal (leading ">") integer limited to the maximum of sectors-on-disk and 4294967295.

NUMBER
Number of sectors to be displayed.

The heading of the display shows the sector number (in hexadecimal) and the device name: both the absolute sector number and the sector number relative to the start of the dump are given, with the latter in parentheses. On each detail line of the display, the first four digits are the hexadecimal byte address within the sector of the leftmost data byte on the line: the next eight groups of four digits are the hexadecimal representations of 16 bytes of data: the rightmost eight pairs of characters are the printable ASCII characters that correspond to the data with nonprintable characters replaced by periods.

Examples:

```
[] SD.
DEVICE: DSO1
SECTOR: 6
```

The command displays the contents of sector six of the disk mounted in device DSO1.

```
[ ] SDUMP
DEVICE: DS02
SECTOR: 34
NUMBER: 2
```

The command displays the contents of sectors 34 and 35 of the disk mounted in device DS02.

SEND <Terminal, Batch>

Function:

SEND sends a file or set of files via a logical link replacing the contents of the remote file with the data sent. Two types of file contents are recognized by the SEND command, coded ASCII (e.g., text data) and image (e.g., program files).

Format:

```
LINK LOGICAL NAME: lname
LOCAL NAME: [ lname | acnm [, lname | acnm ] ... ]
REMOTE NAME: [ lname | acnm ]
MODE: [ CODED | IMAGE ]
```

Where:

LINK LOGICAL NAME
Logical name of the communication link to be used for the file transfer operation. See the CONNECT command description for additional information regarding establishment of the communication link.

LOCAL NAME
Logical name(s) and/or access name(s) of the file or files whose contents are to be concatenated in the order indicated and sent across the link. If these parameters are the correct format for a logical name and the logical file name exists in the partition, then the file associated with the logical name is used. Otherwise, this parameter is used as an access name. If more than one file is specified, none of the files may be program or MDS file organization. The LOCAL NAME may be omitted if and only if the remote end specifies a REMOTE NAME in the corresponding RECEIVE command. If the remote end used an FTS command or is not an RM/COS system, then the LOCAL NAME must be specified in the SEND command.

REMOTE NAME
Logical name or access name of the file at the remote end of the link which is to receive the data sent across the link. If this parameter is the correct format for a logical name and the logical file name exists in the RECEIVEing or FTSing job's partition, then the file associated with the logical name is used. Otherwise, this parameter is used as an access name. The REMOTE NAME may be omitted if and only if the remote end specifies a LOCAL NAME in the RECEIVE command or is not an RM/COS system. If the remote end use an

FTS command or a SEND command without a LOCAL NAME, then the REMOTE NAME must be specified in the SEND command. If the remote end of the link is not an RM/COS system, then the REMOTE NAME should be omitted in the SEND command.

MODE

The transmission mode of the data to be sent. If the file contains only text (e.g., only ASCII graphics and vertical forms control information as in the case of a SPOOL type file), the CODED mode should be used. If the file contains nontext information (e.g., binary data) or if the file's contents are unknown, the IMAGE mode should be used. If the IMAGE mode is used to transmit a SPOOL type file, vertical forms control information associated with the file will be lost. If neither mode is specified, program files and MDS files will be sent in IMAGE mode and all others will be sent in CODED mode. An error will be indicated, and the file not transmitted, if coded mode is specified for a program file or an MDS file. If the remote end specifies the MODE and a REMOTE NAME in the corresponding RECEIVE command, then the SEND command must omit the MODE or specify the same mode. Normally, only one end should specify the transmission mode when two RM/COS systems are communicating; usually the mode may be omitted except to force IMAGE mode when a sequential, relative, or indexed file is transmitted and the file contains binary or computational data items (COBOL data items with usages of COMP, COMP-1, or COMP-3).

See the CONNECT, RECEIVE, and FTS command descriptions and Chapter 6, Data Communications, for additional information on the use of data communications.

Examples:

```
[ ] SEN
LINK LOGICAL NAME: RJELINK
LOCAL NAME: GREEN.REPORT.BILLING
REMOTE NAME:
MODE:
```

The command sends file GREEN.REPORT.BILLING using the link CONNECTed to RJELINK for remote disposition (e.g., printing). If the file is a SPOOL type file, vertical forms control information will be preserved.

```
[ ] SEND
LINK LOGICAL NAME: LINK3780
LOCAL NAME: GREEN.OBJECT.PROGA
REMOTE NAME: RED.OBJECT.PROG1
MODE:
```

The command sends program file GREEN.OBJECT.PROGA using the link CONNECTed to LINK3780. The receiving file is named RED.OBJECT.PROG1 and it must also be a program file.

```
[ ] SEND
LINK LOGICAL NAME: L
LOCAL NAME: .PLOTFILE
REMOTE NAME:
MODE: IMAGE
```

The command sends the file .PLOTFILE using the link CONNECTed to L. The file is sent in IMAGE mode, which provides transparent binary transmission of the file contents. If .PLOTFILE is a spool file, then vertical forms control information will be lost.

SETCOND <Terminal, Batch, Interrupt>

Function:

Batch mode JDL commands may be conditionally executed: the condition code of the partition in which a batch stream executes controls conditional execution. The SETCOND command sets the condition code for a partition.

Format:

```
PARTITION: [ int ]
VALUE: [ string ]
```

Where:

PARTITION
: Number of the partition to which the command applies. When no number is entered, the partition in which the command is executed applies.

VALUE
: New condition code, a single character string, which may be enclosed in quotes. If no value is entered, a blank is the new condition code. A lower-case letter is converted to the corresponding upper-case letter, even if enclosed in quotes.

Initially, the condition code of a partition is a blank but is set to the letter Z after a JDL command detects an error. The user may set the condition code to any (upper-case) letter, or to a blank. Condition codes control conditional execution of batch mode JDL commands.

Examples:

```
[ ] SET
PARTITION:
VALUE: A
```

The command sets the condition code for the user's terminal partition to A.

```
[ ] SETCON
PARTITION: 102
VALUE: Q
```

The command sets the condition code for partition 102 to the letter Q. This command could be used to force the exit of batch processing in partition 102.

```
[ ] SET
PARTITION:
VALUE:
```

The command sets the condition code for the partition executing the command to a blank.

SHOW <Terminal, Batch, Interrupt>

Function:

The user may display the contents of a sequential, relative or indexed file by entering a SHOW command. The file contents are displayed on the screen of the terminal.

Format:

NAME: lname ¦ acnm

Where:

NAME
The logical name associated with a device or a disk or tape file, or the pathname of a disk file. This command is illegal if a disk file does not have sequential, relative, or indexed organization.

The contents of the file are displayed on the terminal screen. Only the first 80 characters of each record are displayed. The records of a sequential file are displayed in the order in which they were written. The records of a relative file are displayed in order of ascending record number. The records of an indexed file are displayed in order of ascending prime record key. When the file contains more records than can be displayed on the screen, a full screen of records are displayed. Succeeding screens are displayed by pressing the Acknowledge Key.

Example:

```
[ ] SH
NAME: GREEN.SOURCE.JOHN.PROGA
```

This command displays the contents of the file GREEN.SOURCE.JOHN.PROGA.

```
[ ] SHOW
NAME: FILE-X
```

This command displays the contents of the file assigned to the logical name FILE-X.

SMODIFY <Terminal, Batch>

Function:

The SMODIFY command alters and optionally verifies data on a sector of a disk. The first word to be altered is specified by the relative byte offset within a sector. The contents of the words to be altered are listed, along with verification data and new data, as applicable. An error message is displayed when the current contents and the verification data are not equal.

Format:

```
DEVICE: name ! DSO1
SECTOR: int
BYTE: hex ! 0
VERIFY: [ hex [ ,hex ]... ]
PATCH: hex [ ,hex ]...
```

Where:

DEVICE
: Device name of the disk drive that contains the disk to be modified or volume name if the disk has been loaded.

SECTOR
: Number of the sector to be modified, specified as a decimal or hexadecimal (leading ">") integer limited to the maximum of sectors-on-disk and 4294967295.

BYTE
: Number of the first byte to be modified within the sector. The number must be an even number not greater than the number of bytes per sector.

VERIFY
: One or more hexadecimal integers each of which represents the contents of a word of data. The data is compared with the contents of the words of the sector, beginning at the specified byte. When no data is entered, no verification is performed.

PATCH
: One or more hexadecimal integers be written in the sector, replacing the current contents.

The SMODIFY command writes the data supplied by the user, after optionally verifying the previous contents of the sector. When all of the verification data does not match the previous data, an error message is displayed, and the patch data are not written.

The SMODIFY command lists the verification data, if any, the current data, and the new data.

The heading of the listing shows the sector number (hexadecimal), and the device name. On each line of the listing, the first four digits are the hexadecimal byte address (within the sector) of the leftmost data byte displayed on the line. The next eight groups of four digits each are the hexadecimal representations of 16 bytes of data. At the right of each line are eight pairs of characters. These are the ASCII characters that correspond to the data. Nonprinting ASCII characters are represented by periods (.).

Examples:

```
[ ] SMO
DEVICE: DS01
SECTOR: 150
BYTE: 0
VERIFY: C034, 30, 2F41
PATCH: C035, 3430
```

The command verifies that the first three words of sector 150 on the disk mounted on drive DS01 contain the specified contents. The command writes the two new words instead of the first two words of the sector, and writes the listing.

```
[ ] SMOD
DEVICE: DS02
SECTOR: 48
BYTE: 3E
VERIFY:
PATCH: 1D10, 1F16, 16F9, 321B
```

The command stores the four words in place of the previous contents of the four words beginning at byte 3E of sector 48 on the disk mounted on drive DS02. No verification of the previous contents is performed.

CAUTION

Use care in modifying the contents of the system disk. Inadvertently modifying the system programs on that disk may cause the system to fail.

Example listing 1:

```
SECTOR: 000008 (0000)   DEVICE: DS01
VERIFICATION DATA:
00B2  3133 0901 3105                              13 .. 1.
CURRENT DATA:
00B2  3133 0901 3105                              13 .. 1.
NEW DATA:
00B2  1000 2000 3000                              ..  . 0.
```

Example listing 2:

```
SECTOR: 000010 (0000)   DEVICE: DS01
VERIFICATION DATA:
0058  1000 2000                                   ..  .
CURRENT DATA:
0058  2020 2020
```

Notice that in example listing 2, an error message is generated and no modification is applied.

SORT <Terminal, Batch>

Function:

The SORT command invokes a file sort/merge utility capable of sorts and merges as available in COBOL, but without input, output or use procedures.

Format:

```
INPUT LOGICAL NAME: lname [ ,lname ]...
OUTPUT LOGICAL NAME: lname
KEY START: int [ ,int ]...
KEY LENGTH: int [ ,int ]...
ASCENDING: YES | NO [ ,YES | NO ]...
KEY TYPE: name | GRP [ ,name ]...
RECORD SIZE: [ int ]
MERGE ONLY: YES | NO
```

Where:

INPUT LOGICAL NAME
: List of one or more logical names which are to be the input files for the sort or merge operation. If one or more sort input files reside on tape, the tape file with the largest block size should be specified first.

OUTPUT LOGICAL NAME
: Logical name of a file which is to receive the final sorted file.

KEY START
: One or more integers defining the first column of each sort field. Keys are specified in the order of decreasing significance.

KEY LENGTH
: One or more integers defining the length of each sort field (for key type NBS, the value of KEY LENGTH is 2; for key type NPS, the value of KEY LENGTH is computed as floor (number-of-digits + 2)/2).

ASCENDING
: One or more YES or NO values indicating whether data is to be sorted from lowest to highest or highest to lowest, respectively, ASCII (for nonnumeric) or numeric order.

KEY TYPE

One or more of the following data type abbreviations (meanings in parentheses):

ABS	(Alphabetic String)
ANS	(Alphanumeric String)
ANSE	(Alphanumeric String Edited)
GRP	(Group)
NBS	(Numeric Signed - COMPUTATIONAL-1)
NCS	(Numeric Signed - COMPUTATIONAL)
NCU	(Numeric Unsigned - COMPUTATIONAL)
NLC	(Numeric Signed LEADING - DISPLAY)
NLS	(Numeric Signed LEADING SEPARATE - DISPLAY)
NPS	(Numeric Signed - COMPUTATIONAL-3)
NPU	(Numeric Unsigned - COMPUTATIONAL-6)
NSE	(Numeric Edited)
NSS	(Numeric Signed - DISPLAY)
NSU	(Numeric Unsigned - DISPLAY)
NTC	(Numeric Signed TRAILING - DISPLAY)
NTS	(Numeric Signed TRAILING SEPARATE - DISPLAY)

These names may be found in the allocation map at the end of the COBOL program compilation listing.

RECORD SIZE

The length of the longest record that may be found on any input file. This parameter is unnecessary if all the input files are disk files. If the largest record size is associated with an input tape file, and the tape file is not the first logical name specified, then its record size must be provided on the RECORD SIZE parameter.

MERGE ONLY

YES or NO indicating whether the data on the sort input files are already sorted or not. When YES is entered for this prompt, there must be more than one logical name for the INPUT LOGICAL NAME response.

Before entering the SORT command, the operator must assign or create three or more sequential files with logical names of the form SW$n, where n is a single digit. These files are not required for a merge only function. The batch stream .SORT has been provided with the system to assist in the creation of these sort work files.

The partition in which SORT is executed should be made as large as possible except in the case of a merge only function. For a sort function, additional available space in the partition is used to increase the efficiency of the sort. See the PARTITION command description for details regarding changing the size of a partition.

Examples:

```
[ ] SORT
INPUT LOGICAL NAME: SORTIN
OUTPUT LOGICAL NAME: SORTOUT
KEY START: 16,12
KEY LENGTH: 6,2
ASCENDING: YES,Y
KEY TYPE: GRP,NBS
RECORD SIZE:
MERGE ONLY: NO
```

This command defines a simple, one file sort with two keys. The most significant key is a group item starting in character position 16 with a length of 6 characters, and the least significant key is a COMPUTATIONAL-1 item starting in character position 12.

```
[ ] SO
INPUT LOGICAL NAME: MERGE1,MERGE2,MERGE3
OUTPUT LOGICAL NAME: MERGEOUT
KEY START: 6
KEY LENGTH: 10
ASCENDING: YES
KEY TYPE: GRP
RECORD SIZE:
MERGE ONLY: YES
```

This command defines a simple merge operation with one key.

STATUS <Terminal, Batch, Interrupt>

Function:

The STATUS command displays status information for the entire system, for a device, for all devices of a given device class, for a partition, or for a specified logical name. Sizes of partitions, condition codes of partitions, logical names assigned to devices, and logical names assigned in a partition can be determined by executing a STATUS command.

Format:

```
PARTITION: [ int ]
DEVICE: [ name ]
LOGICAL NAME: [ lname ]
```

Where:

PARTITION
: Number of a partition, or zero. When zero is entered, the partition in which the command is executed applies. When no number is entered, no single partition is specified.

DEVICE
: The name of either a device or a device suffix. In the latter case, all devices of that class are indicated. When no device name is entered, no single device is specified.

LOGICAL NAME
: Logical name for which status information is to be displayed. When no logical name is entered, no single logical name is specified.

Four types of status displays are available:

System status	Includes status of all partitions and all devices. Enter the command with no parameters.
Partition status	Enter the command with a partition parameter only.
Device status	Enter the command with a device parameter only.
Logical name status	Enter the command with a partition parameter and a logical name parameter.

Much of the information displayed is primarily of interest to the system programmer maintaining the system. However there are some items that concern the user of the system. When the command is entered without specifying a device, partition, or logical name, the status of all partitions, devices, and logical names is displayed. All partitions that were defined when the system was generated are listed in the display. There may be a partition for which only the state and type (unused) are shown. This partition was defined with a size of zero during system configuration. It is not available as a terminal partition or a nonterminal partition for executing a batch stream. Furthermore, such a partition may not have its size increased with the PARTITION command. This type of display also shows the amount of unallocated space, memory that is available but currently unallocated to any partition.

A partition for which the status display shows only the STATE, TYPE, SIZE, and PRIORITY is an inactive partition or the shared file partition. An inactive partition is available for use; if it is a terminal partition, it becomes active when a user logs-in at the terminal. If it is a nonterminal partition, it becomes active when a batch stream executes in the partition.

The status display for an active terminal partition shows the following information in addition to the items displayed for an inactive partition:

MEMORY USED
: The maximum memory used by this partition for JDL commands, programs, data structures and block buffers. Note that certain JDL commands such as COBOL use all of available memory.

PRIVILEGE
: Privilege level of the user logged-in at the terminal.

CONDITION CODE
: Current value of this partition's condition code.

SWITCHES
: The status of the eight switches of the partition, which are set by the SWITCH command. Eight binary digits (0 or 1) correspond to switches 1 through 8. When the digit for a switch is 1, the switch in the ON state.

LOGICAL NAMES
: Logical names assigned in the partition, if any. The number in parentheses preceding each logical name is the partition number.

The status display for each device varies, depending on the device type. Most of the information displayed is system maintenance information. The logical names assigned to a device are shown, with the number of the partition in which the logical name is assigned shown in parentheses preceding the name. Logical names assigned to the files on a disk are listed for a disk drive.

When a partition number and a logical name are entered with a STATUS command, the display shows the device name to which the logical name is assigned.

The display also shows the state, the organization, and type of access. If the file is of indexed organization, then the number of keys, their start position and length are also shown.

Examples:

```
[ ] ST  or [ ] ST.
PARTITION:
DEVICE:
LOGICAL NAME:
```

The command displays the status of the system.

```
[ ] STAT
PARTITION: 0
DEVICE:
LOGICAL NAME:
```

The command displays the status of the user's terminal partition.

```
[ ] STA
PARTITION:
DEVICE: DS01
LOGICAL NAME:
```

The command displays the status of disk drive DS01. The display lists all logical names asigned to all files on the disk.

```
[ ] STA
PARTITION: 2
DEVICE:
LOGICAL NAME: OUT
```

The command displays the status of logical name OUT assigned in partition 2. The display includes the device name, organization, state, and access. When the device name is the name of a disk drive, a file is assigned to the logical name. The organization is sequential for all devices other than disk drives.

SWITCH <Terminal, Batch, Interrupt>

Function:

A COBOL program may define switches in the SPECIAL NAMES paragraph and test the ON or OFF status of the switches by use of the switch condition-name test in conditional expressions. The SWITCH command sets the switches for testing during execution of the program. All switches are set OFF when a partition is activated by log-in to a terminal partition or initiation of batch processing in a nonterminal partition. SWITCH commands are only required to set switches ON or to set a switch OFF when a previous command has set it ON.

Format:

```
STATUS: [ ON | OFF ]
SWITCH: [ int [ ,int ]... ]
PARTITION: [ int ]
```

Where:

STATUS
: ON or OFF, defining the state to which this command sets one or more switches. When neither word is entered, the command sets the switch or switches specified in the SWITCH parameter to OFF.

SWITCH
: One or more switch numbers in the range one to eight inclusive. When two or more numbers are entered, they must be separated by commas. The switches corresponding to the numbers entered are set to the state specified. When no number is entered, all eight switches are set to ON or OFF, depending on the response to the STATUS prompt.

PARTITION
: Partition in which the switches are set. When no number is entered, the partition in which the command is executed applies.

The switch numbers refer to the switches defined in COBOL programs. The range of numbers is one to eight, inclusive.

Examples:

```
[ ] SW
STATUS: ON
SWITCH: 1,3,5
PARTITION: 103
```

The command sets switches 1, 3, and 5 to the ON state, in partition number 103. Switches 2, 4, 6, 7, and 8 remain in their current states.

```
[ ] SWI  or  [ ] SW.
STATUS: OFF
SWITCH:
PARTITION:
```

The command sets switches 1 through 8 to the OFF state.

SYNONYM <Terminal, Batch, Interrupt>

Function:

The SYNONYM command adds, deletes or changes synonyms and their values in the synonym table for the partition.

Format:

```
SYNONYM: lname
VALUE: [ string ]
```

Where:

SYNONYM
Synonym name.

VALUE
String containing at most 68 characters.

If the synonym does not exist, and there is room in the synonym table, the synonym is added. If the synonym exists and there is room in the table, the synonym value is replaced. If there is not room for the replacement value, the current value is not changed. If the VALUE is a null length string ("") or no value is entered, the synonym is deleted.

An attempt to delete a synonym that does not exist does not generate an error message if the SYNONYM command is being executed within a batch stream.

Examples:

```
[ ] SYNONYM
SYNONYM: S
VALUE: "DEMO.SOURCE"
```

The command defines a synonym S with a value DEMO.SOURCE

```
[ ] SYN
SYNONYM: X
VALUE: FILE@$PART
```

The command defines a synonym X with the value FILE001 if executed from terminal one, FILE002 if executed from terminal two, etc.

```
[ ] SYN
SYNONYM: A
VALUE:
```

The command deletes the synonym A if it exists.

SYSTEM-SHUTDOWN <Terminal>

Function:

The SYSTEM-SHUTDOWN command simulates a QUIT command, unloads all loaded disk volumes (including the system disk volume) and terminates the RM/COS operating system. It is the correct method of stopping the system in order to re-IPL or turn the machine off.

If other partitions are active when the SYSTEM-SHUTDOWN command is entered a message, stating that the system is shutting down in 30 seconds, will be broadcast to all terminals. After 30 seconds any partitions still active will be aborted. When no partitions remain active, the shutdown process will proceed.

Format:

This command has no prompts.

SYSTEM-FILE <Terminal, Batch, Interrupt>

Function:

At IPL the files .NEWSFILE and .USRDEFIL are selected as the News File and User Definition File. The SYSTEM-FILE command allows the user to change these selections after IPL.

Format:

```
NAME: acnm
FILE TYPE: NEWSFILE ! USRDEFIL
```

Where:

NAME
The pathname of the file (on the system disk) which is to be used as News File or User Definition File.

FILE TYPE
Specifies the use of the named file as follows:

NEWSFILE	News File
USRDEFIL	User Definition File

RM/COS locates the News File and User Definition File on the system disk during IPL and saves pointers to these files for use during log-in. The System-File command allows a user to change either of these pointers to refer to a different file of the user's choice.

TAPE-ASSIGN <Terminal, Batch>

Function:

An RM/COS tape file behaves as a sequential organization file, and only sequential access is allowed. A COBOL program may open a tape file INPUT and read its records from the beginning, or a program may open a tape file OUTPUT or EXTEND and write records to the file. The open I-O clause and REWRITE statement are illegal on tape files. At this time, vertical forms control information from the AFTER or BEFORE ADVANCING clauses can not be written to tape.

A tape volume under RM/COS may contain one or more than one file. A single file may reside on one or more than one tape volume. The portion of a file on a single volume or track is called a file section. If more than one file resides on a tape volume or series of tape volumes, the collection of all files residing on the tape volume or volumes is called a file set.

There is no directory of files on a tape; the files on a tape are not accessed by name. Instead, the user specifies the position within the file set of the desired file. The first file of the file set is said to be at position one. Succeeding files are at higher numbered positions.

The TAPE-ASSIGN command assigns a logical name to a tape device, and specifies the tape volumes and parameters to be used when accessing a tape file with the given logical name. When several logical names are to be associated with a file set, the first TAPE-ASSIGN command specifies all the volumes of the file set, not merely the volumes containing the first file. Subsequent TAPE-ASSIGN commands to the file set must not specify any volume names.

The logical name defined with the TAPE-ASSIGN command is not necessarily associated with only one file of the file set. A COBOL program may associate several COBOL file names with the same external logical name, thus allowing the one logical name to be accessed with differing file characteristics. Furthermore, with the CLOSE ... WITH NO REWIND and OPEN ... WITH NO REWIND statements, the COBOL program may read or write several tape files with a single COBOL file name.

The FILE-BACKUP, FILE-RESTORE, and FILE-VALIDATE JDL commands are the only JDL commands which can access a tape device directly. All other JDL commands can access a tape file only through a logical name established with the TAPE-ASSIGN command. Commands which do not accept a logical name as a parameter cannot use tape files.

The TAPE-ASSIGN command is the only command needed to make a tape file accessible to a program. The INITIALIZE, LOAD, and UNLOAD commands apply only to disk devices, and are illegal on a tape device.

A single tape device may be assigned to only one user at a time. A single user may assign several file sets to a single tape device, but only one file on a tape device may be open at a time.

Format:

```
LOGICAL NAME: lname
DEVICE: [ name ]
VOLUME: [ string [ ,string ] ... ]
SCRATCH: [ string [ ,string ] ... ]
PREMOUNT: [ string ]
SET NAME: [ string ]
POSITION: [ int ]
BLOCK SIZE: [ int ]
RECORD SIZE: [ int ]
FORMAT: [ FIXED | DECIMAL | SPANNED | UNDEFINED |
          VARIABLE | VARIABLE,SPANNED ]
PADDING: [ hex ]
OFFSET: [ int ]
OPTIONS: [ BACKUP | NONE ]
```

Where:

LOGICAL NAME
: Logical name to be assigned to the tape device.

DEVICE
: Device name of a tape device. If this is not the first logical name of the file set, then the DEVICE parameter is optional, but if specified it must match that used when the file set was initially assigned.

VOLUME
: A list of volume names, optionally with file counts, identifying the volumes on which the file or files currently reside. A volume name is one to six graphic characters excluding quotation mark, slash, comma, semicolon, angle brackets, and square brackets (" / , ; < > [ ]). Each volume name is optionally followed with one or more occurrences of a slash and a file count represented as a decimal integer. When the file set is read, the volumes are mounted in the order they were specified to the VOLUME parameter. When a multiple reel or multiple track file is accessed on hardware that does not support an end of volume mark, the optional file counts specify the number

of file sections on each reel or track. (See the Operator Guide for further information.) The same volume name may not be specified to two or more TAPE-ASSIGN commands; two files with the same volume name are part of the same file set, and must be associated through the SET NAME parameter. The VOLUME and SCRATCH parameters combined may not specify more than forty volumes of one logical track each, or ten volumes of four tracks each. This parameter is illegal if this is not the first logical name assigned to the file set.

SCRATCH

A list of volume names identifying volumes which may be overwritten. A volume name is one to six graphic characters excluding quotation mark, slash, comma, semicolon, angle brackets, and square brackets (" / , ; < > [ ]). If the VOLUME parameter is omitted, the first file is written starting as the first file section on the first scratch volume. If the VOLUME parameter is specified, then the first scratch volume is mounted only when a file being extended exhausts the space remaining on the input volumes. The same scratch volume name may not be specified to two or more TAPE-ASSIGN commands; two files with the same volume name are part of the same file set, and must be associated through the SET NAME parameter. The VOLUME and SCRATCH parameters combined may not specify more than forty volumes of one logical track each, or ten volumes of four tracks each. This parameter is illegal if this is not the first logical name assigned to the file set.

PREMOUNT

The volume name of the tape volume already mounted in the tape drive. If this parameter is specified and the specified volume name is the first volume requested by the application, then the system does not display the initial tape mount message, and does not display an unload message when the tape is released. This parameter is useful in batch streams intended to execute in the middle of the night, where the operator mounts the tape volume in the drive before the day shift ends. This parameter is illegal if this is not the first logical name assigned to the file set.

SET NAME

A one to six character name used to bind several logical names to the same file set. All TAPE-ASSIGN commands to the same file set must specify the same SET NAME; all TAPE-ASSIGN

commands which specify the same SET NAME are associated with the same file set. When assigning several logical names to the same file set, the first TAPE-ASSIGN command establishes all the volumes of the file set, and if necessary, the number of files on each volume. Subsequent TAPE-ASSIGN commands to the same file set must specify the same SET NAME value, and must not specify values for the VOLUME or SCRATCH parameters.

POSITION

The position of the file in the file set. The first file on the first volume is position 1. If no value is specified, a value of 1 is assumed. A COBOL program can override the POSITION specification with a MULTIPLE FILE TAPE clause, or by using the WITH NO REWIND clause on OPEN and CLOSE statements.

BLOCK SIZE

The maximum number of characters to write or expect to read in a physical block on tape. A COBOL program can override the BLOCK SIZE specification with a BLOCK CONTAINS clause. If the record format is not SPANNED or <VARIABLE, SPANNED>, then the block size must allow space for the longest record plus the record or block overhead of the record format used. If no block size is specified, each physical block written will contain only one logical record. The larger the block size specification, the greater the capacity of the tape, and the greater the memory requirements when the file is open. (Some implementations permit the installation to specify a fixed tape block size in the System Definition File, which will override any user block size specification, and will force records to be blocked regardless of whether a block size is specified in the program or to the TAPE-ASSIGN command. See the Operator Guide for further information.)

RECORD SIZE

The maximum number of characters to expect in a logical record. A COBOL program will override this specification with the length of the longest record area associated with the file, so this parameter is needed only when the logical name will be accessed with a JDL command. When the logical name is accessed with a JDL command, and no record size is specified on either the JDL command or the TAPE-ASSIGN command, a record size will be computed from the block size if any. A

record size larger than the block size is legal only if a record format of SPANNED or <VARIABLE, SPANNED> is specified.

FORMAT

The manner in which logical records are organized in each physical block. The operating system is responsible for generating and interpreting the record and block overhead of each format. Only the data characters in the logical record are available to a COBOL program. The available options are:

FIXED - Each logical record of the file contains the same number of characters. A physical block contains one or more contiguous logical records with no record overhead. Format FIXED is compatible with IBM record formats F, FB, and FBS, and ANSI X3.27-1978 record format F. The option FIXED may be abbreviated to F.

DECIMAL - Logical records of the file may vary in length. A physical block contains one or more entire records. Each logical record is preceded by four decimal digits, which specify the length of the record including overhead. Format DECIMAL complies with ANSI X3.27-1978 record format D. The option DECIMAL may be abbreviated to D.

SPANNED - Logical records of the file may vary in length. A physical block contains one or more entire or partial records; a single record may span more than one physical block. Each logical record or segment of a logical record is preceded by five decimal digits, which specify the length of the segment, and whether the segment is the first, last, middle, or entire portion of the record. Format SPANNED complies with ANSI X3.27-1978 record format S. The option SPANNED may be abbreviated to S.

UNDEFINED - Logical records of the file may vary in length. Each physical block is identical to a logical record. Format UNDEFINED is compatible with IBM record format U. The option UNDEFINED may be abbreviated to U.

VARIABLE - Logical records of the file may vary in length. A physical block contains one or more entire records. Each logical record is preceded by a four character field specifying the length of the record. The physical block also contains a four character field specifying the amount of data in the block. Format VARIABLE is compatible with

IBM record formats V and VB. The option VARIABLE may be abbreviated to V.

VARIABLE, SPANNED - Logical records of the file may vary in length. A physical block contains one or more entire or partial records; a single record may span more than one physical block. Each logical record or segment of a logical record is preceded by a four character field, which specifies the length of the segment, and whether the segment is the first, last, middle, or entire portion of the record. The physical block also contains a four character field specifying the amount of data in the block. Format VARIABLE, SPANNED is compatible with IBM record format VBS, but may also be used to read an IBM format V, VB or VS file. The option VARIABLE, SPANNED may be abbreviated to V, S.

If FORMAT is not specified, format FIXED is assumed if all record areas of the file have the same length. If the record areas of the file differ in length, format DECIMAL is assumed if the BACKUP option is specified, otherwise format VARIABLE is assumed. If the record size resulting from these assumptions exceeds the block size, then format SPANNED is assumed if the BACKUP option is specified, otherwise format VARIABLE, SPANNED is assumed.

PADDING

The pad character, specified as a hexadecimal number, which is used to pad blocks to the length required by the hardware controller. On output, if no pad character is specified and the hardware controller requires that blocks be of fixed size, the block is padded with >5E characters. On input, if a pad character is specified or if the hardware controller requires that blocks be of fixed size, and the remainder of the physical block consists entirely of pad characters, it is assumed that the remainder of the block contains no records, and the next block is read from tape. Pad stripping on input is suppressed if the record format is VARIABLE or <VARIABLE, SPANNED> or if the BACKUP option is specified, regardless of whether a PADDING parameter is provided.

OFFSET

The number of overhead characters which precede the data in the physical block. This parameter is primarily intended to ease reading of tapes from other operating systems which may provide a nonstandard block overhead. When reading a file,

this number of characters are skipped before looking for the block or record overhead of the specified record format. When writing a file, this number of characters are left undefined at the beginning of each block.

OPTIONS

Specifies options to control the tape transfer. The available options are:

BACKUP - Specifies that the 14 character block overhead used by FILE-BACKUP and FILE-RESTORE should be generated or expected. This overhead includes the amount of data in the block, a sequence number of the block within the file, a unique identifier of the file, and whether the block is a data block or is the EOF block at the end of the file. The overhead allows the backup and restore processes to validate the order in which tape volumes are mounted and whether to switch tracks or volumes when a tape mark is read, without requiring the use of IBM or ANSI labels. The FILE-BACKUP and FILE-RESTORE commands set the BACKUP option when a tape device name is specified for the BACKUP FILE parameter. The BACKUP option should be specified to TAPE-ASSIGN when a file written directly by FILE-BACKUP is to be read by logical name, or when writing a file by logical name which will later be read directly by FILE-RESTORE.

NONE - No options are selected. Specification of NONE along with other options has no effect, and the other options specified are effective.

Volume Specification

RM/COS can support both nine track tape reels and four track tape cartridges. On a nine track tape, each eight-bit data character is recorded with parity across the nine tracks; thus, the tape is read from end to end in only one pass. On a four track tape cartridge, each data character is recorded serially in a single track, and the tape is accessed in four passes, one for each track. Usually a cartridge tape controller informs the software when the end of a track is approaching, and the software writes a tape mark to terminate the file section and the file is continued in another file section on the next track. Sometimes the cartridge tape controller provides a special tape mark called an end of volume mark to allow the software to distinguish between a tape mark separating files and the tape mark at the end of a track. Other cartridge tape controllers automatically switch tracks, without requiring the software to write or interpret a tape mark at end of

track. RM/COS treats a tape cartridge as a single volume containing one or four logical tracks, depending on whether or not the tape controller supports automatic track switching. Most cartridge tape controllers write each track from beginning of tape to end of tape, but some controllers write tracks 2 and 4 in the reverse direction to avoid rewinding when switching tracks.

Whether a file count is allowed or illegal after a volume name specified on the VOLUME parameter depends on the type of tape hardware in use. If the system can distinguish between a tape mark that separates files and a tape mark which indicates the end of a track or volume, then a file count specification is illegal. Otherwise, when specifying a list of volumes to be read, the number of file sections on each volume must be specified. Furthermore, if a tape mark is written at the end of each cartridge track, then the number of file sections on each track must be specified, regardless of whether more than one or only one volume is being read.

The number of files on a track or volume is sufficient to determine where each file in a file set begins. Except for the last track of the last volume, the last file section on a track or volume is part of a file continued as the first file section on the next track or volume. Except for the first track of the first volume, the first file section of a track or volume is always the continuation of a file, never the initial file section of a file.

When a tape is written or modified under RM/COS, the system maintains a count of the number of file sections on each track or volume. This information is maintained in memory as long as any logical name is assigned to the file set. When the last logical name assigned to the file set is released, the RELEASE JDL command displays to the listing file the names of the volumes containing data. If the user will be required to specify the number of file sections on each track or volume in order to read the data, then the RELEASE JDL command also displays the file count of each track or volume.

The following commands define a file set to be read from a single nine track reel. Because only one reel is being accessed, a file count specification is not required:

```
/TAPE-ASSIGN, LOGICAL=FILE1, POSITION=1, SET NAME=TFILES,
              DEVICE=MT01, VOLUME=MYREEL
/TAPE-ASSIGN, LOGICAL=FILE2, POSITION=2, SET NAME=TFILES
/TAPE-ASSIGN, LOGICAL=FILE3, POSITION=3, SET NAME=TFILES
```

The following commands define the same three files where the second file spans two nine track reels:

```
/TAPE-ASSIGN, LOGICAL=FILE1, POSITION=1, SET NAME=TFILES,
              DEVICE=MT01, VOLUME=<REEL-A/2, REEL-B/2>
/TAPE-ASSIGN, LOGICAL=FILE2, POSITION=2, SET NAME=TFILES
/TAPE-ASSIGN, LOGICAL=FILE3, POSITION=3, SET NAME=TFILES
```

Usually the last file section on a volume is terminated with a double tape mark. This second tape mark is not counted in the file count specification as an empty file section. Thus, REEL-A contains two file sections and REEL-B contains two file sections. When FILE3 is opened, the system recognizes that the first section of REEL-B is part of FILE2, requests that REEL-B be mounted in MT01, and positions past one tape mark in order to access the second file section on REEL-B.

The special case of a single file residing on multiple nine track reels does not require a file count specification:

```
/TAPE-ASSIGN, LOGICAL=MFILE, DEVICE=MT01, VOLUME=<REEL-A,
              REEL-B, REEL-C, REEL-D>
```

Because no file count was specified with any volume name, the system assumes that the single file spans all the volumes specified, and therefore defaults a file count of one with each volume.

The following command defines a file set to be read from a single tape cartridge; the second file spans tracks 1, 2, and 3:

```
/TAPE-ASSIGN, LOGICAL=FILE2, POSITION=2, SET NAME=CFILES,
              DEVICE=MT01, VOLUME=001234/2/1/4
/TAPE-ASSIGN, LOGICAL=FILE4, POSITION=4, SET NAME=CFILES
```

When FILE4 is opened, the system will position past 2 tape marks on track 3, in order to read the third file section of track 3.

When the cartridge tape controller supports end of volume marks, the file count specification is not only unnecessary but illegal. This spares the user the problem of recording the number of files on each track, but it can slow random positioning on the tape. The following commands define the same file set as the previous example, but on hardware that supports end of volume marks:

```
/TAPE-ASSIGN, LOGICAL=FILE2, POSITION=2, SET NAME=CFILES,
              DEVICE=MT01, VOLUME=001234
/TAPE-ASSIGN, LOGICAL=FILE4, POSITION=4, SET NAME=CFILES
```

When FILE4 is opened, the system will read all of tracks 1 and 2 and part of track 3 in order to position to the start of the file. However, the file count information for each track read is remembered as long as a logical name is assigned to the file set. The following command assigns another logical name to the same file set:

```
/TAPE-ASSIGN, LOGICAL=FILE3, POSITION=3, SET NAME=CFILES
```

When FILE3 is opened after processing FILE4, the system can position correctly on track 3, without reading tracks 1 and 2.

A cartridge tape controller which automatically switches tracks appears to the system to have only one track. The volume and file count parameters on such hardware act identically to the nine track reel hardware discussed previously.

When a tape is first written, there are no files already on the tape, and the user may not know how many volumes the first file will require. The user must specify a list of available volumes on the SCRATCH parameter:

```
/TAPE-ASSIGN, LOGICAL=FILE1, DEVICE=MT01, SCRATCH=<REEL1,
              REEL2, REEL3, REEL4, REEL5>
```

When FILE1 is written, the system will write first on REEL1. If the file is larger than one reel, the system will write on REEL2. When the logical name FILE1 is released, the RELEASE JDL command will list the reels actually used:

```
File FILE1     resides on REEL1/1, REEL2/1
```

This information allows the same file to be read later by specifying VOLUME = < REEL1, REEL2 >.

Both the VOLUME and SCRATCH parameters must be specified when extending an existing file to additional volumes. The following commands define a file set on a tape cartridge:

```
/TAPE-ASSIGN, LOGICAL=FILE1, POSITION=1, SET NAME=CFILES,
              DEVICE=MT01, VOLUME=VOLA/2/1/2,
              SCRATCH=<VOLB, VOLC>
/TAPE-ASSIGN, LOGICAL=FILE2, POSITION=2, SET NAME=CFILES
/TAPE-ASSIGN, LOGICAL=FILE3, POSITION=3, SET NAME=CFILES
```

Suppose that after processing FILE3, the COBOL program extends FILE2, thus overwriting FILE3. The additional data appended to FILE2 may require all the remaining tracks on VOLA, and the first and part of the second track on VOLB. When the logical names FILE1, FILE2, and FILE3 are all released, the RELEASE JDL command will list the current file counts on each volume of the file set:

```
Set CFILES  resides on VOLA/2/1/1/1, VOLB/1/1
```

In the above example, after FILE2 was extended, any data in FILE3 or subsequent files was unavailable. An open OUTPUT of FILE3 would have caused new data to be written, starting somewhere on track 2 of VOLB. An open OUTPUT of a file at position 4 or higher would have been illegal.

Volume Mounting

The RM/COS system does not request that a tape volume be mounted until a file is opened on the tape volume. When a file is opened, a message of the following form is displayed in the bottom right corner of the operator's station:

```
Mount VOLNAM into MTxx
```

After placing the tape cartridge in the drive, or mounting the nine track reel, the operator presses the Acknowledge Key to allow the system to begin accessing the tape. If no tape was mounted in the drive, a message of the following form will be displayed:

```
MTxx Not Ready.  Retry or Abort? Retry
```

If the operator presses the Return or Acknowledge Key, the system will redisplay the mount message. If the operator types "Abort" and presses the Return Key, the open request will fail.

When a file is opened OUTPUT or EXTEND, there is no validation that the correct volume is mounted. When the operator acknowledges the tape mount message, the system will begin writing on the tape. When a file is opened INPUT, the volume can be validated if the file is written and read with the BACKUP option; otherwise, there is no validation upon input that the correct volume is mounted.

A tape volume is not unloaded by the system until all logical names associated with the file set are released, or until an open is requested that requires a different volume be mounted. If after a tape file is closed, the same file is opened or a different file on the same volume is opened, the system begins accessing the tape without issuing a tape mount message. If the user opens a file in a different file set, the currently mounted volume is unloaded and a mount message is issued for the new volume. Note that if the currently mounted volume was positioned at the end of a file after a CLOSE ... WITH NO REWIND statement, then the COBOL program may later open a file of the file set WITH NO REWIND and access the following file on the tape, even if volumes of a different file set were accessed on the same drive between the close and the open.

When the system unloads a volume, the tape reel is rewound to beginning of tape, or the tape cartridge is positioned to either beginning of tape or end of tape. When the unload operation is a result of releasing the last logical name of a file set, a message of the following form is displayed:

Unload MTxx

The operator must press the Acknowledge Key to allow the system to continue processing. When the unload operation results from the system requesting a new volume, only the mount message is displayed. The operator must remove the unloaded volume before mounting the requested volume.

The initial volume mount message and final unload message can be suppressed with the PREMOUNT parameter. When a volume name is specified on the PREMOUNT parameter, the system records that the specified volume is already in the drive. When the first file opened on the tape drive resides on the premounted volume, no mount message is displayed, thus avoiding requiring an operator response. If all files accessed reside on the premounted volume, then when all the files are released no unload message is displayed, again avoiding requiring an operator response. If the system needs a different volume mounted, the system resumes the normal displaying of operator messages. This capability of premounting a volume is designed for use in a batch stream when it will be several hours before the application opens the tape file and the operator wishes to mount the tape before leaving the machine unattended.

Record Formats

This section illustrates how records are formatted into blocks for each record format. Further information may be found in the American National Standard on Magnetic Tape Labels and File Structure for Information Interchange, ANSI X3.27-1978, and in the IBM OS/VS1 Data Management Services Guide, GC26-3874.

An understanding of this section is not required in order to use the RM/COS tape subsystem. The only rule that must be remembered is that record format FIXED may be used only when every record is the same length. As long as the program reading a tape file uses the same file descriptor and the same set of record areas as the program which wrote the tape file, the system should choose the same record format and block size, and the data should be accessed correctly.

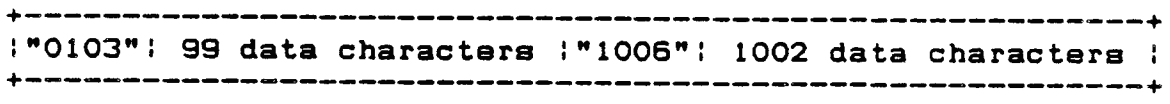
In each of the record formats, the data characters in the record are written to tape as they were presented by the COBOL program, with no compression, no conversion, and no removal of trailing spaces. The record formats differ only in the manner in which the beginning and end of each record

is determined. Record formats FIXED and UNDEFINED do not have overhead information for each record; record formats DECIMAL, SPANNED, VARIABLE, and <VARIABLE, SPANNED> all have overhead information before each record.

When FIXED record format is used, no indication of the length of each record is present within the file. Each block contains an integral number of records, as many records as fit without exceeding the block size. When no block size is specified, each block consists of exactly one record. The last block written to the file may be short. The system accepts blocks shorter than the block size on input, regardless of whether the block is the last block. On hardware controllers that require that all blocks be of fixed length, the system pads each block written with the pad character. On input, when padding is required or specified, the system ignores records where the remainder of the block consists only of pad characters. This may cause a record written to the file to be ignored, and can be avoided by specifying a different record format.

When record format UNDEFINED is used, each physical block is identical to a logical record. On hardware which allows variable length blocks, UNDEFINED format records may vary in length. UNDEFINED record format is especially useful to dump and examine a file of unknown record format. (The C$READS subprogram allows a program to read records of a file and learn the length of each record read.)

Record format DECIMAL allows records to vary in length by preceding each record with four decimal digits, called a Record Control Word, which specify the length of the record. Each block contains an integral number of records, as many records as fit without exceeding the block size. When no block size is specified, each block consists of exactly one record. On hardware controllers that require that all blocks be of fixed length, the system pads each block written with the pad character, and ignores the remainder of any block read which consists only of pad characters. The following is an example block of 1109 characters containing two format DECIMAL records:



Record format SPANNED not only allows records to vary in length, but also allows records to span blocks. Each record is preceded by five digits called a Segment Control Word, which consists of a one digit Spanning Indicator, and four digits specifying the length of the record segment. The Spanning Indicator assumes one of four possible values:

"0": Record begins and ends in this segment
"1": Record begins but does not end in this segment
"2": Record neither begins nor ends in this segment
"3": Record ends but does not begin in this segment

Each physical block except the last will be equal in length to the block size, unless five or fewer character positions are available at the end of the block. The following diagram illustrates three records, of lengths 99, 1002, and 879 characters, written in two blocks of 1000 characters each:

```
+-----------------------------------------------------------+
|"00104"| 99 data characters  |"10896"| 891 data characters |
+-----------------------------------------------------------+

+-----------------------------------------------------------+
|"30116"| 111 data characters |"00884"| 879 data characters |
+-----------------------------------------------------------+
```

Record format VARIABLE allows records to vary in length by preceding each record with a four character field called a Record Descriptor Word. The first two characters contain the binary length of the record, including the four overhead characters. The second two characters contain binary zero. Each block begins with a four character field called a Block Descriptor Word. The first two characters of the Block Descriptor Word contain the binary length of the block including overhead, and the second two characters contain binary zero. Each block contains an integral number of records, as many records as fit without exceeding the block size. When no block size is specified, each block consists of exactly one record. The following diagram illustrates a block of 1109 characters containing two format VARIABLE records:

```
+-----------------------------------------------------------+
|>04590000|>00670000|99 characters|>03EE0000|1002 characters|
+-----------------------------------------------------------+
```

Record format <VARIABLE, SPANNED> allows records to both vary in length and to span blocks. Each record is preceded with a four character field called a Segment Descriptor Word. The first two characters contain the binary length of the record, including the four overhead characters. The third character contains an indicator of whether the segment spans blocks, called a Segment Control Code, which assumes one of four values:

>00: Record begins and ends in this segment
>01: Record begins but does not end in this segment
>02: Record ends but does not begin in this segment
>03: Record neither begins nor ends in this segment

The fourth character of the Segment Descriptor Word is a binary zero. Each block begins with a four character field called a Block Descriptor Word. The first two characters of the Block Descriptor Word contain the binary length of the block including overhead, and the second two characters contain binary zero. Each physical block except the last will be equal in length to the block size, unless four or fewer character positions are available at the end of the block. The following diagram illustrates three records, of lengths 99, 1002, and 875 characters, written in two blocks of 1000 characters each:

```
+-----------------------------------------------------------+
!>03E80000!>00670000! 99 characters!>037D0100!889 characters!
+-----------------------------------------------------------+

+-----------------------------------------------------------+
!>03E80000!>00750200!113 characters!>036F0000!875 characters!
+-----------------------------------------------------------+
```

Backup Option

The FILE-BACKUP, FILE-RESTORE, and FILE-VALIDATE JDL commands can access a tape device without a TAPE-ASSIGN command being entered. When a tape device name is provided to these commands as a BACKUP FILE, the following implicit TAPE-ASSIGN is performed:

```
/TAPE-ASSIGN, LOGICAL NAME=BACKUP, DEVICE=device name,
        VOLUME=<BKUP01,BKUP02,BKUP03,BKUP04,BKUP05,
                BKUP06,BKUP07,BKUP08,BKUP09,BKUP10>,
        RECORD SIZE=record size specified,
        POSITION=1, FORMAT=FIXED, OPTIONS=BACKUP
```

The BACKUP option is used when transfering a backup file between tape and another medium. In Chapter 6, Data Communications, there is an example of copying a directory from one system to another using FILE-BACKUP and FILE-RESTORE. The following command stream illustrates the same example, where the backup file is a tape:

```
/  ASSIGN, LOGICAL NAME=LO, NAME=LP01
/  FILE-BACKUP, BACKUP FILE=MT01, SOURCE NAME=.PROJECTX,
                RECORD SIZE=510
/  RELEASE, LOGICAL NAME=LO
/  TAPE-ASSIGN, LOGICAL NAME=BACKUP, DEVICE=MT01,
                VOLUME=<BKUP01,BKUP02,BKUP03,BKUP04>,
                RECORD SIZE=510, FORMAT=FIXED,
                OPTIONS=BACKUP
/  CONNECT, LOGICAL NAME=LINK, NAME=BL01
/  SEND, LINK LOGICAL NAME=LINK, LOCAL NAME=BACKUP,
        MODE=IMAGE
/  RELEASE, LOGICAL NAME=<BACKUP, LINK>
```

The BACKUP option causes the tape backup block overhead to be removed from the data before sending the records over the link. If the backup block overhead information was not removed, the data would not be valid at the receiving site, even if the file was received to tape. The following command stream could be used to receive and restore the directory:

```
/  TAPE-ASSIGN, LOGICAL NAME=BACKUP, DEVICE=MT01,
                SCRATCH=<BKUP01,BKUP02,BKUP03,BKUP04>,
                RECORD SIZE=510, FORMAT=FIXED,
                OPTIONS=BACKUP
/  CONNECT, LOGICAL NAME=LINK, NAME=BL01, TIMEOUT=0
/  RECEIVE, LINK LOGICAL NAME=LINK, LOCAL NAME=BACKUP
/  RELEASE, LOGICAL NAME=<LINK, BACKUP>
/  ASSIGN, LOGICAL NAME=L0, NAME=LP01
/  FILE-RESTORE, BACKUP FILE=MT01, BACKUP NAME=.PROJECTX,
       DESTINATION NAME=.PROJECTX, RECORD SIZE=510
/  RELEASE, LOGICAL NAME=L0
```

The above command streams do not share any logical names between the backup/restore operation and the send/receive operation. This allows the backup or restore to be performed at a different time than the send or receive. However, it also causes the system to prompt twice for the tapes to be mounted. The following two command streams avoid this inconvenience:

```
/  TAPE-ASSIGN, LOGICAL NAME=BACKUP, DEVICE=MT01,
                SCRATCH=<BKUP01,BKUP02,BKUP03,BKUP04>,
                RECORD SIZE=510, FORMAT=FIXED,
                OPTIONS=BACKUP
/  ASSIGN, LOGICAL NAME=L0, NAME=LP01
/  FILE-BACKUP, BACKUP FILE=BACKUP, SOURCE NAME=.PROJECTX
/  RELEASE, LOGICAL NAME=L0
/  CONNECT, LOGICAL NAME=LINK, NAME=BL01
/  SEND, LINK LOGICAL NAME=LINK, LOCAL NAME=BACKUP,
       MODE=IMAGE
/  RELEASE, LOGICAL NAME=<BACKUP, LINK>

/  TAPE-ASSIGN, LOGICAL NAME=BACKUP, DEVICE=MT01,
                SCRATCH=<BKUP01,BKUP02,BKUP03,BKUP04>,
                RECORD SIZE=510, FORMAT=FIXED,
                OPTIONS=BACKUP
/  CONNECT, LOGICAL NAME=LINK, NAME=BL01, TIMEOUT=0
/  RECEIVE, LINK LOGICAL NAME=LINK, LOCAL NAME=BACKUP
/  RELEASE, LOGICAL NAME=LINK
/  ASSIGN, LOGICAL NAME=L0, NAME=LP01
/  FILE-RESTORE, BACKUP FILE=BACKUP,
       BACKUP NAME=.PROJECTX,
       DESTINATION NAME=.PROJECTX
/<XL> RELEASE, LOGICAL NAME=BACKUP
/  RELEASE, LOGICAL NAME=L0
```

Because these two command streams use the same logical name assignment from the backup to the send, and from the receive to the restore, the BACKUP option is not required at either end. If omitted, backup block overheads are not written to the tape. The presence of backup block overheads provides validation of volumes as they are mounted, and additional error detection when the data is read. Specifying the BACKUP option allows the tapes that are created to be passed directly to a FILE-RESTORE command at a later time.

TEST-SYSDEFIL <Terminal, Batch>

Function:

The TEST-SYSDEFIL command allows the operator to activate and deactivate an alternate System Definition File for testing purposes. (After correct system operation is verified using the System Definition File under test, the INSTALL-SYSTEM JDL command may be used to establish the tested System Definition File as the new permanent System Definition File.)

Format:

```
VOLUME: name
DEVICE: name
TEST SYSDEFIL FILE NAME: [ acnm ]
```

Where:

VOLUME
: The volume name of the disk volume on which the new system is to be installed. The volume need not be loaded. The volume may be the current system disk volume when only one partition is active.

DEVICE:
: The device name of the disk drive on which the volume is mounted or will be mounted. The system disk drive may be used to install a system on the current system disk volume or on another volume when only one partition is active.

TEST SYSDEFIL FILE NAME
: The pathname of the sequential file which is to be the tested System Definition File: the pathname should begin with a period, not a device name or volume name, since the file is assumed to be on the volume specified by the VOLUME and DEVICE parameters. If this parameter is not present, the current test System Definition File (if any) will be deactivated. See Chapter 2, System Configuration, and your Operator Guide for further information about the tested System Definition File.

The tested System Definition File is removed from test status by any subsequent successful INSTALL-SYSTEM command as well as a subsequent TEST-SYSDEFIL with no TEST SYSDEFIL FILE NAME specified.

Example:

```
[ ] TEST
VOLUME: RMCOSSYS
DEVICE: DS01
TEST SYSDEFIL FILE NAME: .TSTDEFIL
```

The command installs the file .TSTDEFIL as the test System Definition File on volume RMCOSSYS in drive DS01. The next IPL of the system will notify the operator of the presence of a test System Definition File. The file .TSTDEFIL will remain in test status until either another TEST-SYSDEFIL or an INSTALL-SYSTEM command is successfully completed.

TIME <Terminal, Batch, Interrupt>

Function:

The TIME command sets the date and time in the system. RM/COS maintains the date and time, once set, and provides date and time information on listings. The system also supplies date and time information to COBOL programs that use ACCEPT from DATE, DAY and TIME.

Format:

```
YEAR: int
MONTH: int
DAY: int
HOUR: int
MINUTE: int
SECOND: int ! 0
```

Where:

YEAR
: Year, represented as four decimal digits, 1982 through 1999, or as two decimal digits, 82 through 99.

MONTH
: Month, represented as a number, 1 through 12.

DAY
: Day of the month, represented as a number, 1 through 31.

HOUR
: Hour, represented as a number, 0 through 23.

MINUTE
: Minute, represented as a number, 0 through 59.

SECOND
: Second, represented as a number, 0 through 59.

RM/COS maintains the date and time for use on listings, and to supply values for DATE, DAY and TIME in COBOL programs. The TIME command initializes the system date and time. No JDL commands other than BATCH, CONTINUE, EXIT, LOOP, QUIT, REPEAT, SETCOND and TIME may be entered until the TIME command has been successfully completed.

NOTE

Time is maintained with a resolution of tenths of seconds.

The TIME command produces a listing header after it sets the date and time. The listing header indicates the date and time so that the operator can verify that they are correctly set. Normally the listing header will be displayed on the terminal screen, but the standard command listing rules apply (see the section Command Listings at the beginning of this chapter). If an error was made in setting the date and time, the operator may reenter the command to specify the correct values.

As part of the user log-in procedures, the system will set the condition code to the letter T if time has not been previously initialized. All initial user batch streams should start with the following JDL command to ensure that the system time is initialized:

```
T/BATCH, NAME=.TIME
```

The batch stream .TIME is provided with the system and it prompts the user to supply the time and date if and only if the system time is not yet initialized. See Appendix E.

Example:

```
[] T
YEAR: 1982
MONTH: 2
DAY: 28
HOUR: 15
MINUTE: 24
SECOND: 0
```

The command initializes the date to February 28, 1982, and the time to 3:24:00 PM.

Example Listing:

```
COS68000  VERSION 2.2     SUNDAY, FEBRUARY 28, 1982  15:24:00
```

This is the listing that would be produced from entering the example command given above. The operator should verify that the correct date and time are indicated.

UNCOUPLE <Terminal, Batch, Interrupt>

Function:

The UNCOUPLE command disassociates a nonterminal partition from its parent terminal partition. The terminal user may then enter a QUIT command, even though the nonterminal batch stream is still executing commands.

This command may be executed in either a terminal or nonterminal partition. If executed in a nonterminal partition, it disassociates the nonterminal partition from its parent terminal partition. If executed in a terminal partition, it disassociates one nonterminal child partition or all nonterminal partitions from the terminal.

Format:

```
PARTITION: [ int ]
```

Where:

PARTITION
: Number of the partition from which the partition executing the command is to be disassociated. If no partition number is specified, then all associated partitions are uncoupled.

The UNCOUPLE JDL command is most often used in a batch stream, such as a print queue server or event logger, which is intended to be independent of any terminal partition. Such a batch stream must be initiated in a nonterminal partition from some other partition. The batch stream should UNCOUPLE itself from its parent terminal partition.

RM/COS internally performs an UNCOUPLE in a terminal partition if a remote terminal is accidentally disconnected from the computer. This allows the terminal partition to be logged-out.

Example:

```
[ ] UNC
PARTITION: 101
```

The command uncouples the current (terminal) partition from (nonterminal) partition number 101.

UNLOAD <Terminal, Batch, Interrupt>

Function:

The UNLOAD command advises the system that the files on the disk volume to be removed are no longer available. Entering an UNLOAD command must precede the physical removal of a disk from the drive. Note that the system disk cannot be loaded or unloaded.

Format:

```
VOLUME: name
```

Where:

VOLUME
: Volume name of the volume or device name that contains the volume to be unloaded.

All logical name assignments must be released prior to unloading the volume; an error message is displayed if a file remains assigned. Otherwise a message is displayed on the terminal screen requesting the removal of the disk. The operator removes the disk, and presses the Acknowledge Key to continue. The message to request removal of the unloaded disk is not suppressed by the NL or NE options in batch streams; the message can be suppressed with the NDM option.

EXCEPTION

Some JDL commands, for example INITIALIZE, allow special use of a drive containing a loaded volume. Such commands request that the currently loaded volume be removed and a different volume mounted. In such cases, the UNLOAD command is not used prior to physically removing the loaded volume. While the JDL command is executing, it prevents access to files contained on the loaded disk by dedicating the disk drive. When the JDL command completes, it guarantees that the user remounts the loaded volume before again allowing access to files on the loaded volume.

Example:

```
[ ] UNL
VOLUME: GREEN
```

The command unloads volume GREEN.

VARY <Terminal, Batch, Interrupt>

Function:

The VARY command removes a peripheral device from the system, or restores to the system a device that has been removed by a VARY command. An inoperative device should be removed while it is being repaired.

Format:

```
DEVICE: name
STATUS: ON ; OFF
```

Where:

DEVICE
: Device name of the device to be removed from or restored to the system. The device cannot have any logical names assigned to it.

STATUS
: When ON is entered, or no value is entered, the device is restored to the system. When OFF is entered, the device is removed from the system.

Using the VARY command to remove an inoperative device from the system prevents attempts to use the device from degrading system performance. Using the VARY command to remove or restore the device may be simpler than reconfiguring and reloading the system.

Examples:

```
[ ] VA
DEVICE: LP01
STATUS: OFF
```

The command makes the line printer unavailable to the system.

```
[ ] VAR
DEVICE: LP01
STATUS: ON
```

The command makes the line printer available to the system.

VCOPY <Terminal, Batch>

Function:

It is a good practice to maintain a copy of every volume to use if the volume becomes unreadable or destroyed. The VCOPY command makes a physical copy of an entire volume, and verifies that the copy is correct. VCOPY will only copy volumes of the same physical format.

Because of conflicts between the RM/COS disk format and the requirements of the system boot logic on some machines, the VCOPY command may be unable to copy the system boot loader. On such machines, the Operator Guide describes how to use the I-BOOT command to install the system boot loader.

Format:

```
SOURCE VOLUME NAME: name
SOURCE DEVICE: name
DESTINATION VOLUME NAME: name
DESTINATION DEVICE: name
```

Where:

SOURCE VOLUME NAME
: Volume name of the source disk.

SOURCE DEVICE
: Device name of the disk that contains the volume to be copied.

DESTINATION VOLUME NAME
: Volume name of the destination disk.

DESTINATION DEVICE
: Device name of the disk drive for the output disk.

To copy the system disk, release any files on the disk that are assigned other than read only (RO). Release any files on the volume in the output disk drive and UNLOAD the volume. The disk volume on which the copy is to be written must be formatted; if a new disk volume is to be used, mount it in the output disk drive and enter an INITIALIZE command.

To copy a volume that is not the system disk, release any files on either disk that remain assigned to logical names. Then UNLOAD the volume on the destination disk drive. If a new disk volume is to be used for the copy, prepare it as described previously by using the INITIALIZE command before entering the VCOPY command.

The VCOPY command begins by displaying a message which requests that the operator mount the source and destination

disk volumes. Press the Acknowledge Key after the disk volumes are mounted. Each allocated ADU of the source disk volume is copied to the corresponding ADU of the destination disk volume. The logical structure and contents of the disk do not affect the copy other than the disk physical error maps.

After each ADU is copied the copy is compared to the original. Error messages relating to the copy operation or the verify operation are displayed on the error line of the terminal screen.

VCOPY generates a listing on logical file LO of informational and error messages. Nonfatal I/O errors and verification errors are listed instead of displayed on the error line. Note that logical name LO cannot be assigned on either the source or destination disk volume.

When the disk drive on which the system disk volume is mounted is used as the source or destination drive, a message is displayed after the copy and verify operation is complete. This message requests that the operator reinstall the system disk in that drive; press the Acknowledge Key after the system disk is mounted.

Example:

```
[ ] VC
SOURCE VOLUME NAME: SYSTEM
SOURCE DEVICE: DS01
DESTINATION VOLUME NAME: SYSTEMBK
DESTINATION DEVICE: DS02
```

The command copies the system disk (DS01) onto a formatted disk mounted on drive DS02, and verifies the copy. The copy disk contained in DS02 is then a backup of the system disk and may be substituted for it.

CHAPTER 4

SCREEN EDITOR

SCREEN EDITOR OVERVIEW

The screen editor is a utility program which aids the user in writing or modifying sequential files of text. This utility is primarily intended for assistance in the writing or modifying of COBOL source programs.

Four logical file names must be assigned or created before executing the screen editor. These are:

INPUT	-	the input file
OUTPUT	-	the output file
WORK	-	editor work file
PROFILE	-	profile save file

Additional logical file names may be assigned or created as necessary (see following descriptions of Insert from File and Save Edit command). The input file must be a sequential file and have a logical record length of 80 characters. The output file must be a sequential file. The editor work file must be a relative file. The profile save file must be an indexed file.

A batch stream with the pathname .SEDIT is provided. This batch stream creates the editor work and output files and assigns the input file, executes the editor, and conditionally replaces the input file with the output file. If additional files are to be used during the edit session, the assignments must be made before the use of the batch stream. See Appendix E, Vendor Supplied Batch Streams, for a description of the .SEDIT batch stream.

The screen editor makes use of several function keys for purposes such as tabbing, inserting lines, deleting lines, and erasing lines. Because RM/COS supports several keyboards, key names referenced in the following text may not match the actual keycap mnemonics of your keyboard. The key names are related to the actual keycap mnemonics of the various supported keyboards in your Terminal User Guide.

SCREEN EDITOR WALKTHROUGH

After entering the BATCH JDL command for the pathname .SEDIT, the following prompt will appear:

```
Name to edit:
```

to which the operator responds with the appropriate file pathname.

If the pathname entered does not exist or is in use by another process an error message will be displayed in the bottom right corner of the screen. Press the Acknowledge Key to clear the error text. The following prompt will appear:

```
Invalid name. Retry? (Y/N): Y
```

A response of Y will cause the previously described prompt to be displayed again; a response of N will cause the batch stream to be terminated.

If a valid pathname was entered, in a few seconds the VDT screen will be erased and the screen editor header will be displayed, identifying the version level of the editor and copyright information. At this point the editor is reading the input file and building the work file; this process may take from a few seconds to several minutes depending upon the number of records contained in the input file and the type of disk in use. When the work file is built, the first 21 lines of the input file will be displayed, and the cursor positioned on the first record. The text editor is now prepared to accept editing keystrokes and commands.

The screen editor uses the strings *BOF* and *EOF* to indicate the bounds of the file being edited. These strings of characters are not actually contained in the file and the cursor cannot be positioned on these lines.

In the special case where the original file is empty, the screen editor will display both the *BOF* and *EOF* with no intervening lines and the cursor will be on the *BOF* line. Any data entered on this line will be discarded. The editor is automatically in the compose mode (see Function 7 Key), i.e. pressing the Return Key will generate a blank line and position the cursor on that line.

Editing of text lines is accomplished by positioning the cursor to the text to be changed and then overtyping with the new data. The cursor may be moved with the Left, Right, Up or Down Arrow Keys, the Tab Right Key, or the Tab Left Key. If the text to be changed is not within the first 21 lines of the input file (i.e., the first "window"), then the

Function 1 and Function 2 Keys may be used to move the window to other 22-line windows of the file. The screen editor command Show Lines may also be used to position the window (some of the other screen editor commands position the window as a secondary result).

Screen editor commands may be entered by placing the screen editor in command mode; this is accomplished by pressing the Command Key. When command mode is entered, the prompt "CMD: ?" will appear at the bottom of the screen. The operator may then enter any of the available screen editor commands. These commands will be described in detail below and in general are used to efficiently move, delete, add or change large quantities of text in one operation. When a screen editor command finishes its action, the screen editor automatically resumes edit mode. Should a menu of available commands be desired, enter command mode and press the Return Key in response to the "CMD: ?" prompt. Should the operator desire to return to edit mode without entering a command, he should press the Command Key.

After all the text has been edited, the operator should enter command mode and type the Quit Edit (QE) command and reply Y to the prompt "Quit Edit? (Y/N)". The screen editor will then copy the updated text to the output file and return control to the JDL batch stream. The batch stream will then REPLACE the input file with the output file, ending the editing session.

NOTE

The screen editor is a COBOL program and as such the operator may interrupt the screen editor by using the Interrupt Key while in the editor command mode or edit mode.

If the screen editor is interrupted in the edit mode and then resumed by use of the CONTINUE JDL command, the operator should press the Send Key to redisplay the edit window. Text displayed on the screen during interrupt mode will replace lines in the current edit window if the cursor is moved over them prior to pressing the Send Key.

If the screen editor is interrupted in the command mode and then continued, the operator should press the Command Key.

SCREEN EDITOR PROFILE

The screen editor profile is the set of parameters which control the operation of the screen editor. Included in the profile are the following parameters:

Profile name,
Right margin,
Tab stops,
Compose mode switch,
Display line numbers switch,
Automatic word wrap at end of line switch,
Word tab switch,
Floating left margin switch,
Window roll value, and
String match character.

These parameters may be changed, saved and fetched by using the Modify Profile, Save Profile and Get Profile commands.

The profile name is used by the Save Profile and Get Profile commands to uniquely identify a particular set of parameters. The name is used as the prime key for the profile save file. The profile name consists of eight alphanumeric characters, all of which may not be equal to spaces. The default profile name is "COBOL".

The right margin value identifies the rightmost column that may be changed on a displayed line. Note that the columns to the right of the right margin may be displayed even though they cannot be changed. The right margin must have a value between 5 and 80. The default right margin value is 72.

The tab stops are a comma-separated list of numbers representing the columns to which the cursor may be positioned by use of the Tab Right and Tab Left keys. The first number in the tab stops list is used as the left margin value. (Note that unlike the right margin, however, the operator may modify the columns to the left of the first tab stop.) The numbers must be in ascending order and must be in the range between 1 and the right margin value inclusive. The default tab stops are 8,12,16,20,24,28,40,50,60,70.

The compose mode switch is used to control the use of compose mode. When the compose mode switch is equal to Y a blank line is automatically added after the current line when the Return key is pressed. The default value for the compose mode switch is N unless the input file is empty, in which case the default value is Y. The compose mode switch may be toggled by using the Function 7 key (q. v.).

The display line numbers switch controls the display format of the lines being editted. If the display line numbers switch is equal to Y the first 75 columns of each line are displayed to the left of the original line number of that line (lines that are added or moved have spaces displayed in place of the line number). If the display line numbers switch is equal to N all 80 columns of each line are displayed. The display line numbers switch is forced to have a value of N if the right margin value is greater than 75. The default value for the display line numbers switch is Y.

The automatic word wrap at end of line switch may be set equal to Y to allow continuous entry of text data without the operator having to press the Return key at the end of each line. The compose mode switch must also equal Y to enable this feature. In word wrap mode, the operator is allowed to type "beyond the right margin". When the right margin is passed, a new line is added after the current line as in standard compose mode. If the character in the right margin column is not a space, all characters to the right of the rightmost space in the current line are moved to the new line, starting at the left margin position, and replaced with spaces in the current line. The cursor is then positioned to the new line on the column immediately to the right of the moved characters. The default value for the automatic word wrap switch is N.

The word tab switch may be set equal to Y to modify the action of the Tab Left, Tab Right, Function 4 and Function 5 keys. In word tab mode, the next tab position to the left is defined to be the first character of the first word to the left of the current cursor position, if such a word exists, otherwise the space immediately to the right of the word in which the cursor is currently positioned, otherwise the next column tab as defined by the tab stop list. The next tab position to the right is defined in a similar manner. Words are defined to be any set of characters delimited by spaces. The default value for the word tab switch is N.

The floating left margin switch is used to determine the positioning of the cursor in cases such as the addition of new lines. When the floating left margin switch is equal to Y and the cursor is to be positioned on the left margin of a line, the left margin is considered to be the same column as the leftmost nonblank character in the line displayed above the line on which the cursor is to be positioned; if the line displayed above is all spaces, the line on which the cursor is to be positioned is the topmost line displayed on the screen, or the floating left margin switch is equal to N, the cursor will be positioned at the column indicated by the first tab stop. The default value for the floating left margin switch is N.

The window roll value is the number of lines the window is moved when using the Function 1 and Function 2 keys (q. v.). The window roll value must be greater than zero and less than 23. The default for window roll value is 11.

The string match character is used in the target and replacement strings described in the Find String and Replace String commands. The string match character may be any character except a digit, space or upper case letter. The default value for the string match character is ^.

The screen editor profile is initialized to the default values at the beginning of an edit session unless the synonym "SE$PFILE" has been defined in the partition in which the editor is running. If this synonym can be resolved to a value which has eight or fewer characters and the value of the synonym is not equal to "SE$PFILE" (i.e. the synonym does not resolve to itself) then the editor will simulate a Get Profile command using the value of "SE$PFILE" as the profile name desired.

EDIT MODE

The screen editor is in edit mode at any time that command mode has not been invoked by the operator. While in edit mode the cursor may be positioned anywhere within the 22-line window; text changed; single lines added and deleted; lines split, joined and centered; and the window moved to adjacent portions of the input file. In this mode, many of the noncharacter keys have special meanings which allow the operator to achieve these functions. The following table describes the action for each noncharacter key by key name; see your Terminal User Guide to relate the key name to the actual keycap mnemonic on your terminal keyboard.

KEY NAME	ACTION
Function 1	Scroll window up (i.e. move toward end-of-file) by the number of lines specified by roll value (see MP command)
Function 2	Scroll window down (i.e. move toward beginning-of-file) by the number of lines specified by roll value (see MP command)
Function 3	Add one blank line before the line upon which the cursor is currently positioned; move cursor to the first tab position of the added line (same as Insert Line Key)
Function 4	Duplicate the previous line (which must be displayed on the screen) from the current cursor position until the next tab stop is encountered; position cursor at that tab stop
Function 5	Replace the characters from the current cursor position until the next tab stop with spaces; position cursor at that tab stop
Function 7	Toggle compose mode on and off (see Return Key); screen editor originally has compose mode off unless input file is empty

Function 8	Bring up as many words as possible from the following line (while preserving spacing) and concatenate them at the end of the current line starting at the current cursor position or after the first space to the right of the rightmost word, whichever is farther right; delete the words moved from the following line; if there are words remaining on the following line, move those words to the left so that the first character of the first remaining word is at the same character position previously occupied by the first word moved; if the following line is equal to spaces (including those columns beyond the right margin), delete it
Function 9	Split the current line; move all characters from the current cursor position to the right margin inclusive to a new line added after the current line; place the first moved character at the left margin of the new line; replace the moved characters with spaces on the current line; do nothing if all characters to the left of the cursor are equal spaces or the cursor is positioned in column one
Function 10	Center the text of the current line between column one and the right margin
Command	Activate editor command mode
Right Arrow	Move cursor one character position to the right
Left Arrow	Move cursor one character position to the left
Up Arrow	Move cursor up one line; if on top line of window scroll window down one line; if on first line of file, no action
Down Arrow	Move cursor down one line; if on bottom line of window, scroll window up one line; if on last line of file, no action
Home	Move cursor to the left margin of the top line of the window
Tab Right	Move cursor to the next tab stop to the right; if on the right margin of the line, no action

Erase Right	Replace all characters from the cursor position to the right margin of the line with spaces
Erase Field	Replace all characters on the current line with spaces; move cursor to the left margin of the line
Delete Line	Delete the current line moving succeeding lines in window; move cursor to the left margin of succeeding line or, if no succeeding line, of preceding line
Insert Line	Add one blank line before the line upon which the cursor is currently positioned; move cursor to the left margin of the added line (same as Function 3 Key)
Insert Char	Add characters before the current cursor position
Delete Char	Delete the character under the cursor, moving characters to the right of the cursor left one character position
Tab Left	Move cursor to the next tab stop to the left; if on column one of a line, no action
Return	If compose mode (see SCREEN EDIT PROFILE section) is off, move cursor to the left margin of the next line, moving window up one line if necessary; if compose mode is on, add a blank line after the current line and move cursor to the left margin of the added line
Send	Redisplay window; used to reenter the screen editor after an interruption by JDL interrupt mode; text displayed on the screen during interrupt mode will replace lines in the current edit window if the cursor is moved over them prior to pressing the Send Key

COMMAND MODE

Command mode of the screen editor is selected by pressing the Command Key. The screen editor will display the current cursor column number in the lower right corner of the screen, and then will prompt with "CMD: ?" in the lower left corner of the screen to which the operator may reply with any of the abbreviations from the following table, followed by pressing the Return Key.

ABBREVIATION	COMMAND
CL	Copy Lines
CM	Clear Marker
DL	Delete Lines
FS	Find String
GP	Get Profile
IF	Insert from File
KE	Kill Edit
ML	Move Lines
MM	Modify Marker
MP	Modify Profile
QE	Quit Edit
RE	Resume Edit
RS	Replace String
SE	Save Edit
SL	Show Lines
SM	Show Markers
SP	Save Profile

Lower case letters may be substituted for upper case letters in the command abbreviation. Each command may then display additional prompts which are specific to that command.

Many of the screen editor commands prompt for line designations. Except for two instances in the Insert from File command, these line designations are of the general form:

[<line number>][± <offset>]

Where:

<line number> is the number of an undeleted original line or a valid line marker character, and

<offset> is the number of lines before (-) or after (+) the line referenced by <line number>

When <line number> is omitted, the current line (i.e. the line upon which the cursor was positioned before command mode was selected) is the line referenced. Each line designation is verified for correctness as it is entered.

Lines may be referenced by line markers. Twenty-six line markers are available, designated by the letters of the alphabet. Two of the markers, B and E, are reserved to indicate the beginning line and ending line, respectively, of the file. The other twenty-four markers may be modified by the operator to indicate any line in the file. Both lower case and upper case letters may be used to refer to line markers.

Violation of the rules specified for the various commands may cause the word "Invalid" to appear in the bottom right corner of the screen. The operator must respond by pressing the Return Key and then reentering the command or datum as necessary, correcting the cause of the invalid condition.

Some of the noncharacter keys have special meaning while responding to command prompts. The following table describes the action for each noncharacter key by key name; see your Terminal User Guide to relate the key name to the actual keycap mnemonic on your terminal keyboard.

KEY NAME	ACTION
Erase Field	The response area for the prompted parameter is erased and the cursor moved to the first position of the area
Left Arrow	Move cursor one character position to the left
Right Arrow	Move cursor one character position to the right
Insert Char	Add characters before the cursor position
Delete Char	Delete the character under the cursor moving all characters in the area to the right of the cursor left one character position
Return	Move cursor to response area for the next prompted parameter if one exists, or execute command
Erase Right	Replace all characters from the cursor position to the rightmost character position of the response area with spaces; move cursor to response area for the next prompted parameter if one exists, or execute command
Command	Return to edit mode

Up Arrow	Move cursor to response area for the previous prompted parameter if one exists, or reprompt entire command
Tab Left	Move cursor to response area for the previous prompted parameter if one exists, or reprompt entire command
Home	Reprompt entire command

The Find String and Replace String commands use two types of strings as parameters which differ in the use of the string match character. Both a <target string> and a <replacement string> use the string match character to delimit trailing spaces, while a <target string> also uses the string match character to designate a "wild card" character position which will match any character. The string match character is initially ^ but may be changed by the Modify Profile editor command. String parameters are processed as follows:

1. All characters to the right of the rightmost nonblank character are truncated from the string, and

2. If the rightmost nonblank character is the string match character, then the rightmost nonblank character is truncated from the string.

For example, if the string match character has been changed to ? (see MP command), the string ABC followed by three blanks would be entered as ABC ?. If a <target string> is to match A followed by any character followed by C, then the string A?C would be entered. If a <target string> is to match A followed by any two characters, then the string A??? would be entered (note that the last string match character will be truncated according to rule 2 above).

On the following pages the screen editor commands are described. For each description, the specific parameters for that command, general rules and action taken are listed. The commands are listed in alphabetical order by command abbreviation.

CL - Copy Lines

Prompts: Begin Line: <line designation>
Thru Line: <line designation>
After Line: <line designation>

Rules:

1. The line designated by Begin Line must not come after the line designated by Thru Line.

Action:

A copy of the lines from Begin Line to Thru Line inclusive is made and inserted after the line designated by After Line. Line markers indicating any of the lines from Begin Line to Thru Line are not modified. Note that the original area which was copied is not deleted.

To copy lines and insert the copy before the beginning line of the file, position the file to line B using the SL command, add a line by pressing the Insert Line Key (see Edit Mode description), copy the lines after line B, and then delete Line B using the DL command or the Delete Line Key.

The starting line of the window is set to After Line and the window is redisplayed.

CM - Clear Markers

Prompt: Marker Letter: <marker designator>

Rules:

1. The <marker designator> may be any upper or lower case letter except B, b, E or e.

Action:

The designated line marker is returned to the "unused" state. The prompt is reissued, allowing the operator to clear more than one marker without retyping the entire command. When the operator has cleared all the markers intended, then the Command Key may be pressed to return to edit mode.

DL - Delete Lines

Prompts: Begin Line: <line designation>
Thru Line: <line designation>

Rules:

1. The line designated by Begin Line must not come after the line designated by Thru Line.

Action:

The lines from Begin Line to Thru Line inclusive are deleted. Line markers indicating any of the lines from Begin Line to Thru Line become invalid until modified.

The starting line of the window is set to the line preceding Begin Line and the window is redisplayed.

FS - Find String

Prompts: Begin Line: <line designation>
Thru Line: <line designation>

(new screen)

Value: <target string>
Start Column: <column number>
End Column: <column number>

Rules:

1. The line designated by Begin Line must not come after the line designated by Thru Line.

2. Start Column must be greater than zero and not greater than the rightmost edit character position (i.e. 72 or 80, depending on the Function 6 toggle; see Edit Mode).

3. End Column must be not less than Start Column and not greater than the rightmost edit character position.

4. The length of the Value string must be greater than zero and not greater than the value (End Column - Start Column + 1).

Action:

The lines from Begin Line to Thru Line are searched sequentially for a line that has the specified Value string totally contained in the columns between Start Column and End Column. When such a line is found, the starting line of the window is set to the line containing the string and the window is displayed (i.e. the top line on the screen contains the desired Value string). The cursor is then positioned in the lower right corner of the screen to await operator action.

If the operator wishes to terminate the search at this point, he must type the Command Key; the cursor will then be positioned at the first tab stop of the top line and the screen editor returns to edit mode. If the operator wishes to continue the search, he must type the Return Key; the search will then be continued starting with the line after the top line. When the end of the line range is encountered, the starting line of the window is set to Thru Line and the window is redisplayed.

GP - Get Profile

Prompts: Profile Name: <profile name>

Rules:

1. The <profile name> must be an eight-character string which was the name of a profile that was previously saved with the Save Profile command.

Action:

The profile save file is opened and read with <profile name> used as the key value. If the profile name exists the screen editor profile is modified, the window is redisplayed and edit mode is resumed.

If the profile name does not exist or any other error occurs a diagnostic message will be displayed at the bottom of the screen. The operator must press the Return key to acknowledge the message, whereupon the the window is redisplayed and edit mode resumed without modification of the screen editor profile.

IF - Insert from File

Prompts:
```
Insert File: <logical file name>
After Line:  <line designator>
Start Line:  <line number>
End Line:    <line number>
```

Rules:

1. The logical file name entered for Insert File must have been ASSIGNed prior to the execution of the screen editor.

2. Start Line and End Line are line numbers of the inserted file; if nonblank, they must be decimal numbers.

3. If Start Line is blank, a value of 1 is assumed.

4. If End Line is blank, EOF of the inserted file is assumed.

5. End Line, if nonblank, must not be less than the (assumed) value for Start Line.

Action:

The Insert File is positioned to the line indicated by Start Line. A copy of the Insert File lines from Start Line to End Line (or EOF, whichever comes first) is made and inserted after the line designated by After Line.

The starting line of the window is set to After Line and the window is redisplayed.

KE - Kill Edit

Prompt: Kill Edit? (Y/N) <"Y" or "N">

Rules:

1. Any response to the Kill Edit prompt except "Y" or "y" will result in a return to edit mode.

Action:

If the response to the Kill Edit prompt is "Y" or "y", all modifications are discarded and the screen editor program is terminated. However, this command does not discard the results of any Save Edit (SE) commands during the edit session. If the last Save Edit command completed successfully to the logical file OUTPUT, the condition code is unchanged; otherwise it is set to the character Z.

When using the standard batch stream (.SEDIT), if the last Save Edit command completed successfully to the file OUTPUT, the input file is replaced by the file OUTPUT. Otherwise the file OUTPUT is left assigned. The user may release logical file OUTPUT (which will discard any edits which may have been saved in that file), or use it to replace a cataloged file via the REPLACE JDL command.

ML - Move Lines

Prompts: Begin Line: <line designation>
Thru Line: <line designation>
After Line: <line designation>

Rules:

1. The line designated by Begin Line must not come after the line designated by Thru Line.

2. The line designated by After Line must not be in the range of lines from Begin Line to Thru Line inclusive.

Action:

The lines from Begin Line to Thru Line are removed from the file and reinserted after the line designated by After Line. Any lines within the range which were original lines will lose that designation. Line markers indicating any of the lines from Begin Line to Thru Line will continue to indicate those lines in the new position.

The starting line of the window is set to After Line and the window is redisplayed.

MM - Modify Marker

Prompt: Marker Letter: <marker designator>

Rules:

1. The <marker designator> may be any upper or lower case letter except B, b, E or e.

Action:

The designated line marker is set to indicate the line upon which the cursor was positioned when command mode was selected.

MP - Modify Profile

Prompt: (new screen)

```
        Profile Name: <profile name>
        Right Margin: <right margin>
           Tab Stops: <tab stop list>
        Compose Mode: <"Y" or "N">
     Display Line #'s: <"Y" or "N">
           Word Wrap: <"Y" or "N">
        Word Tabbing: <"Y" or "N">
    Float Left Margin: <"Y" or "N">
           Roll Value: <roll value>
    String Match Char: <character>
```

Rules:

1. <profile name> is one to eight alphanumeric characters.
2. <right margin> is an integer in the range of 5 through 80.
3. <tab stop list> is a comma-separated list of one to twenty integers. Each integer must be greater than zero, less than <right margin>, and, if not the first integer in the list, greater than the previous integer.
4. <roll value> is an integer which must be greater than zero and less than 23.
5. <character> may be any character except a digit, space, or upper case letter.

Action:

The current editor profile is displayed along with the prompts. The operator may change any or all of the values as needed.

Each parameter value is validated when the operator presses a key which terminates data entry for the field. If a field is found to be incorrect, the word "Invalid" will be displayed in the bottom right corner of the screen; the operator must press the Return key to acknowledge the message.

The operator may press the Command key at any time to terminate the Modify Profile command. The editor profile at termination is as displayed; any changes made will remain in force. The Modify Profile command is terminated automatically when the Return key is pressed in the <character> field.

The operator must enter the Save Profile command to retain the profile for future edit sessions; the Modify

Profile command changes the "in memory" profile only.

QE - Quit Edit

Prompt: Quit Edit? (Y/N) <"Y" or "N">

Rules:

1. Any response to the Quit Edit prompt except "Y" or "y" will cause the screen editor to return to Edit Mode.

Action:

If the response to the Quit Edit prompt is "Y" or "y", the modified file is written to the file with logical file name OUTPUT. The JDL condition code is unchanged and the screen editor program is terminated.

RE - Resume Edit

Prompts: None

Rules: None

Action:

The screen editor returns to edit mode. This command is identical to typing the Command Key in response to the "CMD: ?" prompt, and is included for operator convenience.

RS - Replace String

Prompts:

```
Begin Line:    <line designation>
Thru Line:     <line designation>

(new screen)

Value:         <target string>
Replace:       <replacement string>
Start Column:  <column number>
End Column:    <column number>
```

Rules:

1. The line designated by Begin Line must not come after the line designated by Thru Line.

2. Start Column must be greater than zero and not greater than the rightmost edit character position (i.e. 72 or 80, depending on the Function 6 toggle; see Edit Mode).

3. End Column must be not less than Start Column and not greater than the rightmost edit character position.

4. The length of the Value string must be greater than zero and not greater than the value (End Column - Start Column + 1).

Action:

The lines from Begin Line to Thru Line are searched sequentially for a line that has the specified Value string totally contained in the columns between Start Column and End Column inclusive. When such a line is found, the starting line of the window is set to the line containing the Value string and the window is displayed (i.e. the top line on the screen contains the desired Value string). The same line after the substitution of the Replace string for the Value string is displayed at the bottom of the screen; the line is displayed with the same alignment as those lines in the window. The cursor is then positioned at the first character of the Replace string. The operator may edit the new line and then press one of the following keys, depending upon which action the operator desires.

KEY	ACTION
Command	Do not apply change, terminate search
Function 1	Do not apply change, continue search
Function 2	Apply change, terminate search
Return	Apply change, continue search
Function 3	Apply change, continue search, and make subsequent substitutions without operator intervention, continue to display changed lines at the bottom of the screen

If the operator has chosen to terminate the search the window is redisplayed (showing the changed line, if the change was applied), the cursor placed at the first tab position of the top line, and the screen editor returns to the edit mode. If the operator has chosen to continue the search, the search will continue from the first character after the Value string; thus, multiple substitutions on the same line may be made, but recursive substitutions are not possible. When the end of the line range is encountered, the starting line of the window is set to Thru Line and the window is redisplayed.

When the Value string and the Replace string do not have the same length, then all characters to the right of the substituted string until the last edit character position are moved left with blank fill or right with truncation as necessary.

SE - Save Edit

Prompts: Save File: <logical file name>

(after successful save)

Renumber Source? (Y/N) <"Y" or "N">

Rules:

1. The logical file name entered for Save File must have been ASSIGNed Exclusive All (EA) prior to the execution of the screen editor.

2. Any response to the Renumber Source prompt except "Y" or "y" will result in a return to the edit mode without renumbering.

Action:

The modified file is written to the file specified by the response to the Save File prompt. After successful completion of the save, the Renumber Source prompt is displayed. If the response to Renumber Source is "Y" or "y" the saved file is used to reinitialize the screen editor work file, thus assigning new line numbers to each line of the file and clearing all line markers.

When using the standard batch stream (.SEDIT), the logical file OUTPUT is a scratch file. Thus, if the system were to stop due to loss of power, edits saved on this file would be lost, even though the Save Edit command completed successfully.

SL - Show Lines

Prompt: Begin Line: <line designation>

Rules: None

Action:

The starting line of the screen window is set to the line entered as the response to the Begin Line prompt. The window is redisplayed.

SM - Show Markers

Prompt: None

Rules: None

Action:

The current value of all line markers is displayed, and the message "Press RETURN to Acknowledge" is displayed in the lower right corner. Each marker may be in one of three states: UNUSED, DELETED LINE or active. UNUSED indicates that no MM command has been entered for that marker since initialization, a Save Edit with renumbering or a CM command was entered for that marker. DELETED LINE indicates that the line which that marker was set to indicate has been deleted by a Delete Lines command or by the Delete Line Key. If a marker is active then a value for that marker will be shown in the following form:

<line number>[+<offset>]

where <line number> is an original line number or the character B and <offset> is the number of lines beyond <line number> of a nonoriginal line. If the character B is used, then no original line exists between the marked line and the first line of the file. To return to edit mode, the operator presses the Return Key.

SP - Save Profile

Prompt: None

Rules: None

Action:

The current editor profile is saved on the profile save file using the current profile name as its identifier. If a profile with the same name already exists on the profile save file, it will be replaced with the current profile.

CHAPTER 5

LINE EDITOR

LINE EDITOR OVERVIEW

The line editor is a utility program which aids the user in writing or modifying sequential files of text. The utility is most useful for assistance in writing or editing source code files; it may be used to assist in writing or editing any sequential file containing textual material.

The line editor uses the following logical names:

EWF	Editor work file
CI	Command input device
CO	Command output device
List	Listing file or device
Input	Input file
Output	Output file

The logical names for input, output and list are supplied by the user and there may be more than one file for each purpose, but each must have a unique logical name. These files must be sequential organization and have a logical record length of 80 characters. All input and output records are blank padded to 80 characters. It is recommended that these files be of COMPRESSED type when possible.

Before executing the line editor, the editor work file must be created, and all the output and list files must be created if necesary. All logical names must be assigned before the editor is invoked. A batch JDL file has been supplied for this purpose (see Appendix E, Vendor Supplied Batch Streams).

The editor work file must be a relative record file with a record length of 990 (decimal) characters. It may be WORK or SCRATCH type, but must not be COMPRESSED or SPOOL type. It is recommended that the editor work file be of SCRATCH type.

The size of the file which can be edited depends on the memory space available to the line editor in the partition after it is loaded and the work file, command input and command output buffers are allocated. Memory space is reserved for the largest block size of all the remaining unopened sequential files assigned with write access in the partition. The largest remaining block of available memory in the partition is then allocated to table space by the editor. Every four bytes of table space allocated will accomodate 12 records (one work file record) under optimal conditions. Line insertions, deletions, moves and duplications may fragment the editor work file and reduce the maximum number of records available. String operations do not affect work file fragmentation; saves and concatenates do not recover unused work file space.

EDITOR COMMAND

The EDITOR command is the JDL command that calls the text editor utility. It has no parameters. The command transfers control to the text editor, which displays a heading identifying the utility, and the prompt for a command:

```
C?
```

If the batch stream .EDIT was used and an existing file is to be edited, the command C0 EDITFILE should be entered at this time.

COMMAND OPERANDS

Most of the line editor commands have one or more operands. The types of operands specified for the commands are described in the following paragraphs.

Line Number

A line number refers to a numbered line in the work file. A line number is an integer, 1 through the number of lines that have been read into the file.

Lines are read into the work file by entering a CONCATENATE command. When the first lines are read into the empty work file, they are numbered consecutively. If the file is a COBOL source file, the numbers correspond to those in the COBOL compiler listing. Any of these numbers may be used as a line number operand.

START, END, and $ are all valid line number operands. START refers to the first line in the work file. END refers to the last line in the work file. The $ refers to the current line. The current line is the line being operated on. The current line is displayed by each command prior to the prompt for another command.

The nonnumeric line number operands are necessary because not all lines have numbers. The lines entered by an INSERT command do not have numbers; neither do additional lines read into the work file by CONCATENATE commands. The character $ may be used in expressions as line number operands. For example:

$+2	The second line below the current line
$-5	The fifth line above the current line

Line numbers in this form are called relative line numbers in this chapter. Line numbers that are integers are called absolute line numbers.

Line number pair

A line number pair defines a range of line numbers beginning with the left number and ending with the right number inclusive. The two line numbers of a line number pair are separated by a comma (,), a hyphen (-) or a period (.). Two absolute line numbers or two relative line numbers may be used as a pair, but not one of each. START and END are considered to be absolute line numbers. That is, START,35 is a valid line number pair, as is 23,END. However, $,END is not valid.

A single line number may be used as a line number pair, and it defines a range of one line. ALL is a valid line number pair. It is equal to START,END.

Zone Number Pair

A zone number pair defines a range of columns in a line. It consists of two column numbers, separated by a comma (,) a hyphen (-) or a period (.). The zone includes both specified columns. For example, 1,6 entered as a zone number pair defines the sequence number field of a COBOL source statement. The first number in a zone number pair must be less than the second and both must lie in the range of one to eighty inclusive.

Number

A number operand is a positive integer used to define a tab stop, or to specify a number of lines in the work file.

Logical Name

A logical name operand is an RM/COS logical name that identifies an input or output file, or a printing device. A logical name used as an operand must have been assigned prior to calling the line editor.

COMMANDS

The first two letters of any command name may be used as a valid abbreviation for that command. The operands for the command, if any, are shown after the command name. Some operands are optional. These operands are enclosed in brackets ([ ]). Parentheses are required syntax when shown enclosing command operands. The operand type is specified for each operand. Operands are separated by a blank. A command may be entered whenever the command prompt is displayed.

When a required operand is omitted, or when a line number pair contains mixed line number types, the line editor displays the following message:

```
** ILLEGAL COMMAND **
```

Re-enter the command correctly.

ABORT

The ABORT command is entered as follows:

```
AB
```

The ABORT command terminates the line editor. Any editing performed since the most recently executed SAVE command is lost.

The ABORT command is appropriate to use to terminate the line editor when no change in the file being edited has been made.

CHANGE

The CHANGE command is entered as follows:

```
CH [line number]
```

The CHANGE command displays the specified line, enters the insert mode, and displays the insert prompt, as follows:

```
I?
```

The operator enters one line of text and presses the RETURN key. The newly entered line replaces the displayed line, and the line editor returns to the command mode.

If no line number operand is entered, the current line is replaced by the newly entered line.

The following are examples of the CHANGE command:

```
CH 22
```

This command displays line 22, and replaces it with a line entered by the user.

```
CH $-5
```

This command displays the fifth line before the current line, and replaces it with a line entered by the user.

CONCATENATE

The CONCATENATE command is entered as follows:

```
CO [(line number)] logical name [(line number pair)]
```

The CONCATENATE command copies the file assigned to the logical name to the work file.

If a line number operand is entered, the lines (records) read from the file are written to the work file following the specified line. Otherwise, the lines are written following the current line. If the work file is empty, the first line read becomes the first line in the work file and the lines are numbered as they are written to the work file.

If the line number pair operand is entered, only the lines defined by the pair are read into the work file. Otherwise, the entire file is read.

Lines from several files may be read into the work file. Each file must be assigned to a unique logical name and the appropriate logical name must be entered in each CONCATENATE command. When a CONCATENATE command reads lines into a work file that already contains lines the added lines are not numbered.

The following are examples of CONCATENATE commands:

```
CO INPUT
```

This command reads all the lines of the file assigned to logical name INPUT. The lines are read into the work file following the current line. If the work file is empty the first line in the file becomes line 1 in the work file. This form of the CONCATENATE command issued in editing an existing file.

```
CO ($+15) IN1 (38-41)
```

This command reads lines 38, 39, 40, and 41 from the file assigned to IN1 and places them following the 15th line below the current line. If the 15th line below the current line is not the last line in the work file, the added lines are placed between the 15th and 16th lines below the current line.

DELETE

The DELETE command is entered as follows:

```
DE [line number pair]
```

The DELETE command removes the specified lines from the work file. If no line number pair is entered, the current line is deleted.

When the line number pair entered with a DELETE command is ALL (or the equivalent) the line editor displays the following message:

```
CANCEL = 'X'
```

The operator may cancel the command (leaving the contents of the work file intact) by entering an X and pressing the Return Key, or the operator may press only the Return Key and execute the command.

The following are examples of the DELETE command:

```
DE 6,9
```

This command removes the lines beginning with line 6 and ending with line 9, from the work file. Unnumbered line between lines 6 and 9 will also be removed.

```
DE
```

The command deletes the current line.

DISPLAY

The DISPLAY command is entered as follows:

```
DI [line number pair]
```

The DISPLAY command displays the specified lines on the screen of the terminal. When no line number pair is entered, the current line is displayed.

The following are examples of the DISPLAY command:

```
DI ALL
```

This command displays all the lines in the work file. The lines are displayed beginning with the first line in the file, one screen at a time. The following message is displayed at the bottom of each screen.

```
CANCEL = 'X'
```

The operator may terminate the command by entering an X and pressing the Return Key or continue the display by pressing only the Return Key.

```
DI $,$+5
```

The command displays the current line and the five lines following the current line.

DUPLICATE

The DUPLICATE command is entered as follows:

```
DU [(line number pair) [line number]]
```

The DUPLICATE command copies the lines in the work file specified by the line number pair. The copy of the lines is written in the work file following the specified line number. If no line number is entered, the copy is placed in the work file following the current line. If no line number pair is entered, the current line is duplicated. The copied lines are not deleted and the duplicate lines inserted into the work file have no line numbers.

Inserting duplicated lines within the range of lines being duplicated is allowed. Only the original lines specified by the line number pair are duplicated to prevent an endless loop situation.

The following are examples of the DUPLICATE command:

```
DU (1,15) 25
```

This command copies lines 1 through 15 following line 25 in the work file. If line 25 is not the last line in the file, the copies of the lines are inserted between lines 25 and 26 of the file. Lines 1 through 15 are not deleted or altered in any way.

```
DU ($-10,$-5)
```

This command copies six lines starting with the tenth line before the current line, placing the copies following the current line.

FIND

The FIND command is entered as follows:

```
FI line number
```

The FIND command locates and displays the specified line.

The following are examples of the FIND command:

```
FI START
```

This command displays the first line in the work file.

```
FI $+50
```

This command finds and displays the 50th line after the current line.

INSERT

The INSERT command is entered as follows:

```
IN [line number]
```

The INSERT command places the line editor in the insert mode, and it displays the insert prompt:

```
I?
```

Whatever the operator enters in response to the insert prompts is written in the work file on the line following the specified line number. If no line number is entered, the new lines are written following the current line. If the work file is empty and an INSERT command is entered with no line number, the first new line is written as the first line of the work file.

As many lines may be entered as required. Pressing the Return Key begins a new line. The insert prompt is then repeated. When the last line has been entered, the operator must enter an exclamation point (!) as the first and only character of a line to return to the command mode.

When composing a file, the INSERT command with no line number is the first command entered. The operator enters the lines required, then enters an exclamation point to leave the insert mode.

When editing a file, the INSERT command is entered with an absolute or relative line number. The lines entered are written in the work file following the specified line.

The following are examples of the INSERT command:

```
IN 13
```

This command places the line editor in the insert mode. The text lines entered following the insert prompt are written as unnumbered lines following line 13 in the work file. If line 13 is not the last line in the file, the new lines are placed between line 13 and the subsequent lines of the file.

```
IN $
```

This command places the line editor in the insert mode, to insert one or more lines following the current line. When editing an area of a file that has no numbered lines use a POSITION or FIND command to locate the current line at the desired point and then use this form of the INSERT command to insert lines.

MOVE

The MOVE command is entered as follows:

```
MO (line number pair) [line number]
```

The MOVE command moves the lines specified by the line number pair, placing them in the work file following the specified line number. If a line number is not specified, the lines are moved to follow the current line. The moved lines have no absolute line numbers after being moved. The command removes the specified lines from the original position.

The following are examples of the MOVE command:

```
MO (1-5) END
```

This command moves lines 1 through 5 to follow the last line in the work file. The first line in the file is now line 6 (or the unnumbered line after line 5.)

```
MO ($+3)
```

This command moves the third line following the current line to a position between the current line and the line after the current line.

The destination line number may not be within the range of lines to be moved. A destination line number immediately preceding or following the range specified, while not actually disallowed, serves only to reduce work file efficiency and remove line numbers and should be avoided.

POSITION

The POSITION command is entered in one of the following forms:

```
PO [+|-]number
PO +|-
+|-[number]
```

The POSITION command displays a line relative to the current line, which then becomes the current line. If the command name and a number are entered, the number is considered to be positive. If the sign alone is entered, the adjacent line is displayed, and becomes the current line. A plus sign (+) moves the current line designation forward and a minus sign (-) moves the current line designation backward in the work file by the number of lines indicated or implied.

The following are examples of the POSITION command:

```
PO 3
```

This command displays the third line following the current line.

```
- 5
```

This command displays the fifth line before the current line.

```
PO+
+
```

Either command displays the line after the current line.

PRINT

The PRINT command is entered as follows:

PR [(line number pair)] logical name

The PRINT command prints one or more lines from the work file on the printer or spool file assigned to the logical name with pagination headers. If a line number pair is entered, the specified lines are printed. If no line number pair is entered, the current line is printed.

The following are examples of the PRINT command:

PR (ALL) LO

This command prints all the lines in the work file on the printer assigned to logical name LO.

PR ($,$+6) LO

This command prints the seven lines beginning with the current line.

QUIT

The QUIT command is entered as follows:

```
QU [logical name]
```

The QUIT command writes the lines in the work file to the file assigned to the logical name and terminates the line editor. The logical name must be entered if no SAVE command has been executed during the editing session. If no SAVE commands have been executed and a QUIT command is entered without a logical name, the command is not accepted and the line editor does not terminate. If a SAVE command has been executed and a QUIT command is entered without a logical name, the line editor terminates without writing the work file contents.

The previous contents of the file assigned to the logical name are replaced by the work file contents.

NOTE

All lines in the work file not included in a previous SAVE command will be lost if the QUIT command is entered without a logical name.

The following is an example of a QUIT command:

```
QU OUT
```

This command writes the current work file contents to the file assigned to logical name OUT and then the line editor terminates.

SAVE

The SAVE command is entered as follows:

SA [(line number pair)] logical name

The SAVE command writes the specified line of the work file to the file assigned to the logical name. When the line number pair is not entered, all the lines in the work file are written. Lines written by the SAVE command remain in the work file.

Any previous contents of the file assigned to the logical name, including those resulting from a previous SAVE command, are replaced by the lines written from the work file.

The following are examples of SAVE commands:

SA (1,250) OUT

This command writes lines 1 through 250 of the work file to the file assigned to logical name OUT.

SA ($,$+100) OUT

This command writes the current line and the next 100 lines to the file assigned to logical name OUT.

SEARCH

The SEARCH command is entered as follows:

```
SE [(line number pair)] delimiter  string  delimiter
   [(zone number pair)]
```

The SEARCH command searches a specified group of lines for a specified string. The line number pair specifies the lines to be searched. If no line number pair is entered, only the current line is searched.

The string to be searched for is enclosed between single character delimiters. The delimiter may be any character that is not in the string other than blank and left parenthesis.

The search may be limited to only a portion of the line by specifying a zone number pair. Only the columns included in the pair are searched.

When the string is found (in the zone, if one is specified), the line is displayed and becomes the current line.

When the search does not find the specified string, the line editor displays the following message:

```
** ITEM NOT FOUND **
```

The last line searched is then displayed and becomes the current line. The following are examples of the SEARCH command:

```
SE (ALL) 'PROCEDURE DIVISION.' (8,26)
```

This command searches the entire work file for the heading of the procedure division. The search is limited to columns 8 through 26 of each line.

```
SE (8-50) *03*
```

This command searches lines 8 through 50 for the string "03" in columns 1 through 80 of those lines.

STRING

The STRING command is entered as follows:

```
ST [(line number pair)] delimiter string-1 delimiter
  string-2 delimiter [(zone number pair)] [G]
```

The STRING command searches for a string of characters and replaces it with another string of characters. The line number pair defines the lines to be searched. If no line number pair is entered the current line is searched.

The string to be searched for (string-1) is enclosed between single character delimiters. The delimiter may be any character that is not in either string other than blank or left parenthesis. The replacement string (string-2) may be shorter or longer than string-1. If string-2 is long enough to cause one or more characters of string-2 to be truncated at column 80 in a line, the replacement is not made. The replacement string (string-2) may be empty, causing string-1 to be deleted. String-1 may also be empty, causing string-2 to be inserted at the start of the zone, if one was specified, or at the start of each line of the indicated range.

The zone number pair defines a zone of columns in which the search is to be made. The letter G specifies the global option. If the global option is in effect, all occurrences of string-1 in a line are replaced; otherwise only the first occurrence of string-1 in a line is replaced.

When the search does not find the specified string, the line editor displays the following message:

```
** ITEM NOT FOUND **
```

The last line searched is then displayed and becomes the current line. The following are examples of the STRING command:

```
ST (1,75) "FIEL"FILE" G
```

This command searches lines 1 through 75 of the work file, replacing all occurrences of the string FIEL in each line with the string FILE.

```
ST (35,END) /WORK1/WORK2/ (16,72)
```

This command searches columns 16 through 72 of the work file, beginning with line 35, changing the first occurrence of WORK1 on a line to WORK2.

TAB

The TAB command is entered as follows:

```
TA [tab character[,number[,number]...]]
```

The TAB command, with both operands, specifies a tab character and sets the tab stops. With only a tab character operand, the TAB command changes only the tab character; the tab stops in effect remain set. The TAB command with no operands clears all tab stops.

Tab characters are displayed on the screen as entered but the character to the right of the tab character is written to the work file at the position of the next tab stop. A character that is not used in the text being entered should be chosen as the tab character. Tabs only apply to the insert mode of the line editor.

At least one column number but no more than eight column numbers may be specified as tab stops. Column numbers must be entered in ascending order. Tab stops are similar to the tab stops on a typewriter; the stop with a column number higher than the column number of the tab character applies. If more tab characters are entered than are defined, the remainder of the insert line is discarded.

The following is an example of a TAB command:

```
TA #,7,8,12,16
```

This command specifies the pound sign (#) as the tab character, and sets tab stops at columns 7, 8, 12, and 16. These represent the indicator column, the first column of the A area, the first column of the B area and the first column of a four-column indentation into the B area of the COBOL source format.

The following are examples of the use of these tab stops:

```
##IDENTIFICATION DIVISION.
```

The statement is written in the work file without the tab characters and with the division header starting in area A:

```
1      7    12  16      (column positions)
       IDENTIFICATION DIVISION.
```

A comment could be entered as follows:

```
#* COBOL COMMENTS HAVE A * IN THE INDICATOR COLUMN
```

The record in the file is as follows:

```
1      7    12  16      (column positions)
       * COBOL COMMENTS HAVE A * IN THE INDICATOR COLUMN
```

An example with four tab characters:

```
    ####03  ITEM-3A   PIC X.
```

This record description entry is written in the work file as follows:

```
1      7    12  16      (column positions)
                03   ITEM-3A   PIC X
```

The following example is not a COBOL source statement, but illustrates the operation of tab characters:

```
    TERMINATE#ALL#RECORDS
```

The first tab character does not cause the next character to be written in column 7 or column 8 because the tab character itself is in column 10. The word ALL begins in column 12. The next tab character has no effect except to leave a blank, because the next tab stop, column 16, is also the next character position:

```
1      7    12  16
TERMINATE  ALL RECORDS
```

The following are more TAB command examples:

```
    TA :
```

This command makes the colon the tab character instead of the current tab character. The current tab stops remain in effect.

```
    TA
```

This command clears all tab stops. The tab character is now available for normal use.

CHAPTER 6

DATA COMMUNICATIONS

INTRODUCTION

RM/COS supports communication between processors and terminals using a set of communication JDL commands and a low-level user-programmable communication subroutine.

Both of these functions are designed to provide a basic set of functional communication primitives which can be used by the RM/COS user to produce a communication capability tailored to a specific application or set of applications. The JDL command set (CONNECT, FTS, SEND and RECEIVE) can be combined with other commands and user COBOL programs in order to provide a particular function. Similarly, the "C$COMM" subroutine can be called from a user COBOL program in order to provide link level protocol support for a specific communication environment.

File Transfer

The JDL commands allow "file transfer" between machines in a manner consistent with a layered communication architecture. In order to use the RM/COS communication system, several concepts should be considered.

A RM/COS file transfer dialog between two systems consists of three major components:

1) A file transfer user process

2) A file transfer server process

3) A transport mechanism (logical link) between the two processes

Regardless of the physical link(s) and protocol(s) employed, the file transfer function can be reduced to these three components.

The file transfer user process is the set of JDL commands that are requesting that an explicit file transfer action take place (e.g., send a specific file to the remote processor). The file transfer server process is a set of JDL commands (e.g., the SEND JDL command) that performs the action directed by the corresponding user process. Note that in some instances the SEND command is the user process and in other instances the RECEIVE command is the user process.

The logical link currently must be an IBM 2780/3780 type connection established between two RM/COS "BL" type devices or between a RM/COS "BL" device and a real (or emulated) IBM 2780, 3780 or mainframe.

The sequence of events required to effect a RM/COS file transfer are:

1) The logical link must be established at each end of the file transfer operation by connecting both processes to a common physical link.

2) The link must be CONNECTed to a logical name on both ends.

3) A user/server process pair (SEND/RECEIVE, SEND/FTS, or RECEIVE/FTS) must be established at the endpoints of the link.

The particular user/server process pair chosen will then determine the file transfer function performed. See the FTS, RECEIVE, and SEND command descriptions for additional information.

User Link

The "C$COMM" COBOL callable subroutine provides direct user-program control over the capabilities of an asynchronous, bit-serial communication port in the RM/COS environment. The programmer can specify speed and framing characteristics, control link connection and disconnection, and read or write data messages terminated by user specified codes. In addition, facilities are provided for both character and longitudinal integrity checking of the data. The "C$COMM" subroutine is intended to allow the system designer with some knowledge of data communications to use a custom bit serial link interface. This package is, therefore, intended to satisfy a different need from that of the communication JDL commands, which allow communicating systems to be built with little or no programmer knowledge of data communications.

RM/COS 3780 EMULATOR

The RM/COS file transfer capability is based on the emulation of the popular IBM 3780 Data Communication Terminal [1]. The 3780 is a non-programmable remote job entry (RJE) terminal which is comprised of a high-speed line printer, card reader and controller. The 3780 communicates with either a host computer system or another 3780 terminal over a bit-serial communication interface.

The RM/COS emulator system is designed to appear functionally equivalent to the IBM 3780 terminal as viewed through the bit-serial communication link. As such, the RM/COS emulator does not require that the system be configured with either a line printer or card reader; both of these devices are emulated through the standard RM/COS file system interfaces. That is, the operator may direct printer output to any accessible file or device and obtain card reader input from any accessible file or device. The operation of the RM/COS emulator is directed by the SEND, RECEIVE, and FTS JDL commands. The SEND command is equivalent to reading a set of cards on the IBM 3780 terminal while the RECEIVE command is equivalent to accepting a file for remote printing. The FTS (File Transfer Server) command may be used when communicating with another RM/COS system; it is equivalent to a RECEIVE command when the remote system executes a SEND command and it is equivalent to a SEND command when the remote system executes a RECEIVE command.

In many applications, the RM/COS 3780 emulator will be communicating with other RM/COS systems. If this is the case, the operator need not be concerned with the details of the IBM 3780 or the various features or capabilities of the emulator. Only the functionality of the SEND/RECEIVE process must be considered. If the operator's system is communicating with a non-RM/COS 3780 emulator system, then some details of the RM/COS emulator operation must be considered in order to select communication parameters on the remote system.

A somewhat more limited file transfer capability is also provided by RM/COS based on an IBM 2780 emulation mode. Due to significant constraints on the functions provided in this mode, it should be used only if the 3780 mode is inappropriate (e.g., communication with an actual IBM 2780 terminal).

---

1. "IBM", "IBM 3780" and "IBM 2780" are registered trademarks of International Business Machines Corporation.

2780/3780 EMULATOR CHARACTERISTICS

The RM/COS 2780/3780 Emulator exhibits the following characteristics. For an explanation of the meaning of these characteristics see the related IBM publications [1,2,3].

2780 Characteristics	Supported	Not Supported
Multipoint Line Control		X
Printer Horizontal Format Control	X	
Auto Answer	X	
Multiple Record Transmission	X	
Multiple Record Reception	X	
EBCDIC Transparency		X
Auto Turnaround		X
Terminal Identification		X
144-Character Print Line	X	
Selective Character Set		X
Component Selection		X
EBCDIC Code Structure	X	
ASCII Code Structure	X	
Six-bit Transcode Code Structure		X

3780 Characteristics	Supported	Not Supported
Multipoint Data Link Control		X
EBCDIC Transparency	X	
Automatic Answer	X	
Automatic Disconnect	X	
Terminal Identification		X
Additional Print Positions	X	
Component Selection		X
EBCDIC Code Structure	X	
ASCII Code Structure	X	
Space Compression/Expansion	X	
Audible Alarm		X
Processor Interrupt		X
Conversational Mode		X

---

1. IBM, "General Information-Binary Synchronous Communications", GA27-3004-2.
2. IBM, "Component Description: IBM 2780 Data Transmission Terminal", GA27-3005-3.
3. IBM, "Component Information for the IBM 3780 Data Communication Terminal", GA27-3063-3.

PHYSICAL LINKS SUPPORTED

The RM/COS emulator provides support for any point-to-point bit-serial synchronous communication link conforming to EIA specification RS-232C. In addition, the emulator supports asynchronous RS-232C links operating at selected speeds (see Operator Guide).

In most cases, the physical link employed will operate over a pair of Modulator-Demodulators (MODEMs) utilizing the switched telephone network. For this configuration, a MODEM such as the Bell 201C or equivalent is probably most appropriate. This type of MODEM operates at a speed of 2400 bits/second in a half-duplex mode over 2-wire lines. Other synchronous MODEMs can be used at either higher or lower speeds as required. In some local link environments, synchronous line drivers (SLDs) operating over local metallic circuits are appropriate. If this is the case, a "controlled carrier" capability should be selected. In any case, the communication link need only operate in a half-duplex mode as the procedures employed form an inherently two-way-alternate (TWA) protocol.

DATA FORMATS SUPPORTED

Code Structure

The RM/COS emulator provides support of both the EBCDIC (Extended Binary Coded Decimal Interchange Code) and ASCII (American National Standard Code for Information Interchange) code structures. The choice of code structure determines the values of the communications link control characters, so both parties to the link must agree on the code structure in use. The EBCDIC code comprises the internal code structure of the IBM System/360 and System/370, and is the default selection for 2780/3780 communications. The ASCII code option allows communications with IBM mainframes and 2780/3780 installations which have the USASCII option installed.

Coded Mode

The RM/COS emulator provides support for the transfer of files containing only coded text characters using the most efficient means possible within the constraints of the 3780 protocol. Coded text is defined to be data containing only ASCII graphics characters and noncommunication control characters; the disallowed characters are the communication control characters with hexadecimal character codes 01-06, 0E-10, 15-17, 19, 1C, 1D, and 1F. Additionally, other control characters (i.e., hexadecimal character values less than 20) may be interpreted functionally for transmission over the link. If disallowed characters are sent in coded mode, link control errors will be indicated.

In coded mode transmission, the text characters are transmitted as a non-transparent message. If the EBCDIC code structure is selected, the text characters are translated to EBCDIC before transmission. When receiving non-transparent messages, if the EBCDIC code structure is selected, the incoming text characters are translated from EBCDIC to ASCII. The translation functions used are described in Appendix J.

If an RM/COS SPOOL type file is transmitted to a remote system in coded mode, vertical forms control information associated with the file is coded and transferred intact.

Sequences of three or more ASCII blank characters are compressed on transmission and expanded on reception, thus maintaining the precise record structure and length while more effectively utilizing the communication link. The emulator does not provide this capability when operating in a 2780 compatible mode.

The emulator provides support of a method of electronic tab stops compatible with the technique used in the IBM 3780 terminal. For more information on the use of this capability, the appropriate IBM publication should be consulted.

The emulator provides support of vertical forms control in addition to the normal RM/COS forms control in a manner consistent with that used in the IBM 3780 terminal. Since RM/COS does not support vertical forms control (VFC) tapes or programs for its printers, the VFC "tape" used by the 3780 emulator system is fixed and not operator selectable. Currently, only the channel 1 and channel 12 information is defined within the RM/COS VFC image. For additional information on vertical forms positioning within the IBM 3780 context see the appropriate IBM publication. Note that for RM/COS to RM/COS configurations, knowledge of the vertical forms positioning techniques used by the emulator is not necessary for proper use of the emulator.

Image Mode

In the image mode of file transfer, any RM/COS file can be transferred without restriction to the type of data contained within the file. No blank compression/expansion, horizontal format control or imbedded vertical forms control is provided. In addition, any forms positioning information retained by SPOOL type files will be discarded without being transmitted to the remote system.

In image mode transmission, the text characters are transmitted as a transparent message, with no character translation regardless of the code structure. When receiving transparent messages, RM/COS performs no

translation of the received characters regardless of the code structure.

Of particular use is the ability to transfer RM/COS program files (COBOL object programs) and MDS files (such as the System Image File and the JDL Image File) in image mode between RM/COS systems using the 3780 emulation capability.

Image mode may not be used when operating in a 2780 compatible mode.

ASCII Vertical Redundancy Check

When the ASCII code structure is selected, RM/COS performs a vertical redundancy check on every control character and coded mode text character which is received. This check is an even parity check if the physical link is asynchronous, and is an odd parity check if the physical link is synchronous. The parity check performed may be modified with options on the CONNECT command.

ASCII Block Check

When the EBCDIC code structure is selected, or when image (transparent) ASCII transmissions are used, RM/COS employs a two character cyclic redundancy check on each text block tranferred, using the CRC-16 generator polynomial. For ASCII coded transmissions, two forms of block checking are possible: the CRC-16 algorithm, and a longitudinal redundancy check (LRC) consisting of a single character exclusive-OR of the text characters. ANSI X3.28-1976 specified use of the LRC for non-transparent operation. IBM hardware may use either the LRC or CRC block check, depending on the hardware involved. Usually the hardware employs a CRC check all the time if the transparency option is installed, regardless of whether the transmission itself is transparent or non-transparent. RM/COS will use the LRC check on ASCII coded transmissions unless overridden on the CONNECT command.

2780/3780 EMULATOR PARAMETERS

The disconnect timeout, response timeout, and ENQ/NAK retry limit control the behavior of the 2780/3780 Emulator. These parameters are set at IPL time with values provided in the System Definition File on the unit specification record (U record) which defines the link device.

Disconnect Timeout

The disconnect timer causes the link to be disconnected when no productive activity occurs within the time allowed. Productive activity is defined as a successful transfer of a data block. There may be no productive activity because one or both parties to the link have not entered a SEND, RECEIVE, or FTS command. There may be no productive activity because during the file transfer the destination device goes busy, e.g. the destination is a line printer which is out of paper. Regardless of why there is no productive activity, the disconnect timeout allows an installation using a telephone line to disconnect the line when it is not being used.

Response Timeout

The response timeout causes a receiving station to stop a file transfer if no transmission is received within the specified interval. It is primarily intended to return control to the operator when the sending station has terminated the transfer but the receiving station has missed the end of transmission message. When using a 3780 protocol, the value should be set to at least 6 seconds longer than the amount of time required to transfer 512 characters at the baud rate in use. Thus, on a 2400 baud synchronous link, the response timer must be set to at least 8 seconds; on a 300 baud asynchronous link, the response timer must be set to at least 24 seconds.

When using a 2780 protocol, there are no supervisory messages that maintain the link during a transfer when one party is busy for a prolonged period. Thus, the 2780 response timer places a limit on how long the receiving station is willing to wait for the sending station. The 2780 response timer also places a limit on how long the sending station is willing to wait for data from the source device before terminating the transfer. For example, if performing a SEND operation with a local name of ME, the response timer at both the sending and receiving station must be set long enough to allow for operator input of each record.

Retry Limit

The retry limit set at IPL limits both the number ENQ messages that are sent and the number of NAK messages that the Emulator is willing to receive. The ENQ limit restricts how many times the sending station tries to verify a response from the receiving station when there was no response or the response was wrong. The NAK limit restricts how many times the sending station is willing to send the same data block when the receiving station indicates that it did not receive the block successfully. Typical values for the retry limit range from 3 to 15; when using a 3780 protocol a value of 5 is recommended.

When using a 2780 protocol, there are no supervisory messages that maintain the link during a transfer when one party is busy for a prolonged period. During 2780 operation, the ENQ limit restricts how long the sending station is willing to wait for the receiving station. ENQ messages are sent every 3 seconds. Thus, if the destination device at the receiving station is a buffered line printer which is busy for 30 second intervals, the retry limit must be set to at least 10.

FILE TRANSFER OPERATION

Prior to attempting the CONNECT JDL command, the physical communication link should be established. If the switched telephone network is to be used, the calling system operator should dial the remote system number and wait for the answer tone. At the called location, the system operator should insure that the MODEM is connected to the telephone line and that the "auto-answer" mode is enabled. After the physical link has been established, the operators should proceed with the processing of the CONNECT JDL command.

An emulator time-out interval begins with the execution of the CONNECT command. The value used for this timer is that specified on the CONNECT command. If the communication link does not respond with a "Data Set Ready" indication within this interval, the link connection will be aborted with an error condition. If the link connection is successful, the operators may engage in any number of file transfer dialogs using the connected link.

While the link is connected, productive activity, defined as a successful transfer of a message text block, must occur within the disconnect time-out value. If no productive activity occurs, the link will be disconnected and "Data Terminal Ready" turned off. After this occurs, subsequent SEND, RECEIVE and FTS commands performed using the disconnected link will be aborted with an error indication. In order to reestablish the connection, the link must be RELEASEd and reCONNECTed.

A file transfer dialog consists of a pair of SEND and RECEIVE commands operating over a CONNECTed communication link. An FTS command may be substituted for either the SEND or RECEIVE command at one end of the communication link. If a SEND command attempts to communicate with another SENDer or a RECEIVE command attempts to communicate with another RECEIVEr, the commands will be aborted with an error indication. If this occurs, the link remains CONNECTed.

If the link quality is insufficient to support reasonably efficient and reliable communication, the SEND or RECEIVE commands will abort with an error indication. The determination of line quality is controlled by the device retry limit associated with the link device.

After completion of the last file transfer operation, the link may be disconnected and the telephone line released by entering a RELEASE JDL command specifying the link device.

EXAMPLES

Auto-Answer Remote Printer

The following example illustrates the coding of a JDL command sequence to implement an auto-answering remote printing terminal.

```
     /       LOOP
     /       CONNECT, LOGICAL NAME=LINK, NAME=BL01, TIMEOUT=0
 Z   /       EXIT
 S   /       SETCOND
     /<NE>   RECEIVE, LINK=LINK, LOCAL=LP01
     /       SETCOND, VALUE="S"
 S   /       REPEAT
 Z   /       SETCOND
     /       RELEASE, LOGICAL NAME=LINK
     /       REPEAT
```

In this example, the command stream begins by CONNECTing the link. The time allowed for the connection is indefinitely large. Once the connection occurs, the commands then receive one or more printer jobs from the remote system. The link is RELEASEd when the RECEIVE operation fails (usually an "on hook" condition) and the command sequence starts over.

Auto-Answer Transmission

The following example illustrates the coding of a JDL command sequence to answer the phone and transmit a transaction file to a remote host system.

```
     /       LOOP
     /<NE>   CONNECT, LOGICAL NAME=LINK, NAME=BL01
     /       SETCOND, VALUE="S"
 Z   /       SETCOND
     /       REPEAT
 S   /       SETCOND
     /<NE>   SEND,    LINK=LINK, LOCAL=.XACTION
     /       RELEASE, LOGICAL NAME=LINK
     /       DELETE,  NAME=.XACTION
     /<NE>   CREATE,  NAME=.XACTION, ALL=10, SEC=10
     /       REPEAT
 !   /       RELEASE
 !   /       SETCOND
     /       REPEAT
```

The command sequence begins by answering the telephone with the CONNECT command. If this operation fails, it is repeated. When the link has been successfully CONNECTed, the accumulated transaction file is sent to the calling host computer. If an error occurs, the transaction file is retained and the link RELEASEd. If the file was sent

without error, the transaction file is emptied and the link RELEASEd.

Remote JDL Command Execution

The following example illustrates the coding of JDL command sequences to allow the execution of JDL command sequences on a remote RM/COS system.

The batch command file on the controlling (master) RM/COS system consists of:

```
     /       MESSAGE, TEXT=
                "Dial the remote system, enter RETURN",
                REPLY=NO,STATION=0
     /       LOOP
     /<NE>   CONNECT, LOGICAL NAME=LINK, NAME=BLO1, TIMEOUT=5
 Z   /       SETCOND, VALUE="C"
 S   /       SETCOND
     /<NE>   SEND,    LINK=LINK,
                LOCAL("Enter command file pathname")=()
     /<NE>   RECEIVE, LINK=LINK,
                LOCAL("Enter listing file pathname")=()
 Z   /       SETCOND, VALUE="L"
     /       MESSAGE, TEXT=
                "Done. Do you have any more commands? (Y/N)",
                REPLY=YES, STATION=0
 Y   /       SETCOND, VALUE="S"
 S   /       REPEAT
 L   /       SETCOND
     /       MESSAGE, TEXT=
                "Error. Retry the exchange? (Y/N)",
                      REPLY=YES, STATION=0
 Y   /       SETCOND, VALUE="S"
 S   /       REPEAT
 N   /       RELEASE, LOGICAL NAME=LINK
 N   /       SETCOND, VALUE="C"
 C   /       MESSAGE, TEXT=
                "Do you wish to retry the connection? (Y/N)",
                REPLY=YES, STATION=0
 Y   /       SETCOND
     /       REPEAT
 N   /       EXIT
```

This command file begins by issuing a message in order to allow the system operator to complete the call to the remote site. If this positive acknowledgement of the connection is not included, the connect timeout must be set to a large enough value to allow the connection to take place after the batch stream is started. Following the "go-ahead" by the operator, the CONNECT command is processed. Assuming the telephone connection has already been made, this command will complete quickly. If not, the operator is given a chance to re-execute the CONNECT command. Following

completion of the CONNECT command, the operator is prompted for the pathname of a JDL command file to be sent to the remote system. Note that a pathname of "ME" in this case allows the operator to key-in the remote JDL commands at the time the SEND command is processed. After the prompted pathname has been supplied to the SEND command, the two systems establish a data transfer dialog and the commands are transmitted to the remote system. The RECEIVE command following the SEND command awaits the reply to the remote batch stream. The disconnect timer must be set to a value large enough to allow the remote system to process the JDL commands submitted. After the receipt of the reply, the SEND/RECEIVE process is repeated as long as the operator desires. When no additional command transfers are desired, the operator indicates this and the link is disconnected (RELEASEd).

The batch control command file on the remote (slave) RM/COS system might look like this:

```
!   /       RELEASE
!   /<NE>   DELETE,  NAME=.LSTFILE
!   /<NE>   DELETE,  NAME=.CMDFILE
!   /       SETCOND
    /<NE>   CREATE,  NAME=.CMDFILE, ALL=10, SEC=10
    /<NE>   CREATE,  NAME=.LSTFILE, ALL=10, SEC=10
Z   /       SETCOND, VALUE="X"
    /       LOOP
    /<NE>   CONNECT, LOGICAL NAME=LINK, NAME=BL01
    /       SETCOND, VALUE="S"
Z   /       SETCOND
    /       REPEAT
S   /       SETCOND
    /<NE>   RECEIVE, LINK=LINK, LOCAL=.CMDFILE
Z   /       SETCOND, VALUE="L"
    /<NE>   BATCH,   NAME=.CMDFILE, LIST NAME=.LSTFILE
Z   /       SETCOND
    /<NE>   SEND,    LINK=LINK, LOCAL=.LSTFILE
Z   /       SETCOND, VALUE="L"
    /       SETCOND, VALUE="S"
S   /       REPEAT
L   /       SETCOND
    /       RELEASE, LOGICAL NAME=LINK
    /       REPEAT
X   /<NE>   DELETE,  NAME=.LSTFILE
X   /<NE>   DELETE,  NAME=.CMDFILE
```

The remote stream begins by creating temporary files for the incoming batch command file and the listing file to be returned to the master system after which the CONNECT command is performed until it succeeds. The command file is then RECEIVEd and executed in the current partition. Note that output directed to the terminal by JDL commands in the master batch stream (e.g., SHOW) will be output to the slave

terminal if this command sequence is being executed in a terminal partition. After completion of the batch stream, the listing file is returned to the master system via the SEND command. If an error condition is detected during the SENDing or RECEIVEing process, the link is disconnected and re-connected. If the condition code is set to "X" by the master command sequence, the slave command sequence will terminate.

Directory Copy

The following example illustrates the coding of JDL command sequences to copy a directory structure from one RM/COS system to another.

The batch command file on the transmitting RM/COS system, where the directory .PROJECTX already exists, consists of:

```
/       CREATE, LOGICAL NAME=BACKUP, RECORD SIZE=510,
             ALLOCATION=100, SECONDARY=100,
             TYPE=<SCRATCH,COMPRESSED>
/       ASSIGN, LOGICAL NAME=LO, NAME=LP01
/       FILE-BACKUP, BACKUP FILE=BACKUP,
             SOURCE NAME=.PROJECTX
/       RELEASE, LOGICAL NAME=LO
/       CONNECT, LOGICAL NAME=LINK, NAME=BL01
/       SEND, LINK LOGICAL NAME=LINK,
             LOCAL NAME=BACKUP, MODE=IMAGE
/       RELEASE, LOGICAL NAME=<BACKUP, LINK>
```

This command file begins by creating a scratch, compressed sequential file on the system disk which will contain a backup image of the directory to be transferred. The record size of 510 is chosen because it is the optimal record size for an image mode transfer. The logical name LO is assigned to a printer so a hardcopy of any errors during the backup will be available. The directory .PROJECTX is then backed up to the logical name given the scratch file. The succeeding commands establish the link, send the file in image mode, and release both the link and the scratch file. The backup file is sent in image mode because it contains binary values that are not ASCII graphic characters.

The batch command file on the receiving RM/COS system consists of:

```
/       CREATE, LOGICAL NAME=BACKUP, RECORD SIZE=510,
            ALLOCATION=100, SECONDARY=100,
            TYPE=<SCRATCH,COMPRESSED>
/       CONNECT, LOGICAL NAME=LINK, NAME=BL01,
            TIMEOUT=0
/       RECEIVE, LINK LOGICAL NAME=LINK,
            LOCAL NAME=BACKUP
/       RELEASE, LOGICAL NAME=LINK
/       ASSIGN,  LOGICAL NAME=LO, NAME=LP01
/       FILE-RESTORE, BACKUP FILE=BACKUP,
            BACKUP NAME=.PROJECTX,
            DESTINATION NAME=.PROJECTX
/       RELEASE, LOGICAL NAME=<BACKUP, LO>
```

This command file begins by creating a scratch, compressed sequential file on the system disk which will contain a backup image of the directory to be transferred. The record size specified must match that specified at the transmitting site. The CONNECT command establishes the link, and the TIMEOUT value of zero allows the command to wait indefinitely while the transmitting site is creating the file. The backup image is received into the logical name of the scratch file, and the link is released. The logical name LO is assigned to a printer so a hardcopy of any errors during the restore will be available. The scratch file is then used as the input file to the FILE-RESTORE command, restoring all files of .PROJECTX to the same pathname on the system disk of the receiving system.

USER-PROGRAMMABLE LINK SUBROUTINE

The user-programmable link facility is provided through the COBOL callable subroutine C$COMM which is fully described in Appendix B, COBOL Subroutine Library. The following is an annotated example of a COBOL program which uses the "C$COMM" subroutine for a specific application.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. APOLLO.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.
SOURCE COMPUTER. RM/COS.
OBJECT COMPUTER. RM/COS.

INPUT-OUTPUT SECTION.

FILE-CONTROL.
    SELECT PRINT-FILE
        ASSIGN TO OUTPUT, "TICKETS".

DATA DIVISION.

FILE SECTION.

FD  PRINT-FILE
    LABEL RECORDS ARE OMITTED.
01  MESSAGE-LINE.
    02  FILLER                  PIC X OCCURS 1 TO 3000 TIMES
                                DEPENDING ON MESSAGE-SIZE.
01  MESSAGE-BUFFER              PIC X(3000).
```

Note that the item MESSAGE-LINE is variable length and the item MESSAGE-BUFFER is fixed length. This is the item into which the program will read the data characters from the bit-serial link.

```
      WORKING-STORAGE SECTION.

      01  CCT-AREA.
          02  CCT-LINK-FLAGS          PIC 9(5) COMP-1.
          02  CCT-RATE                PIC 9(5) COMP-1 VALUE 0.
          02  FILLER                  PIC X.
          02  CCT-TERMINATOR-CHAR     PIC X.
          02  FILLER                  PIC 9(5) COMP-1 VALUE -1.
          02  CCT-RECEIVE-BUFFER-SIZE
                                      PIC 9(5) COMP-1 VALUE 0.
          02  CCT-TRANSMIT-BUFFER-SIZE
                                      PIC 9(5) COMP-1 VALUE 0.
          02  CCT-CONNECT-TIMEOUT     PIC 9(5) COMP-1 VALUE 100.
          02  CCT-READ-TIMEOUT        PIC 9(5) COMP-1 VALUE 100.
          02  CCT-WRITE-TIMEOUT       PIC 9(5) COMP-1 VALUE 100.

      01  CCT-FLAG-CONSTANTS          COMPUTATIONAL-1.
          02  CCT-PRESET-FLAG         PIC 9(5) VALUE 1.
          02  CCT-CONNECT-FLAG        PIC 9(5) VALUE 2.
          02  CCT-ENABLE-TRANSMITTER-FLAG
                                      PIC 9(5) VALUE 4.
          02  CCT-ENABLE-RECEIVER-FLAG
                                      PIC 9(5) VALUE 8.
          02  CCT-ENABLE-TERMINATOR-FLAG
                                      PIC 9(5) VALUE 32.
          02  CCT-ENABLE-PARITY-FLAG
                                      PIC 9(5) VALUE 64.
          02  CCT-STRIP-TERMINATOR-FLAG
                                      PIC 9(5) VALUE 256.
          02  CCT-ODD-PARITY-FLAG     PIC 9(5) VALUE 512.
          02  CCT-FLUSH-TRANSMITTER-FLAG
                                      PIC 9(5) VALUE 4096.
          02  CCT-FLUSH-RECEIVER-FLAG
                                      PIC 9(5) VALUE 8192.

      01  FUNCTION-CODES              COMPUTATIONAL-1.
          02  CONTROL-FUNCTION-CODE       PIC 9 VALUE 0.
          02  READ-FUNCTION-CODE          PIC 9 VALUE 1.
          02  WRITE-FUNCTION-CODE         PIC 9 VALUE 2.

      01  LOCAL-CONSTANTS.
          02      COMP-1-ZERO         PIC 9    COMP-1 VALUE 0.
          02      PORT-NAME           PIC X(8) VALUE "DATALINK".

      01  MESSAGE-SIZE                PIC 9(5) COMP-1.
      01  READ-MESSAGE-COUNT          PIC 9(5) COMP-1.
      01  WRITE-MESSAGE-COUNT         PIC 9(5) COMP-1.

      01  MISCELLANEOUS-VARIABLES.
          02  END-OF-SESSION          PIC 9(5) COMP-1 VALUE 0.
          02  RETURN-STATUS           PIC 99.

      01  STRANGE-CHARACTERS.
          02  NUL-ETX-VALUE           PIC 9(5) COMP-1 VALUE 3.
```

```
    02  NUL-ETX                REDEFINES NUL-ETX-VALUE.
        03  NUL-CHAR           PIC X.
        03  ETX-CHAR           PIC X.
```

The Communication Control Table is represented by the item CCT-AREA. Several compile-time values are set in this section. The speed of the link transmitter/receiver is specified as 0; this will allow the speed selected at IPL time to be used. The receiver and transmitter buffer size are also specified as 0. This will not alter the logical size of the buffers. All three timeout intervals are set to 10 seconds.

The STRANGE-CHARACTERS item is constructed to allow specifying the non-graphic ETX terminator character.

```
PROCEDURE DIVISION.

THE-MAIN SECTION.

A.
    PERFORM OPEN-FILES.
    PERFORM DISCONNECT-PORT.
    PERFORM PRESET-CCT-AREA.
    PERFORM CONNECT-PORT.

B.
    PERFORM READ-APOLLO-MESSAGE.
    IF END-OF-SESSION NOT EQUAL 0
        GO TO C.
    PERFORM WRITE-APOLLO-MESSAGE.
    GO TO B.

C.
    PERFORM DISCONNECT-PORT.

END-OF-PROGRAM.
    STOP RUN.
```

This section requires no comment.

```
PRESET-STORAGE SECTION.

OPEN-FILES.
    OPEN OUTPUT PRINT-FILE.

PRESET-CCT-AREA.
    ADD CCT-CONNECT-FLAG,
        CCT-PRESET-FLAG,
        CCT-ENABLE-TRANSMITTER-FLAG,
        CCT-ENABLE-RECEIVER-FLAG,
        CCT-ENABLE-PARITY-FLAG,
        CCT-ENABLE-TERMINATOR-FLAG,
        CCT-STRIP-TERMINATOR-FLAG,
        CCT-FLUSH-RECEIVER-FLAG,
        CCT-FLUSH-TRANSMITTER-FLAG GIVING CCT-LINK-FLAGS.
    MOVE ETX-CHAR TO CCT-TERMINATOR-CHAR.

CONNECT-PORT.
    CALL "C$COMM" USING
        CONTROL-FUNCTION-CODE,
        RETURN-STATUS,
        PORT-NAME,
        CCT-AREA,
        COMP-1-ZERO, COMP-1-ZERO.
    IF RETURN-STATUS NOT EQUAL 0
        DISPLAY "LINK CONNECT ERROR "
           LINE 12 POSITION 30 ERASE
        STOP RUN.
    DISPLAY "LINK ESTABLISHED"
        LINE 12 POSITION 31 ERASE.

DISCONNECT-PORT.
    MOVE 0 TO CCT-LINK-FLAGS.
    CALL "C$COMM" USING
        CONTROL-FUNCTION-CODE,
        RETURN-STATUS,
        PORT-NAME,
        CCT-AREA,
        COMP-1-ZERO, COMP-1-ZERO.
    DISPLAY "LINK DISCONNECTED"
        LINE 12 POSITION 31 ERASE.
```

The PRESET-CCT section forms the CCT flags word value by adding all of the flag values to be asserted. Note that the two "flush" flags do not have to be explicitly turned off. The CONNECT-PORT section issues a control function using the preset value of the CCT. This causes the link to be connected and the speed set. The DISCONNECT-PORT section clears all of the CCT flags resulting in the link disconnecting when the control function is issued.

```
READ-APOLLO-MESSAGE SECTION.

A.
     MOVE 0 TO READ-MESSAGE-COUNT.

B.
     CALL "C$COMM" USING
          READ-FUNCTION-CODE,
          RETURN-STATUS,
          PORT-NAME,
          MESSAGE-BUFFER,
          READ-MESSAGE-COUNT,
          COMP-1-ZERO.
     GO TO
          TIMEOUT-ERROR,
          PARITY-ERROR,
          ON-HOOK-ERROR,
          FATAL-ERROR,
          FATAL-ERROR DEPENDING ON RETURN-STATUS.
     IF READ-MESSAGE-COUNT EQUAL 0
          GO TO B.
     MOVE READ-MESSAGE-COUNT TO MESSAGE-SIZE.
     MOVE 0 TO WRITE-MESSAGE-COUNT.

C.
     CALL "C$COMM" USING
          WRITE-FUNCTION-CODE,
          RETURN-STATUS,
          PORT-NAME,
          MESSAGE-LINE,
          WRITE-MESSAGE-COUNT,
          MESSAGE-SIZE.
     IF RETURN-STATUS GREATER THAN 1
          GO TO EXIT-SECTION.
     IF MESSAGE-SIZE NOT EQUAL WRITE-MESSAGE-COUNT
          GO TO C.
     GO TO EXIT-SECTION.

PARITY-ERROR.
TIMEOUT-ERROR.
     GO TO B.

ON-HOOK-ERROR.
FATAL-ERROR.
     MOVE 1 TO END-OF-SESSION.

EXIT-SECTION.
     EXIT.
```

The READ-MESSAGE-COUNT item is used as an offset into MESSAGE-BUFFER for the read function. If the read operation times out, the function is simply reissued with the new value of READ-MESSAGE-COUNT. Only when the terminator character is read is the operation complete. At that time,

the WRITE-MESSAGE-COUNT offset value is cleared and the message echoed back over the communication link. If an error occurs, the END-OF-SESSION value is set, causing the program to terminate.

```
WRITE-APOLLO-MESSAGE SECTION.

A.
    WRITE MESSAGE-LINE AFTER ADVANCING 0.
```

The message read from the link is written on an output file. The size of the record written is a function of the size of the message received.

CHAPTER 7

SYSTEM STATUS CODES

INPUT/OUTPUT STATUS CODES

Input/Output status codes for RM/COS are of the form:

XXYY

where,

XX is the file status: two digits defined by the ANSI X3.23-1974 COBOL Standard. The file status is returned to a COBOL program in the FILE STATUS data item associated with the file.

YY is an additional information code. This code is available to a COBOL program by calling the subroutine C$RERR (see Appendix B, COBOL Subroutine Library Package).

When an I/O error occurs in a COBOL program with no applicable use procedure or in a JDL command processor, the RM/COS I/O status code is displayed on the error line along with the logical name assigned to the file on which the error occurred.

0000 SUCCESSFUL COMPLETION

1) Normal completion of I/O operation.

0200 SUCCESSFUL, DUPLICATE KEY EXISTS

1) A successful read statement on an indexed organization file. Key value for the current key of reference is equal to the value of the same key in the next logical record within the current key of reference.
2) For a write or rewrite statement, the written record created a duplicate key value for at least one of the alternate record keys for which duplicate keys are allowed.

1000 END OF FILE

1) The end of the currently referenced file has been reached.
2) Sequential read statement on a sequential, relative or indexed organization file failed as a result of an attempt to read a record when no next logical record exists in the file.
3) The occurrence of this error causes an At End condition in a COBOL program.

The following prefix 2xxx codes refer to indexed and relative organization files. The occurrence of these errors causes an Invalid Key condition in a COBOL program.

2100 INVALID KEY SEQUENCE

1) A sequential write statement on an indexed organization file was unsuccessful as a result of a violation of the ascending sequence requirement for successive prime record key values.
2) A sequential rewrite statement on an indexed organization file was unsuccessful as a result of a violation of the requirement that the key be the same as the key returned by the preceding successful read (i.e. the prime record key value was altered between a read and a rewrite to the file).

2200 INVALID KEY, DUPLICATE

1) Attempt to randomly write or rewrite a record to a relative or indexed organization file is unsuccessful because it would create a duplicate key for which duplicates are not allowed.

2300 INVALID KEY, NO RECORD FOUND

1) Random read, delete or rewrite to a relative or indexed organization file was unsuccessful as a result of an attempt to reference a record identified by a key that does not exist in the file.
2) A START statement to a relative or indexed organization file failed as a result of the stated comparison not being satisfied by any record in the file.

2402 INVALID KEY, BOUNDARY (EXHAUSTED EXTENTS)

1) File referenced has reached the maximum permissible number of discontiguous allocations (47).
2) FCOPY the volume to another volume thereby reorganizing the volume's file allocations.

2403 INVALID KEY, BOUNDARY (EXHAUSTED ADUS)

1) File referenced cannot continue expansion due to lack of allocatable disk units on disk where file currently resides.
2) RECLOSE disk to recover any unused scratch space.
3) Delete unneeded files on disk volume.
4) FCOPY file to another disk volume.

2404 INVALID KEY, BOUNDARY (FILE NOT EXPANDABLE)

1) Consumed primary file allocation and file as currently defined disallows secondary allocations.
2) Redefine file via CHANGE permitting secondary allocations.
3) Check program to ensure proper reference of intended record.

2405 INVALID KEY, BOUNDARY (EXHAUSTED BLOCKS)

1) Attempt to write more than 65535 blocks to the file.
2) Recreate file with a larger block size.

2406 INVALID KEY, BOUNDARY (DISK TOO FRAGMENTED)

1) Unable to find sufficient contiguous disk for one block.
2) Recreate file with a smaller block size.
3) FCOPY volume to another volume thereby reorganizing the volume's file allocations.

2407 INVALID KEY, BOUNDARY (ACCESS PAST LAST EXTENT)

1) This error is caused by an attempt within the operating system to access past the last ADU. It indicates a system error, and the vendor should be contacted.
2) Copying the file to another disk, and changing the block size, may get around the problem.

2408 INVALID KEY, BOUNDARY (INVALID RECORD NUMBER)

1) A write statement to a relative organization file attempted to create a record 0 or a record that would lie past block 65535.

2450 INVALID KEY, BOUNDARY (INVALID MEMBER)

1) This error is caused by an attempt within the operating system to access a nonexistent member of an MDS file. It indicates a system error, and the vendor should be contacted.
2) Copying the file to another disk, and changing the block size, may get around the problem.

2451 INVALID KEY, BOUNDARY (ACCESS PAST END OF MEMBER)

1) This error is caused by an attempt within the operating system to access past the end of a member of an MDS file. It indicates a system error, and the vendor should be contacted.
2) Copying the file to another disk, and changing the block size, may get around the problem.

30XX PERMANENT ERROR

1) All 30xx error codes indicate unsuccessful attempt as a consequence of a permanent I/O error: data check, parity error, transmission error etc. The specific cause or condition is categorized by the last two digits.

3000 BAD INDEX STRUCTURE

1) Index structure of indexed organization file is invalid because:
   a) Leaf node of index tree points to a nonexistent data record.
   b) On a delete or rewrite operation, the data record being changed cannot be found in an index tree.
   c) An offspring or sibling node block did not have the expected node level or block type.
   d) Incorrect index tree structure caused tree search to attempt to search more than 16 levels.
   e) Tree structure indicates a sibling node block is present without any parent node block.
   f) Tree structure indicates an offspring block is not pointed to by its parent.
2) Perform RECLOSE operation on file to repair tree structure.

3001 RECORD INTEGRITY ERROR

1) The record structure of an MVD organization disk file or a tape file is invalid because:
   a) End of file was encountered before reading the last segment of a spanned record.
   b) The segments of a spanned record did not occur in the correct order.
   c) The length in the header of a variable length record was too small or exceeded the length of the block read.
   d) The length in the header of a format V or <V, S> block was too small or exceeded the length of the block read.
   e) The header of a format D or S record contained nonnumeric characters or specified a size greater than 65535.

3002 VOLUME NOT MOUNTED

1) The operator aborted a mount request instead of mounting the requested volume.

3010 ILLEGAL OPCODE

1) This error is caused by a device request from within the operating system which specified an operation illegal on this device.
2) Refer problem to system support personnel.

3015 DEVICE DISABLED

1) Device has been disabled via the VARY command.
2) Device has been disabled during IPL by specification of the Ignore field of a D record in .SYSDEFIL.
3) Determine why the device has been disabled; edit the System Definition File to reflect the correct initial state of the device.
4) If the device is present and in working order it may be enabled with the VARY command.

3017 OUTPUT KILLED

1) A KPRINTER command was issued for the line printer.

3018 UNABLE TO START PROCESS

1) An I/O request was rejected because there was no entry available in the process table.
2) Reduce the number of partitions active and retry the request.

3020 DEVICE ERROR

1) Hardware error detected.

3021 DEVICE TIMED OUT

1) Device has not responded within the required time limit. The time limit on some devices may be changed by the System Definition File. The current time limit of a device may be displayed with a STATUS command.

3022 DEVICE DISCONNECTED

1) A station disconnected, either by logging off or by disconnecting the communications line.

3023 DEVICE PARAMETER ERROR

1) The format, sector size, or interleave factor entered on an INITIALIZE command was not valid for the specified drive or its controller.

3024 DEVICE NOT READY

1) An I/O request to a device was rejected because there was no medium mounted in the device. For example, there was no cartridge in a tape drive, or the cartridge was removed.

3780/2780 ERROR CODES

3030 LINE DISCONNECTED

1) The 3780/2780 communication line was disconnected prior to the start of this request.
2) Usually a 3031 or 3032 error code will have been displayed previously to indicate the reason the line was disconnected. The 3031 or 3032 error code will have been lost if the line was disconnected while processing the End of Transmission (EOT) message at the end of the preceding receive operation.
3) RELEASE and reCONNECT link, then retry the command.

3031 DISCONNECT RECEIVED

1) The remote station disconnected the communication line by sending a disconnect message (DLE EOT), or the data communication circuit went "on hook".
2) This error will occur immediately upon entering a command if the line was disconnected between commands.
3) RELEASE and reCONNECT link, then retry the command.

3032 DISCONNECT TIMER ELAPSED

1) The local station disconnect timer elapsed, indicating that no text blocks were transferred in either direction within the disconnect timeout.
2) This error will occur immediately upon entering a command if more time elapsed since the preceding command than is allowed by the disconnect timeout.
3) If attempting a SEND command, or any command with a remote pathname, this error will occur if the remote operator never entered a RECEIVE or FTS command. This explanation is likely if the "NAKs received" on the STATUS display is non-zero, and the "Blocks transferred" on the STATUS display is zero.
4) If attempting a SEND command, or any command with a remote pathname, the disconnect timeout may disguise a 3035 error if the disconnect timeout is set to less than 3 seconds multiplied by the retry limit. This explanation is likely if the "ENQs sent" on the STATUS display is non-zero, and the "Blocks transferred" on the STATUS display is zero.
5) If attempting a RECEIVE command, this error will occur if the remote operator never entered a SEND command. In this case, the "Blocks transferred" on the STATUS display will be zero.

6) If attempting an FTS command, this error will occur if the remote operator never entered a RECEIVE command with a remote pathname. In this case, the "Blocks transferred" on the STATUS display will be zero.
7) During a file transfer, this error will occur if the local or remote operator interrupts the command for longer than the disconnect timeout. This error will also occur during transfers destined to a printer if the printer is busy (e.g., out of paper) for longer than the disconnect timeout. In these cases, the "Blocks transferred" on the STATUS display will be non-zero, and the "ENQs sent" and "NAKs received" on the sending station's STATUS display will be zero.
8) During a SEND command, or an FTS command that is sending a file to the remote station, the disconnect timeout may disguise a 3039 error if the disconnect timeout is set to less than 3 seconds multiplied by the retry limit. This explanation is likely if the "ENQs sent" and "Blocks transferred" on the STATUS display are both non-zero.
9) During a SEND command, or an FTS command that is sending a file to the remote station, the disconnect timeout may disguise a 3040 error if the disconnect timeout is set too low. This explanation is likely if the "NAKs received" and "Blocks transferred" on the STATUS display are both non-zero.
10) RELEASE and reCONNECT link, then retry the command.

3033 RESPONSE TIMER ELAPSED

1) A receiving station will generate this error after sending an acknowledgement if the sending station does not respond with a valid transmission block within the response timeout specified on the unit specification record.
2) Verify that the response timeout set on the unit specification record is at least 6 seconds longer than the time required to send 512 characters at the baud rate in use.
3) If a receiving station generates this error, but the file appears to be complete, this error may be caused by loss of the End of Transmission (EOT) message from the sending station. This could result from setting too short a Clear to Send Delay in the sending station's modem.
4) A receiving station will generate this error if the sending station attempts to send a transparent heading block (a block beginning with DLE SOH). ANSI X3.28-1976 provides for transparent heading blocks in Subcategory D1, but IBM's 3780/2780 procedures do not include them.

5) A sending or receiving station will generate this error when emulating a 2780 if no message text is available within the response timeout interval after the acknowledgement of the previous text block. The likely explanation for this is that the SEND command was interrupted by the operator for longer than the response timeout.

3034 BID ERROR, BID CONTENTION

1) The local station attempted to SEND while the remote station was waiting to SEND.
2) The local station must first receive the waiting transmission from the remote station, or the remote operator must interrupt and exit his command.
3) This error can result from the local operator specifying a local pathname on a SEND command and the remote operator specifying a remote pathname on a RECEIVE command. It will also result if both operators specify remote pathnames.

3035 BID ERROR, ENQ LIMIT

1) The remote station failed to respond positively or negatively to the local station's attempts to send.
2) If the remote station is using manual connection over a telephone circuit, this error will result if the SEND command is entered before the remote operator has switched control of the communication circuit from his telephone to his modem. Increasing the retry limit on the unit specification record will allow the remote operator more time to complete the connection procedure.
3) This error will occur if the remote station is responding too quickly to the local station. This can result from setting too short a Clear to Send Delay in the remote station's modem.
4) If the local and remote stations are not using the same baud rate or transmission method this error will result. Verify that the local and remote stations agree as to whether to use ASCII or EBCDIC control characters; if ASCII, verify that the local and remote stations agree on the parity sense: ODD, EVEN, or NONE. Verify that the local and remote stations agree as to whether to use asynchronous or synchronous communications. If asynchronous, verify that the local and remote stations are using the same baud rate. If synchronous, verify that the local and remote modems support the same baud rate. Verify that the local and remote modems are using the same modulation technique.

5) Check the cabling at both the local and remote stations. This error will occur if Transmitted Data, Received Data, Request to Send, Clear to Send, or Received Line Signal Detector signals are not wired correctly. When attempting synchronous communications, this error will occur if modem clock signals are not wired correctly to the computer port.

3036 UNEXPECTED DATA BLOCK

1) A sending station will generate this error if it receives a message text block (a block beginning with SOH, STX, or DLE STX) from the remote station.
2) A receiving station will generate this error if it receives a message text block after sending a Wait Acknowledge (WACK) response.
3) A receiving station will generate this error if it receives a message text block after receiving an end of text block (a message text block ending in ETX or DLE ETX). This usually indicates that an End of Transmission (EOT) message was lost, and a subsequent transmission began before the response timer expired (see error 3033).

3037 TTD ERROR, ENQ LIMIT

1) The sending station generates this error when the receiving station does not respond to a Temporary Text Delay transmission (STX ENQ or DLE STX DLE ENQ).
2) This error may result from attempting to use 3780 emulation to communicate with a remote station which is using 2780 emulation.

3038 RECEIVING STATION ABORT

1) The remote receiving station terminated the file transfer for one of the following reasons:
   a) The remote station detected a protocol error.
   b) A device error occurred when writing to the destination file at the remote station.
   c) The REMOTE NAME specified on the local operator's SEND or RECEIVE command was in error, or the MODE specified on the local operator's RECEIVE command when specifying a REMOTE NAME conflicted with the MODE specified on the remote operator's command. In this case, the "Blocks transferred" on the STATUS display will equal 1.
   d) The remote operator interrupted and terminated the operation.

3039 WRITE ERROR, ENQ LIMIT

1) The sending station generates this error when the receiving station does not respond to a message text block within the allowed number of retries.
2) If the receiving station is using 2780 emulation, this error will result at the sending station if the destination device at the receiving station is busy for longer than 3 seconds multiplied by the sending station's retry limit.

3040 WRITE ERROR, NAK LIMIT

1) The sending station generates this error when the receiving station does not positively acknowledge a message text block within the allowed number of retries.
2) This error can indicate poor quality communication facilities between the two stations. On a switched circuit connection, this may be improved by releasing and reconnecting the link.
3) Too much activity, especially disk activity, at the remote station can cause this error by preventing the system from processing the incoming characters. This situation may be improved by using a lower baud rate or by using IMAGE mode transmission instead of CODED mode.
4) When using CODED mode transmission with ASCII control characters, this error will occur if the local and remote stations do not agree on whether the block check sequence is an LRC or CRC computation. Note that IBM systems use CRC or LRC for non-transparent (CODED) messages depending on whether or not the transparency option is installed. In this case, the "ENQs sent" on the STATUS display will be non-zero, and the "Blocks transferred" on the STATUS display will equal 1.

3041 WRITE ERROR, WRONG ACKNOWLEDGEMENT

1) The remote station responded to a message text block with the wrong value of the alternating acknowledgement, i.e., ACK0 when ACK1 was expected, or ACK1 when ACK0 was expected.
2) This error can occur when sending two files in close succession, when the remote station misses the End of Transmission (EOT) at the end of the first file, and the second file transmission begins before the remote station's response timer expires (see error 3033).
3) Retry the command.

3042 SENDING STATION ABORT

1) The remote sending station terminated the file transfer for one of the following reasons:
   a) The remote station detected a protocol error.
   b) The remote station exhausted its retry limit.
   c) A device error occurred when reading the source file at the remote station.
   d) The remote operator interrupted and terminated the operation.
   e) In 2780 emulation, the local receiving station was interrupted or busy too long. This would have caused a 3039 error at a remote RM/COS sending station.
   f) In 2780 emulation, the remote sending station attempted to send more than one record in a transparent message text block. This can only occur if the remote station is not an RM/COS installation.

3043 START OF HEADING NOT SUPPORTED

1) A message text block was received that began with a Start of Heading (SOH) character. RM/COS does not support message headings.

3044 UNABLE TO SET LINK CONFIGURATION

1) A CONNECT command failed, probably as a result of the baud rate specified for the link device. Further information may be found in the Operator Guide for the model computer in use.

DISK DEVICE ERROR CODES

3050 OFF-LINE

1) Selected disk drive not available for operation as a result of:
   a) An invalid unit number.
   b) Diskette or disk not mounted.
   c) Door not latched.
   d) Power cable or controller cable disconnected.
   e) Disk index not sensed; indicates defective drive or disk.
2) Eliminate cause; if caused by power-down the error will clear within two disk revolutions after power-up.

3051 NOT READY

1) Causes are partially overlapped with the previous error code, viz., the specified disk drive is offline.
2) The specified drive has not reached operational rotation speed; retry.

3052 WRITE PROTECTED

1) The disk mounted in selected drive is write protected.
2) Data cannot be recorded or modified on a write protected disk.
3) Verify that the correct disk volume is being accessed; defeat the write protection with the method appropriate to the drive.

3053 UNIT CHECK (FAULT)

1) The specified unit is inhibited from further disk data transfer operations as a result of error conditions in the disk drive detected by the drive hardware.

3054 ILLEGAL ADDRESS

1) Sector address specified for a disk operation falls outside valid range of sectors on disk volume.
2) Probable software origin; invalid parameters used to access the disk. Refer problem to system support personnel with ample documentation.

3055 SEEK INCOMPLETE

1) Indicates disk drive is unable to locate a cylinder as a result of:
   a) Cylinder address is out of range.
   b) No Restore operation performed since last power cycle or disk change.
   c) A conflict between the disk IDs and cylinder number requested in command.
   d) End of recording surface reached without word count decrementing to zero.
2) Possible software origin; invalid parameters used to access the disk. Refer problem to system personnel with ample documentation.

3056 ILLEGAL COMMAND / UNKNOWN ERROR

1) Indicates operation is illegal or unknown on disk device.
2) Probable software origin; invalid parameters used to access the disk. Refer problem to system personnel with ample documentation.

3057 UNKNOWN ERROR

1) Disk drive indicated unknown error, possibly caused by power fluctuation.
2) Retry operation.
3) If problem persists, refer to problem to system personnel with ample documentation.

3058 DATA ERROR

1) Indicates detection of an error during data transfer from the disk to memory.
2) VCOPY the disk to another disk. VCOPY will tolerate a few errors and in any event copy whatever was read to the destination disk. At least some of the affected file(s) may be recoverable by this method.
3) If problem persists beyond software retries reformatting or replacement of the disk media is required; hardware service is required if the problem persists after disk media changed.

3060 ID ERROR

1) Indicates sector referenced by sector ID not found on specified disk; operation terminated.
2) Disk media not INITIALIZEd, degradation or hardware failure indicated; reformat or replace disk media. Hardware service is required if problem persists.

3061 DATA RATE OVERRUN/UNDERRUN

1) Data transfer rate from/to the disk during a read/write operation is fixed while the data transfer rate from the controller to memory is a function of bus activity and priority.
2) Indicates a lower/upper bound discrepancy between the two rates of transfer such that the integrity of the data transferred is questionable; operation terminates.
3) Investigate compatability of current system configuration; high speed/long burst devices may interact unfavorably.

3062 COMMAND TIMEOUT

1) Each controller command is allotted a preset amount of time to complete; otherwise the command operation terminates.
2) If the command timeout is specified in System Definition File, the value may be too small.
3) Indicates hardware service required for disk controller.

3063 SEARCH ERROR

1) Synchronization field of specified disk sector not found within allotted time and preset controller retries.
2) Indicates disk media hardware degradation; reformat or replace disk media. If problem persists, hardware service required.

3064 COMMAND REJECTED

1) Indicates disk mounted in specified drive not an RM/COS format diskette.
2) Substitute an RM/COS formatted diskette.

3065 NO STATUS

1) Status request returned no status information.
2) Indicates disk not properly attached to computer or is not powered.

TAPE DEVICE ERROR CODES

3070 END OF TAPE

1) An unexpected end of tape status was received.

3071 END OF VOLUME

1) An unexpected end of volume was encountered.

3072 BLOCK TOO LONG

1) The block read from tape was longer than the block size specified for the file.

3073 WRONG BLOCK READ

1) When reading a file with backup block overhead areas, the overhead of a block read was invalid because:
   a) The block type indicator was neither data nor end-of-file.
   b) The file identifier did not match that of the preceding blocks.
   c) The block length found in the overhead was too small or exceeded the amount of data read from tape.
   d) The file section number did not match that expected.
   e) The block number did not match that expected, indicating that some blocks were missed.
   f) The checksum verification failed.
2) Backup block overhead verification can be suppressed by specifying an offset of 14 on the TAPE-ASSIGN command instead of specifying the backup option.

3074 BEGINNING OF TAPE

1) An unexpected beginning of tape status was received.

3075 WRITE PROTECTED

1) The tape mounted in the selected drive is write protected.
2) Data cannot be recorded on a write protected tape.
3) Defeat the write protect by rotating the file protect knob of the tape cartridge, or by inserting a write enable ring in the tape reel.

3076 WRITE AFTER READ NOT SUPPORTED

1) Some tape controllers do not allow a write operation to immediately follow a read operation.

3080-309F Controller Dependent Errors

1) These errors are unique to a particular tape controller. Further information may be found in the Operator Guide for the computer model in use.

3402 BOUNDARY VIOLATION, EXHAUSTED EXTENTS

1) File referenced has reached the maximum permissible number of discontiguous allocations (47).
2) FCOPY the volume to another volume thereby reorganizing the volume's file allocations.

3403 BOUNDARY VIOLATION, EXHAUSTED ADUS

1) File referenced cannot continue expansion due to lack of allocatable disk units on disk where file currently resides.
2) RECLOSE disk to recover possible unused scratch space.
3) Deallocate disk space via DELETE of unused files on current disk.
4) FCOPY file to another disk.

3404 BOUNDARY VIOLATION, FILE NOT EXPANDABLE

1) Consumed primary file allocation and file as currently defined disallows secondary allocations.
2) Redefine file via CHANGE permitting secondary allocations.
3) Check program to ensure proper reference of intended record.

3405 BOUNDARY VIOLATION, EXHAUSTED BLOCKS

1) Attempt to write more than 65535 blocks to the file.
2) Recreate file with a larger block size.

3406 BOUNDARY VIOLATION, DISK TOO FRAGMENTED

1) Unable to find sufficient contiguous disk for one block.
2) Recreate file with a smaller block size.
3) FCOPY the volume to another volume thereby reorganizing the volume's file allocations.

3407 BOUNDARY VIOLATION, ACCESS PAST LAST EXTENT

1) This error is caused by an attempt within the operating system to access past the last ADU. It indicates a system error, and the vendor should be contacted.
2) Copying the file to another disk, and changing the block size, may get around the problem.

3409 BOUNDARY VIOLATION, EXHAUSTED VOLUMES

1) A write request to a tape file or MVD organization disk file failed because the file filled up the volumes allocated it or because the user aborted the mount request.
2) Specify more volumes on the SCRATCH parameter to TAPE-ASSIGN and write the tape file again.
3) Use the INITIALIZE command to prepare more disk volumes and write the disk file again.

JDL ROOT ERROR CODES

4001 INSUFFICIENT MEMORY IN PARTITION

1) Memory size requirements of program exceed memory available in partition.
2) Assigning a file with shared, read only or exclusive write access has exhausted memory available in the shared file partition.
3) The memory needed to load the disk resident portion of a JDL command processor is not available in the partition.
4) The COPY BLOCK SIZE parameter on the COBOL command was insufficient to allow copying of all of the reference library text.
5) Use STATUS command to verify partition sizes and amount of memory used.
6) Reorganize program with more sophisticated techniques: segmentation, overlays, CALL/CANCEL, etc.
7) Specify a larger COPY BLOCK SIZE parameter on the COBOL command.
8) Elect to run in another larger partition if one available in current system configuration.
9) Manually increase size of partition via PARTITION.

4003 SYNTAX ERROR

1) JDL command cannot be parsed due to mistake in syntax.
2) If in interactive mode enter Acknowledge Key, correct syntax and reenter.
3) A syntax error causes termination in batch stream. Refer to the batch listing file, if one exists, to find command in error.
4) This error will also occur if user prompting is requested in a batch stream which has been UNCOUPLEd from its initiating partition.

4004 PARAMETER OF TYPE YESNO NOT ENTERED CORRECTLY

1) For this parameter type valid values are an alphabetic character string beginning with a 'Y' or 'N'.

4005 COMMAND NOT ALLOWED IN INTERRUPT MODE

1) Attempted to enter a JDL command forbidden in interrupt mode.
2) Only a subset of JDL commands are valid in interrupt mode:

CONTINUE	MAP-FLAWS	STATUS
FDUMP	MAP-KEYS	SWITCH
EXIT	MAP-PROGRAMS	SYNONYM
HALT	MAP-SYNONYMS	SYSTEM-FILE
KILL-PARTITION	MESSAGE	TIME
KPRINTER	PARTITION	UNCOUPLE
KTASK	SDUMP	UNLOAD
LIST	SETCOND	VARY
LOAD	SHOW	

BATCH is also a valid command in interrupt mode but must be initiated into an idle partition.

4006 COMMAND NOT ENABLED

1) Entry of a JDL command supplied in factory system but disabled under current system configuration.
2) Edit the JDL Definition file to enable the command (see Chapter 2, System Configuration).

4008 COMMAND EXCEEDS USER PRIVILEGE LEVEL

1) Assigned privilege level of user less than that currently assigned to JDL command entered; command attempt rejected.
2) User privilege level is set by User Definition File entry for user ID at log-in and JDL command privilege is set by JDL Definition File at IPL (see Chapter 2, System Configuration).

400A PARAMETER TOO LONG

1) Parameter value entered exceeds length allowed for for its parameter type.

400B UNRECOGNIZABLE FORMAL PARAMETER FOUND IN BATCH

1) Self-explanatory: the usual format for a JDL command parameter is <keyword>=<type>. In interactive mode the prompt is <keyword>:. In batch streams the command, the keyword and the '=' must be included and syntactically correct.
2) Correct the offending JDL command.

400C TIME NOT INTIALIZED

1) No values entered for TIME parameters at system initialization.
2) Listings, headings of COBOL programs and other functions performed by the system require current real time values.
3) Enter TIME with current values for parameters.

400D PARAMETERS EXCEEDED ROLL MEMORY

1) A parameter value longer than 58 characters was specified.
2) A parameter prompt in batch mode longer than 58 characters was specified.
4) The values specified for all parameters of a command could no be represented in 198 characters of memory.
5) The command attempted too many levels of recursive synonym evaluation.

400E SYNONYM EXPANSION TOO LONG

1) As a result of synonym expansion, a single record of JDL input would exceed 154 characters in length.
2) A record in batch mode indicates user prompting and as a result of synonym expansion the length of the record exceeds 80 characters.
3) In batch mode, rewrite the JDL command to occupy more records, placing fewer parameters on each record.

4020 INVALID OBJECT FILE

1) The object of a JDL command processor was found to be invalid at the time the JDL processor was loaded into memory.
2) Contact system support personnel.

4042 SYNONYM FILE NOT SEQUENTIAL

1) The name specified for a synonym file for this user ID is not a sequential file.
2) Delete and recreate the file as a sequential file with a logical record length of 78.
3) Use the Modify User Identification process to correct the synonym file parameter (see Chapter 2, System Configuration).

4043 SYNONYM FILE RECORD LENGTH NOT 78

1) The name specified for a synonym file for this user ID does not have a logical record length of 78 bytes.
2) Synonym file must be sequential with a logical record length of 78 bytes.
3) Delete and recreate the file with the correct logical record length.
4) Use the Modify User Identification process to correct the synonym file parameter (see Chapter 2, System Configuration).

4044 SYNONYM TABLE TOO LARGE

1) The size requested for the synonym table is so large as to not leave enough room for required JDL processors.
2) The user is logged-in, but no synonym space is provided.
3) Increase the partition size by use of the PARTITION command and then log-out and log-in again.
4) Use the Modify User Identification process to reduce the synonym table size (see Chapter 2, System Configuration).

4051 SYNONYM TABLE FULL

1) The number of synonyms defined in the synonym file exceeds the space requested for a synonym table.
2) The user is logged-in with as many of the synonyms that will fit in the synonym table. However, no synonyms are saved at log-out time.
3) The number after MEM in the error message indicates the size required to hold all of the synonyms defined in the synonym file.
4) Use the Modify User Identification process to correct the synonym table size parameter to at least that indicated by the error message (see Chapter 2, System Configuration).

4052 INVALID SYNONYM RECORD

1) The data in the specified synonym file is not consistent with a synonym file.
2) The user is logged-in, but no synonyms are provided.
3) Verify that the name specified for the synonym file associated with the user ID is correct (see Chapter 2, System Configuration).

PROGRAM LOADER ERROR CODES

4101 ZERO LENGTH PROCEDURE SEGMENT

1) The procedure segment of a COBOL program is the instruction (constant value) portion. The compiler generates the finished object program into nine logical parts. One of these, the header, contains information required by the loader. The load fails if the program header claims a zero length procedure segment.
2) Check program source.
3) Possible compiler error; call upon system support personnel with documentation.

4102 ZERO LENGTH DATA SEGMENT

1) The data segment of a COBOL program is the nonreentrant, read/write portion.
2) All valid COBOL programs have a nonzero length data segment. Recompile program.

4103 PROGRAM ACTIVE OR NOT FOUND

1) CALL statement invokes a subprogram that is already active.
2) Program not found in any assigned program file which is not assigned with Exclusive All access.
3) Check PROGRAM-ID in source program; confirm program file disk is mounted, and program file is assigned.
4) MAP-PROGRAMS will list all program-names currently available in the partition.
5) Programs contained in program files assigned with Exclusive All access are not available for execution. Release and reassign the program file with Read Only access.

RUNTIME DEBUG ERROR CODES

4301 UNRECOGNIZED BREAKPOINT

1) Entered breakpoint not matched by existing breakpoint.
2) All arguments of the form [xx/]yyyy should indicate program location. If xx/ is omitted, it implies that yyyy is within the resident segment. Check DEBUG column of compilation source listing.

4302 TOO MANY BREAKPOINTS

1) Attempt to set additional breakpoints while three breakpoints remain effective; COBOL object Debug supports a maximum of 3 breakpoints effective simultaneously.
2) Clear some of currently effective breakpoints and retry setting breakpoint.

4303 DUPLICATE BREAKPOINT

1) Attempt to reenter breakpoint while breakpoint effective from previous entry.
2) No action required; breakpoint currently effective.

4304 UNRECOGNIZED COMMAND

1) Error in entering command syntax.
2) Reenter command correctly.

4305 INVALID LINE NUMBER

1) Set line number parameter in Ln command outside logical screen range.
2) Check on available lines for screen I/O and reset; the default is line twenty-four.

4306 INVALID LOCATION

1) The command location format (xx/yyyy) specified a four digit location optionally preceded by a two digit overlay number. If the overlay is specified the subsequent four digits are the hexadecimal location within the overlay.
2) Check overlay number if specified then confirm the validity of the location within that overlay.
3) Check four digit location if overlay unspecified.
4) In either case use the compiler source listing and allocation map.

4307 SYNTAX ERROR

1) Mistake in command syntax entry.
2) Reenter command after confirming syntax.

4308 TRUNCATION OF MODIFY DATA

1) Signal that in the course of processing the Modify command, the bytes occupied by data newly assigned to the specified location exceeds the size of that location after type conversion and has accordingly been truncated.

4309 INVALID DATA TYPE

1) An attempted Display or Modify command failed as a result of an invalid data type value.
2) Valid data type entries for both commands are listed in Appendix H.

4310 INVALID NUMERIC VALUE

1) Command entered rejected as a result of an invalid numeric value.
2) A numeric parameter value must not contain nondigit characters. A decimal parameter may not contain the hexadecimal digits A-F.
3) Note that all numeric values must be within proper range (e.g. overlay number xx is limited to range 0 to 7F).

4311 WAITING FOR ACKNOWLEDGEMENT OF DISPLAY

1) Display command outputs results to screen one line at a time. The displayed line remains until positively acknowledged.
2) Press the Acknowledge Key to acknowledge the displayed data. Next line will be displayed, or, if no more lines of data exist, the command prompt will be displayed.

COBOL RUNTIME INTERPRETER ERROR CODES

4400 PROGRAM LOCATION OF PREVIOUS I/O ERROR

1) Displays program location counter and name of COBOL program that caused previous I/O error.

4401 COMPILATION ERROR ENCOUNTERED

1) Attempted execution of source statement failed as a result of statement having been compiled in error.
2) Correct statement syntax and recompile.

4402 TRACEBACK

1) Informative as to location of program source module where error tagged by previous error code occurred; an indication of where in the subroutine calling sequence the previous error occurred.

4403 DATA REFERENCE ERROR

1) The value of a subscript is zero or exceeds the maximum value defined by an OCCURS clause.
2) A group item containing an OCCURS ... DEPENDING ON clause is referenced, and the DEPENDING ON variable has a value of zero.
3) A formal parameter (linkage USING item) is referenced in a COBOL subprogram and the item is not fully contained within the actual parameter passed to the subprogram as the corresponding argument. The data description of the formal parameter defines more character positions than the data description of the actual parameter.

4404 ILLEGAL PROCEDURE NESTING

1) Illegal attempt to PERFORM a procedure, either because the procedure is already active and the requested PERFORM is recursive, or because the requested procedure is in an independent segment different from the currently active segment.
2) The COBOL language definition prohibits recursive procedure calls. It also prohibits invoking one independent segment from another, either directly or indirectly.
3) The detection of this error can be suppressed by setting the PERFORM attribute with the FLAG-PROGRAM command.

4405 INDEPENDENT SEGMENT LOAD ERROR

1) Disk I/O error occurred attempting to load an independent segment of the COBOL program. The preceding error message indicated the specific I/O error which occurred along with the logical name assigned to the program file.
2) Review cause of I/O error, correct problem (e.g., if disk drive not ready, ready the disk drive) and reexecute program.

4406 ADVANCING VALUE ERROR

1) Value specified in ADVANCING clause invalid.
2) ADVANCING value must be in the range 0 to 255, inclusive.

4407 ILLEGAL INSTRUCTION ENCOUNTERED

1) Runtime interpreter has encountered an undefined object instruction.
2) This error should occur only as a result of a compiler or a runtime internal defect; consult system support personnel with proper documentation.
3) This error may result when a program was compiled on a later release than the one on which it is being executed.
4) Program file may be corrupted. If so, recompilation should correct problem.

4408 ILLEGAL LABEL REFERENCE

1) Attempt within source program to transfer control to an undefined paragraph or section (compilation listing indicates error).
2) If error is in source program then correct and recompile. If error is in either compiler or runtime interpreter then consult system support personnel with adequate documentation.

4409 ILLEGAL DATA DESCRIPTOR

1) Error detected during data descriptor vector (dope) decoding at execution time.
2) Should occur only as a result of a compiler or runtime interpreter internal defect; consult system support personnel with adequate documentation.
3) Program file may be corrupted. If so, recompilation should correct problem.

4410 ILLEGAL EDIT PICTURE

1) Source statement implies editing of data and associated PICTURE clause is in error.
2) Should occur only as a result of an internal defect of the compiler or runtime interpreter; consult system support personnel with adequate documentation.
3) Program file may be corrupted. If so, recompilation should correct problem.

4411 ERROR OCCURRED WHILE USING STATION

1) Indicates error during performance of logical I/O to screen via ACCEPT and DISPLAY clauses.
2) ACCEPT and DISPLAY are not allowed from a nonterminal partition.

4412 ILLEGAL ARGUMENT

1) Indicates an attempt to reference more arguments in a called program than were passed by the calling program. Arguments are the linkage data items listed in the USING phrase of the Procedure Division header. This error occurs in a called program when a statement is executed referencing one of the arguments for which no corresponding data item was passed in the USING phrase of the CALL statement in the calling program.
2) Can occur during an I/O statement if FILE STATUS or relative KEY linkage data items are used.
3) When occurring from CALL to a subroutine from the C$SUBS COBOL subroutine library may indicate an improper number of arguments was specified or that the size of one the arguments was invalid. Can also indicate that the User Link logical filename is improperly CONNECTed when using C$COMM.
4) Program file may be corrupted. If so, recompilation should correct the problem.
5) Can be an indication of an internal defect of the system. Consult system support personnel with adequate documentation.

4413 ATTEMPT TO CANCEL ACTIVE PROGRAM

1) Indicates that CANCEL statement references a program currently active and as a result the CANCEL statement is invalid.
2) Signals a defect in the run unit organization: reorganize run unit by modifying affected source programs and recompiling them.

4414 ILLEGAL RETURN OR RELEASE

1) RELEASE or RETURN statement execution failed because:
   a) No sort-merge operation was in progress.
   b) The associated file is not the correct sort-merge file for the sort-merge operation which is in progress.
2) Correct the source program so that RELEASE or RETURN is executed only in an input or output procedure invoked by a SORT or MERGE statement for the same sort-merge file.

4415 UNSUCCESSFUL SORT OR MERGE INITIALIZATION

1) SORT or MERGE statement execution failed because:
   a) A sort-merge operation was already in progress for this program.
   b) The sort-merge processor is not available in the system load library.
   c) The sort-merge processor could not be successfully loaded.
   d) The sort-merge processor returned an error code which was previously displayed.
2) Verify that, in the system being used:
   a) The program does not attempt to nest sort-merge functions; sort-merge input and output procedures are not allowed to execute a SORT or MERGE statement.
   b) The sort-merge processor does exist and is enabled.
   c) The partition size is large enough to accomodate the sort-merge function in addition to the calling program and all associated file overhead requirements.

4416 PROCEDURE REFERENCE ERROR

1) Error detected when entering or exiting a USE procedure, or when performing a procedure.
2) Error should occur only as a result of a compiler or runtime interpreter internal defect; consult system support personnel with adequate documentation.
3) Program file may be corrupted. If so, recompilation should correct the program.

4417 ALTER STATEMENT NOT YET EXECUTED FOR PARAGRAPH

1) The program attempted to execute a GO TO statement with no destination that was not yet modified by an ALTER statement.
2) Correct the source program so that the GO TO statement is given a destination before it is executed.

4418 ILLEGAL EXPONENT VALUE

1) An exponent used in a COMPUTE statement had a nonzero fractional part.
2) Only integer exponent values are supported.

ASSIGN ERROR CODES

5001 ATTEMPT TO ASSIGN DIRECTORY

1) Attempt to assign a logical name to a directory file rejected.
2) Check pathname.

5002 LOGICAL NAME ALREADY EXISTS

1) Attempt to ASSIGN a logical name to a file or device was rejected because the logical name was already ASSIGNed to a file or device.
2) Choose another logical name or RELEASE the existing logical name assignment.

5003 BAD PATHNAME

1) System unable to access the desired pathname because:
   a) An intermediate directory level was missing or misspelled in pathname or
   b) The file itself (the identifier in last position) is missing or misspelled.
2) Check pathname for these errors.

5004 ASSIGN REQUIRES HIGHER PRIVILEGE

1) Conflict between privilege level of file and user's privilege level.

5005 FILE CANNOT BE ASSIGNED WITH REQUESTED ACCESS

1) Conflict between access type applied in a previous ASSIGN and the current access type ASSIGNment.
2) Example: previous ASSIGNment applied exclusive all access type and current ASSIGNment attempts to apply shared access type to the file.
3) Previous access type must be revised by original assigner, or current assigner must alter his access type requirements.
4) No shared file partition exists to allow sharing of files. Reconfigure system to include an adequate shared file partition.
5) During terminal log-in, this error indicates the synonym file is already assigned to another user, probably because another station is logged in with the same user id.

5006 INVALID DISK OR DEVICE NAME

1) Attempt to reference a nonexistent device by device name or by device or volume name in a pathname.
2) Execute STATUS to confirm names of devices and loaded volumes.

5007 VOLUME NOT LOADED

1) Attempt to assign to a pathname on a specific disk drive in which no volume is loaded.
2) Correct syntax or LOAD volume.
3) Execute STATUS to confirm names of loaded volumes.

5008 FILE NOT RECLOSED

1) Indicates a file that was opened and not successfully closed, probably as a result of power failure or removing disk improperly.
2) Perform RECLOSE operation on this file.

5009 DEVICE VARIED OFF

1) Attempt to reference a disabled device.
2) If device is operational, use VARY command to make it available.

5010 FILE RENAME IN PROGRESS

1) Indicates a file being renamed from one directory to another at the time of a power failure.
2) Perform RECLOSE operation on volume containing this file.

5011 DIFFERENT VOLUME FOUND

1) Current disk volume is not the volume loaded.
2) Place correct disk volume in drive.
3) UNLOAD previous volume and LOAD correct volume.

5012 DEVICE DEDICATED

1) Attempt to access a file on a volume for which the drive is dedicated to an ongoing process.
2) A disk device may be dedicated by the I-BOOT, INITIALIZE, INSTALL-SYSTEM, RECLOSE, and VCOPY JDL commands, and by assigning an MVD organization file.

5014 INVALID INDEXED FILE

1) Block zero of the file is not the Key Information Block.
2) Key Information Block specifies that the file contains more than 14 alternate keys.
3) Disk must be recreated by copying all files except the file that has been destroyed.

5015 INVALID PROGRAM FILE

1) Number of program entries is not consistent with the size of the program directory.
2) A program entry has been overwritten.
3) The program file must be recreated.

5016 INVALID FILE DIRECTORY ENTRY

1) Invalid file organization in file entry on disk.
2) Invalid number of extents in file entry on disk.
3) Disk must be recreated by copying all files except the file that has been destroyed.

5017 PROGRAM FILE IS EMPTY

1) An attempt was made to assign with other than Exclusive All (EA) access to a program file that does not contain any program object.

5018 WRONG TYPE OF DEVICE

1) The type of device specified is not accepted by the JDL command that was entered.
2) Communications devices should be accessed only with the CONNECT command. A Bisync Link (BLnn) device may be accessed with a CONNECT command followed by a SEND, RECEIVE, or FTS command. A User Link (ULnn) device may be accessed only with a CONNECT command followed by a COBOL program execution (see C$COMM subroutine description in Appendix B, COBOL Subroutine Library Package).
3) A tape device is assigned with the TAPE-ASSIGN command, not with the ASSIGN command. The only commands which may access a tape drive directly (without a logical name assignment) are FILE-BACKUP, FILE-RESTORE, and FILE-VALIDATE.

5019 INCORRECT PATHNAME SYNTAX

1) The pathname specified contained two consecutive periods, a trailing period, an edgename with more than 8 characters, an edgename that did not start with an upper case alphabetic character, or an edgename that contained characters other than upper case alphabetic characters, numeric characters, dollar sign ($), or hyphen (-).

5020 LOGICAL IO MODULE NOT FOUND

1) ASSIGN was unable to locate the organization-dependent system component responsible for I/O to the assigned file.
2) Contact your system support representative.

5021 UNABLE TO DEDICATE DEVICE

1) Attempt to ASSIGN an MVD file on a volume failed because the disk drive could not be dedicated to this logical name.
2) An MVD file may be ASSIGNed only if no other files are assigned on the same volume and only if that volume is not the current system disk.

TAPE-ASSIGN ERROR CODES

5031 DEVICE OR SET NAME REQUIRED

1) No value was specified for the DEVICE or SET NAME parameters of a TAPE-ASSIGN command.
2) When assigning another logical name to an already defined file set, the SET NAME parameter must be specified. When assigning a logical name to a new file or file set, the DEVICE parameter must be specified.

5032 SET NAME NOT DEFINED

1) No value was specified for the DEVICE parameter of a TAPE-ASSIGN command, and the SET NAME specified did not identify an already defined file set.
2) If attempting to assign another logical name to a file set, verify the spelling of the SET NAME, and verify that a logical name associated with that file set is still assigned.
3) When defining a new file set, the device to use when accessing files of the file set must be provided by the DEVICE parameter.

5033 DEVICE SET CONFLICT

1) The SET NAME specified identified an already defined file set, but the device name did not match that used when defining the file set.
2) If attempting to associate a logical name with the already defined file set, correct or omit the DEVICE specification.
3) If attempting to define a new file set, use a different SET NAME or release all files associated with the already defined file set.

5034 INVALID SET NAME SYNTAX

1) The SET NAME specified contained more than 6 characters.

5035 VOLUME PARAMETER REQUIRED

1) No values were specified for the VOLUME or SCRATCH parameters when assigning a new logical name to a tape device.
2) If the file already exists, a value must be specified for the VOLUME parameter. If the file does not exists, a value must be specified for the SCRATCH parameter.

5036 INVALID VOLUME SYNTAX

1) A volume name with more than 6 characters was specified for the VOLUME, SCRATCH or PREMOUNT parameter.
2) More than one file count was specified for a volume of an unlabelled tape on a device with one logical track.
3) More than four file counts were specified for a volume of an unlabelled tape on a device with four logical tracks.

5037 LOGICAL NAME ALREADY EXISTS

1) The logical name specified on a TAPE-ASSIGN command is already used to assign a logical name to a file or device.
2) Choose another logical name, or release the existing logical name assignment.

5038 FILE COUNT ILLEGAL

1) A file count was specified with a volume of a labelled tape set.
2) A file count was specified when assigning to a tape device which supports an End of Volume tape mark.
3) A file count was specified with a volume for the SCRATCH parameter. Since a scratch tape does not yet contain any files, it is illegal to specify the number of files already on the tape.

5039 INVALID FILE COUNT SYNTAX

1) The file counts on the tracks of a tape were not separated by slash (/) characters.
2) A file count specified for a volume contained characters other than decimal digits.
3) A file count specified for a volume exceeded 65535.

5040 TOO MANY VOLUMES

1) More than forty volumes were specified on the VOLUME and SCRATCH parameters when assigning to a tape device with one logical track.
2) More than ten volumes were specified on the VOLUME and SCRATCH parameters when assigning to a tape device with four logical tracks.

5041 VOLUME CONFLICT

1) A volume name specified for the VOLUME or SCRATCH parameter matches that specified for a different file or file set of the same user.

5042 INVALID LABELS SPECIFICATION

1) A value other than NONE was specified for the LABELS parameter.

5043 LABELS CONFLICT

1) The value specified for the LABELS parameter did not match that specified when the file set was defined.

5044 INVALID FILE NAME

1) The value specified for the FILE NAME parameter was longer than 17 characters.

5045 VOLUME PARAMETER ILLEGAL

1) A value was specified for the VOLUME, SCRATCH or PREMOUNT parameter when assigning to an already defined file set.
2) When assigning more than one logical name to a file set, the entire set of volume names must be specified on the first TAPE-ASSIGN command, even if the first file does not occupy all of the volumes.

5046 INVALID FORMAT SPECIFICATION

1) A value other than F, D, S, V, U or <V, S> was specified for the FORMAT parameter.
2) FORMAT = <V, S> was specified for a tape with ANSI labels.
3) A format of D or S was specified for a tape with IBM labels.

5047 INVALID PRINT CONTROL

1) A value other than NONE was specified for the PRINT CONTROL parameter.

5048 INVALID CODE SET

1) A value other than ASCII was specified for the CODE SET parameter.

5049 INVALID PADDING CHARACTER

1) The value specified for the padding character exceeded 255.

5050 INVALID RETENTION PERIOD

1) A nonzero retention period was specified for an unlabelled tape.

5051 DENSITY MISMATCH

1) The value specified for the DENSITY parameter did not match that specified when the file set was defined.

5052 BACKUP OPTION AND OFFSET PARAMETER CONFLICT

1) OPTIONS = BACKUP was specified, and a nonzero value was specified for the OFFSET parameter.
2) A backup format tape automatically has a block offset of 14 characters.
3) Omit either the BACKUP option or the OFFSET parameter.

5053 PREMOUNT VOLUME NOT FOUND

1) The volume name specified for the PREMOUNT parameter was not any of the volume names specified to the VOLUME and SCRATCH parameters.

5054 PREMOUNT CONFLICT WITH LOADED VOLUME

1) The PREMOUNT parameter was specified, but the system knows there is a volume for a different file set still mounted in the drive.
2) Releasing the logical names associated with the file set in the drive will cause the currently mounted tape volume to be unloaded.

5055 INVALID BLOCK OFFSET

1) The value of the OFFSET parameter was zero or greater than 99.

BATCH and CHAIN ERROR CODES

5101 BATCH ALREADY ACTIVE IN USER'S PARTITION

1) Attempt to execute Batch in user's partition and user partition not idle. BATCH can only execute in an idle partition, either terminal or nonterminal.
2) Execute STATUS to find an idle partition. If memory assigned to the partition is smaller than that required, then execute PARTITION and adjust accordingly.

5102 BATCH CANNOT BE NESTED DEEPER THAN 4 LEVELS

1) A batch command can initiate a batch stream which contains another BATCH command. The initiating batch stream suspends execution until the invoked batch stream terminates. This nesting of BATCH commands is permitted up to a level of 4.
2) Note that a BATCH command that initiates a batch stream in another partition is not nesting the batch streams. These batch streams process concurrently.

5103 PARTITION NUMBER ILLEGAL

1) Partition parameter value entered refers to a nonexistent partition, an unused partition, a terminal partition, or the shared file partition.
2) Execute STATUS command and revise partition parameter value accordingly so as to refer to an idle nonterminal partition.

5104 LIST = YES SPECIFIED WITHOUT A LIST NAME

1) Listing of commands in batch stream desired (YES value elected for LIST JDL parameter) but no file pathname entered for LIST NAME. LIST NAME is a required parameter with no default when LIST = YES is specified in interactive mode.
2) Reenter the command and specify an existing file pathname or a listing device (e.g. LP01, ME, MYLP).

5105 SPECIFIED PARTITION IS BUSY

1) Partition value specified in BATCH command is the partition where the batch stream is to be executed (distinction should be made between the partition where the BATCH command itself executes and the partition where the batch stream specified in the command will execute). This partition whether terminal or nonterminal must always be idle.
2) Check partition state via STATUS command and revise partition number value accordingly.

5106 OVERLAY LOAD ERROR

1) Disk error received while loading an overlay of the Batch Command processor.

5107 INSUFFICIENT MEMORY IN REQUESTED PARTITION

1) The amount of memory allocated to the requested nonterminal partition is too small to execute a BATCH command.
2) Increase the destination partition size by using the PARTITION command.

5108 BATCH INTO ANOTHER PARTITION ILLEGAL IF TIME NOT ENTERED

1) If time is not initialized. BATCH is only allowed in the invoking partition.
2) Initialize time by using the TIME command.

CHANGE ERROR CODES

5202 SYSTEM FILE MAY NOT BE CHANGED

1) The CHANGE command may not be used on the current System Image File or JDL Image File on a disk volume.
2) The INSTALL-SYSTEM or REMOVE-SYSTEM command may be used to remove the system protection from these files.

COBOL COMPILER ERROR CODES

All references to Roll and Roll pointer(s) refer to stack data structures utilized by the the Compiler.

5300 COMPILATION TERMINATED BECAUSE OF PREVIOUS ERROR

1) Informing that the compilation cannot continue as a result of a fatal compilation error.
2) The specific error has previously been tagged as to type by the appropriate error code prior to termination.

5301 NULL SOURCE PROGRAM OR FATAL SYNTAX ERROR

1) A syntax error in COBOL source code caused the COBOL compilation to terminate.
2) Verify contents of file assigned for source.
3) Review source statements and revise defective syntax; recompile; focus on possibility of blank line at beginning or end of source code.

5302 NO MEMORY AVAILABLE FOR ROLL MEMORY

1) Insufficient user memory to allow compiler to allocate memory for processing stacks.
2) Increase partition size.

5303 ROLL POINTER OVERFLOW

1) Compiler maintains references to roll (stack) entries by means of single word pointers; indicates excessive number of a particular type of entry (i.e. data-names, paragraph-names, etc.).
2) Indicates source program requirements excessive; reorganize program and recompile. Segmentation reduces pointer requirements.

5304 ROLL MEMORY OVERFLOW

1) Insufficient system memory to dynamically allocate roll (stack) memory during compiler processing.
2) Indicates source program processing memory requirements are excessive:
   (a) increase partition size;
   (b) delete cross reference option; or
   (c) reorganize source program.
3) Segmentation reduces memory requirements for compilation as well as execution.

5305 SOURCE PROGRAM SIZE OVERFLOW

1) Memory size conflict between source program memory requirements and total system memory available for the specified COBOL source program sections: data must reference < 65K and COBOL code/instructions < 32K.
2) COBOL source program must be reorganized and confined within COBOL compiler limitations; recompile. Segmentation will reduce code size only.

5306 COMPILER INTERNAL LOGIC FAILURE

1) Compiler unable to complete the compilation due to a problem internal to the compiler.
2) Refer problem to system support personnel with ample documentation.

5307 COPY STATEMENT NESTING LIMIT EXCEEDED

1) COPY statement nesting within library-text (copied source files) exceeds five levels.
2) Modify the original source and/or library-text such that no more than five levels of library-text files need be open at any one time.

CONTINUE ERROR CODES

5402 PARTITION NUMBER ILLEGAL

1) Partition value entered for partition parameter unrecognized because it references a nonexistent partition, an unused partition, or a terminal partition other than the one in which the command is executing.

5403 PARTITION NOT SUSPENDED

1) Partition value references an active partition (i.e., the partition has not been HALTed or interrupted).

CREATE ERROR CODES

5501 FILE ALREADY EXISTS

1) File pathname specified for the NAME parameter already exists.
2) CREATE new file with same directory levels and different file name or different directory levels and same file name.

5502 SCRATCH TYPE CONFLICT

1) A disk device name was specified but SCRATCH was omitted from the file type.
2) A pathname was specified for a file of type SCRATCH.

5503 LOGICAL NAME ALREADY EXISTS

1) Value entered for LOGICAL NAME parameter is currently assigned to a file or device.

5504 TYPE COMPRESSED NOT ALLOWED

1) File type COMPRESSED is not allowed with relative, program, or MDS organization files.

5505 TYPE SPOOL NOT ALLOWED

1) File type SPOOL is allowed only with sequential and MVD organization files.

5506 UNABLE TO ALLOCATE IN SINGLE EXTENT

1) The disk volume contained insufficient contiguous storage to allocate the file in a single extent and the SINGLE file type attribute was specified.
2) Delete unneeded files from the volume, use a different volume, or do not specify the SINGLE file type attribute. Note that the System Image File and the JDL Image File must be allocated as a single extent; so, for these files, the last remedy indicated is not appropriate.

5507 TYPE AUTO NOT ALLOWED

1) The AUTO type may not be specified with the WORK type.
2) File type AUTO is not allowed with MDS or program organization files.

5508 MULTIVOLUME DISK FILE NOT AT TOP LEVEL

1) An organization of MVD was specified, but the pathname did not refer to the volume directory.
2) An MVD organization file must be a direct descendent of the volume directory.

5509 LOGICAL IO MODULE NOT FOUND

1) CREATE was unable to locate the organization-dependent system component responsible for I/O to the created file.
2) Contact your system support representative.

5510 UNABLE TO DEDICATE DEVICE

1) Attempt to CREATE and assign an MVD file on a volume failed because the disk drive could not be dedicated to this logical name.
2) An MVD file may be assigned only if no other files are assigned on the same volume and only if that volume is not the current system disk.
3) The file is CREATEd but no logical name is assigned to the file.

SCRATCH ERROR CODES

5521 TEMPLATE NAME DOES NOT EXIST

1) No file is assigned with the logical name specified for the TEMPLATE NAME parameter.
2) The template file must be assigned prior to entering the SCRATCH command.

5522 DEVICE NOT DISK

1) Logical name entered for the TEMPLATE NAME parameter is not assigned to a disk file.
2) Only disk files may be used as a template file for the SCRATCH command.

5523 LOGICAL NAME ALREADY EXISTS

1) Logical name entered for the LOGICAL NAME parameter is currently assigned to a file or device.
2) Release the logical name or choose a different logical name for the SCRATCH command.

5524 UNABLE TO ALLOCATE IN SINGLE EXTENT

1) The disk volume contained insufficient contiguous storage to allocate the scratch file in a single extent and the SINGLE file type attribute applies.
2) Delete unneeded files on disk volume, use a different disk volume, or specify NOT-SINGLE in TYPE parameter. Note that the System Image File and the JDL Image File must be allocated in a single extent; so, for these files, the last remedy indicated is not appropriate.

5525 MULTIVOLUME DISK ORGANIZATION ILLEGAL

1) The logical name entered for the TEMPLATE NAME parameter is associated with a multivolume disk organization file.

DELETE ERROR CODES

5601 FILE NOT FOUND

1) The last edge of the pathname entered was misspelled or does not exist.
2) Use a LIST or SHOW command to verify directory or file name.

5603 DELETE PROTECTED

1) File/directory entered for NAME parameter of DELETE command is delete protected.
2) FILE-RESTORE attempted to replace to delete protected file.
3) Execute CHANGE command and alter the protection of the file or directory.
4) Recall that:
   a) a write protected file is automatically delete protected.
   b) files with the system file attribute cannot be deleted.

5605 DIRECTORY NOT EMPTY

1) A directory could not be deleted because one of the files in the directory could not be deleted.

5606 DIRECTORY TYPE MISMATCH

1) The pathname specified identified a directory and DIRECTORY = NO was specified.
2) The pathname specified did not identify a directory and DIRECTORY = YES was specified.

DIRECTORY ERROR CODES

5701 ESTIMATED NUMBER OF FILES ERROR

1) Zero value specified for estimated number of files parameter.

5702 BAD PATHNAME

1) The pathname entered was the volume name or device name of a disk device; the pathname did not specify the name of the directory to be created.

5703 FILE ALREADY EXISTS

1) Directory referenced by pathname already exists and cannot be recreated.
2) Alter directory name entered as last value in pathname or create new directory level.

SDUMP and SMODIFY ERROR CODES

5801 UNABLE TO OBTAIN BUFFER MEMORY

1) Not enough memory space remains in partition to accomodate the size of one sector.
2) Increase partition size by at least the size of one sector.

5802 ERROR OPENING LISTING FILE

1) Error encountered during opening of file or device assigned to LO.
2) Ensure valid file or device assignment.

5803 INVALID DEVICE NAME

1) Device name entered for DEVICE parameter required for both SMODIFY and SDUMP unrecognized either because device name referenced is not a disk drive or is currently disabled.
2) Execute STATUS to determine currently enabled disk devices. Execute VARY to enable device.

5804 INVALID SECTOR NUMBER

1) Sector value entered for SECTOR parameter required for both SDUMP and SMODIFY is out of range for disk referenced.

5805 BYTE DISPLACEMENT NOT EVEN

1) Byte value entered for BYTE parameter of SMODIFY command is not an even number and therefore does not reference a byte beginning on a word boundary within the sector.
2) Add or subtract 1 from byte value entered.

5806 MODIFICATION SPANS SECTOR BOUNDARY

1) Indicates that an entry for the BYTE, VERIFY, or PATCH parameters of the SMODIFY command overflows beyond the cited sector. All parameter values are confined to one sector. A modification may not span a sector boundary.

5807 VERIFICATION ERROR

1) Current contents at the specified disk address cited in SMODIFY command do not match values entered for VERIFY parameter. Current contents are not altered.
2) Check if values entered for DISK, SECTOR and BYTE parameters are the intended values.
3) Execute SDUMP command to confirm disk contents.

5808 INVALID NUMBER OF SECTORS

1) Zero was specified for the number of sectors to be dumped.
2) The number of sectors to be dumped when added to the starting sector number exceeded the number of sectors on the disk.
3) A number of sectors other than 1 was specified when dumping starting at sector zero. When dumping sector zero, only one sector may be displayed.

EXECUTE ERROR CODES

6001 NO PROGRAM FOUND

1) Value entered for PROGRAM NAME parameter could not be located in any of the assigned program files.
2) Ensure that name specified corresponds to the program name in the PROGRAM-ID clause of the COBOL program to be executed.
3) MAP-PROGRAMS will list all program names currently available in the partition.

6005 DEBUG NOT PRESENT

1) The DEBUG option was requested but the JDL Image File does not include a debug processor.

EXIT and QUIT ERROR CODES

6101 EXIT INVALID IF PARTITION IDLE

1) EXIT entered in user's terminal partition and no active program or batch stream exists in the partition.
2) Recall these points: (a) in batch mode EXIT terminates the batch stream; (b) in interactive mode EXIT is valid only in interrupt mode; (c) in interactive mode EXIT applies only to user's terminal partition.

6102 CANNOT QUIT WHILE BATCH STILL ACTIVE

1) QUIT command entered while a batch stream is still active in a nonterminal partition initiated by the user's terminal.
2) Batch streams initiated by the user in a nonterminal partition display any errors on the user's terminal; therefore, the user is not permitted to log-off until the batch streams terminate.
3) Wait until batch stream(s) terminate or use the UNCOUPLE JDL command.

6103 LOST MEMORY

1) QUIT command verifies that all dynamically assigned memory has been released.
2) Indicates system error. Refer problem to vendor with ample documentation.

FILE-BACKUP ERROR CODES

6121 PATHNAME EXCEEDS RECORD CAPACITY

1) The complete pathname of a file to be backed up exceeded the backup record size less 114 characters.
2) The backup operation will continue unless this is the initial node being backed up.
3) Use a backup file with a larger record size to backup this file.

6122 BAD CUTOFF DATE SYNTAX

1) The CUTOFF DATE parameter was not "TODAY", nor of the form YYMMDD or YY/MM/DD, where YY, MM, and DD are each two decimal digits.
2) The year specified was less than 80.
3) The month specified was zero or greater than 12.
4) The day specified was zero or greater than 31.
5) The backup operation was not started.

6123 INVALID FILE DIRECTORY ENTRY

1) A file directory entry was found that did not contain a valid RM/COS file organization.
2) The minimum logical record size in the file directory entry exceeded the maximum logical record size.
3) The extents in the file directory entry claimed to include more ADUs than are on the disk.
4) The backup process skipped the file with the invalid file directory entry.

6124 OPTIONS SPECIFICATION CONFLICT

1) Both BEFORE and AFTER options were specified.
2) The BEFORE option was specified without specifying a CUTOFF DATE.
3) The NONE option was specified with another option.

6125 OPERATOR REQUESTED TERMINATION

1) The backup process was aborted, either by the user entering interrupt mode and executing an EXIT command when executing in a terminal partition, or by a user entering a KILL-PARTITION or KTASK command.

6126 RECORD SIZE TOO SHORT

1) The backup file record size was less than 256. The backup operation was not started.
2) The backup file record size may be specified on the FILE-BACKUP command, or when the backup disk file is created or the backup tape file is assigned.
3) Specify a value greater than 255 for the RECORD SIZE parameter of the FILE-BACKUP command.

FILE-RESTORE ERROR CODES

6141 FILE NOT PRESENT OR MISSED

1) The backup INDEX specified contains no files.
2) The BACKUP NAME specified does not include nor is included in the SOURCE NAME of the specified backup INDEX.
3) The BACKUP NAME specified included the SOURCE NAME of the specified backup INDEX, but the BACKUP NAME did not match any file backed up in the specified backup INDEX.
4) No files were restored. Correct the BACKUP NAME and restart the restore process.

6142 FIRST FILE RECORD NOT NODE RECORD

1) This informative error indicates that the backup record after the index record did not identify a file.

6143 UNEXPECTED END OF FILE

1) The INDEX specified exceeded the number of backups that were appended to the backup file. The restore process was not started.
2) The backup INDEX specified contained no files and was the last backup written to the backup file. The restore process was not started.
3) The last file of the backup was only partially backed up. FILE-RESTORE either left the file undefined, or marked the file as needing RECLOSE.

6144 TOO MANY ADUS

1) A file was not restored because it would require more than 65535 ADUs of the specified disk.
2) The file may be restored to a disk with a larger ADU size.

6145 GENERATED ACCESS NAME ERROR

1) Pathname of the file that was backed up, when combined with the DESTINATION NAME specified to FILE-RESTORE, exceeded 65535 characters.

6146 NONDATA RECORD ENCOUNTERED

1) The restore process found less of the file data than it expected, as a result of the being only partially backed up.
2) FILE-RESTORE either left the file undefined, or marked the file as needing RECLOSE.

6147 DIRECTORY FILE CONFLICT

1) The pathname of the file backed up, when combined with the DESTINATION NAME specified to FILE-RESTORE, matches that of an existing directory.
2) The pathname of the directory backed up, when combined with the DESTINATION NAME specified to FILE-RESTORE, matches that of a nondirectory file.
3) The restore process skipped the file in error.

6148 OPERATOR REQUESTED TERMINATION

1) The restore process was aborted, either by the user entering interrupt mode and executing an EXIT command when executing in a terminal partition, or by a user entering a KILL-PARTITION or KTASK command.

6149 RECORD SIZE TOO SHORT

1) The backup file record size was less than 256. The restore operation was not started.
2) The backup file record size may be specified on the FILE-RESTORE command, when the backup tape file is assigned, or is determined from the file characteristics of a backup disk file.
3) The backup file record size specified to FILE-RESTORE must match that used when the backup file was created.

6150 FUTURE FILE STRUCTURE LEVEL

1) The organization of the file backed up is not recognized as a valid file organization for the release of RM/COS performing the restore operation.
2) The release of RM/COS performing the restore operation is not allowed to write a file with the structure level of the file backed up.
3) The restore process skipped the rejected file.

6151 BACKUP NAME REQUIRED

1) Because the index record of the backup file could not be read, the file restore process required a BACKUP NAME specification to determine how much of a file's pathname to replace with the DESTINATION NAME value.
2) The restore process was not started.

6152 ADD OR REPLACE REQUIRED

1) Neither the ADD nor the REPLACE option was specified to the FILE-RESTORE command.
2) The restore process was not started.

6153 INTERNAL ERROR IN FILE-RESTORE

1) Contact system support personnel. Provide ample documentation.

VCOPY ERROR CODES

6201 DEVICE NOT DISK

1) Device name entered for SOURCE or DESTINATION parameter for VCOPY command is not a disk device.
2) VCOPY recognizes only disk devices as valid.

6206 DESTINATION AND SOURCE ARE SAME DEVICE

1) Same device name entered for SOURCE and DESTINATION parameters.
2) Since VCOPY processes entire volumes two separate disk drives are required.

6207 SOURCE AND DESTINATION ARE NOT INITIALIZED TO THE SAME PHYSICAL OR LOGICAL FORMAT

1) Conflict between SOURCE format and DESTINATION format.
2) SOURCE and DESTINATION physical/logical formats must match (e.g. disk drives must be identical and only RM/COS format disks are allowed).
3) SOURCE and DESTINATION disk volumes must both have been INITIALIZEd as "old" or "new" format disks. RM/COS systems prior to 2.4 made "old" format disks. Releases commencing with 2.4 make "new" format disks.
4) INITIALIZE destination disk and reenter command.

6208 BAD ADU ON DESTINATION DISK CONFLICTS WITH SOURCE DISK ALLOCATION

1) A mismatch between ADU allocations on SOURCE disk and defective ADUs on DESTINATION disk; processing discontinued.
2) Replace DESTINATION disk and reenter command.

6211 VOLUME NAME DOES NOT MATCH NAME ENTERED

1) Volume name entered for either source or destination disk does not match the name with which the disk volume was initialized.
2) The error text indicates either the source or destination disk and the correct disk volume name by "S/name" or "D/name" respectively.
3) Verify the disk volumes as well as the device and volume names entered; correct as necessary and reenter the command.

6212 EXCESSIVE NUMBER OF VERIFY ERRORS

1) Indicates the number of times a discrepancy between the SOURCE disk ADU and the DESTINATION disk ADU copied are so excessive that VCOPY processing has terminated.
2) Hardware origin: replace defective DESTINATION disk or service hardware.

FCOPY ERROR CODES

6220 OPERATOR REQUESTED TERMINATION

1) The copy process was aborted, either by the user entering interrupt mode and executing an EXIT command when executing in a terminal partition, or by a user entering a KTASK command when executing in a nonterminal partition.

6221 INVALID DESTINATION FILE

1) File organization specified for DESTINATION parameter conflicts with file organization specified for SOURCE parameter.
2) Note these points:
   a) A program file is copied only to a program file.
   b) A sequential file can be copied to any of the the three COBOL file organizations.
   c) A relative organization file can be copied to any of the three COBOL file organizations.
   d) An indexed organization file can be copied to any of the three COBOL file organizations.
   e) An MDS file may be copied to an MDS file or a sequential file. A sequential file may be copied to an MDS file if it is a copy of an MDS file.
   f) If the DESTINATION is an indexed file, the SOURCE must also be an indexed file, or the keys must be defined prior to FCOPY execution.
   g) A directory can only be copied to a directory (optionally at a different level) or a device.
   h) If a device other than a disk is entered it is treated as a sequential file and can only be copied to a file or another nondisk device. If a disk device is entered all the files on the disk are copied.

6222 TOO MANY DIRECTORY LEVELS

1) Indicates the FCOPY stack generated to process through directory trees has overflowed: the level at which the stack overflowed and all deeper levels are not processed.
2) Begin copy at the level where overflow occurred to complete copy.
3) Revise the directory structure to minimize stack depth requirements.
4) Increase the partition size.

6223 INVALID SOURCE FILE

1) The source file is not a valid file for copying to an MDS destination file. The source file must be an MDS file or a sequential copy of an MDS file.
2) Ensure that source and destination were properly specified. If so, source file has been corrupted and cannot be copied to an MDS file.

FLAG-PROGRAM ERROR CODES

6241 PROGRAM NOT FOUND

1) The program name specified cannot be found on the program file.

6242 INCOMPLETE OR INCONSISTENT PARAMETERS SPECIFIED

1) The same option may not be specified on both the ON-FLAGS and OFF-FLAGS parameter.
2) At least one of the parameters ON-FLAGS, OFF-FLAGS or CODE must be nonnull.

FLAW-ADU, FLAW-TRACK and MAP-FLAWS ERROR CODES

6260 INVALID DEVICE

1) The device name specified for the DEVICE parameter is not a disk device name or loaded volume name.

6261 VOLUME NOT LOADED

1) The device specified for the DEVICE parameter does not contain a loaded disk volume.

6262 DEVICE DEDICATED

1) The device specified for the DEVICE parameter is dedicated to an ongoing process.

6263 DISK VOLUME IS OLD FORMAT

1) The referenced disk volume was INITIALIZEd by a version of RM/COS prior to release 2.4.
2) Only disk volumes INITIALIZEd by release 2.4 or later of RM/COS have the necessary disk structures to permit FLAW-ADU and FLAW-TRACK to work.

6264 INVALID ADU NUMBER

1) The ADUS parameter specifies an ADU number larger than the highest numbered ADU.
2) Use the STATUS JDL command to determine the number of ADUs on the disk.

6265 INVALID CYLINDER OR HEAD NUMBER

1) The CYLINDER or HEAD parameter specifies an invalid physical disk track.

INITIALIZE, RECLOSE, VCOPY COMMON ERROR CODES

6281 DESTINATION DRIVE CONTAINS LOADED VOLUME

1) Requested operation not allowed on a loaded volume.
2) Unload volume and reissue command.

6282 DRIVE IS ALREADY DEDICATED TO ANOTHER PROCESS

1) Requested operation not allowed because another process is using the device.
2) Wait until the current process completes and reissue command.

6283 DRIVE CANNOT BE DEDICATED BECAUSE FILES ARE ASSIGNED

1) Requested operation not allowed while files are assigned.
2) Release assigned files and reissue command.

6284 CONFIGURED BIT MAP TOO SMALL FOR THIS DISK

1) The amount of space used by the disk to perform disk allocation exceeds the memory reserved by the system.

6285 DRIVE MAY NOT RECEIVE LISTING OUTPUT

1) The batch listing file output would be to one of the drives that contain a volume that may be removed.
2) Change destination of batch listing file to the printer, or a volume not involved in the process, or request no batch listing.

6286 REQUESTED DISK NOT REMOUNTED

1) The volume mounted does not correspond to the volume that was removed.
2) Disk I/O errors can inhibit recognition of the mounted volume.
3) Verify that the correct volume is remounted and acknowledge the error message.

6287 SYSTEM DRIVE NOT DEDICATED BECAUSE OTHER PARTITIONS ARE ACTIVE

1) The system drive may not be used for processes requiring its removal if other partitions are active.
2) Use another drive, or wait until the system is idle.

6288 DISK NOT INITIALIZED OR FORMAT UNKNOWN

1) Specified device contains a disk that is not RM/COS format.
2) Verify correct volume or perform INITIALIZE command to establish an RM/COS format disk.

HALT and KTASK ERROR CODES

6301 SYSTEM-SHUTDOWN IS IN PROGRESS

1) HALT, KILL-PARTITION and KTASK are not permitted if a SYSTEM-SHUTDOWN command has been entered.

6302 COMMAND ABORTED BY EXIT COMMAND

1) The user aborted the command before completion by entering interrupt mode and executing an EXIT command.

6303 SPECIFIED PARTITION IS IDLE

1) Partition identified by integer value entered for PARTITION parameter specifies an idle partition.

6304 PARTITION NUMBER ILLEGAL

1) Integer value entered for PARTITION parameter identifies a nonexistent partition, an unused partition, a terminal partition, or the shared file partition.

6305 HALT ALREADY PENDING ON SPECIFIED PARTITION

1) A HALT command specifies a partition which some other partition is already attempting to halt.

INITIALIZE ERROR CODES

6401 DEVICE NOT DISK

1) Device referenced by value entered for DEVICE parameter is not a disk device.
2) Verify device reference. INITIALIZE only valid for disk volumes.

6404 INVALID BAD ADU PARAMETER VALUE

1) One or more of the values entered for bad ADUs is outside the range of the maximum number of ADUs for the disk volume.
2) A disk volume with a bad index track cannot be used as an RM/COS disk and must be replaced by another volume.

6406 INDEX TRACK MARKED AS BAD

1) A track detected as unusable during initialization overlaps the disk overhead area.
2) A disk volume with a bad index track is unusable as an RM/COS volume.
3) Replace disk volume.

6407 INVALID EST NUMBER OF VCATALOG FILES

1) EST NUMBER OF VCATALOG FILES entered as zero or greater than 32,767.
2) Reenter command with valid specification.

6408 INVALID VOLUME NAME

1) Volume names that are four characters in length cannot have a valid device prefix as the first two characters and two digits as the last two characters.
2) Volume names that are two characters in length cannot have the volume name ME.
3) Reenter command with valid volume name.

6501 OBJECT RECORD CHECKSUM INCORRECT

1) The checksum on an input object record did not equal the computed checksum for the object record, indicating that the object file has been corrupted.
2) Reconstruct the object file.

6502 ERROR WHEN CLOSING INPUT OBJECT FILE

1) An I/O error occurred when closing the file accessed by the logical name entered for the OBJECT parameter. The message preceding this message indicated the I/O error code.
2) Confirm that disk containing object file is mounted and that disk drive is operating correctly.

6503 ERROR WHEN CLOSING OUTPUT PROGRAM FILE

1) An I/O error occurred when closing the file accessed by the logical name entered for the PROGRAM parameter. The message preceding this message indicated the I/O error code.
2) Confirm that disk containing program file is mounted and that disk drive is operating correctly. Error may be caused by insufficient storage availability on the disk volume.

6504 ERROR WHEN PROCESSING DATA AREA IDENTIFICATION

1) The input object file is invalid for installation in a program file. The error was detected while processing the data area identification tag. Tag was not present when expected or contained invalid fields. Previous message indicates further detail regarding the error.
2) Reconstruct object file according to rules for installation in a program file.

6505 ERROR WHEN TERMIMATING BECAUSE OF PREVIOUS ERROR

1) Command processor attempted a graceful termination as a result of a previous error and encountered another error during termination. Immediately preceding message indicates specific error encountered.
2) The error is likely to have been a result of the previous error.
3) Refer to previous error and correct accordingly.

6506 INVALID IDENTIFICATION TAG IN INPUT OBJECT FILE

1) Command processor expected a valid identification tag in the object input file. The tag was not found, indicating an invalid object file for installation in a program file.
2) Reconstruct object file according to rules for installation in a program file.

6507 ERROR WHEN INITIALIZING DATA AREA INPUT

1) An error occurred while preparing for data area input from object file. Immediately preceding error message indicates specific error (e.g., an I/O error when reading first object record).
2) Review previous error and correct cause.

6508 ERROR WHEN INITIALIZING INPUT OBJECT FILE

1) Input object file is empty or an I/O error occurred when the first object record was input.
2) Reconstruct object file.

6509 ERROR WHEN INITIALIZING OUTPUT PROGRAM FILE

1) An I/O error occurred when preparing for output to the file accessed by the logical name entered for the PROGRAM parameter.
2) Review previous error message and correct cause of I/O error.

6510 ERROR WHEN INSTALLING DATA AREA OBJECT

1) An error occurred while processing the data area object. Immediately preceding message indicates specific error.
2) Review previous error message and correct cause of error. It may be necessary to reconstruct the object file.

6511 ERROR WHEN INSTALLING PROCEDURE AREA OBJECT

1) An error occurred while processing the procedure area object. Immediately preceding message indicates specific error.
2) Review previous error mesage and correct cause of error. It may be necessary to reconstruct the object file.

6512 INVALID OBJECT TAG IN INPUT OBJECT FILE

1) Invalid tag on tagged object field found in input from object file.
2) Object file is not correct for installation in a program file. Reconstruct object file.

6513 ERROR WHEN PROCESSING IDENTIFICATION LENGTH FIELD

1) Object file contains an invalid identification length field.
2) Reconstruct object file.

6514 ERROR WHEN PROCESSING LOAD ADDRESS VALUE IN OBJECT INPUT FILE

1) Object file contains an invalid load address value field.
2) Reconstruct object file.

6515 ERROR WHEN PROCESSING OBJECT DATA VALUE IN OBJECT INPUT FILE

1) Object file contains an invalid data value field.
2) Reconstruct object file.

6516 ERROR WHEN TERMINATING NORMALLY

1) Failure after processing of output (PROGRAM) file completed and before successful close of the file; as a result the output file is not a valid program file.
2) Review previous error message for cause and correct before retrying command.

6517 INVALID HEXADECIMAL VALUE IN OBJECT INPUT FILE

1) Object file contains an invalid hexadecimal value, indicating an invalid object file structure.
2) Reconstruct object file before retrying command.

6518 ERROR WHEN OPENING INPUT OBJECT FILE

1) An I/O error occurred while trying to open the input object file. Previous message indicates specific error.
2) Review previous error message and correct cause of the error.

6519 ERROR WHEN OPENING OUTPUT PROGRAM FILE

1) An I/O error occurred while trying to open the output program file. Previous message indicates specific error.
2) Review previous message and correct the cause of the error.

6520 ERROR WHEN WRITING OUTPUT PROGRAM FILE

1) An I/O error occurred while trying to write a record to the output program file. Previous message indicates specific error.
2) Review previous message and correct the cause of the error.

6521 ERROR WHEN PROCESSING PROCEDURE AREA IDENTIFICATION

1) The input object file is invalid for installation in a program file. The error was detected while processing the procedure area identification tag. Tag was not present when expected or contained invalid fields. Previous message indicates further detail regarding the error.
2) Reconstruct object file according to rules for installation in a program file.

6522 ERROR WHEN READING INPUT OBJECT FILE

1) An I/O error occurred when attempting to read the next object record. The previous message indicates the specific error.
2) Review previous message and correct the cause of the error.

6523 INPUT OBJECT FILE CONTAINS DATA RELOCATABLE VALUE

1) Data relocatable data values are not allowed.
2) Reconstruct object file without data relocatable data values.

6524 ERROR WHEN PROCESSING SYMBOL DEFINITION TAG

1) Input object file contains a symbol definition tag with invalid fields.
2) Reconstruct object file.

COMBINE ERROR CODES

6531 TOO MANY PROGRAMS

1) The number and size of the programs being combined cause more than 65535 records to be written to the new program file.
2) Combine fewer or smaller programs.

I-BOOT ERROR CODES

6560 DEVICE NOT DISK

1) The device name specified for the DEVICE parameter is not a disk device.
2) A boot program may only be installed upon or removed from a disk volume. Specify a valid disk device name.

6561 INVALID VOLUME

1) The volume name specified for the VOLUME parameter does not match the volume name of the disk volume mounted in the disk device specified via the DEVICE parameter.
2) Ensure that the VOLUME and DEVICE parameter values were correctly entered and that correct disk volume is mounted in the specified device.

6562 INVALID FILE ORGANIZATION

1) The pathname specified for the BOOT FILE NAME parameter refers to a file which in not MDS organization.
2) The System Image File should be specified for the BOOT FILE NAME parameter. This should be an MDS file.

6564 INVALID MEMBER NAME

1) The member name specified for the BOOT MEMBER NAME parameter is invalid.
2) A member name must be from 1 to 8 characters in length optionally followed by a slash (/) and a modifier value. The modifier value is specified as a hexadecimal integer with or without a leading greater than symbol (>).

6566 FILE NOT ALLOCATED AS SINGLE EXTENT

1) The pathname specified for the BOOT FILE NAME parameter refers to a file which is allocated as two or more extents.
2) Copy the offending file to a file created with the SINGLE file type attribute and I-BOOT the copy.

6568 UNKNOWN BOOT TYPE

1) The boot image type is invalid.
2) Verify that the System Image File being installed was correctly constructed and that the BOOT FILE NAME and BOOT MEMBER NAME parameter values were properly specified.

6569 INSUFFICIENT MEMORY

1) The memory available in the user's partition is not sufficient to load the entire boot program, or else the memory is too fragmented.
2) RELEASE any unnecessary logical names.
3) Elect to run in another larger partition if one is available in the current system configuration.
4) Manually increase the size of partition via PARTITION.

INSTALL, REMOVE and TEST-SYSDEFIL ERROR CODES

6580 DEVICE NOT DISK

1) The device name specified for the DEVICE parameter is not a disk device.
2) A system may only be installed upon or removed from a disk volume. Specify a valid disk device name.

6581 INVALID VOLUME

1) The volume name specified for the VOLUME parameter does not match the volume name of the disk volume mounted in the disk device specified via the DEVICE parameter.
2) Ensure that the VOLUME and DEVICE parameter values were correctly entered and that correct disk volume is mounted in the specified device.

6582 INVALID FILE ORGANIZATION

1) One or more of the pathnames specified for the SYSTEM FILE NAME, JDL FILE NAME, or SYSDEFIL FILE NAME parameters refers to a file with incorrect organization.
2) The System Image File and the JDL Image File must be MDS files. The System Definition File must be a sequential file.

6583 INVALID RECORD LENGTH

1) The pathname specified for the SYSDEFIL FILE NAME parameter refers to a file which does not have a logical record length of eighty bytes.
2) The System Definition File must have a logical record length of eighty bytes.

6584 INVALID MEMBER NAME

1) A member name specified for the SYSTEM NAME or BOOT NAME parameter is invalid.
2) A member name must be from 1 to 8 characters in length optionally followed by a slash (/) and a modifier value. The modifier value is specified as a hexadecimal integer with or without a leading greater than symbol (>).

6585 FILE UNDEFINED

1) The pathname specified for the SYSDEFIL FILE NAME parameter refers to an undefined file.
2) The System Definition File must be defined so that it may be opened for input during IPL.

6587 INVALID BOOT SIZE

1) The size of the boot image is invalid.
2) Verify that the System Image File being installed was correctly constructed and that the BOOT NAME parameter value was properly specified.

6588 UNKNOWN BOOT TYPE

1) The boot image type is invalid.
2) Verify that the System Image File being installed was correctly constructed and that the BOOT NAME parameter value was properly specified.

6589 BAD VERIFICATION

1) The pathname specified for the SYSTEM FILE NAME or the JDL FILE NAME parameter references a file which is not a valid System Image File or a valid JDL Image File.
2) Ensure that the proper pathnames were specified or reconstruct the offending image file.

6590 JDL IMAGE FILE CHANGED (IPL REQUIRED)

1) The INSTALL command changed the JDL Image File to a different file than the one indicated at IPL time.
2) An IPL is required to setup the in-memory JDL directory so as to use the new JDL Image File for obtaining JDL processor images.

6591 NO SYSTEM INSTALLED ON VOLUME

1) A TEST-SYSDEFIL command has been attempted for a volume which has no system currently installed.

6592 TEST SYSDEFIL SAME AS INSTALLED SYSDEFIL

1) A TEST-SYSDEFIL command attempted to establish the installed System Definition File as a test System Definition File.

6593 SYSTEM IMAGE FILE TOO FRAGMENTED

1) The system image in the System Image File was spread over too many discontiguous extents.
2) Copy the System Image File to a new file. Be sure to CREATE the new file with an initial allocation equal to the size of the current file. Use the LIST command with DETAIL = YES to obtain the current allocation.

6594 INVALID BLOCK SIZE

1) Either the System Image File or the JDL Image File has a logical block size which is not a multiple of the disk sector size.
2) Copy the file to a new file with a satisfactory block size. The STATUS JDL command may be used to obtain the disk sector size.

KEY ERROR CODES

6601 INCORRECT PARAMETER COUNT

1) The same number of parameters must be entered to each of the key description prompts.

6602 INCORRECT COLUMN ORDER

1) The alternate record keys must be described in ascending order of starting column.
2) No two keys may have the same starting column.

6603 INCORRECT KEY LENGTH

1) A key length is zero or greater than 255 characters.
2) The sum of a key length and its starting column exceeds the logical record length.

6604 DUPLICATES ILLEGAL IN PRIME RECORD KEY

1) DUPLICATES ALLOWED=YES was illegally specified for the prime record key.
2) The prime record key values must be unique among records of the file.

KPRINTER ERROR CODES

6701 DEVICE NOT PRINTER

1) Device name value entered for DEVICE parameter is not a printer name.
2) Execute STATUS to view current printer device names and reenter value.

6702 PRINTER IDLE

1) Command ignored because no user has the printer assigned.

LOAD ERROR CODES

6901 VOLUME NAME IN USE

1) Two volumes with the same name cannot be LOADed at the same time.
2) The RENAME command may be used to modify the volume name.

6902 VOLUME LOADED

1) Device entered for DEVICE parameter currently has a volume loaded.
2) Unload current volume or verify correct device name.

6903 DEVICE NOT DISK

1) Value entered for DEVICE parameter was not a valid disk device name.
2) Reenter with valid disk device name.

6904 VOLUME LABEL DOES NOT MATCH USER SUPPLIED LABEL

1) Value entered for VOLUME parameter conflicts with label value on volume.
2) The name after this error is the actual volume name.

6905 CONFIGURED BIT MAP TOO SMALL FOR THIS DISK

1) Total ADUs on the disk exceed the preallocated capacity of the system bit map buffer.

6906 DEVICE DEDICATED

1) An attempt was made to load a volume on a drive that was dedicated to an ongoing process.
2) A disk device may be dedicated by the I-BOOT, INITIALIZE, INSTALL-SYSTEM, and VCOPY JDL commands.

6907 VOLUME NOT RM/COS FORMAT

1) No valid RM/COS disk label was found on the disk.

6908 VOLUME NOT USABLE ON THIS SYSTEM

1) The disk volume was INITIALIZEd on a different computer and can not be used on this computer.
2) Usually due to an incompatibility between the disk controller used on the computer where the disk was INITIALIZEd and the disk controller on the computer where the LOAD failed.

LOOP and REPEAT ERROR CODES

7002 REPEAT NOT PRECEDED BY LOOP

1) REPEAT in batch stream without a previous LOOP. These two commands operate in tandem. The LOOP acts as a label for the REPEAT command.
2) Check batch stream file and edit to include a LOOP in the appropriate place.

7003 LOOP ILLEGAL IF BATCH INPUT FILE NOT DISK FILE

1) A LOOP command was entered from a batch input file that was not a disk file.
2) The LOOP command is allowed only from disk files, because only disk files can be repositioned to the record following the LOOP command.

MESSAGE ERROR CODES

7101 ILLEGAL PARTITION

1) Value entered for STATION parameter references an unrecognized partition or a nonterminal partition.
2) Zero entered for STATION parameter and terminal partition has been UNCOUPLEd from the executing partition.
3) Confirm intended partition and reenter.

7103 OPERATOR REQUESTED TERMINATION

1) A broadcast message to all terminals was aborted before all terminals received the message, either by the user entering interrupt mode and executing an EXIT command when executing in a terminal partition, or by a user entering a KTASK command when executing in a nonterminal partition.

7104 SYNTAX ERROR

1) Reply not allowed on broadcast message.

7105 MESSAGE TEXT TOO LONG

1) Text string value entered for TEXT parameter exceeds 55 character limit or 52 character limit when reply was requested.
2) Edit text string value and reenter.

FDUMP and FMODIFY ERROR CODES

7201 CHECKSUM INCORRECT

1) The value entered for the CHECKSUM parameter does not match the checksum calculated for the patch data by FMODIFY.
2) The checksum is the 16 bit arithmetic sum of all the specified patch values plus one for each relocatable patch value.
3) Verify that the correct patch data was entered and then recalculate the specified checksum and reenter the command.

7202 BYTE DISPLACEMENT NOT EVEN

1) The byte offset value entered for the OFFSET parameter is not an even number and therefore does not reference a byte beginning on a word boundary within the specified block, member, or FDE.
2) Add or subtract one from the byte value entered and reenter the command.

7203 RELOCATION INVALID

1) An apostrophe (') was entered to indicate relocation following one or more of the VERIFY or PATCH parameters entered but the member which is being patched is not relocatable.
2) The resident system is absolute, thus no relocation map exists to check or modify relocatable values. Only members in the JDL Image File, or a copy of it, are relocatable.
3) Reference a member in the JDL Image File or remove relocation references from the VERIFY and PATCH parameters and reenter the command.

7206 INVALID BLOCK NUMBER

1) The logical block number entered for the MEMBER-BLOCK parameter is too large. The referenced block is not within the file.

7208 INVALID MEMBER NAME

1) The member name specified for the MEMBER-BLOCK parameter was invalid.
2) The member name must be one to eight alphanumeric characters possibly followed by a slash (/) or period (.) and a modifier value. The modifier value is specified as a hexadecimal integer with or without a leading greater than symbol (>).

7209 OFFSET OUT OF RANGE

1) The byte offset value entered for the OFFSET parameter is greater than the length of the logical block, member, or FDE.
2) Check the value entered for the OFFSET parameter and reenter the command with a valid value.

7211 VERIFICATION DATA DOES NOT MATCH CURRENT DATA

1) Neither the values and relocation states entered for the VERIFY parameters nor the PATCH parameters in an FMODIFY command are the same as the current contents and relocation states of the member entered for the MEMBER-BLOCK parameter at the byte displacement entered for the OFFSET parameter.
2) Check the values entered for all parameters and reenter the command. An FDUMP command may be used to review the current contents.

7212 OBJECT FILE HEADER INVALID

1) The header table in the specified member was found to be invalid.
2) The file is corrupted.

7213 OBJECT FILE RELOCATION TABLE INVALID

1) The relocation table in the specified member was found to be invalid.
2) The file is corrupted.

7214 OFFSET GREATER THAN 131068

1) The byte offset value entered for the OFFSET parameter is greater than 131,068.

7215 INVALID ATTEMPT TO CHANGE RELOCATION

1) The relocation of each word in the patch data, indicated by the presence or absence of an apostrophe ('), did not match the relocation of the data in the file.
2) The relocation table cannot be changed.

PARTITION ERROR CODES

7301 SPECIFIED PARTITION IS NOT IDLE

1) A SIZE parameter was specified, and the value entered for the PARTITION parameter identifies an active partition.
2) Note that the size of the shared file partition may not be changed if any files are assigned with an access other than exclusive all.

7302 PARTITION NUMBER ILLEGAL

1) Value entered for PARTITION parameter identifies a nonexistent partition.
2) Execute STATUS command to confirm current partition numbers.

7303 A FOLLOWING PARTITION IS NOT IDLE

1) The partition identified by the PARTITION parameter value is adjacent to one or more active partitions. Note the following:
   a) for an intended increase in size all higher numbered adjacent partitions required to perform the expansion must be idle
   b) for an intended decrease in size only the immediately adjacent higher numbered partition must be idle.

7304 REQUESTED SIZE ILLEGAL

1) Value entered for SIZE parameter is not a valid signed or unsigned decimal or hexadecimal integer.
2) Value entered for a size reduction exceeds the current size of the specified partition.
3) The size requested for the specified partition is less than 12 bytes.
4) Insufficient free memory is available to increase the size of the specified partition to the requested size.
5) Requested size reduction leaves insufficient memory to quit, or attempts to reduce the size to less than that required to log-in.
6) Because of memory fragmentation, a request to change the size of the current user's partition was rejected because there was not sufficient memory adjacent to the memory cell containing the PARTITION command to fulfill the request.
7) Because of memory fragmentation, a request to change the size of the current user's partition was rejected because the PARTITION command was not loaded at the top of the user's memory. Quit and log-in again to eliminate memory fragmentation.

7305 SPECIFIED PARTITION NOT IN SYSTEM DEFINITION FILE

1) Value entered for PARTITION parameter identified a partition whose size was zero in the System Definition File.
2) If a partition is given a size of zero in the System Definition File, it is treated as nonexistent and may not be given any memory.

7306 DECREASE IN SIZE ILLEGAL IN INTERRUPT MODE

1) Initiating partition is in interrupt mode. A partition size may not be decreased in interrupted mode.
2) EXIT to return to noninterrupt mode and reenter command; or CONTINUE to allow interrupted process to complete and then reenter command.

PRINT ERROR CODES

7402 INVALID FILE ORGANIZATION

1) Pathname referenced by value entered for NAME parameter is not a directory or a sequential, relative, or indexed organization file.

RECLOSE ERROR CODES

7602 FILE NOT OPEN

1) The pathname entered identifies a specific file which does not need to be RECLOSEd.

RELEASE ERROR CODES

7701 BAD LOGICAL NAME

1) Value entered for LOGICAL NAME parameter unrecognized as a logical name currently assigned.
2) Confirm current logical name assignments via STATUS command.

RENAME ERROR CODES

7801 BAD PATHNAME

1) Pathname value entered for OLD or NEW parameters incorrect for one of the following reasons: (a) old pathname lacks intermediate directory level (b) old pathname file name miscited (c) new pathname cites directories not previously CREATEd (d) new pathname lacks intermediate directory level.
2) Confirm pathname references via LIST.

7802 VOLUME MISMATCH

1) Pathname values entered for OLD and NEW parameters specify different disk volumes.

7803 FILE ALREADY EXISTS

1) Pathname value entered for NEW parameter matches an existing pathname.
2) Reenter command with different pathname value for NEW or delete the existing file having the same pathname.

7804 PATHNAMES SAME

1) Pathname values entered for NEW and OLD parameters match on every level.
2) Confirm pathnames via LIST and reenter command.

7805 ATTEMPT TO RENAME A SYSTEM FILE

1) The referenced system file cannot be renamed with the RENAME command.

7806 OLD AND NEW PARAMETERS MIX VOLUME AND FILE NAMES

1) Both the OLD and NEW parameters must specify the same type: volume or file names.

7807 NEW VOLUME NAME IS INVALID

1) Volume names that are four characters in length cannot have a valid device prefix as the first two characters and two digits as the last two characters.
2) Volume names that are two characters in length cannot have the value ME.
3) Reenter command wih valid volume name.

7808 DEVICE IS DEDICATED

1) An attempt was made to rename a volume on a drive that is dedicated to an ongoing process.
2) A disk device may be dedicated by the I-BOOT, INITIALIZE, INSTALL-SYSTEM, RECLOSE, and VCOPY JDL commands, and by assigning an MVD organization file.

7810 NEW VOLUME NAME OF A LOADED VOLUME IS ALREADY LOADED

1) An attempt was made to change the name of a loaded volume to that of a currently loaded volume.
2) Each LOADed volume must have a unique volume name. UNLOAD the conflicting volume if it is no longer needed by the system or select another volume name for the volume to be RENAMEd.

7811 DISK NOT INITIALIZED OR FORMAT UNKNOWN

1) Specified device contains a disk that is not RM/COS format.
2) Verify correct volume or perform INITIALIZE command to establish an RM/COS format disk.

7812 DISK DOES NOT MATCH LOADED VOLUME

1) The volume name specified in the OLD parameter does not correspond to the volume name of the disk currently mounted in that drive.
2) Verify volume in specified drive.
3) UNLOAD previously loaded volume and reenter.

7813 ATTEMPT TO RENAME AN MVD FILE

1) The referenced MVD organization file cannot be renamed with the RENAME command.

REPLACE ERROR CODES

7821 LOGICAL NAME NOT FOUND

1) Value entered for the SCRATCH NAME or LOGICAL NAME parameter does not specify a currently assigned file.

7822 DEVICE NOT DISK

1) Value entered for the SCRATCH NAME or LOGICAL NAME parameter is not assigned to a disk file.

7823 FILE NOT ASSIGNED EXCLUSIVE ALL

1) File specified by the SCRATCH NAME or LOGICAL NAME parameter is not assigned with Exclusive All access.

7824 DESTINATION FILE IS SCRATCH

1) File specified by the LOGICAL NAME parameter is a scratch file.

7825 SOURCE FILE IS NOT SCRATCH

1) File specified by the SCRATCH NAME parameter must be a scratch file.

7826 SOURCE FILE IS UNDEFINED

1) File specified by the SCRATCH NAME parameter may not be an undefined file.

7827 DESTINATION FILE IS WRITE PROTECTED

1) File specified by the LOGICAL NAME parameter is write protected.

7828 VOLUME MISMATCH

1) Files specified by the LOGICAL NAME and SCRATCH NAME parameters must reside on the same disk volume.

7829 SYSTEM FILE MISMATCH

1) The file specified by the LOGICAL NAME parameter is a system file, and the file specified by the SCRATCH NAME parameter does not have the same organization, record size, and block size.

SETCOND ERROR CODES

7901 ILLEGAL PARTITION

1) Value entered for PARTITION parameter references a nonexistent partition or another terminal partition
2) Confirm partition assignments via STATUS or recall that terminal partitions are assigned values that coincide with station terminal numbers.
3) Can only set condition code in another terminal partition as a response to a message sent by that partition.

7902 PARTITION NOT ACTIVE

1) Value entered for PARTITION parameter references an idle partition.
2) Condition code setting valid only for a currently active partition.
3) Confirm partition status via STATUS command.

SHOW ERROR CODES

8001 INVALID FILE ORGANIZATION

1) File pathname entered for NAME parameter specifies a file not organized as a sequential, relative, MVD, or indexed file (e.g. a program or directory file).

SYSTEM-SHUTDOWN error codes

8081 SYSTEM-SHUTDOWN ALREADY IN PROGRESS

1) A SYSTEM-SHUTDOWN command is already in progress in another partition.

8082 SYSTEM DISK IS RESERVED

1) A process is running which has reserved the system disk drive.
2) Wait until the process reserving the system disk completes.

STATUS ERROR CODES

8101 PARAMETERS AMBIGUOUS

1) Values entered for parameters incompatible or ambiguous.
2) Note the following parameter combinations:
   a) if system status intended enter no parameter values
   b) if partition status intended enter value for partition parameter only
   c) if device status intended enter value for device parameter only
   d) if logical name status intended enter values for both partition parameter and logical name parameter.

8102 ILLEGAL PARTITION

1) Value entered for PARTITION parameter references a nonexistent partition.
2) Confirm current partition configuration via STATUS command and recall that terminal partitions are assigned values equal to station terminal numbers.

8103 NAME DOES NOT EXIST

1) Value entered for DEVICE parameter is not a valid device name or device prefix for the current system configuration.
2) Value entered for LOGICAL NAME parameter is not a logical name of the specified partition.

8104 SYSTEM DESCRIPTION ERROR

1) The system description overlay for the current hardware was missing in the JDL Image File or contained the wrong hardware type.
2) System disk was improperly constructed for hardware on which it is being used. Contact system personnel with documentation of problem and information on hardware being used.

SWITCH ERROR CODES

8201 TOO MANY SWITCHES SPECIFIED

1) Number of values entered for SWITCH parameter exceeds eight.
2) Only eight switches are implemented.

8202 ILLEGAL PARTITION

1) Value entered for PARTITION parameter references undefined partition, unused partition, another terminal partition, or shared file partition.
2) Confirm currently valid partitions via STATUS command.

8203 INVALID SWITCH NUMBER

1) One or more of the values entered for SWITCH parameter is greater than eight.
2) Only eight switches are implemented and each is referenced by an integer corresponding to its ordinal rank.
3) Confirm switch identity via STATUS command.

8204 PARTITION NOT ACTIVE

1) Value entered for PARTITION parameter references an idle partition.
2) SWITCH setting valid only for currently active partition.
3) Confirm partition states via STATUS command.

TIME ERROR CODES

8301 INVALID PARAMETER VALUE

1) Value entered for one or more of the parameters of the TIME command out of range boundaries.
2) The range boundaries are: YEAR: four decimal digits in the range 1982 through 1999 or two decimal digits in the range 82 through 99; MONTH: 1 through 12; DAY: 1 through 31; HOUR: 0 through 23; SECOND 0 through 59.
3) Reenter command with correct values.

UNCOUPLE ERROR CODES

8320 INVALID PARTITION NUMBER

1) Value entered for the PARTITION parameter identifies a nonexistent partition.
2) The partition specified was not started by the terminal partition processing the UNCOUPLE command.
3) The partition specified to an UNCOUPLE command in a nonterminal partition does not identify the terminal partition which started the nonterminal partition.

UNLOAD ERROR CODES

8401 VOLUME NOT LOADED

1) Value entered for VOLUME parameter references a volume not currently loaded.

8402 DEVICE NOT DISK

1) Value entered for VOLUME parameter is not a disk device.
2) Both LOAD and UNLOAD operate only an disk volumes.

8403 FILES ASSIGNED

1) File(s) residing on referenced disk volume have logical name assignments in effect.
2) Disk volume cannot be UNLOADed while files are assigned indicating a possibility of current or future references.
3) Confirm file assignments via STATUS command and RELEASE all logical names assigned to files resident on the disk volume or delay UNLOADing.

8404 SYSTEM DISK

1) Value entered for VOLUME parameter references system disk.
2) System disk cannot be LOADed or UNLOADed via a JDL command.
3) System disk is loaded via IPL (Initial Program Load) and can only be unloaded by the SYSTEM-SHUTDOWN JDL command.

8405 DEVICE DEDICATED

1) An attempt was made to unload a volume on a drive that was dedicated to an ongoing process.
2) A disk device may be dedicated by the I-BOOT, INITIALIZE, INSTALL-SYSTEM, RECLOSE, and VCOPY JDL commands, and by assigning an MVD organization file.

VARY ERROR CODES

8501 DEVICE ALREADY DISABLED

1) Currently disabled device referenced by DEVICE parameter and OFF value entered for STATUS parameter.
2) Confirm intended device reference, correct and reenter.

8502 DEVICE ALREADY ENABLED

1) Currently enabled device referenced by DEVICE parameter and ON value entered for STATUS parameter.
2) Confirm intended device reference, correct and reenter.

8503 DEVICE ACTIVE

1) Device referenced by value entered for DEVICE parameter is currently active.
2) Device cannot be set to OFF status until device is inactive.

8504 DEVICE ASSIGNED

1) Device referenced by value entered for DEVICE parameter is currently assigned or contains files assigned to logical names and OFF value entered for STATUS parameter.
2) Confirm current assignments via STATUS command and RELEASE assignments or delay disabling device.

8505 DEVICE IS SYSTEM DISK

1) Device referenced by the DEVICE parameter is the device name associated with the system disk.
2) System disk drive cannot be disabled.

8506 DEVICE HAS ACTIVE USER

1) Value entered for DEVICE parameter identifies a station device that is logged-in.

MAP-KEYS ERROR CODES

8602 LOGICAL NAME NOT INDEXED

1) The logical file name entered is not of indexed organization.
2) Check logical file name or reenter with no logical name to map all indexed files.

SORT-MERGE ERROR CODES

8700 KEY ERROR

1) The sort-merge initialization function detected either:
   a) An invalid key data type.
   b) A key which is not contained entirely within the sort-merge file's record area.
2) Verify that the source program compiled without errors.
3) Contact system support personnel with complete documentation.

8701 INSUFFICIENT MEMORY

1) Insufficient memory available for sort function.
2) Increase available memory by:
   a) Increasing the partition size.
   b) Reducing the number of files assigned.
   c) Reducing the number of files open.
   d) CANCELing subprograms no longer required in memory.
   e) Segmenting the COBOL program.
   f) Reducing the block size of the files required to be open.

8702 INSUFFICIENT SORT WORK FILES

1) The sort initialization function could not find at least three sequential files with logical names SW$n, where n is a single digit.
2) Ensure that prior to initiating the sort function, three or more sequential files with logical names SW$n, where n is a single digit, are assigned for use as intermediate sort files. Refer to Appendix E, Vendor Supplied Batch Streams, for a discussion of the batch stream provided for this purpose.

8703 RELEASE RECORD MISMATCH

1) The record released from a sort input procedure is not a record of the current sort file.
2) Correct the source program so that the input procedure releases a record of the sort file named in the SORT statement which references the input procedure.

8704 RECORD TOO SHORT

1) A record obtained from a sort or merge USING input file is too short to contain all the sort-merge key items.
2) Ensure that the input file was correctly contructed.
3) Ensure that the source program properly describes the records and the key data items to be used in ordering the records.

8705 RUN NUMBER OVERFLOW

1) A sort process attempted to generate too many runs or sequences of records in sorted order.
2) Increase the amount of memory available to the sort process.
3) Split the data to be sorted into several files, sort each file, and then merge the resulting sorted files.

8710 INCORRECT PARAMETER COUNT

1) The parameters to the JDL SORT command do not properly describe all the key data items.
2) Reenter the command, ensuring that the parameter lists for KEY START, KEY LENGTH, ASCENDING, and KEY TYPE each contain the same number of entries.

8711 INCORRECT KEY START

1) A response to the KEY START prompt specifies a starting value larger than the sort record size.
2) Correct the response to the KEY START prompt such that all keys are completely contained within the sort record and reenter.

8712 INCORRECT KEY LENGTH

1) A response to the KEY LENGTH prompt specifies a key size value such that the end of the key is beyond the end of the sort record.
2) Correct the response to the KEY LENGTH prompt such that all keys are completely contained within the sort record and reenter.

8713 INCORRECT KEY TYPE

1) A response to the KEY TYPE prompt specifies an unrecognized key type abbreviation.
2) Correct the response to the KEY TYPE prompt such that all keys are described with a valid key type abbreviation. See the description of the SORT JDL command description for a list of valid key type abbreviations.

8714 INCORRECT MERGE INPUT FILE COUNT

1) Only one input file was specified for a merge only function.
2) Specify two or more input files for a merge only function.

8715 RECORD SIZE REQUIRED

1) No value was given to the RECORD SIZE parameter of the SORT command, and the maximum record size of the files being sorted could not be computed.
2) The SORT command computes the sort record size from any disk input files specified, and from opening the first input file specified. If no file being sorted is a disk file, and the first file being sorted does not have a record size, then a value must be specified for the RECORD SIZE parameter.

SYNONYM ERROR CODES

8801 SYNONYM TABLE FULL

1) Insufficient space exists in the Synonym Table for the specified synonym and value.
2) Delete some unused synonyms and reenter the command.
3) Use the Modify User Identification procedure to increase the Synonym Table size (see Chapter 2, System Configuration); QUIT and log-in to obtain the increased size.

8802 SYNONYM VALUE TOO LONG

1) The synonym value specified is greater than 68 characters.

8803 SYNONYM NOT FOUND

1) The synonym requested to be deleted does not exist.
2) Use MAP-SYNONYMS to verify synonym's existence.

8804 NO SYNONYM TABLE

1) No user synonyms are allowed with this user ID.
2) See Chapter 2, System Configuration, for information on enabling synonyms for the user ID.

CONNECT ERROR CODES

8903 INVALID TYPE OPTION

1) A value other than USER was entered for the TYPE parameter when CONNECTing to a User Link device.
2) A value of USER was entered for the TYPE parameter when CONNECTing to a Bisync device.
3) Both 3780 and 2780 were entered for the TYPE parameter.
4) Both ASCII and EBCDIC were entered for the TYPE parameter.
5) LRC, ODD, or EVEN were entered for the TYPE parameter when CONNECTing to an EBCDIC Bisync device.
6) Both CRC and LRC were entered for the TYPE parameter.
7) More than one of the values ODD, EVEN, or NONE were entered for the TYPE parameter.

RECEIVE and FTS ERROR CODES

8911 UNDEFINED PATHNAME

1) The RECEIVE command omitted the LOCAL NAME parameter value or an FTS command was used, and the remote end of the communication link did not specify a pathname for the REMOTE NAME parameter of a SEND command.
2) Either a RECEIVE command must be used which specifies a pathname for the LOCAL NAME parameter, or the corresponding remote SEND command must specify a pathname for the REMOTE NAME parameter.

8912 INVALID PROGRAM FILE FORMAT

1) The record length received when receiving a program file was not 126.
2) All but the first 4 characters received as the first record of a program file must be, but were not, binary zero.

8913 INVALID MDS FILE FORMAT

1) The file being received when receiving into an MDS file is not logically structured as an MDS file.
2) A file which is received into an MDS file must be an MDS file or a sequential copy of an MDS file.

8914 CHARACTERS NOT TRANSLATED

1) EBCDIC characters with no ASCII equivalent were encountered when receiving a coded file. The characters were converted to ASCII substitute characters (>1A). The number of such characters is indicated in the message.
2) This error can occur when a Horizontal Tab (HT) character is received without receiving a horizontal tab setting ("Electronic tab stop") record.

SEND and FTS ERROR CODES

8921 INVALID MODE OPTION

1) CODED mode was specified for SENDing a program file or an MDS file.
2) Use IMAGE mode.

8922 UNDEFINED PATHNAME

1) The SEND command omitted the LOCAL NAME parameter value, and the remote end of the communication link did not specify a pathname for the REMOTE NAME parameter of a RECEIVE command or entered a SEND command.
2) The FTS command was used and the remote end of the communication link did not specify a pathname for the REMOTE NAME parameter of a SEND command.
3) The SEND command omitted the LOCAL NAME parameter value, or an FTS command was used, and the remote end of the communication link specified a transmission mode that conflicted with that specified on the SEND or FTS command or conflicted with the IMAGE mode implicit in sending an MDS or program organization file.

8923 TOO MANY FILES SPECIFIED

1) More than nine files were specified for the LOCAL NAME parameter of a SEND command or the REMOTE NAME parameter of a RECEIVE command.

8924 CHARACTERS NOT TRANSLATED

1) ASCII characters with no EBCDIC equivalent were encountered when sending a coded file. The characters were converted to EBCDIC substitute characters (>3F). The number of such characters is indicated in the message.
2) This error can result from improperly sending a file containing binary data in CODED mode. If the destination system is an ASCII system (e.g., another RM/COS installation), use image mode instead of CODED mode when the contents of the file might include binary data.

3780 I/O ERROR CODES

8931 UNEXPECTED DATA IN PRINTER SEQUENCE

1) Invalid construction of a 2780/3780 vertical forms positioning sequence.
2) See appropriate IBM reference document.

8933 ERROR IN TAB STOP IMAGE

1) "Electronic tab stop" image too large.
2) Consult appropriate IBM document.
3) Revise remote program or system accordingly.

8937 CONNECT TIMEOUT

1) Link failed to go "off hook" (i.e., connect) within the timeout interval specified on the CONNECT command.
2) Check the cabling at both the local and remote stations. This error will result if Data Set Ready or Data Terminal Ready signals are not wired correctly.

8938 CODED RECORD TOO LONG

1) In CODED mode, records longer than a transmission block may not be sent. The maximum transmission block size of a 3780 link is 512 characters, and the maximum transmission block size of a 2780 link is 400 characters.
2) If the destination system is another RM/COS installation, use IMAGE mode instead of CODED mode.

IMPLEMENTOR DEFINED COBOL I/O ERROR CODES

9000 ILLEGAL I/O FUNCTION

1) Unsuccessful I/O attempt caused by one of the following:
   a) Operation attempted is not consistent with the open mode specified when the file was opened.
   b) Omission of a successful read prior to a requested sequential delete or rewrite operation.
   c) Attempt to write or rewrite a write protected file opened for I-O. An OPEN I-O of a write protected file is treated as an OPEN INPUT.
   d) Attempt to rewrite a compressed sequential file.
2) Revise COBOL source accordingly, use CHANGE to remove write protection from the file, or delete and recreate the file without the compressed attribute.
3) Attempt to access a multivolume disk file (MVD organization) without the I/O support routine being loaded.

9100 FILE NOT OPENED

1) Omission of file OPENing prior to I/O request referencing the file.
2) Revise COBOL source accordingly.

9201 FILE OPEN, DUPLICATE LOGICAL NAME

1) An attempt to open a file has failed because the logical name (COBOL external file name) is already open.
2) Revise the COBOL source program so that each ASSIGN clause specifies a unique logical name.

9202 FILE OPEN, COBOL FILE NAME NOT CLOSED

1) A COBOL program execution of an OPEN statement has failed because the same COBOL file name was previously opened with the same program and has not been successfully closed by the execution of a CLOSE statement without the REEL, UNIT or LOCK phrase.
2) Revise the COBOL source program to either omit the OPEN or provide a CLOSE prior to the second OPEN.

9301 FILE NOT AVAILABLE, LOGICAL NAME NOT FOUND

1) Open attempted on a logical name which is not associated with any file.

9303 FILE NOT AVAILABLE, FD LOCKED

1) The File Descriptor for this file was previously closed with lock, preventing any further access by this program.

9304 FILE NOT AVAILABLE, ACCESS CONFLICT

1) A logical name assigned to a disk file could not be assigned because the requested open mode conflicts with the mode in which the file is already opened:
   a) Open I-O or EXTEND is illegal if the disk file is already open OUTPUT.
   b) Open OUTPUT is illegal if the disk file is already open.
2) A file assigned to a printer or tape drive could not be opened because the device is already open under a different logical name.

9401 INVALID OPEN, INSUFFICIENT MEMORY

1) Insufficient memory for file buffers. Increase memory available by:
   a) Reduce number of buffers requested for this or other open files.
   b) Reduce the block size of this or other open files.
   c) Reduce the number of files open.
   d) Reduce the number of logical names assigned.
   e) Increase segmentation in the program.
   f) Increase the partition size.
2) For an indexed file, insufficient memory for buffers to contain longest key and a copy of the longest record area.

9402 INVALID OPEN, EXHAUSTED EXTENTS

1) File referenced has reached the maximum permissible number of discontiguous allocations (47).
2) Recovery requires an FCOPY of the disk file which in the process of copying will reorganize disk space allocation.

9403 INVALID OPEN, EXHAUSTED ADUS

1) File referenced cannot continue expansion due to lack of ADUs on the disk.
2) RECLOSE disk to recover possible unused scratch space.
3) Deallocate ADUs via DELETE of no longer needed files on the disk.
4) FCOPY file to another disk.

9404 INVALID OPEN, FILE NOT EXPANDABLE

1) Consumed primary file allocation and file as currently defined disallows secondary allocations.
2) Redefine file via CHANGE permitting secondary allocations.

9405 INVALID OPEN, EXHAUSTED BLOCKS

1) Attempt to write more than 65535 blocks to the file.
2) Recreate file with a larger block size.

9406 INVALID OPEN, DISK TOO FRAGMENTED

1) Unable to find sufficient contiguous disk for one block.
2) Recreate file with a smaller block size.
3) FCOPY volume to another volume thereby reorganizing the volume's file allocations.

9407 INVALID OPEN, ACCESS PAST LAST EXTENT

1) This error is caused by an attempt within the operating system to access past the last ADU. It indicates a system error, and the vendor should be contacted.

9410 INVALID OPEN, DISK DEDICATED

1) An attempt was made to open logical name LO or a batch listing file on a volume in a drive that was dedicated to an ongoing process.
2) A disk device may be dedicated by the I-BOOT, INITIALIZE, INSTALL-SYSTEM, RECLOSE, and VCOPY JDL commands, and by assigning an MVD organization file.

9411 INVALID OPEN, RECORD SIZE MISMATCH

1) Open failed because the length of the record area did not match the record size of the already existing file.
2) If the open which fails is an OPEN I-O, assign the logical name to a freshly created file and execute the program. An OPEN I-O of an undefined file defines the file characteristics identically to an OPEN OUTPUT. After program execution, use a LIST command to confirm the record size the program actually expects.

9412 INVALID OPEN, BLOCK SIZE MISMATCH

1) Open failed because the size specified in the BLOCK CONTAINS clause did not match the block size of the already existing file.
2) If the open which fails is an OPEN I-O, assign the logical name to a freshly created file and execute the program. An OPEN I-O of an undefined file defines the file characteristics identically to an OPEN OUTPUT. After program execution, use a LIST command to confirm the block size the program actually expects.

9413 INVALID OPEN, ORGANIZATION MISMATCH

1) Open failed because the organization specified in the SELECT clause did not match that of the file.
2) Files assigned to devices must be described as ORGANIZATION SEQUENTIAL.

9414 INVALID OPEN, NUMBER OF KEYS MISMATCH

1) Open of an indexed file failed because the number of alternate record keys specified did not match those of the already existing file.
2) If the open which fails is an OPEN I-O, assign the logical name to a freshly created file and execute the program. (Do not use a KEY command to define the keys.) An OPEN I-O of an undefined file defines the file characteristics identically to an OPEN OUTPUT. After program execution, use a MAP-KEYS command to confirm the key characteristics the program actually expects.

9415 INVALID OPEN, FILE UNDEFINED

1) Open INPUT failed because the file has never been opened OUTPUT.

9416 INVALID OPEN, BLOCK SIZE INVALID

1) Open failed because the block size computed from the BLOCK CONTAINS n RECORDS clause exceeded 65535 bytes.
2) Open of a sequential organization disk file failed because the block size was less than 6 characters.
3) Open of a relative or program organization disk file failed because the block size was less than 4 characters.
4) Open of an indexed organization disk file failed because the block size was less than 14 characters.
5) Open of an MVD organization disk file failed because the block size was less than 8 characters.
6) Open of an indexed file failed because the block size was too small to contain the Key Information Block. The block size must not be less than 10 * (number-of-alternate-keys + 2).
7) Open of an indexed file failed because the block size was too small for an index block. The block size must not be less than 6 + 3 * (length of longest key + 4).
8) Open of a tape file failed because the block size specified is not acceptable to the tape controller.

9) Open of a tape file failed because the block size was less than the block offset.
10) Open of a tape file without spanned records failed because the block size was less than the sum of the record size, the record and block overheads, and the block offset.
11) Open of a tape file with record format S failed because the block size and the record size exceeded 9999 characters.

9417 INVALID OPEN, NO CHARACTERISTICS SPECIFIED

1) Open of a tape file failed because neither a record size nor a block size was specified.
2) Open OUTPUT attempted with no File Descriptor. This is an internal system error and the vendor should be contacted.

9418 INVALID OPEN, OLD FILE FORMAT

1) Because of new features, the type of open attempted is not supported for older file formats. Further information should be available from release documentation.
2) Usually an old file may be opened for input, thus allowing FCOPY or other utility to copy the file.
3) An open output normally will update the file format to support all new features.

9419 INVALID OPEN, NEW FILE FORMAT

1) An open was attempted on a file created by a later release of RM/COS. Because of changes in the file structure, an open of the file is not allowed.
2) An open output normally will update the file format to match the current release.

9420-942E INVALID OPEN, KEY STARTING POSITION MISMATCH

1) Open of an indexed file failed because the starting position specified for the prime record key or for an alternate record key did not match that of the already existing file.
2) The last digit of the error code is the hexadecimal number of the record key in error, where 0 is the prime record key, and 1 through E correspond to the first through fourteenth alternate record keys.
3) If the open which fails is an OPEN I-O, assign the logical name to a freshly created file and execute the program. (Do not use a KEY command to define the keys.) An OPEN I-O of an undefined file defines the file characteristics identically to an OPEN OUTPUT. After program execution, use a MAP-KEYS command to confirm the key characteristics the program actually expects.

9430-943E INVALID OPEN, KEY LENGTH MISMATCH

1) Open of an indexed file failed because the length specified for the prime record key or for an alternate record key did not match that of the already existing file.
2) The last digit of the error code is the hexadecimal number of the record key in error, where 0 is the prime record key, and 1 through E correspond to the first through fourteenth alternate record keys.
3) If the open which fails is an OPEN I-O, assign the logical name to a freshly created file and execute the program. (Do not use a KEY command to define the keys.) An OPEN I-O of an undefined file defines the file characteristics identically to an OPEN OUTPUT. After program execution, use a MAP-KEYS command to confirm the key characteristics the program actually expects.

9441-944E INVALID OPEN, KEY DUPLICATES MISMATCH

1) Open of an indexed file failed because DUPLICATES ALLOWED was specified for an alternate record key and duplicates are not allowed in the already existing file, or DUPLICATES ALLOWED was not specified and duplicates are allowed in the already existing file.
2) The last digit of the error code is the hexadecimal number of the record key in error, where 1 through E correspond to the first through fourteenth alternate record keys.
3) If the open which fails is an OPEN I-O, assign the logical name to a freshly created file and execute the program. (Do not use a KEY command to define the keys.) An OPEN I-O of an undefined file defines the file characteristics identically to an OPEN OUTPUT. After program execution, use a MAP-KEYS command to confirm the key characteristics the program actually expects.

9450 INVALID OPEN, NO MEMBER

1) An open INPUT or I-O on an MDS file did not specify a member name of a member existing in the file; or an open EXTEND on an MDS file did not specify the member name of the member to be added to the file.

9451 INVALID OPEN, DUPLICATE MEMBER

1) An open EXTEND on an MDS file specified a member name of a member already existing in the file.

9452 INVALID OPEN, DIRECTORY FULL

1) An open EXTEND on an MDS file was attempted when the file directory was full. No new members may be added to the file.

9453 INVALID OPEN, NO REWIND NOT CORRECTLY POSITIONED

1) An open WITH NO REWIND was attempted on a tape device which was not positioned at the end of a file.

9454 INVALID OPEN, FILE POSITION NOT FOUND

1) An open OUTPUT or EXTEND of a tape file on a scratch volume was attempted with a file position other than 1.
2) An open of a tape file was attempted with a file position past the last file written on the tape.
3) An open EXTEND of a tape file failed because the file being opened extends past the last reel assigned.
4) Open was unable to position to the tape file specified because the file position specified lies past the last reel assigned.

9455 INVALID OPEN, RECORD SIZE INVALID

1) A record size of zero was requested for a disk file.
2) The record size requested for a tape file without spanned records exceeded the block size less the block offset and the block and record overheads.
3) The record size requested for a tape file with spanned records exceeded 65530 characters.
4) The record size requested for a tape file with record format D exceeded 9995 characters.

9456 INVALID OPEN, FILE COUNT NOT SPECIFIED

1) Open failed on unlabelled tape that does not support a hardware end of volume indicator. The file position specified was past the highest position file accessed since the tape was assigned, and the highest position file accessed was not on the last reel, causing the tape subsystem to be unable to determine whether a tape mark is an end of file or end of volume indication.
2) When reading an unlabelled multivolume tape, it is necessary to specify the number of files on each reel. When reading an unlabelled cartridge tape without hardware end of volume indicators, it is necessary to specify the number of files on each track. Change the VOLUME parameter of the TAPE-ASSIGN command to specify the number of files on each reel or track.

9457 INVALID OPEN, DENSITY INVALID

1) The density specified for a tape file was not acceptable to the tape controller.

9458 INVALID OPEN, EXTEND NOT SUPPORTED

1) An open EXTEND was attempted on a tape file on a streaming tape controller which does not allow a write operation after a read operation.

9500 ILLEGAL DEVICE TYPE

1) READ statement attempted on a file assigned to a printer.
2) OPEN statement attempted on a file assigned to a Bisync or User Link communications device.
3) OPEN INPUT or I-O attempted on a file assigned to a printer.
4) OPEN I-O attempted on a file assigned to a tape drive.
5) REWRITE statement attempted on a file assigned to a device.

9600 UNDEFINED RECORD POINTER

1) Sequential read statement rejected as a result of a previous unsuccessful read or start statement having left the current record pointer in an undefined state.
2) Write or CLOSE REEL statement to a tape file opened OUTPUT rejected as a result of a previous unsuccessful write statement having left the tape at an unknown position.

9701 INVALID RECORD LENGTH, RECORD AREA TOO SHORT

1) Write statement to an indexed organization file rejected because the record length did not include all the record keys.

9702 INVALID RECORD LENGTH, RECORD AREA TOO LONG

1) Write statement to a disk or tape file rejected because the record length exceeded the record size of the file.
2) Indexed file write or rewrite rejected because the actual record size > block size - 8. Refer to Appendix D, File Size Calculations, for the definition of actual record size.
3) This error can occur when the record area is described with an OCCURS ... DEPENDING ON clause, and the value of the depending on variable exceeds the maximum described for the file.

9703 INVALID RECORD LENGTH, RECORD TOO LONG

1) The length of the record read from a disk or tape file exceeded the length of the record area.
2) The record area used to read a member directory entry of an MDS organization file was less than 20 characters long.
3) The length of the record received from a Bisync link device exceeded the record length of the local file or device.

9704 INVALID RECORD LENGTH, RECORD SIZE MISMATCH

1) Rewrite statement rejected because the record length as rewritten does not match the previous record length.

9705 INVALID RECORD LENGTH, FIXED LENGTH REQUIRED

1) Write statement to a tape file with record format F rejected because the length of the record does not match the record size of the file. In a format F file, all records must be the same length.

9800 WRITE PROTECTED

1) OPEN OUTPUT or EXTEND statement rejected because the file or device was assigned with Read Only access or because a disk file was write protected.
2) The system assigns program organization disk files with Read Only access unless Exclusive All access was specified.
3) The write protect attribute of a disk file may be removed with the CHANGE command.

9900 RECORD LOCKED

1) An I/O statement to a relative or indexed organization file rejected because referenced record was locked by another user's program.
2) A statement will generate this code only if both a USE procedure and a FILE STATUS data item are in effect for the file associated with the referenced record; otherwise the operation waits until the record is available.

SYSTEM STOP CODES and MESSAGES

The following codes and messages appear on the debug terminal. If a terminal is not attached to the computer at the debug terminal location, then the system will stop for any of the following reasons without being able to indicate the specific cause. Consult your Operator Guide regarding attachment of a debug terminal.

In the case of several IPL errors, the message is displayed and then the system waits approximately ten seconds to give the user time to see it, after which IPL continues. In such cases, a default action is indicated in the following descriptions.

A002 P RECORDS REQUEST MORE MEMORY THAN IS AVAILABLE

1) Sum of memory requests in memory size parameters for all P records exceeds memory available for user partitions.
2) Adjust memory size requirements on P records in System Definition File until total memory requested is less than total available user partition memory.
3) If the error was encountered on a P record for a terminal partition, the minimum memory needed for terminal log-in is allocated to each terminal partition (i.e., for each station device defined in the System Definition File).
4) If the error was encountered after allocating memory to all terminal partitions, memory is allocated to any remaining partitions for which sufficient memory is available. Other partitions are marked "unused" and allocated no memory.

A004 INTEGER VALUE ERROR

1) At IPL time the System Definition File or the JDL Definition File was found to contain an integer parameter with nondigit characters. A hexadecimal integer without the leading hexadecimal indicator (>) will cause this error.
2) Confirm proper integer values entered in System Definition File and JDL Definition File.
3) A default value is used, if there is one.

A005 S RECORD SPECIFIES INVALID YEAR

1) The system specification record in the System Definition File contains a year field which is not blank and contains nonnumeric characters.
2) On hardware with a battery powered clock which does not supply the year, the year may be specified as a parameter on the S record in the System Definition File. Ensure that the year parameter is blank or contains a valid decimal integer (see your Operator Guide).
3) The invalid year specification is ignored.

A006 U RECORD SPECIFIES INVALID DEVICE PREFIX

1) The leading two characters of a device name on a unit specification record in the System Definition File do not match those of any device type defined in the generated system. See Chapter 2 of this manual for a list of the allowed device prefixes.
2) Check that the device name starts in column three of the U record.
3) The U record is ignored.

A007 D RECORD SPECIFIES INVALID DEVICE NAME

1) A device definition record (D record) in the System Definition File specifies an undefined device name.
2) Correct D record device name field or delete D record.
3) The D record is ignored.

A008 INSUFFICIENT MEMORY FOR JDL NAMES

1) IPL is unable to build the JDL command directory due to insufficient free memory.
2) Unless the System Definition File contains far too many device definitions, this is probably an error in the system software.
3) The system initialization process stops. This is an from which IPL is unable to recover.

A00B ERROR OPENING JDL IMAGE FILE

1) The system initialization process (IPL) was unable to open the file containing the JDL processors to build the JDL directory. Probably due to a disk read error.
2) The system initialization process stops. This is an error from which IPL is unable to recover.
3) Restore the JDL image file from a backup.

A00C NO VALID TERMINAL DEFINED

1) No terminal partition was found whose station was enabled and whose size was sufficient to permit log-in.
2) The minimum memory needed for terminal log-in is allocated to each terminal in the configuration (i.e., to each station device defined in the System Definition File).

A00D U RECORD DEFINES DUPLICATE DEVICE NAME

1) The device name specified on a unit specification record in the System Definition File has already been defined on a previous U record.
2) Correct the device names on U records such that each device is specified only once within the System Definition file.
3) The U record is ignored.

A00E U RECORD DEFINES DEVICE FOR WHICH THERE IS NO DEVICE SERVICE ROUTINE

1) A unit specification record in the System Definition File defines a device for which there is no device service routine in the generated system.
2) The device name on the U record should be changed or the U record deleted from the System Definition File.
3) The U record is ignored.

A010 U RECORD SPECIFIES INVALID STATION ID

1) A unit specification record in the System Definition file specifies an invalid station device suffix.
2) Correct the U record station device suffix. Station device suffices must correspond to the paritition number of the partition to be used by the terminal.
3) The U record is ignored.

A012 P RECORD SPECIFIES INVALID PRIORITY VALUE

1) A partition specification record in the System Definition File specifies an nonnumeric or zero priority value.
2) Priority values must be numeric with values from 1 to 65535. Correct any invalid priority values on P records.
3) IPL assumes a priority value of 1 (highest priority) for the partition.

A013 C RECORD SPECIFIES INVALID BOARD TYPE

1) A controller specification record in the System Definition File specifies a board type which is unknown to the generated system.
2) Correct the C records in the System Definition File so that only board types defined in the system are referenced. See your Operator Guide for valid board types.
3) The C record is ignored. Following U cards are also ignored.

A014 U RECORD SPECIFIES INVALID CONTROLLER

1) A unit specification record in the System Definition File specifies a controller type or controller number which is not valid at that point.
2) The controller identified by the controller type and controller number must be one of those associated with the last preceding C record. Check your Operator Guide for valid controller names and numbers.
3) The U record is ignored.

A015 U RECORD SPECIFIES INVALID UNIT NUMBER

1) A unit specification record in the System Definition File specifies a unit number which is too large for the indicated controller.
2) Correct the unit number or indicate a different controller number.
3) IPL ignores the U record.

A016 U RECORD SPECIFIES DUPLICATE UNIT NUMBER

1) Attempt to add a device to an existing controller with a unit number equal to the unit number of a device already defined for the controller.
2) IPL ignores the U record.

A017 ERROR OPENING JDL DEFINITION FILE

1) The system initialization process (IPL) was unable to open the JDL Definition File (.JDLDEFIL). This is probably due to a disk read error.
2) The system initialization process stops. This is an error from which IPL is unable to recover.
3) Restore the JDL Definition File from a backup.

A018 SYSTEM DISK NOT FOUND

1) The system disk device was not defined by any of the unit specification records in the System Definition File.
2) Add a U record which defines the system disk device.
3) IPL cannot continue after this error. Re-IPL with the system disk mounted in a disk device which is defined in the System Definition File.

A019 ERROR READING SYSTEM DEFINITION FILE

1) The system initialization process (IPL) was unable to read the System Definition File. This is probably due to a disk read error.
2) The system initialization process stops. This is an error from which IPL is unable to recover.
3) Restore the System Definition File from a backup.

A01A ERROR OPENING SYSTEM DEFINITION FILE

1) The system initialization process (IPL) was unable to open the System Definition File. This is probably due to a disk read error.
2) The system initialization process stops. This is an error from which IPL is unable to recover.
3) Restore the System Definition File from a backup.

A01B SYSTEM DEFINITION FILE IS EMPTY

1) The system initialization process (IPL) received an end of file status on the first read from the System Definition File.
2) The system initialization process stops. This is an error from which IPL is unable to recover.
3) Restore the System Definition File from a backup.

A01C ERROR OPENING SYSTEM IMAGE FILE

1) The system initialization process (IPL) was unable to open the System Image File to read terminal definitions. This is probably due to a disk read error.
2) The system initialization process stops. This is an error from which IPL is unable to recover.
3) Restore the System Image File from a backup.

A01D ERROR READING SYSTEM IMAGE FILE

1) The system initialization process (IPL) received an error reading a terminal definition from the System Image File. This is probably due to a disk read error.
2) The terminal definition file is not read. Any terminals of this terminal type are changed to type TTY.
3) Restore the System Definition File from a backup.

A01E ERROR CLOSING SYSTEM IMAGE FILE

1) The system initialization process (IPL) was unable to close the System Image File after reading terminal definitions. This is probably due to a disk read error.
2) The system initialization process stops. This is an error from which IPL is unable to recover.
3) Restore the System Image File from a backup.

A01F TERMINAL TYPE NOT RECOGNIZED

1) One of the terminal types specified on one of the station U records in the System Definition File was not found on the System Image File.
2) Any terminals of this terminal type are changed to type TTY.
3) Check all U records for stations in the System Definition File. The valid terminal types are listed in your Operator Guide.

A020 COMMAND DIRECTORY FULL

1) The JDL image file contains too many JDL processor names. The memory-resident table of names has overflowed.
2) Some JDL processors will not be accessible.

A021 S RECORD CONTAINS INVALID TYPE-AHEAD BUFFER SIZE

1) The system specification record in the System Definition File contains a type-ahead buffer size which is not blank and contains nonnumeric characters.
2) The invalid size specification is ignored.

A022 S RECORD CONTAINS INVALID EVENT LOG BUFFER SIZE

1) The system specification record in the System Definition File contains an event log buffer size which is not blank and either contains nonnumeric characters or is too small or is too large.
2) The invalid size specification is ignored. No event log buffer is allocated. Event logging is disabled.

COOO-CFFF DEVICE DEPENDENT C RECORD ERROR

1) An attempt was made to configure a controller using parameters that are unacceptable to the device service routine. See the Operator Guide for further information.

DOOO-DFFF DEVICE DEPENDENT U RECORD ERROR

1) An attempt was made to configure a device using parameters that are unacceptable to the device service routine. See the Operator Guide for further information.
2) A device service routine encountered a logic failure or unknown permanent error. Contact system support personnel with documentation of the problem.

---> System Failure <---

System crash
PC=pppppp

1) The system encountered an internal logic error. The location at which the error was detected is indicated as "PC=pppppp".
2) Contact system personnel with documentation of the problem.

APPENDIX A

NUMERIC LISTING OF RM/COS

SYSTEM STATUS CODES

COBOL INPUT/OUTPUT ERROR CODES

0000	Successful completion
0200	Successful, duplicate key exists
1000	End of file
2100	Invalid key sequence
2200	Invalid key, duplicate
2300	Invalid key, no record found
2402	Invalid key, boundary (exhausted extents)
2403	Invalid key, boundary (exhausted ADUs)
2404	Invalid key, boundary (file not expandable)
2405	Invalid key, boundary (exhausted blocks)
2406	Invalid key, boundary (disk too fragmented)
2407	Invalid key, boundary (access past last extent)
2408	Invalid key, boundary (invalid record number)
2450	Invalid key, boundary (invalid member)
2451	Invalid key, boundary (access past end of member)
30	Permanent Error Codes
3000	Bad index structure
3001	Record integrity error
3002	Volume not mounted
3010	Illegal opcode
3015	Device disabled
3017	Output killed
3018	Unable to start process
3020	Device error
3021	Device timed out
3022	Device disconnected
3023	Device parameter error
3024	Device not ready

3780/2780 Error Codes

3030	Line disconnected
3031	Disconnect received
3032	Disconnect timer elapsed
3033	Response timer elapsed
3034	Bid error, bid contention
3035	Bid error, ENQ limit
3036	Unexpected data block
3037	TTD error, ENQ limit
3038	Receiving station abort
3039	Write error, ENQ limit
3040	Write error, NAK limit
3041	Write error, wrong acknowledgement
3042	Sending station abort
3043	Start-of-heading not supported
3044	Unable to set link configuration

Disk Device Error Codes

3050	Off-line
3051	Not ready
3052	Write protected
3053	Unit check (fault)
3054	Illegal address
3055	Seek incomplete
3056	Illegal command / unknown error
3057	Unknown error
3058	Data error
3060	ID error
3061	Data rate overrun/underrun
3062	Command timeout
3063	Search error
3064	Command rejected
3065	No status

Tape Device Error Codes

3070	End of tape
3071	End of volume
3072	Block too long
3073	Wrong block read
3074	Beginning of tape
3075	Write protected
3076	Write after read not supported

3080-309F	Controller dependent errors

34	File Management Error Codes
3402	Boundary violation, exhausted extents
3403	Boundary violation, exhausted ADUs
3404	Boundary violation, file not expandable
3405	Boundary violation, exhausted blocks
3406	Boundary violation, disk too fragmented
3407	Boundary violation, access past last extent
3409	Boundary violation, exhausted volumes

JDL COMMAND ERROR CODES

40	JDL Root Error Codes
4001	Insufficient memory in partition
4003	Syntax error
4004	Parameter of type YESNO not entered correctly
4005	Command not allowed in interrupt mode
4006	Command not enabled
4008	Command exceeds user privilege level
400A	Parameter too long
400B	Unrecognizable formal parameter found in batch
400C	TIME not initialized
400D	Parameters exceeded roll memory
400E	Synonym expansion too long
4020	Invalid object file
4042	Synonym file not sequential
4043	Synonym file record length not 78
4044	Synonym table too large
4051	Synonym table full
4052	Invalid synonym record
41	Program Loader Error Codes
4101	Zero length procedure segment
4102	Zero length data segment
4103	Program active or not found
43	Runtime DEBUG Error Codes
4301	Unrecognized breakpoint
4302	Too many breakpoints
4303	Duplicate breakpoint
4304	Unrecognized command
4305	Invalid line number
4306	Invalid location
4307	Syntax error
4308	Truncation of MODIFY data
4309	Invalid data type
4310	Invalid numeric value
4311	Waiting for acknowledgement of display

44	COBOL Runtime Interpreter Error Codes
4400	Program location of previous I/O error
4401	Compilation error encountered
4402	Traceback
4403	Data reference error
4404	Illegal procedure nesting
4405	Independent segment load error
4406	ADVANCING value error
4407	Illegal instruction encountered
4408	Illegal label reference
4409	Illegal data descriptor
4410	Illegal edit picture
4411	Error ocurred while using station
4412	Illegal argument
4413	Attempt to CANCEL active program
4414	Illegal RETURN or RELEASE
4415	Unsuccessful SORT or MERGE initialization
4416	Procedure reference error
4417	ALTER statement not yet executed for paragraph
4418	Illegal exponent value
50	ASSIGN Error Codes
5001	Attempt to assign directory
5002	Logical name already exists
5003	Bad pathname
5004	Assign requires higher privilege
5005	File cannot be assigned with requested access
5006	Invalid disk or device name
5007	Volume not loaded
5008	File not reclosed
5009	Device varied off
5010	File rename in progress
5011	Different volume found
5012	Device dedicated
5014	Invalid indexed file
5015	Invalid program file
5016	Invalid file directory entry
5017	Program file is empty
5018	Wrong type of device
5019	Incorrect pathname syntax
5020	Logical IO module not found
5021	Unable to dedicate device

50	TAPE-ASSIGN Error Codes
5031	Device or set name required
5032	Set name not defined
5033	Device set conflict
5034	Invalid set name syntax
5035	Volume parameter required
5036	Invalid volume syntax
5037	Logical name already exists
5038	File count illegal
5039	Invalid file count syntax
5040	Too many volumes
5041	Volume conflict
5042	Invalid labels specification
5043	Labels conflict
5044	Invalid file name
5045	Volume parameter illegal
5046	Invalid format specification
5047	Invalid print control
5048	Invalid code set
5049	Invalid padding character
5050	Invalid retention period
5051	Density mismatch
5052	Backup option and offset parameter conflict
5053	Premount volume not found
5054	Premount conflict with loaded volume
5055	Invalid block offset
51	BATCH and CHAIN Error Codes
5101	Batch already active in user's partition
5102	Batch cannot be nested deeper than 4 levels
5103	Partition number illegal
5104	LIST=YES specified without a list name
5105	Specified partition is busy
5106	Overlay load error
5107	Insufficient memory in requested partition
5108	BATCH into another partition illegal if TIME not entered
52	CHANGE Error Codes
5202	System file may not be changed

53	COBOL Compiler Error Codes
5300	Compilation terminated because of previous error
5301	Null source program or fatal syntax error
5302	No memory available for roll memory
5303	Roll pointer overflow
5304	Roll memory overflow
5305	Source program size overflow
5306	Compiler internal logic failure
5307	COPY statement nesting limit exceeded
54	CONTINUE Error Codes
5402	Partition number illegal
5403	Partition not suspended
55	CREATE Error Codes
5501	File already exists
5502	Scratch type conflict
5503	Logical name already exists
5504	Type COMPRESSED not allowed
5505	Type SPOOL not allowed
5506	Unable to allocate in single extent
5507	Type AUTO not allowed
5508	Multivolume disk file not at top level
5509	Logical IO module not found
5510	Unable to dedicate device
55	SCRATCH Error Codes
5521	Template name does not exist
5522	Device not disk
5523	Logical name already exists
5524	Unable to allocate in single extent
5525	Multivolume disk organization illegal
56	DELETE Error Codes
5601	File not found
5603	Delete protected
5605	Directory not empty
5606	Directory type mismatch
57	DIRECTORY Error Codes
5701	Estimated number of files error
5702	Bad pathname
5703	File already exists

58	SDUMP and SMODIFY Error Codes
5801	Unable to obtain buffer memory
5802	Error opening listing file
5803	Invalid device name
5804	Invalid sector number
5805	Byte displacement not even
5806	Modification spans sector boundary
5807	Verification error
5808	Invalid number of sectors
60	EXECUTE Error Codes
6001	No program found
6005	Debug not present
61	EXIT and QUIT Error Codes
6101	EXIT invalid if partition idle
6102	Cannot QUIT while batch still active
6103	Lost memory
61	FILE-BACKUP Error Codes
6121	Pathname exceeds record capacity
6122	Bad cutoff date syntax
6123	Invalid file directory entry
6124	Options specification conflict
6125	Operator requested termination
6126	Record size too short
61	FILE-RESTORE Error Codes
6141	File not present or missed
6142	First file record not node record
6143	Unexpected end of file
6144	Too many ADUs
6145	Generated access name error
6146	Nondata record encountered
6147	Directory file conflict
6148	Operator requested termination
6149	Record size too short
6150	Future file structure level
6151	Backup name required
6152	Add or replace required
6153	Internal error in FILE-RESTORE

62 Copy Utility Error Codes

VCOPY Error Codes

6201	Device not disk
6206	Destination and source are same device
6207	Source and destination are not initialized to the same physical or logical format
6208	Bad ADU on destination disk conflicts with source disk allocation
6211	Volume name does not match name entered
6212	Excessive number of verify errors

FCOPY Error Codes

6220	Operator requested termination
6221	Invalid destination file
6222	Too many directory levels
6223	Invalid source file

62 FLAG-PROGRAM Error Codes

6241	Program not found
6242	Incomplete or inconsistent parameters specified

62 FLAW-ADU, FLAW-TRACK and MAP-FLAWS Error Codes

6260	Invalid device
6261	Volume not loaded
6262	Device dedicated
6263	Disk volume is old format (before RM/COS release 2.4)
6264	Invalid ADU number
6265	Invalid cylinder or head number

62 INITIALIZE, RECLOSE, and VCOPY Common Error Codes

6281	Destination drive contains loaded volume
6282	Drive is already dedicated to another process
6283	Drive cannot be dedicated because files are assigned
6284	Configured bit map too small for this disk
6285	Drive may not receive listing output
6286	Requested disk not remounted
6287	System drive not dedicated because other partitions active
6288	Disk not initialized or format unknown

63	HALT and KTASK Error Codes
6301	SYSTEM-SHUTDOWN is in progress
6302	Command aborted by EXIT command
6303	Specified partition is idle
6304	Partition number illegal
6305	HALT already pending on specified partition
64	INITIALIZE Error Codes
6401	Device not disk
6404	Invalid bad ADU parameter value
6406	Index track marked as bad
6407	Invalid EST NUMBER OF VCATALOG FILES
6408	Invalid volume name
65	IPROGRAM Error Codes
6501	Object record checksum incorrect
6502	Error when closing input object file
6503	Error when closing output program file
6504	Error when processing data area identification
6505	Error when terminating because of previous error
6506	Invalid identification tag in input object file
6507	Error when initializing data area input
6508	Error when initializing input object file
6509	Error when initializing output program file
6510	Error when installing data area object
6511	Error when installing procedure area object
6512	Invalid object tag in input object file
6513	Error when processing identification length field
6514	Error when processing load address value in object input file
6515	Error when processing object data value in object input file
6516	Error when terminating normally
6517	Invalid hexadecimal value in object input file
6518	Error when opening input object file
6519	Error when opening output program file
6520	Error when writing output program file
6521	Error when processing procedure area identification
6522	Error when reading input object file
6523	Input object file contains data relocatable value
6524	Error when processing symbol definition tag
65	COMBINE Error Codes
6531	Too many programs

65 I-BOOT Error Codes

6560 Device not disk
6561 Invalid volume
6562 Invalid file organization
6564 Invalid member name
6566 File not allocated as single extent
6568 Unknown boot type
6569 Insufficient memory

65 INSTALL, REMOVE and TEST-SYSDEFIL Error Codes

6580 Device not disk
6581 Invalid volume
6582 Invalid file organization
6583 Invalid record length
6584 Invalid member name
6585 File undefined
6587 Invalid boot size
6588 Unknown boot type
6589 Bad verification
6590 JDL Image File changed (IPL required)
6591 No system installed on volume
6592 Test SYSDEFIL same as installed SYSDEFIL
6593 System Image File too fragmented
6594 Invalid block size

66 KEY Error Codes

6601 Incorrect parameter count
6602 Incorrect column order
6603 Incorrect key length
6604 Duplicates illegal in prime record key

67 KPRINTER Error Codes

6701 Device not printer
6702 Printer idle

69 LOAD Error Codes

6901 Volume name in use
6902 Volume loaded
6903 Device not disk
6904 Volume label does not match user supplied label
6905 Configured bit map size too small for this disk
6906 Device dedicated
6907 Volume not RM/COS format
6908 Volume not usable on this system

70	LOOP and REPEAT Error Codes
7002	REPEAT not preceded by LOOP
7003	LOOP illegal if batch input file not disk file
71	MESSAGE Error Codes
7101	Illegal partition
7103	Operator requested termination
7104	Syntax error
7105	Message text too long
72	FDUMP and FMODIFY Error Codes
7201	Checksum incorrect
7202	Byte displacement not even
7203	Relocation invalid
7206	Invalid block number
7208	Invalid member name
7209	Offset out of range
7211	Verification data does not match current data
7212	Object file header invalid
7213	Object file relocation table invalid
7214	Offset greater than 131068
7215	Invalid attempt to change relocation
73	PARTITION Error Codes
7301	Specified partition is not idle
7302	Partition number illegal
7303	A following partition is not idle
7304	Requested size illegal
7305	Specified partition not in System Definition File
7306	Decrease in size illegal in interrupt mode
74	PRINT Error Codes
7402	Invalid file organization
76	RECLOSE Error Codes
7602	File not open
77	RELEASE Error Codes
7701	Bad logical name

78	RENAME Error Codes
7801	Bad pathname
7802	Volume mismatch
7803	File already exists
7804	Pathnames same
7805	Attempt to rename a system file
7806	OLD and NEW parameters mix volume and file names
7807	New volume name is invalid
7808	Device is dedicated
7810	New volume name of a loaded volume is already loaded
7811	Disk not initialized or format unknown
7812	Disk does not match loaded volume
7813	Attempt to rename an MVD file
78	REPLACE Error Codes
7821	Logical name not found
7822	Device not disk
7823	File not assigned exclusive all
7824	Destination file is scratch
7825	Source file is not scratch
7826	Source file is undefined
7827	Destination file is write protected
7828	Volume mismatch
7829	System file mismatch
79	SETCOND Error Codes
7901	Illegal partition
7902	Partition not active
80	SHOW Error Codes
8001	Invalid file organization
80	SYSTEM-SHUTDOWN Error Codes
8081	SYSTEM-SHUTDOWN already in progress
8082	System disk is reserved
81	STATUS Error Codes
8101	Parameters ambiguous
8102	Illegal partition
8103	Name does not exist
8104	System description error

82	SWITCH Error Codes
8201	Too many switches specified
8202	Illegal partition
8203	Invalid switch number
8204	Partition not active
83	TIME Error Codes
8301	Invalid parameter value
83	UNCOUPLE Error Codes
8320	Invalid partition number
84	UNLOAD Error Codes
8401	Volume not loaded
8402	Device not disk
8403	Files assigned
8404	System disk
8405	Device dedicated
85	VARY Error Codes
8501	Device already disabled
8502	Device already enabled
8503	Device active
8504	Device assigned
8505	Device is system disk
8506	Device has active user
86	MAP-KEYS Error Codes
8602	Logical name not indexed

87	SORT-MERGE Error Codes
8700	Key error
8701	Insufficient memory
8702	Insufficient sort work files
8703	RELEASE record mismatch
8704	Record too short
8705	Run number overflow
8710	Incorrect parameter count
8711	Incorrect key start
8712	Incorrect key length
8713	Incorrect key type
8714	Incorrect merge input file count
8715	Record size required
88	SYNONYM Error Codes
8801	Synonym table full
8802	Synonym value too long
8803	Synonym not found
8804	No synonym table
89	Communications Error Codes
	CONNECT Error Codes
8903	Invalid TYPE option
	RECEIVE and FTS Error Codes
8911	Undefined pathname
8912	Invalid program file format
8913	Invalid MDS file format
8914	Characters not translated
	SEND and FTS Error Codes
8921	Invalid mode option
8922	Undefined pathname
8923	Too many files specified
8924	Characters not translated

3780 I/O Error Codes

8931	Unexpected data in printer sequence
8933	Error in tab stop image
8937	Connect timeout
8938	Coded record too long

IMPLEMENTOR DEFINED COBOL I/O ERROR CODES

9000	Illegal I/O function
9100	File not opened
9201	File open, duplicate logical name
9202	File open, COBOL file name not closed
9301	File not available, logical name not found
9303	File not available, FD locked
9304	File not available, access conflict
9401	Invalid open, insufficient memory
9402	Invalid open, exhausted extents
9403	Invalid open, exhausted ADUs
9404	Invalid open, file not expandable
9405	Invalid open, exhausted blocks
9406	Invalid open, disk too fragmented
9407	Invalid open, access past last extent
9410	Invalid open, disk dedicated
9411	Invalid open, record size mismatch
9412	Invalid open, block size mismatch
9413	Invalid open, organization mismatch
9414	Invalid open, number of keys mismatch
9415	Invalid open, file undefined
9416	Invalid open, block size invalid
9417	Invalid open, no characteristics specified
9418	Invalid open, old file format
9419	Invalid open, new file format
9420-942E	Invalid open, key starting position mismatch
9430-943E	Invalid open, key length mismatch
9441-944E	Invalid open, key duplicates mismatch
9450	Invalid open, no member
9451	Invalid open, duplicate member
9452	Invalid open, directory full
9453	Invalid open, NO REWIND not correctly positioned
9454	Invalid open, file position not found
9455	Invalid open, record size invalid
9456	Invalid open, file count not specified
9457	Invalid open, density invalid
9458	Invalid open, extend not supported
9500	Illegal device type
9600	Undefined record pointer
9701	Invalid record length, record area too short
9702	Invalid record length, record area too long
9703	Invalid record length, record too long
9704	Invalid record length, record size mismatch
9705	Invalid record length, fixed length required

9800	Write protected
9900	Record locked

SYSTEM STOP CODES AND MESSAGES

IPL Errors

A002	P records request more memory than is available
A004	Integer value error
A005	S record specifies invalid year
A006	U record specifies invalid device prefix
A007	D record specifies invalid device name
A008	Insufficient memory for JDL names
A00B	Error opening JDL Image File
A00C	No valid terminal defined
A00D	U record defines duplicate device name
A00E	U record defines device for which there is no device service routine
A010	U record specifies invalid station ID
A012	P record specifies invalid priority value
A013	C record specifies invalid board type
A014	U record specifies invalid controller
A015	U record specifies invalid unit number
A016	U record specifies duplicate unit number
A017	Error opening JDL Definition File
A018	System disk not found
A019	Error reading System Definition File
A01A	Error opening System Definition File
A01B	System Definition File is empty
A01C	Error opening System Image File
A01D	Error reading System Image File
A01E	Error closing System Image File
A01F	Terminal type not recognized
A020	Command directory full
A021	S record contains invalid type-ahead buffer size
A022	S record contains invalid event log buffer size

DSR Errors

C000-CFFF	Device dependent C record error
D000-DFFF	Device dependent U record error

System Stop Message

---> System Failure <---

APPENDIX B

COBOL SUBROUTINE

LIBRARY PACKAGE

INTRODUCTION

This appendix contains a general explanation for each of the subroutines in the COBOL subroutine library. Each explanation details the function of the routine, the COBOL calling sequence, a description of each parameter, and the error codes returned.

The following subroutines are available:

C$CLOSE - Close an arbitrary file
C$COMM - Control RS232 communication port
C$CURSOR - Return the relative position of the cursor
C$DELAY - Suspends program for a time period
C$FILEI - Return information about an assigned file
C$MAPS - Map and return a synonym value
C$OFFSET - Specify the initial cursor position
C$OPENI - Open a file for input
C$READS - Read the next record on a file
C$RERR - Return last I/O completion status
C$SCC - Set condition code
C$SCRD - Read terminal screen
C$SETS - Define a synonym
C$TTSL - Terminate time slice
EV$CLEAR - Clear event code
EV$SET - Set event code
EV$SGNL - Signal event occurrence
EV$WAIT - Wait for event occurrence

All of these subroutines reside on the program file .C$SUBS. In order to use these routines, the following JDL command must be performed prior to executing the COBOL program containing the references to these routines:

```
/ASSIGN, LOGICAL NAME = C$SUBS, NAME = .C$SUBS, ACCESS=RO
```

WARNING

These subroutines frequently receive source fixes or enhancements from one RM/COS release to the next. Therefore the C$SUBS program file should not be COMBINEd with application programs.

C$CLOSE

Function: Close an arbitrary file

Calling sequence:

```
CALL "C$CLOSE" USING LOGICAL-NAME, CLOSE-OPTIONS,
                     RETURN-STATUS.
```

LOGICAL-NAME

An alphanumeric data item which contains the logical name of the file to be read. Only the first eight characters of LOGICAL-NAME and only those characters to the left of any space character occurring in LOGICAL-NAME will be used to determine the logical name of the file.

CLOSE-OPTIONS

A one character alphabetic item which must be set to "N" to cause a CLOSE WITH NO REWIND or a " " to cause a close with rewind.

RETURN-STATUS

A four character alphanumeric data item used for returning the RM/COS I/O completion code and the information code. The I/O completion code is returned in the first two character positions, and the information code is returned in the last two character positions. These codes are in ASCII representation and correspond to the RM/COS status codes. Note that the RM/COS I/O completion code is the same code returned to the FILE STATUS item on completion of the I/O. The RM/COS information code provides additional detail as to why the error occurred.

C$COMM

Function: Perform various communication utility functions under the direction of the user program.

Calling Sequence:

Format 1 - Control Functions:

```
CALL "C$COMM" USING FUNCTION-CODE, RET-STATUS,
     LINK-NAME, CCT-AREA, COMP-1-ZERO, COMP-1-ZERO.
```

Format 2 - Read/Write Functions:

```
CALL "C$COMM" USING FUNCTION-CODE, RET-STATUS,
     LINK-NAME, DATA-AREA, DATA-COUNT, AREA-SIZE.
```

Format 3 - Check Functions:

```
CALL "C$COMM" USING FUNCTION-CODE, RET-STATUS,
     CHECK-VALUE, DATA-AREA, DATA-COUNT, AREA-SIZE.
```

FUNCTION-CODE

A COMP-1 data item which specifies the user link communication function to be performed. The allowable values and the resulting actions are:

Value	Action
0	Perform link control operation(s) as specified by the Communication Control Table (CCT-AREA) parameter. This function uses the Format 1 calling sequence.
1	Read data from the user link specified by the LINK-NAME parameter into the storage locations described by the DATA-AREA, DATA-COUNT and AREA-SIZE parameters. This function uses the Format 2 calling sequence. The function will complete with a RET-STATUS of "05" if the ENABLE-RECEIVER flag was not set in the LINK-FLAGS of the last control function.

2 Copy data from the storage locations described by the DATA-AREA, DATA-COUNT and AREA-SIZE parameters to the transmitter buffer of the communication link specified by the LINK-NAME parameter, and pass all characters in the transmitter buffer to the hardware interface. This function uses the Format 2 calling sequence. The function will complete as soon as the characters are copied from DATA-AREA, unless the WAIT-WRITE-COMPLETE flag was set in the LINK-FLAGS of the last control function. The function will complete with a RET-STATUS of "05" if the ENABLE-TRANSMITTER flag was not set in the LINK-FLAGS of the last control function.

3 Perform a "CRC-16" block check operation on the data specified by the DATA-AREA, DATA-COUNT and AREA-SIZE parameters. This function uses the Format 3 calling sequence.

4 Perform an "LRC" block check operation on the data specified by the DATA-AREA, DATA-COUNT and AREA-SIZE parameters. This function uses the Format 3 calling sequence.

5 Copy data from the storage locations described by the DATA-AREA, DATA-COUNT and AREA-SIZE parameters to the transmitter buffer of the communication link specified by the LINK-NAME parameter, but do not pass all characters in the transmitter buffer to the hardware interface because more data in the same message will follow. This function uses the Format 2 calling sequence. The function is useful during synchronous transmission to reduce the likelihood of transmitter underrun causing the hardware interface to perform sync character insertion. The function will complete with a RET-STATUS of "05" if the ENABLE-TRANSMITTER flag was not set in the LINK-FLAGS of the last control function.

RET-STATUS

A two character alphanumeric data item used for returning the user communication link status code. These codes are in ASCII representation and correspond to the following values:

"00" Normal completion
"01" Timeout occurred
"02" Parity error or buffer overrun occurred
"03" Disconnection occurred (Data Set Ready dropped)
"04" No DATA-AREA to be read or written
"05" Control function error

See also the CCT-AREA parameter description.

LINK-NAME

A one to eight character alphanumeric data item representing the logical name of the user programmable communication link (See also the CONNECT JDL command description).

CCT-AREA

The Communication Control Area for the user link. This data item is a group consisting of the following elementary data items. Changes in the values of items in this group take affect only when a control function (FUNCTION-CODE = 0) is issued.

LINK-FLAGS

A COMP-1 data item which controls the state and operation of the link. See the LINK-FLAGS operation section.

RATE

A COMP-1 data item whose value describes the data rate in bits per second of the asynchronous transmitter and receiver. This value is used only when the PRESET flag is initially set in the LINK-FLAGS. The set of allowed values depends on the hardware interface, and is described in the RM/COS Operator Guide for the machine. If a value of 0 is provided, the data rate assumes the value specified in the System Definition File. If the data rate is not allowed by the hardware, an error status (RET-STATUS = "05") will be returned.

SYNC-CHAR

A one character alphanumeric data item whose value specifies the user's synchronization character. If the ENABLE-SYNC flag is set in

LINK-FLAGS, received characters are discarded on a read function until a character is read which, when masked by the MASK-SYNC byte in MASK-VALUE, equals SYNC-CHAR. If the ENABLE-SYNC flag is not set in LINK-FLAGS, then the value of SYNC-CHAR is ignored and synchronization is assumed upon reading the first character. If the System Definition File specifies that the link is synchronous, the value of this one character data item when the PRESET flag is set in LINK-FLAGS is used as the hardware synchronization character. For example, synchronization upon any ASCII control character being received is indicated by setting the ENABLE-SYNC flag in LINK-FLAGS, setting SYNC-CHAR to a value of zero, and setting MASK-VALUE in the range of decimal -8192 to -7937 (hexadecimal E000 to E0FF).

TERMINATOR-CHAR

A one character alphanumeric data item whose value specifies the user's message terminator character. If the ENABLE-TERMINATOR flag is set in LINK-FLAGS and synchronization has been established (see SYNC-CHAR) the reading of a character which, when masked by the MASK-TERMINATOR byte in MASK-VALUE, equals TERMINATOR-CHAR causes immediate normal termination (RET-STATUS = "00") of the read function. If the ENABLE-TERMINATOR flag is not set in LINK-FLAGS, then the value of TERMINATOR-CHARACTER is ignored.

MASK-VALUE

A COMP-1 data item treated as two one-byte masks. The high order byte is the MASK-SYNC mask value; it is logically ANDed with received characters during a read operation and the result is compared with the SYNC-CHAR when ENABLE-SYNC is set in LINK-FLAGS. The low order byte is logically ANDed with received characters during a read function and the result is compared with the TERMINATOR-CHAR when ENABLE-TERMINATOR is set in LINK-FLAGS. A decimal value of -1 (hexadecimal FFFF) is equivalent to no mask. A decimal value of -7968 (hexadecimal E0E0) and the use of zero for SYNC-CHAR and TERMINATOR-CHAR is equivalent to testing for any ASCII control character. The MASK-VALUE is ignored when neither ENABLE-SYNC nor ENABLE-TERMINATOR is set in LINK-FLAGS. In any case, the masking affects only the

comparisons made for synchronization and termination; characters placed in the user's DATA-AREA are not masked.

RECEIVE-BUFFER-SIZE
A COMP-1 data item whose value specifies the "logical" size of the First-In/First-Out (FIFO) receiver buffer. This field is used only when the FLUSH-RECEIVER flag is set in LINK-FLAGS. The value is ignored if it exceeds the physical receiver buffer size specified in the System Definition File. A value of 0 indicates that the current value of this parameter should be retained.

TRANSMIT-BUFFER-SIZE
A COMP-1 data item whose value specifies the "logical" size of the FIFO transmitter buffer. This field is used only when the FLUSH-TRANSMITTER flag is set in LINK-FLAGS. The value is ignored if it exceeds the physical transmitter buffer size specified in the System Definition File. A value of 0 indicates that the current value of this parameter should be retained.

CONNECT-TIMEOUT
A COMP-1 data item whose value specifies the amount of time in tenths of seconds that the system will wait for a link connection operation to complete. A value of 0 will cause the control function (FUNCTION-CODE = 0) to always complete without delay. A timeout status (RET-STATUS = "01") will be returned if the connection of the link is not completed.

READ-TIMEOUT
A COMP-1 data item whose value specifies the amount of time in tenths of seconds that the system will wait for a read function (FUNCTION-CODE = 1) to complete. A value of 0 will cause the read function to complete without delay. A timeout status (RET-STATUS = "01") will be returned if the read function is not terminated for some other reason before the end of the timeout interval.

WRITE-TIMEOUT
A COMP-1 data item whose value specifies the amount of time in tenths of seconds that the system will wait for a write function (FUNCTION-CODE = 2) to complete. A value of 0 will cause the write function to complete without delay. A timeout status (RET-STATUS = "01") will be returned if the write function is not terminated for some other reason before the end of the timeout interval.

DATA-AREA
A user specified data item whose value (in conjunction with DATA-COUNT and AREA-SIZE) determines data to be read from or written to the user link or checked by a block checking function.

DATA-COUNT
A COMP-1 data item which reflects the number of characters in the DATA-AREA already processed by a read, write or check function. For a read function, the value of DATA-COUNT represents the offset in DATA-AREA which determines the storage location which will be used to contain the first character of data read from the link. For a write function, the value of DATA-COUNT represents the offset in DATA-AREA which determines the storage location of the first data character to be written to the link. For a check function, the value of DATA-COUNT represents the offset in DATA-AREA which determines the storage location of the first data character to be checked. In all cases, the value of DATA-COUNT upon completion of the requested function is adjusted by the number of characters processed by the function. If the value of DATA-COUNT is not less than the value of AREA-SIZE (or the actual size of DATA-AREA if the value of AREA-SIZE is zero), an argument error is declared.

AREA-SIZE
A COMP-1 data item whose value limits the size of the DATA-AREA item to the length specified. A value of 0 indicates that the actual size of the DATA-AREA item should be used.

CHECK-VALUE
In the case of the CRC-16 (Cyclic Redundancy Check) block check function (FUNCTION-CODE = 3), the value of this two character alphanumeric data item represents the accumulated remainder of the CRC-16 polynomial division applied to user specified data. In the case of the LRC (Longitudinal Redunduncy Check) block check function (FUNCTION-CODE = 4), the value of this one character alphanumeric data item represents the accumulated exclusive-or of each character of the user specified data.

COMP-1-ZERO
A COMP-1 data item with a value of zero. This argument is not altered by C$COMM.

LINK-FLAGS Operation

The value of the LINK-FLAGS data item contained in the CCT-AREA group determines the state of the user data link and the operational characteristics of functions performed on the link. The value of this item is determined by the arithmetic sum of the following flag values. Note that the state of the link is determined by the LINK-FLAGS value of the latest control function performed on the link.

For example, to connect (assert Data Terminal Ready and wait for Data Set Ready) a link control function should be performed with a LINK-FLAGS value which contains the CONNECT flag term. The link will remain connected as long as the CONNECT flag term is present in every subsequent control function. When a control function is performed which does not contain the CONNECT flag, the link will be disconnected (Data Terminal Ready turned off).

A description of each LINK-FLAGS flag term follows:

PRESET (value 1)
The initial setting of this flag causes the user link to be prepared for operation and the data rate to be set according to the RATE item in the CCT-AREA group. The data rate of the link may be changed only by clearing this flag and then setting it.

CONNECT (value 2)
The initial setting of this flag causes Data Terminal Ready to be asserted and Data Set Ready to be detected. If Data Set Ready is not detected within the CONNECT-TIMEOUT interval, the control function will terminate with a RET-STATUS value of "01", without turning off Data Terminal Ready.

Subsequent presence of the flag will maintain the link. Absence of the flag will cause link disconnection (Data Terminal Ready will be turned off). If this flag is set and the PRESET flag is off, the control function will terminate with a RET-STATUS value of "05".

ENABLE-TRANSMITTER (value 4)
When this flag is set, the link transmitter will be enabled, Request to Send will be asserted, and the system will wait for Clear to Send. Once Clear to Send is on, data present in the transmitter buffer will be transmitted. If this flag is set and the CONNECT flag is off, the control function will terminate with a RET-STATUS of "05". When this flag is reset, the link transmitter will be disabled and Request to Send will be turned off; disabling the transmitter will wait for characters already written to be transmitted if the WAIT-WRITE-COMPLETE flag is set in LINK-FLAGS. When this flag is not set, subsequent write requests (FUNCTION-CODE = 2 or 5) will terminate with a RET-STATUS of "05".

ENABLE-RECEIVER (value 8)
When this flag is set, data received on the link will be placed in the receiver buffer for subsequent processing by a read function. If this flag is set and the CONNECT flag is off, the control function will terminate with a RET-STATUS of "05". When this flag is not set, subsequent read requests (FUNCTION-CODE = 1) will terminate with a RET-STATUS of "05".

ENABLE-TERMINATOR (value 32)
If this flag is set, data removed from the receiver buffer to be placed in the user's DATA-AREA as a result of a read function are compared under mask (see MASK-VALUE) with the TERMINATOR-CHAR. If a match occurs, the read function is immediately terminated. The matching data character is discarded if the STRIP-TERMINATOR flag is set. This testing of received characters does not begin until synchronization occurs (see ENABLE-SYNC).

ENABLE-PARITY (value 64)
On input, if this flag is set, data characters removed from the receiver buffer to be placed in the user's DATA-AREA as a result of a read function are checked for proper parity (even or odd as determined by the ODD-PARITY flag). If the data character has incorrect parity, it is discarded and the read function is terminated

immediately with a RET-STATUS value of "02". If the parity of the character is correct the high-order bit is cleared (set to 0) and the resulting character is placed in the DATA-AREA. If the ENABLE-PARITY flag is not set, all eight bits of each data character are passed to the user program unchecked. Note that the comparison of incoming data to the SYNC-CHAR and/or TERMINATOR-CHAR is made after parity checking and stripping. Thus, the SYNC-CHAR and TERMINATOR-CHAR character values must not have their high-order bit set if ENABLE-PARITY is set.

On output, if this flag is set, the proper parity bit is generated and replaces the high-order data bit before the data character is transmitted. If ENABLE-PARITY is not set, the data characters are transmitted unchanged.

STRIP-SYNC (value 128)
See ENABLE-SYNC description.

STRIP-TERMINATOR (value 256)
See ENABLE-TERMINATOR description.

ODD-PARITY (value 512)
See ENABLE-PARITY description.

DOUBLE-STOP (value 1024)
If set when the PRESET operation takes place, this flag forces two stop bits to be transmitted at all data rates, if this capability is supported by the hardware interface. If this bit is not set, the number of stop bits transmitted will be one for all data rates.

ENABLE-SYNC (value 2048)
If this flag is set, data removed from the receiver buffer to be placed in the user's DATA-AREA as a result of a read function are first compared under mask (see MASK-VALUE) with the SYNC-CHAR. Until a match occurs, the received characters are discarded without testing for terminators (see ENABLE-TERMINATOR). When a match occurs, the ENABLE-SYNC flag is turned off and remains off unless set by another control function call of C$COMM. The matching data character is discarded if the STRIP-SYNC flag is set. If the STRIP-SYNC flag is not set and the ENABLE-TERMINATOR flag is set, the matching character is tested to see if it is a terminator (see ENABLE-TERMINATOR). For example, use of ENABLE-SYNC, a MASK-VALUE in the range decimal -8192 to -7937 (hexadecimal E000 TO E0FF), and

SYNC-CHAR with a value of zero allows the receiver to synchronize on receipt of a control character; this is useful when synchronization has been lost and message characters are still being received.

FLUSH-TRANSMITTER (value 4096)
If this flag is set, all data characters currently stored in the transmitter buffer will be discarded. Additionally, the value of the TRANSMIT-BUFFER-SIZE item will be used to establish the size of the transmitter buffer.

FLUSH-RECEIVER (value 8192)
If this flag is set, all data characters currently stored in the receiver buffer will be discarded. Additionally, the value of the RECEIVE-BUFFER-SIZE item will be used to establish the size of the receiver buffer.

WAIT-WRITE-COMPLETE (value 16384)
If this flag is set, the system will wait until all characters from the transmit buffer have been transmitted by the hardware interface before completing a write operation or disabling the transmitter. If this flag is not set before a write operation, the write operation completes as soon as all characters from DATA-AREA have been copied to the transmitter buffer. If this flag is not set when the ENABLE-TRANSMITTER flag is reset, the transmitter is stopped immediately.

An annotated COBOL program which demonstrates the use of C$COMM for one application is given in Chapter 6, Data Communications.

C$CURSOR

Function: Return the relative position of the cursor at input termination of an ACCEPT statement.

Calling Sequence:

```
CALL "C$CURSOR" USING CURSOR-VALUE.
```

CURSOR-VALUE
: A COMP-1 data item which is set to indicate the one relative position of the cursor within the input field of the immediately preceding ACCEPT statement. "One relative" means the first character position of an ACCEPT field is represented by a value of one. If no ACCEPT statement has been executed, the contents of CURSOR-VALUE is undefined. Any intervening DISPLAY statement causes the contents of CURSOR-VALUE to be undefined. If the ACCEPT statement did not contain the TAB clause and the ACCEPT operation was terminated by typing a character in the last position of the field (i.e. the operator "typed out" of the field), CURSOR-VALUE will be set to the size of the input field plus one.

This subroutine is used in conjunction with the C$OFFSET subroutine.

C$DELAY

Function: Suspend program for a specified time period.

Calling Sequence:

```
CALL "C$DELAY" USING TIME.
```

TIME

A COMP-1 data item which specifies the time delay in seconds. The maximum time delay allowed is 6553 seconds (about 109 minutes or 1.8 hours).

C$FILEI

Function: Obtain information about an assigned file.

Calling Sequence:

```
CALL "C$FILEI" USING LOGICAL-NAME, INFO-AREA,
                     RETURN-STATUS.
```

LOGICAL-NAME
An alphanumeric data item which contains the logical name of the file for which information is desired. Only the first eight characters of LOGICAL-NAME and only those characters to the left of any space character occurring in LOGICAL-NAME will be used to determine the logical name of the file.

INFO-AREA
A 42-character group item defined as follows:

```
01  INFO-AREA.
    02  DEVICE-TYPE                    PIC X(1).
        88  DISK-DEVICE                VALUE "D".
        88  STATION-DEVICE             VALUE "S".
        88  PRINTER-DEVICE             VALUE "P".
        88  BISYNC-DEVICE              VALUE "B".
        88  USER-LINK-DEVICE           VALUE "U".
        88  TAPE-DEVICE                VALUE "T".
        88  DUMY-DEVICE                VALUE "Y".
        88  OTHER-DEVICE               VALUE " ".
    02  FILE-ORGANIZATION              PIC X(1).
        88  DEVICE-ORGANIZATION        VALUE "D".
        88  SEQUENTIAL-ORGANIZATION    VALUE "S".
        88  RELATIVE-ORGANIZATION      VALUE "R".
        88  INDEXED-ORGANIZATION       VALUE "I".
        88  PROGRAM-ORGANIZATION       VALUE "P".
        88  MDS-ORGANIZATION           VALUE "M".
        88  MVD-ORGANIZATION           VALUE "V".
        88  OTHER-ORGANIZATION         VALUE " ".
    02  FILE-FLAGS.
        03  FILLER                     PIC X(1).
            88  BLANK-SUPPRESS         VALUE "X".
        03  FILLER                     PIC X(1).
            88  DEFAULT-BLOCK-SIZE     VALUE "X".
        03  FILLER                     PIC X(1).
            88  DELETE-PROTECT         VALUE "X".
        03  FILLER                     PIC X(1).
            88  SCRATCH                VALUE "X".
        03  FILLER                     PIC X(1).
            88  SINGLE-EXTENT          VALUE "X".
        03  FILLER                     PIC X(1).
            88  SPOOLING               VALUE "X".
        03  FILLER                     PIC X(1).
```

```
            88  UNDEFINED              VALUE "X".
        03  FILLER                     PIC X(1).
            88  WRITE-PROTECT          VALUE "X".
        03  FILLER                     PIC X(1).
            88  WORK                   VALUE "X".
        03  FILLER                     PIC X(1).
            88  AUTO                   VALUE "X".
        03  FILLER                     PIC X(6).
    02  FILE-BLOCK-SIZE                PIC 9(8).
    02  FILE-RECORD-SIZE               PIC 9(8).
    02  FILE-RECORD-COUNT              PIC 9(8).
```

The appropriate condition (level 88) names will be true and the file block size, record size and record count elementary items will be set upon return from the subroutine.

RETURN-STATUS

A four character alphanumeric data item used for returning the RM/COS I/O completion code and the information code. The I/O completion code is returned in the first two character positions, and the information code is returned in the last two character positions. These codes are in ASCII representation and correspond to the RM/COS status codes. Note that the RM/COS I/O completion code is the same code returned to the FILE STATUS item on completion of the I/O. The RM/COS information code provides additional detail as to why the error occurred.

Note that INFO-AREA is valid only if RETURN-STATUS is equal to "00xx" (where xx is any value).

C$MAPS

Function: Map and return a synonym value

Calling Sequence:

```
CALL "C$MAPS" USING STATUS-CODE, SYNONYM,
                    SYNONYM-VALUE, SYNONYM-COUNT.
```

STATUS-CODE
A two character alphanumeric data item for returning a status code. Status codes are as follows:

00	Successful completion
21	Synonym does not match
33	Synonym value longer than receiving data item
FD	No synonym table present
FE	Synonym too long

SYNONYM
An alphanumeric data item or a nonnumeric literal value. The value specified must be no more than eight characters in length, begin with a letter (A-Z), and contains only letters, digits (0-9), the hyphen (-), and the dollar sign ($). Trailing blanks are removed if present.

SYNONYM-VALUE
An alphanumeric data item of sufficient length to contain the mapped value of the synonym. If the length of the synonym is longer than SYNONYM-VALUE, as much of the value is transferred as possible and a status code is returned. If the length of the synonym is shorter than SYNONYM-VALUE, the value is returned left justified in the data item with the remainder blank filled.

SYNONYM-COUNT
A COMP-1 data item which is set to indicate the actual number of characters transferred.

C$OFFSET

Function: Specify the initial cursor position of an ACCEPT statement to other than the beginning of the field.

Calling Sequence:

```
CALL "C$OFFSET" USING OFFSET-VALUE.
```

OFFSET-VALUE
: A COMP-1 data item which contains the one relative offset to be used to define the initial cursor position on the subsequent ACCEPT statement. OFFSET-VALUE is used modulo 256. Values beyond the field size default to position one.

For example, a program wishes to display a default and accept user modifications starting at the 5th position within the field:

```
      .
      .
      .
DISPLAY CURRENT-VALUE POSITION 10 LINE 5 HIGH.
MOVE 5 TO OFFSET-VALUE.
CALL "C$OFFSET" USING OFFSET-VALUE.
ACCEPT NEW-VALUE POSITION 10 LINE 5.
CALL "C$CURSOR" USING OFFSET-VALUE.
      .
      .
      .
```

C$OPENI

Function: Open a file for input (using C$READS)

Calling sequence:

```
CALL "C$OPENI" USING LOGICAL-NAME, OPEN-OPTIONS,
                     RETURN-STATUS.
```

LOGICAL-NAME

An alphanumeric data item which contains the logical name of the file to be opened. Only the first eight characters of LOGICAL-NAME and only those characters to the left of any space character occurring in LOGICAL-NAME will be used to determine the logical name of the file.

OPEN-OPTIONS

A one character alphabetic item which must be set to "N" to cause an OPEN WITH NO REWIND or a space to cause an open with rewind.

RETURN-STATUS

A four character alphanumeric data item used for returning the RM/COS I/O completion code and the information code. The I/O completion code is returned in the first two character positions, and the information code is returned in the last two character positions. These codes are in ASCII representation and correspond to the RM/COS status codes. Note that the RM/COS I/O completion code is the same code returned to the FILE STATUS item on completion of the I/O. The RM/COS information code provides additional detail as to why the error occurred.

C$READS

Function: Read the next record from a file

Calling sequence:

```
CALL "C$READS" USING LOGICAL-NAME, SLEW-CONTROL,
     RECORD-AREA, CHARACTER-COUNT, RETURN-STATUS.
```

LOGICAL-NAME

An alphanumeric data item which contains the logical name of the file to be read. Only the first eight characters of LOGICAL-NAME and only those characters to the left of any space character occurring in LOGICAL-NAME will be used to determine the logical name of the file.

SLEW-CONTROL

A nine character group-item as follows:

```
01  SLEW-CONTROL.
    02  FILLER                          PIC X(1).
        88  SPOOL-FILE                  VALUE "S".
    02  FILLER                          PIC X(1).
        88  PAGE-BEFORE-PRINT           VALUE "P".
    02  FILLER                          PIC X(1).
        88  PAGE-AFTER-PRINT            VALUE "P".
    02  FILLER                          PIC X(1).
        88  LINE-SLEW-BEFORE-PRINT      VALUE " ".
        88  CHANNEL-SLEW-BEFORE-PRINT   VALUE "C".
    02  FILLER                          PIC X(1).
        88  LINE-SLEW-AFTER-PRINT       VALUE " ".
        88  CHANNEL-SLEW-AFTER-PRINT    VALUE "C".
    02  LINE-CHANNEL-SLEW-BEFORE-PRINT  PIC S9(5)
                                        COMP-1.
    02  LINE-CHANNEL-SLEW-AFTER-PRINT   PIC S9(5)
                                        COMP-1.
```

If the file indicated by LOGICAL-NAME is a sequential file with spooling information, the slew control information for the record returned will be interpreted and placed in SLEW-CONTROL: SPOOL-FILE will be true. In all other cases, SPOOL-FILE will be false.

RECORD-AREA

A data item into which the record is to be read. If the record is shorter than RECORD-AREA, the record will be left-justified but the remaining characters at the right of RECORD-AREA will be unchanged.

CHARACTER-COUNT

A COMP-1 data item which is set to the size of the record that was read.

RETURN-STATUS

A four character alphanumeric data item used for returning the RM/COS I/O completion code and the information code. The I/O completion code is returned in the first two character positions, and the information code is returned in the last two character positions. These codes are in ASCII representation and correspond to the RM/COS status codes. Note that the RM/COS I/O completion code is the same code returned to the FILE STATUS item on completion of the I/O. The RM/COS information code provides additional detail as to why the error occurred.

Note that other return parameters are valid only if RETURN-STATUS is equal to "00xx" (where xx is any value).

A special "soft" end-of-file status of "1099" indicates another process still has the file open in output mode.

C$RERR

Function: Retrieve the last I/O completion status.

Calling sequence:

```
CALL "C$RERR" USING RET-STATUS.
```

RET-STATUS

A four character alphanumeric data item used for returning the RM/COS I/O completion code and the information code. The I/O completion code is returned in the first two character positions, and the information code is returned in the last two character positions. These codes are in ASCII representation and correspond to the RM/COS status codes. Note that the RM/COS I/O completion code is the same code returned to the FILE STATUS item on completion of the I/O. The RM/COS information code provides additional detail as to why the error occurred.

C$SCC

Function: Set the condition code.

Calling sequence:

```
CALL "C$SCC" USING CONDITION-CODE.
```

CONDITION-CODE

A one character alphanumeric data item which specifies the new condition code of the partition in which the program is executing. The condition code is not updated until the run unit completes execution, thus only the last value set, whether by C$SCC or runtime error processing, will be used to set the partition condition code. Runtime error processing sets the condition code to the letter Z if the program is terminated because of an error, regardless of any prior C$SCC calls. A value of space does not change the existing partition condition code and is the default value used in the absence of C$SCC calls and error termination.

C$SCRD

Function: Read the terminal display screen.

Calling Sequence:

```
CALL "C$SCRD" USING SCREEN-BUFFER [, BUFFER-SIZE
                [, SCREEN-LINE [, SCREEN-POSITION ]]].
```

SCREEN-BUFFER
: An alphanumeric data item to receive the characters read from the terminal display screen. Only the first 1920 characters are used; any excess is not blank filled. Control characters, if any, are translated to spaces and intensity control bits are set to zero so that all characters returned are not less than space and not greater than tilde (~).

BUFFER-SIZE
: A COMP-1 data item whose value is the number of characters to be read. If the value of BUFFER-SIZE is zero or the parameter is omitted, the actual size of SCREEN-BUFFER is used.

SCREEN-LINE
: A COMP-1 data item whose value is the line at which the cursor is to be positioned prior to the screen read. If the parameter is omitted a value of 1 is assumed. The value of this parameter is otherwise subject to the default and modulo rules described for the LINE clause of ACCEPT and DISPLAY.

SCREEN-POSITION
: A COMP-1 data item whose value is the position on the line at which the cursor is to be positioned prior to the screen read. If the parameter is omitted a value of 1 is assumed. The value of this parameter is otherwise subject to the default and modulo rules described for the POSITION clause of ACCEPT and DISPLAY.

Cursor positioning after the call to C$SCRD obeys the rule described in ACCEPT and DISPLAY.

C$SETS

Function: Define a synonym

Calling Sequence:

```
CALL "C$SETS" USING STATUS-CODE, SYNONYM,
                    SYNONYM-VALUE, SYNONYM-COUNT.
```

STATUS-CODE
A two character alphanumeric data item for returning a status code. Status codes are as follows:

00	Successful completion
21	Synonym does not match
FB	Synonym value too long
FC	Synonym not LNAME type
FD	No Synonym table present
FE	Synonym too long
FF	Synonym table full

SYNONYM
An alphanumeric data item or a nonnumeric literal value. The value specified must be no more than eight characters in length, begin with a letter (A-Z), and contain only letters, digits (0-9), the hyphen (-), and the dollar sign ($). Trailing blanks are removed if present.

SYNONYM-VALUE
An alphanumeric data item or a nonnumeric literal value.

SYNONYM-COUNT
A COMP-1 data item which contains the number of characters to be used (starting from the left) from SYNONYM-VALUE. If SYNONYM-COUNT is a zero the SYNONYM is deleted.

WARNING

The C$SETS allows binary data for SYNONYM-VALUEs. Care should be exercised with the MAP-SYNONYMS command if the synonym values contain binary data since some values may translate into control information.

C$TTSL

Function: Terminate the CPU scheduling time slice of the program

Calling Sequence:

```
CALL "C$TTSL".
```

When a partition is active and the system scheduling program determines that the partition is to be allocated CPU time for execution, the partition is allocated a unit of time called a time slice in which to execute. If the partition does not relinquish control of the CPU by requesting I/O, a time delay or terminating, CPU control is rescheduled after the time slice has expired. This subroutine allows a partition to voluntarily relinquish control of the CPU for rescheduling without requesting I/O or delaying. This is useful when it is necessary to wait for an event to occur, such as a locked record being released, but it is not desirable to delay for a full second nor to occupy the CPU when no useful work can be done.

EV$CLEAR

Function: Clear the user event code.

Calling Sequence:

```
CALL "EV$CLEAR".
```

The user event code is also cleared by the subroutine C$DELAY and at the termination of the run unit (i.e., at STOP RUN).

If an event for which this process (i.e. run unit) has previously set its user event code has occurred, but for which this process has not waited, the pending event will be discarded.

EV$SET

Function: Set the user event code of interest to the process.

Calling Sequence:

```
CALL "EV$SET" USING EVENT-CODE.
```

EVENT-CODE
: A data item or literal the value and length of which are used to determine a 32-bit event code by the use of a CRC-32 algorithm. This item or literal may be of any length and value. However, since both length and value are used in determining the event code used by RM/COS, care must be taken that EVENT-CODE used by EV$SET and EV$SGNL match in both length and value.

If an event for which this process (i.e., run unit) has previously set its user event code has occurred, but for which this process has not waited, the pending event will be discarded.

EV$SGNL

Function: Signal the occurrence of a user event to all processes interested in that event.

Calling Sequence:

```
CALL "EV$SGNL" USING EVENT-CODE, STATUS-CODE.
```

EVENT-CODE
A data item or literal the value and length of which are used to determine a 32-bit event code by the use of a CRC-32 algorithm. This item or literal may be of any length and value. However, since both length and value are used in determining the event code used by RM/COS, care must be taken that EVENT-CODE used by EV$SET and EV$SGNL match in both length and value.

STATUS-CODE
A COMP-1 data item in which a status code is returned. Status codes are as follows:

0 Successful completion
1 Event ID error (system error)

Signalling an event causes the system to search for all processes which have set a matching event code. For each process found, one of the following actions occurs:

if the process is currently waiting for the event, the process is restarted;

otherwise, the occurrence of the event is recorded and the process will be immediately restarted when it waits for the event.

EV$WAIT

Function: Wait for the occurrence of the event of interest or for a specified time period, whichever occurs first.

Calling Sequence:

```
CALL "EV$WAIT" USING DELAY, STATUS-CODE.
```

DELAY

A COMP-1 data item which specifies the time delay in seconds. The maximum time delay allowed is 6553 seconds (about 109 minutes or 1.8 hours). A time delay of zero is permitted to allow testing for the occurrence of an event without any delay.

STATUS-CODE

A COMP-1 data item in which a status code is returned. Status codes are as follows:

0 Event occurred
1 No event code has been set
2 Time delay expired before occurrence of the desired event

APPENDIX C

MEMORY REQUIREMENTS

INTRODUCTION

This appendix provides general information regarding memory requirements when executing JDL commands or COBOL programs in an RM/COS partition. The sizes given are approximate and some are likely to change with future versions of the system. The sizes are meant as guidelines in allocating memory to partitions.

Note that it would be inflexible to allocate only just sufficient memory to any particular partition based on the numbers given in this appendix. It is especially important to consider the problems of memory fragmentation which can occur in a partition if a process is actively calling and canceling programs. especially if this activity is intermixed with opening And closing of files.

JDL COMMANDS

The JDL commands and their memory requirements for execution in typical circumstances are as shown in Table C-1. Command execution in a terminal partition requires at least an additional 2K bytes.

ALPHABETICAL ORDER

Command	Size
ASSIGN	7K
BATCH	7K
CHAIN	7K
CHANGE	7K
COBOL	28K
COMBINE	3K
CONNECT	10K
CONTINUE	2K
CREATE	9K
DELETE	11K
DIRECTORY	7K
EDITOR	12K
EXECUTE	2K
EXIT	2K
FCOPY	14K
FDUMP	13K
FILE-BACKUP	21K
FILE-RESTORE	22K
FILE-VALIDATE	21K
FLAG-PROGRAM	7K
FLAW-ADU	2K
FLAW-TRACK	2K
FMODIFY	13K
FTS	15K
HALT	2K
I-BOOT	10K
INITIALIZE	7K
INSTALL-SYSTEM	13K
KEY	2K
KILL-PARTITION	2K
KPRINTER	2K
KTASK	2K
LIST	12K
LOAD	3K
LOGIN	9K
LOGOUT	6K
LOOP	2K
MAP-FLAWS	6K
MAP-KEYS	7K
MAP-PROGRAMS	5K
MAP-SYNONYMS	5K
MESSAGE	2K

SIZE ORDER

Command	Size
CONTINUE	2K
EXECUTE	2K
EXIT	2K
FLAW-ADU	2K
FLAW-TRACK	2K
HALT	2K
KEY	2K
KILL-PARTITION	2K
KPRINTER	2K
KTASK	2K
LOOP	2K
MESSAGE	2K
REPEAT	2K
SETCOND	2K
SWITCH	2K
SYNONYM	2K
UNCOUPLE	2K
VARY	2K
COMBINE	3K
LOAD	3K
PARTITION	3K
REPOINT	3K
SYSTEM-SHUTDOWN	3K
UNLOAD	3K
TIME	4K
MAP-PROGRAMS	5K
MAP-SYNONYMS	5K
LOGOUT	6K
MAP-FLAWS	6K
QUIT	6K
RELEASE	6K
REPLACE	6K
SCRATCH	6K
SYSTEM-FILE	6K
ASSIGN	7K
BATCH	7K
CHAIN	7K
CHANGE	7K
DIRECTORY	7K
FLAG-PROGRAM	7K
INITIALIZE	7K
MAP-KEYS	7K

PARTITION	3K	SDUMP	7K
PRINT	11K	SMODIFY	7K
QUIT	6K	RENAME	8K
RECEIVE	15K	SHOW	8K
RECLOSE	16K	CREATE	9K
RELEASE	6K	LOGIN	9K
REMOVE-SYSTEM	13K	SORT	9K
RENAME	8K	TAPE-ASSIGN	9K
REPEAT	2K	CONNECT	10K
REPLACE	6K	I-BOOT	10K
REPOINT	3K	DELETE	11K
SCRATCH	6K	PRINT	11K
SDUMP	7K	VCOPY	11K
SEND	15K	EDITOR	12K
SETCOND	2K	LIST	12K
SHOW	8K	STATUS	12K
SMODIFY	7K	FDUMP	13K
SORT	9K	FMODIFY	13K
STATUS	12K	INSTALL-SYSTEM	13K
SWITCH	2K	REMOVE-SYSTEM	13K
SYNONYM	2K	TEST-SYSDEFIL	13K
SYSTEM-FILE	6K	FCOPY	14K
SYSTEM-SHUTDOWN	3K	FTS	15K
TAPE-ASSIGN	9K	RECEIVE	15K
TEST-SYSDEFIL	13K	SEND	15K
TIME	4K	RECLOSE	16K
UNCOUPLE	2K	FILE-BACKUP	21K
UNLOAD	3K	FILE-VALIDATE	21K
VARY	2K	FILE-RESTORE	22K
VCOPY	11K	COBOL	28K

Table C-1

MEMORY STRUCTURES

The following paragraphs specify the sizes of the various system structures allocated in the user's partition. These are in addition to the size of the program and JDL commands.

Each partition has an overhead area. This overhead area is in addition to the size indicated in the STATUS display or the size specified on the P record. Partitions whose size is zero on the P record have zero bytes of overhead. The shared file partition has an overhead of 26 bytes. All other partitions have an overhead of 1256 bytes.

There are Process Control Blocks (PCBs) allocated for each active terminal or nonterminal partition. The PCB's are allocated in the partition when it is activated, either by log-in to a terminal partition or batch processing initiation in a nonterminal partition. A terminal partition requires 552 bytes for PCBs. A nonterminal partition requires 336 bytes for PCBs. In addition, each partition has a stack area for use by Logical I/O. This stack area occupies 124 bytes.

There is a Physical File Definition Table (PFDT) for each file or device assigned. If the shared file partition size is nonzero, the PFDTs for disk files not assigned with Exclusive All access are placed in the shared file partition. Otherwise the PFDT is in the user's partition. The size of a tape PFDT is 482 bytes; the size of other device PFDTs is 24 bytes. The size of a disk file PFDT is 298 bytes. Note that batch command input files are assigned with Read Only access when a BATCH command is entered; so, if the shared file partition size is nonzero, there must be sufficient remaining storage in the shared file partition to allocate the PFDT for the batch command input file.

There is a Logical File Definition Table (LFDT) in the user's partition for each logical name assigned. The size of a tape LFDT is 82 bytes; the size of other device LFDTs is 46 bytes. The size of a disk file LFDT is 84 bytes. An additional 34 bytes are allocated during COBOL program execution in a terminal partition for ACCEPT and DISPLAY access to the station.

There is a Program File Descriptor Record in the user's partition for each program file assigned. The length in bytes of this structure is 14 + (30 * number of programs in the program file).

For each logical disk file that is open, blocking buffers are allocated in the user's partition. The size of each buffer is (18 + logical block size) bytes. The number of buffers is specified in the RESERVE integer AREAS clause in the COBOL program, or on the CREATE or ASSIGN JDL commands. If not specified, two buffers are allocated to each file, except that sequential or relative files opened INPUT need only one buffer. Indexed files opened I-O or OUTPUT require at least two buffers.

For each tape file that is open, one blocking buffer is allocated in the user's partition. The size of this buffer is (4 + block size) bytes, or (4 + record size) bytes if no block size is specified.

There is a Key Information Table for each physical indexed file that is assigned. This space is allocated in the same partition as the PFDT for the indexed file. A Key Information Table is 196 bytes in length.

For each logical Indexed File that is open, additional memory is allocated. The size of this memory is (logical record length + length of longest key + 4) if open mode is I-O; otherwise the size is (length of longest key + 4).

For each station device defined, a screen buffer is allocated in the corresponding terminal partition. The length in bytes of this structure is 4 + ((number of rows + 2) * number of columns). For a 24x80 terminal, this is 2084 bytes.

SYSTEM MEMORY

Devices are defined during system configuration (see Chapter 2, System Configuration). The specification of any devices of a given type requires an overhead of system memory. Each controller and unit requires its own piece of system memory. The memory required for the CPU controller is allocated during IPL whether or not its controller specification record is included in the System Definition File. Any increase in system memory reduces the amount of user memory available.

GLOBAL MEMORY REQUIREMENTS

The following formulas allow a programmer to determine how much user memory is available on a particular system configuration.

The physical shared file partition size is the amount of memory allocated to the shared file partition. If the partition size of the shared file partition is 0, the physical shared file partition size is 0. Otherwise,

physical shared file partition size =
shared file partition size + 26

The physical partition size is the amount of memory allocated to a terminal or nonterminal partition. If the size of the partition is set to zero at IPL time, the physical partition size is zero. Otherwise,

physical partition size = partition size + 1256

The amount of memory required for the operating system is the sum of the sizes of the vector table, the system stack, the resident system, and the system data structures. The size of the system structures is dependent on the number of partitions and upon the device configuration. The amount of memory available to partitions is:

available user memory = physical memory - system memory

The size of system memory for a particular configuration is not easily computed. An easier technique is to use the STATUS JDL command to determine the available memory. First configure the system with all devices and partitions desired. Since the amount of available memory is not known, set the sizes for most partitions to very small values. IPL the system with this configuration and execute the STATUS JDL command. The amount displayed as "Unallocated space" is the amount of memory not assigned to any partition. The sum of the physical partition sizes of all terminal and nonterminal partitions, the physical shared file partition size, and "unallocated space" is the amount of memory available to partitions.

MINIMUM REQUIREMENTS

The minimum memory a terminal partition must have to permit log-in is approximately 9K bytes. The System Configuration process and PARTITION JDL command ensure that this much memory is available for at least one terminal partition.

The minimum memory a nonterminal partition must have to be used for BATCH processing is approximately 7K bytes plus the memory for the open batch files (see Memory Structures).

The minimum memory for the shared file partition is 300 bytes. This much memory is necessary to hold one disk PFDT, and any attempt to set the size to less than this amount causes the size to be zero. When the shared file partition size is zero, no two users can assign the same file, unless all assigns are for Read Only access.

Approximately 1.5K contiguous bytes must be available in a terminal partition in order to enter interrupt mode. Any attempt to enter interrupt mode with less memory available is ignored until the contiguous memory becomes available.

Besides assuring a large enough partition to log-in again, the PARTITION JDL command requires that approximately 4K bytes of free memory be available in the partition after execution to permit another PARTITION command or a QUIT command.

COMPILER COPY BLOCK SIZE

The COBOL compiler reserves memory for COPY statement file structures at initiation and uses the remaining available memory in the partition for compilation tables. The values considered in determining how much memory is reserved are: (1) the largest logical block size of all the potential copy files assigned logical names, (2) the Copy Block Size parameter and in a terminal partition, (3) the minimum memory required to enter interrupt mode (see above, Minimum Requirements). No memory is reserved for COPY statement processing in a nonterminal partition if the Copy Block Size parameter is explicitly set to zero.

In the following discussion, references to PFDT, LFDT and buffer overhead are the sizes of those data structures for the target system (see above, Memory Structures). Annotated PFDT references below (e.g. PFDT(1)) have a value of zero when a shared file partition is allocated. The term block szie refers to the logical block size of the copy file for which memory requirements are being calculated.

In terminal partitions, the memory reserved is always at least the minimum required to enter interrupt mode. Since this is larger than nominal block sizes it is usually the controlling factor in the amount of memory reserved. However, unless additional memory is reserved beyond what is needed for copy files the compilation cannot be interupted while a copy file is assigned by the compiler since its data structures use the memory needed for interrupt mode. The actual memory size that is compared to the minimum required to enter interrupt mode is calculated by the compiler as the greatest of:

```
Largest-assigned-copy-block + PFDT(1) + LFDT
    + buffer-overhead

Copy-Block-Size + PFDT(1) + LFDT + buffer-overhead

2 * (PFDT + LFDT) + 128 + buffer-overhead
```

These expressions can be restated in terms of the Copy Block Size by subtracting common terms from each expression. Only when the largest block size of the copy files to be assigned by the compiler exceeds all of the following expressions for a terminal partition, or only the first two expressions for a nonterminal partition, does the Copy Block Size need to be specified:

```
Largest-assigned-copy-block

PFDT + LFDT + 128

minimum-interrupt-memory - (PFDT(1) + LFDT
```

+ buffer-overhead)

When copy files contain COPY statments, called nesting, the total memory requirement increases with each level of nesting. The following paragraphs detail how to calculate the memory required for each nested copy file.

Case 1: When a pathname or a synonym is used that resolves to a volume directory level file (e.g. volume.file), the memory required is the greater of the following expressions:

block-size + PFDT(1) + LFDT + buffer-overhead

128 + PFDT + LFDT + buffer-overhead

Case 2: When a pathname or synonym is used that resolves to a subdirectory level file (e.g. volume.directory.file, volume.directory.directory.file, etc.) the memory required is the greater of the following expressions:

block-size + PFDT(1) + LFDT + buffer-overhead

128 + 2 * (PFDT + LFDT) + buffer-overhead

The total memory required for copy file processing is the sum of the requirements for each level in the worst case copy path. The minimum value for the Copy Block Size parameter in that case is:

total-requirements - PFDT(1) - LFDT - buffer-overhead

While using both preassigned and compiler assigned copy files in a single compilation is not forbidden, it is best to use one technique throughout a program to avoid confusion. Allowing the compiler to assign copy files is the more memory efficient method since only one copy file per nesting level is assigned at any time.

Copy Block Size Examples

Assume a system with not shared file partition allocation and compilation in a terminal partition of a program which copies file DISK.A.B. This file copies C and the synonym COPYLIB is assigned the volume name "DISK". That file copies ZFILE and the synonym ZFILE is assigned the pathname ".X.Y.Z". The file DISK.A.B has a block size of 512 bytes, DISK.C has a block size of 256 bytes and .X.Y.Z has a block size of 288 bytes. The minimum memory required to enter interrupt mode is 1512 bytes, PFDT size is 298 bytes, LFDT size is 84 bytes and buffer overhead size is 18 bytes.

The first level COPY (DISK.A.B) corresponds to Case 2 above. The memory required is the greater of:

```
    512 + (298 + 84 + 18) = 912
128 + 2 * (298 + 84) + 18 = 910
```

The second level COPY (@COPYLIB.C) corresponds to Case 1 above. The memory required is the greater of:

```
  256 + (298 + 84 + 18) = 656
    128 + 298 + 84 + 18 = 528
```

The third level COPY (@ZFILE) corresponds to Case 2 above. The memory required is the greater of:

```
    288 + (298 + 84 + 18) = 688
128 + 2 * (298 + 84) + 18 = 910
```

The total memory required is:

```
912 + 656 + 910 = 2478
```

Since this is greater than the minimum memory required to enter interrupt mode (1512 bytes) the Copy Block Size parameter for this example must be at least:

```
2478 - (298 + 84 + 18) = 2078
```

Note that if only the first level COPY existed the memory required would be 912 bytes and the Copy Block Size would be 512 bytes, but the actual memory reserved would be 1512 bytes, the minimum memory required to enter interrupt mode.

If only the first two COPY levels existed then the memory required would be:

```
912 + 656 = 1568
```

Since this is greater than the minimum memory required to enter interrupt mode (1512 bytes) the Copy Block Size parameter in this case must be at least:

```
1568 - (298 + 84 + 18) = 1168
```

APPENDIX D

FILE SIZE CALCULATIONS

Sequential, Relative and Indexed Files

All file allocation is in Allocatable Disk Units (ADUs) even though the allocation parameters on the CREATE command is in records. The system attempts to make worst-case assumptions on the size of the records when converting records to ADUs. This will frequently cause the system to allocate more disk space than is really needed, particularly on compressed files or files with variable length records.

Further complications occur with indexed files. These files have overhead blocks to maintain the index. These blocks are not included in the computation at the time of the CREATE command, because the number of keys and lengths of the keys are not known.

The following formulas include the computations used by CREATE to convert records to ADUs, and to compute the number of records to add to include space for indexed file overhead blocks.

The actual record size is the worst-case amount of space required for records of a given logical record size.

Sequential, uncompressed:

```
actual-record-size =
   ceiling (logical-record-size / 2) * 2 + 4

if the file is type SPOOL, add 8
```

(Formula D-1)

Sequential, compressed:

```
actual-record-size =
   ceiling ((ceiling (logical-record-size / 127) +
      logical-record-size) / 2) * 2 + 4

if the file is type SPOOL, add 10
```

(Formula D-2)

Relative:

```
actual-record-size =
   ceiling (logical-record-size / 2) * 2 + 2
```

(Formula D-3)

Indexed, uncompressed:

```
actual-record-size =
   ceiling (logical-record-size / 2) * 2 + 4
                                          (Formula D-4)
```

Indexed, compressed:

```
actual-record-size =
   ceiling ((ceiling (logical-record-size / 127) +
      logical-record-size) / 2) * 2 + 4
                                          (Formula D-5)
```

For sequential and relative files, there are no restrictions on the minimum logical block size. For indexed files, the block size must be large enough to contain the indexed file overhead, one logical record, or three keys. Thus, the logical block size must be larger than each of the three following expressions:

1) actual-record-size + 8
2) (2 + number-of-alternate-keys) * 12
3) 3 * ceiling ((length-of-longest-key + 6) / 2) * 2 + 6

The actual block size is the amount of space in a logical block that is available for records.

Sequential or Relative:

```
actual-block-size = logical-block-size
                                          (Formula D-6)
```

Indexed:

```
actual-block-size =
   floor ((logical-block-size - 8) /
      actual-record-size) * actual-record-size
                                          (Formula D-7)
```

The block requirement is the number of logical blocks required to contain the requested number of records.

```
block-requirement =
   ceiling ((number-of-records * actual-record-size) /
      actual-block-size)
                                          (Formula D-8)
```

The following calculates the number of ADUs required. This is a function of the disk sector size and the number of sectors in an ADU. A STATUS command for a disk device with a loaded volume will display sector size and sectors per ADU.

```
number-of-ADUs =
   ceiling (block-requirement /
      floor (ADU-size-in-sectors /
      ceiling (logical-block-size /
      sector-size-in-bytes)))
```

(Formula D-9)

The overhead of a key indexed file consists of a tree structure of blocks called Node Blocks. The order of a Node Block is the number of keys which can fit in the block. The following formulas define the various characteristics of a worst-case tree structure.

```
order-of-tree-for-key-i =
   floor ((logical-block-size - 8) /
      ceiling (length-of-key-i / 2) * 2 + 6)
```

(Formula D-10)

```
depth-of-tree-for-key-i <=
   1 + log (base (ceiling (order-of-tree-i / 2))
      ceiling (number-of-records / 2)
```

(Formula D-11)

```
number-of-node-blocks-for-tree-i <=
   ceiling (number-of-records /
      ceiling (order-of-tree-i / 2) - 1)
```

(Formula D-12)

```
number-of-overhead-blocks <=
   1 + sum (number-of-node-blocks-for-each-tree-i)
```

(Formula D-13)

```
number-of-additional-records-to-specify-at-CREATE =
   (number-of-overhead-blocks * actual-block-size) /
      actual-record-size
```

(Formula D-14)

The following formulas may be used in place of the above formulas to approximate the average behavior of an index structure:

```
depth-of-tree-for-key-i =
   1 + log (base (ceiling (.75 * order-of-tree-i))
      ceiling (.75 * number-of-records)
                                                (Formula D-15)

number-of-node-blocks-for-tree-i =
   ceiling (number-of-records /
      floor (.75 * order-of-tree-i))
                                                (Formula D-16)
```

The following example illustrates the creation of a key indexed file containing 1000 compressed records. Each record is 80 bytes long of which, on the average, 22 are blank. There are two keys, one is 5 bytes long and one is 20 bytes long. The logical block size is 256 bytes.

```
actual record size = 80 - 22 + 4 = 62

actual block size = (256 - 8 / 62) * 62 = 248

block requirement = 1000 * 62 / 248 = 250

order of tree 1 = (256 - 8) / (((5 + 1) / 2) * 2 + 6)
                = 20

number of node blocks in tree 1 = 1000 / (.75 * 20) = 67

order of tree 2 = (256 - 8) / (((20 + 1) / 2) * 2 + 6)
                = 9

number of node blocks in tree 2 = 1000 / (.75 * 9) = 166

number of overhead blocks = 1 + 67 + 166 = 234

total number of blocks = 250  + 234 = 484

number of records = 484 * 248 / 62 = 1936
```

```
[ ] CR
LOGICAL NAME: INX-FILE
NAME:
RECORD SIZE: 80
BLOCK SIZE: 288
BUFFERS:
ALLOCATION: 1936
SECONDARY: 100
ORGANIZATION: INX
PRIVILEGE:
DELETE PROTECTION: NO
TYPE: SCRATCH,COMPRESSED
```

Directories

Directory allocation is in Allocatable Disk Units (ADUs) even though the allocation parameter on the DIRECTORY and INITIALIZE commands is the estimated number of entries needed in the directory or volume directory being created. Therefore, the actual number of entries available without expansion beyond the initial allocation may be greater than the estimated number of entries. The following formulas include the computations performed by DIRECTORY and INITIALIZE.

```
est-sectors-required =
   ceiling (est-number-of-entries / 16)
      + est-number-of-entries
```

(Formula D-17)

```
ADUs-allocated =
   ceiling (est-sectors-required / sectors-per-ADU)
```

(Formula D-18)

```
sectors-allocated =
   ADUs-allocated * sectors-per-ADU
```

(Formula D-19)

```
actual-number-of-entries =
   sectors-allocated -
      ceiling (sectors-allocated / 16)
```

(Formula D-20)

APPENDIX E

VENDOR SUPPLIED BATCH STREAMS

EXECUTE COBOL COMPILER

To execute the COBOL compiler in a terminal partition the following JDL command may be used:

```
[ ] B.
NAME: .COBOL
```

To execute the COBOL compiler in a nonterminal partition the following JDL command may be used:

```
[ ] B
NAME: .COBOL
LIST JDL:
LIST NAME:
PARTITION: enter number of nonterminal partition
```

The batch stream begins with the following prompts in sequence:

```
Source name:
Object name:
List name:
```

In response to each prompt, the operator should enter the appropriate file name or device name. A device name of DUMY may be entered if either an object file or a listing file is not desired. If an invalid name is entered, an error message will be displayed on the system error line. After the message has been acknowledged by pressing the Acknowledge Key, the following prompt will appear:

```
Invalid name. Retry? (Y/N): Y
```

Press the Return Key to cause the name prompt to be redisplayed. Press N and then the Return Key to terminate the batch stream and return to JDL interactive mode or, if .COBOL was invoked in a batch stream, to JDL batch mode following the BATCH command in the controlling batch stream (with a condition code of N).

After these prompts have been successfully completed, the two compiler prompts will appear:

```
Compile options:
COPY Block Size (0 if no COPY statements):
```

Press the Return Key if the default options are required, otherwise enter one or more comma separated options and press the Return Key. Press the Return Key to accept the default block size of 512, or enter 0 and press the Return Key if there are no COPY statements in the source. (Refer to Chapter 3, Job Description Language, for a description of the COBOL compiler parameters).

After the compilation completes, one of the following two messages will appear:

Compilation errors occurred

Compilation completed without errors

Press the Return Key to clear the message and return to JDL interactive mode or, if .COBOL was invoked in a batch stream, to JDL batch mode following the BATCH command in the controlling batch stream (with a condition code of Z if errors occurred).

Additional logical names required for COPY statements should be assigned before starting the .COBOL batch stream.

EXECUTE COBOL COMPILER BATCH STREAM

```
! /RELEASE, LOGICAL NAME=<SI,BO,LO,EM>
  /LOOP
Y /SETCOND
  /ASSIGN, LOGICAL NAME=SI, NAME("Source name")=(), ACCESS=RO
Z /SETCOND,
       VALUE("Invalid name. Retry? (Y/N)")=(Y)
Y /REPEAT
  /LOOP
Y /SETCOND
  /ASSIGN, LOGICAL NAME=BO, NAME("Object name")=(), ACCESS=EA
Z /SETCOND,
       VALUE("Invalid name. Retry? (Y/N)")=(Y)
Y /REPEAT
  /LOOP
Y /SETCOND
  /ASSIGN, LOGICAL NAME=LO, NAME("List name")=(LP01), ACCESS=EA
Z /SETCOND,
       VALUE("Invalid name. Retry? (Y/N)")=(Y)
Y /REPEAT
  /ASSIGN, LOGICAL NAME=EM, NAME=.CBLERRTX, ACCESS=RO
Z /SETCOND, VALUE=N
  /COBOL, OPTIONS("Compile options")=(),
          COPY("COPY Block Size (0 if no COPY statements)")=()
  /MESSAGE, TEXT="Compilation completed without errors", ST=0
Z /MESSAGE, TEXT="Compilation errors occurred", ST=0
! /RELEASE, LOGICAL NAME=<SI,BO,LO,EM>
```

EXECUTE LINE EDITOR

To execute the Line Editor in a terminal partition, the following JDL command should be used:

```
[ ] B.
NAME: .EDIT
```

The following prompts will then appear in sequence:

```
List name: ME
```

Pressing the Return Key will direct the output generated by the PRint editor command to the terminal. Enter a device or file name if another listing destination is required. This statement assigns the logical name LO.

```
Name to edit:
```

Enter the pathname of the file to be edited. This statement will assign logical name EDITFILE. If an invalid name is entered, an error message will be displayed on the system error line. After the message has been acknowledged by pressing the Acknowledge Key, the following prompt will appear:

```
Invalid name. Retry? (Y/N): Y
```

Press the Return Key to cause the name prompt to be redisplayed. Press N and then the Return Key to terminate the batch stream and return to JDL interactive mode or, if .EDIT was invoked in a batch stream, to JDL batch mode following the BATCH command in the the controlling batch stream (with a condition code of N).

After successful completion of these prompts, control will be transferred to the Line Editor. (Refer to Chapter 5, Line Editor, for a description of the Line Editor.)

If additional logical names are to be used during the edit session, the assignments should be done before use of the .EDIT batch stream.

EXECUTE LINE EDITOR BATCH STREAM

```
! /REL, LOGICAL NAME=<CI,CO,LO,EWF,EDITFILE>
  /CR,  LOGICAL NAME=EWF, RECORD SIZE=990, ALLOCATION=1,
        SECONDARY=1, BUFFERS=3, BLOCK SIZE=512,
        ORGANIZATION=REL, TYPE=SCRATCH
  /AS,  LOGICAL NAME=CI, NAME=ME, ACCESS=EA
  /AS,  LOGICAL NAME=CO, NAME=ME, ACCESS=EA
  /AS,  LOGICAL NAME=LO, NAME("List name")=(ME), ACCESS=EA
Z /SETCOND, VALUE="Q"
  /LOOP
Y /SETCOND, VALUE=" "
  /AS,  LOGICAL NAME=EDITFILE,
        NAME("Name to edit")=(), ACCESS=EA
Z /SETCOND,
        VALUE("Invalid name. Retry? (Y/N)")=(Y)
Y /REPEAT
  /EDITOR
! /REL, LOGICAL NAME=<CI,CO,LO,EWF,EDITFILE>
```

Notes:

1. The editor work file may be moved from the system disk by using the keyword parameter NAME= on the CR command to specify a different disk.

2. The first and second AS commands may have sequential files as the value of the NAME= parameters, allowing the line editor to operate in a noninteractive mode.

EXECUTE MODIFY USER IDENTIFICATIONS

To execute the modify user identification program in a terminal partition, the following JDL command should be used:

```
[ ] B.
NAME: .MODUSER
```

The following prompt will then appear:

```
List name: ME
```

Pressing the Return Key will direct the output generated by the modify user identification program to the terminal. Enter a device or file name if another listing destination is required. If an invalid name is entered, an error message will be displayed on the system error line. After the message has been acknowledged by pressing the Acknowledge Key, the following prompt will appear:

```
Invalid name. Retry? (Y/N): Y
```

Press the Return Key to cause the name prompt be redisplayed. Press N and then the Return Key to terminate the batch stream and return to JDL interactive mode or, if .MODUSER was invoked in a batch stream, to JDL batch mode following the BATCH command in the controlling batch stream (with a condition code of N).

After successful completion of this prompt, control will be transfered to the modify user identification program. (Refer to Chapter 2, System Configuration, for a description of this program.)

EXECUTE MODIFY USER IDENTIFICATIONS BATCH STREAM

```
  /CHANGE,
        NAME           = .USRDEFIL,
        WRITE PROTECT= NO
  /ASSIGN,
        LOGICAL NAME = USRDEFIL,
        NAME           = .USRDEFIL,
        ACCESS         = EA
  /LOOP
Y /SETCOND
  /ASSIGN,
        LOGICAL NAME = LO,
        NAME("List name") = (ME),
        ACCESS         = EA
Z /SETCOND,
       VALUE("Invalid name. Retry? (Y/N)")=(Y)
Y /REPEAT
  /ASSIGN,
       LOGICAL NAME = MODUSER,
       NAME           = .S$MODUSR,
       ACCESS         = RO
  /EXECUTE,
       PROGRAM        = MODUSER
! /RELEASE, LOGICAL NAME=<USRDEFIL,LO,MODUSER>
! /CHANGE, NAME=.USRDEFIL, WRITE PROTECT=YES
```

EXECUTE SCREEN EDITOR

To execute the Screen Editor in a terminal partition, the following JDL command should be used

```
[] B.
NAME: .SEDIT
```

The following prompt will then appear:

```
Name to edit:
```

Enter the pathname of the file to be edited. If an invalid name is entered, an error message will be displayed on the system error line. After the message has been acknowledged by pressing the Acknowledge Key, the following prompt will appear:

```
Invalid name. Retry? (Y/N): Y
```

Press the Return Key to cause the name prompt to be redisplayed. Press N and then the Return Key to terminate the batch stream and return to JDL interactive mode or, if .SEDIT was invoked in a batch stream, to JDL batch mode following the BATCH command in the controlling batch stream (with a condition code of N).

After successful completion of the prompt, control will be transferred to the Screen Editor. (Refer to Chapter 4, Screen Editor, for a description of the Screen Editor.)

If additional logical names are to be used during the edit session, the assignments should be done before using the .SEDIT batch stream.

EXECUTE SCREEN EDITOR BATCH STREAM

```
!   /RELEASE, LOGICAL NAME=<INPUT,OUTPUT,WORK,P0,P1,PROFILE>
    /LOOP
Y   /SETCOND
    /ASSIGN, LOGICAL NAME=INPUT, ACCESS=EA,
         NAME("Name to edit")=()
Z   /SETCOND, VALUE("Invalid name. Retry? (Y/N)")=(Y)
Y   /REPEAT
    /ASSIGN, LOGICAL NAME=P0, NAME=.C$SUBS,      ACCESS=RO
    /ASSIGN, LOGICAL NAME=P1, NAME=.S$EDITOR,    ACCESS=RO
    /ASSIGN, LOGICAL NAME=PROFILE, NAME=.PROFILE, ACCESS=SH
    /CREATE, LOGICAL NAME=WORK, ALLOCATION=100,
             SECONDARY=100, ORGANIZATION=REL,
             BLOCK SIZE=512, BUFFERS=10, TYPE=SCRATCH
    /SCRATCH, LOGICAL NAME=OUTPUT, TEMPLATE NAME=INPUT
    /EXECUTE, PROGRAM=EDIT
    /REPLACE, SCRATCH NAME=OUTPUT, LOGICAL NAME=INPUT
!   /RELEASE, LOGICAL NAME=<INPUT,WORK,P0,P1,PROFILE>
```

Notes:

1. The editor work file may be allocated on a disk other than the system disk by use of the NAME= parameter on the CREATE command.

SETUP SORT WORK FILES

To setup Sort/Merge work files in a terminal partition, the following JDL command may be used:

```
[] B.
NAME: .SORT
```

The following prompt will then appear:

```
Enter number of work files (3-9): 3
```

Press the Return Key to cause three sort work files to be created. Otherwise enter a number between 3 and 9 inclusive. If an invalid number is entered, the following message will appear:

```
Invalid number of work files
```

Press the Return Key to clear the message and cause the original prompt to be redisplayed.

After successful completion of the prompt, control will return to the JDL interactive mode or, if .SORT was invoked in a batch stream, to JDL batch mode following the BATCH command in the controlling batch stream (with a condition code of blank if no errors and a condition code of Z if errors occurred, such as insufficient disk space).

SETUP SORT WORK FILES BATCH STREAM

```
        /LOOP
        /SETCOND,
           VALUE("Enter number of work files (3-9)")=(3)
#9876543/ME, ST=0, TEXT="Invalid number of work files"
#9876543/SETCOND
        /REPEAT
9       /CR, LOGICAL=SW$9,A=200,S=100,TYPE=[SC,C]
98      /CR, LOGICAL=SW$8,A=200,S=100,TYPE=[SC,C]
987     /CR, LOGICAL=SW$7,A=200,S=100,TYPE=[SC,C]
9876    /CR, LOGICAL=SW$6,A=200,S=100,TYPE=[SC,C]
98765   /CR, LOGICAL=SW$5,A=200,S=100,TYPE=[SC,C]
987654  /CR, LOGICAL=SW$4,A=200,S=100,TYPE=[SC,C]
9876543 /CR, LOGICAL=SW$3,A=200,S=100,TYPE=[SC,C]
9876543 /CR, LOGICAL=SW$2,A=200,S=100,TYPE=[SC,C]
9876543 /CR, LOGICAL=SW$1,A=200,S=100,TYPE=[SC,C]
9876543 /SETCOND
```

The sort work files may be allocated on a disk other than the system disk by use of the NAME= parameter on the CR commands.

INITIALIZE TIME OF DAY

The .TIME batch stream is provided to automatically prompt the user for date and time information. Either the .TIME batch stream should be specified as the initial batch stream of a user identification or the following statement should be the first executable statement in the initial batch stream:

```
T/BATCH, NAME = .TIME
```

If an invalid parameter is specified, an error message will be displayed on the system error line. After the message has been acknowledged by pressing the Acknowledge Key, the TIME prompts will be redisplayed.

INITIALIZE TIME OF DAY BATCH STREAM

```
#T /EXIT
*
*   Set time of day if initial login
*
T  /LOOP
T  /TIME
T  /SETCOND
Z  /SETCOND, VALUE="T"
T  /REPEAT
*
*   Insert further initialization here
*          (such as loading of fixed disks)
*
```

APPENDIX F

BATCH JDL SYNTAX SUMMARY

INTRODUCTION

This appendix contains the batch syntax descriptions for all JDL commands available in the batch mode. The syntax rules are as follows:

* any entity surrounded by square brackets ([ ]) is optional;
* a vertical bar (|) is used to separate items of a list from which one item must be chosen;
* the default that will be used when an optional keyword is not included on a command is underlined; and
* items within angle brackets (<>) are a comma separated list.

See Chapter 3, Job Description Language, for further details with regard to batch stream syntax rules.

NOTE

The following commands are shown with a separate line for each keyword and a comma before the keyword. This convention is used for syntactical clarity only. If multiple lines are to be used for a command within a batch stream, all but the last line of the command must end with a comma or equal sign (see Chapter 3, Job Description Language).

NOTE

Some of the following commands are context sensitive or are too complicated to allow underlining to indicate a default value for an optional keyword.

BATCH JDL SYNTAX SUMMARY

```
/ASSIGN
      ,LOGICAL NAME = lname
      ,NAME = acnm
   [  ,ACCESS = RO ¦ SH ¦ EW ¦ EA ]
   [  ,BUFFERS = int ]


/BATCH
      ,NAME = acnm
   [  ,LIST JDL = YES ¦ NO ]
   [  ,LIST NAME = acnm ]
   [  ,PARTITION = int ]


/CHAIN
      ,NAME = acnm
   [  ,LIST JDL = YES ¦ NO ]
   [  ,LIST NAME = acnm ]
   [  ,RETURN COUNT = int ¦ 0 ]


/CHANGE
      ,NAME = acnm
   [  ,PRIVILEGE = int ]
   [  ,DELETE PROTECTION = YES ¦ NO ]
   [  ,WRITE PROTECTION = YES ¦ NO ]
   [  ,WORK FILE = YES ¦ NO ]
   [  ,AUTO FILE = YES ¦ NO ]
   [  ,EXPANDABLE = YES ¦ NO ]


/COBOL
   [  ,OPTIONS = < [ ANSI ] [ ,DEBUG ] [ ,NOLIST ] [ ,NONE ]
                   [ ,NOOBJ ] [ ,OBJLIST ] [ ,PROCX ]
                   [ ,RESEQUENCE ] [ ,XREF ] > ]
   [  ,COPY BLOCK SIZE = int ]


/COMBINE
      ,NEW PROGRAM FILE = lname
   [  ,EXCLUDE PROGRAMS = < name [ ,name ]... > ]


/CONNECT
      ,LOGICAL NAME = lname
      ,NAME = name
   [  ,TYPE = < [ 2780 ¦ 3780 ¦ USER ] [ ,EBCDIC ¦ ,ASCII ]
                [ ,CRC ¦ ,LRC ]   [ ,ODD ¦ ,EVEN ¦ ,NONE  ] > ]
   [  ,TIMEOUT = int ]
```

```
/CONTINUE
   [ ,PARTITION = int ]


/CREATE
   [ ,LOGICAL NAME = lname ]
   [ ,NAME = acnm ]
   [ ,RECORD SIZE = int ! 80 ]
   [ ,BLOCK SIZE = int ]
   [ ,BUFFERS = int ]
     ,ALLOCATION = int
   [ ,SECONDARY = int ! 0 ]
   [ ,ORGANIZATION = SEQ ! REL ! INX ! PGM ! MDS ! MVD ]
   [ ,PRIVILEGE = int ]
   [ ,DELETE PROTECTION = YES ! NO ]
   [ ,TYPE = < [ SCRATCH ] [ ,SPOOL ] [ ,COMPRESSED ]
               [ ,WORK ] [ ,AUTO ] [ ,SINGLE ] > ]


/DELETE
     ,NAME = acnm
   [ ,DIRECTORY = YES ! NO ]


/DIRECTORY
     ,NAME = acnm
   [ ,EST NUMBER OF FILES = int ! 5 ]
   [ ,EXPANDABLE = YES ! NO ]
   [ ,DELETE PROTECTION = YES ! NO ]


/EDITOR


/EXECUTE
     ,PROGRAM NAME = name
   [ ,OPTIONS = DEBUG ! NONE ]


/EXIT


/FCOPY
     ,SOURCE = acnm
     ,DESTINATION = acnm


/FDUMP
     ,NAME = acnm
     ,MEMBER-BLOCK = string ! int
   [ ,OFFSET = hex ]
   [ ,COUNT = hex ]
```

```
/FILE-BACKUP
      ,BACKUP FILE = lname ! acnm
   [  ,SOURCE NAME = acnm ]
   [  ,CUTOFF DATE = string ! TODAY ]
   [  ,RECORD SIZE = int ]
   [  ,BLOCK SIZE = int ]
   [  ,OPTIONS = < [ AFTER ! BEFORE ] [ ,VALIDATE ]
                 [ ,APPEND ] [ ,NONE ] [ ,PREMOUNT ] > ]


/FILE-RESTORE
      ,BACKUP FILE = lname ! acnm
   [  ,BACKUP NAME = acnm ]
   [  ,DESTINATION NAME = acnm ]
   [  ,OPTIONS = < [ ADD ] [ ,REPLACE ] [ ,TRUNCATE ]
                 [ ,PREMOUNT ] > ]
   [  ,RECORD SIZE = int ]
   [  ,BLOCK SIZE = int ]
   [  ,INDEX = int ! 1 ]


/FILE-VALIDATE
      ,BACKUP FILE = lname ! acnm
   [  ,RECORD SIZE = int ]
   [  ,BLOCK SIZE = int ]
   [  ,OPTIONS = < [ PREMOUNT ] > ]


/FLAG-PROGRAM
      ,NAME = acnm
      ,PROGRAM = name
   [  ,ON-FLAG = < [ PERFORM ] [ ,DIVIDE ] > ]
   [  ,OFF-FLAG = < [ PERFORM ] [ ,DIVIDE ] > ]
   [  ,CODE = int ]


/FLAW-ADU
      ,DEVICE = name
      ,ADUS = < int [ ,int ]... >


/FLAW-TRACK
      ,DEVICE = name
      ,CYLINDER = int
      ,HEAD = int


/FMODIFY
      ,NAME = acnm
      ,MEMBER-BLOCK = string ! int
      ,OFFSET = hex
   [  ,VERIFY =< hex ['] [ ,hex ['] ]... > ]
      ,PATCH = < hex ['] [ ,hex ['] ]... >
   [  ,CHECKSUM = hex ]
```

```
/FTS
      ,LINK LOGICAL NAME = lname
    [ ,MODE = CODED ¦ IMAGE ]


/HALT
      ,PARTITION = int


/I-BOOT
      ,VOLUME = name
      ,DEVICE = name
      ,BOOT FILE NAME = acnm ¦ .SYSFILE
      ,BOOT MEMBER NAME = name


/INITIALIZE
      ,VOLUME = name
      ,DEVICE = name
    [ ,BAD ADU NUMBERS = <  int [ ,int ]...  > ]
    [ ,EST NUMBER OF VCATALOG FILES = int ¦ 5 ]
    [ ,EXPANDABLE = YES ¦ NO ]
    [ ,FORMAT = SINGLE ¦ DOUBLE ¦ MIXED ¦ QUAD ]
    [ ,BYTES PER SECTOR = int ]
    [ ,INTERLEAVE FACTOR = int ]


/INSTALL-SYSTEM
      ,VOLUME = name
      ,DEVICE = name
    [ ,SYSTEM FILE NAME = acnm ¦ .SYSFILE ]
    [ ,JDL FILE NAME = acnm ¦ .JDLFILE ]
    [ ,SYSDEFIL FILE NAME = acnm ¦ .SYSDEFIL ]
    [ ,SYSTEM NAME = name ¦ SYSTEM ]
    [ ,BOOT NAME = name ¦ BOOT ]


/KEY
      ,LOGICAL NAME = lname
    [ ,STARTING COLUMN = < int ¦ 1 [ ,int ]... > ]
      ,KEY LENGTH = < int [ ,int ]... >
    [ ,DUPLICATES ALLOWED = < NO [ ,YES ¦ NO ]... > ]


/KPRINTER
    [ ,DEVICE = name ¦ LP01 ]


/LIST
    [ ,NAME = acnm ]
    [ ,LEVELS TO DISPLAY = int ]
    [ ,DETAIL INFORMATION = YES ¦ NO ]
```

```
/LOAD
      ,VOLUME = name
      ,DEVICE = name


/LOOP


/MAP-FLAWS
   [ ,DEVICE = name ]


/MAP-KEYS
   [ ,LOGICAL NAME = lname ]
   [ ,DETAIL INFORMATION = YES ! NO ]


/MAP-PROGRAMS


/MAP-SYNONYMS


/MESSAGE
      ,TEXT = string
   [ ,STATION = int ]
   [ ,REPLY = YES ! NO ]


/PARTITION
   [ ,PARTITION = int ]
   [ ,SIZE = int ]
   [ ,PRIORITY = int ]


/PRINT
   [ ,NAME = acnm ]
   [ ,DELETE AFTER PRINT = YES ! NO ]


/QUIT
   [ ,LOGOFF = YES ! NO ]


/RECEIVE
      ,LINK LOGICAL NAME = lname
   [ ,LOCAL NAME = lname ! acnm ]
   [ ,REMOTE NAME = < lname ! acnm [, lname ! acnm ]... > ]
   [ ,MODE = CODED ! IMAGE ]


/RECLOSE
   [ ,NAME = acnm ]
```

```
/RELEASE
   [ ,LOGICAL NAME = < lname [ ,lname ]... > ]


/REMOVE-SYSTEM
      ,VOLUME = name
      ,DEVICE = name


/RENAME
      ,OLD = acnm
      ,NEW = acnm


/REPEAT


/REPLACE
      ,SCRATCH NAME = lname
      ,LOGICAL NAME = lname


/REPOINT


/SCRATCH
      ,LOGICAL NAME = lname
      ,TEMPLATE NAME = lname
   [ ,RECORD SIZE = int ]
   [ ,BLOCK SIZE = int ]
   [ ,BUFFERS = int ]
   [ ,ALLOCATION = int ]
   [ ,SECONDARY = int ]
   [ ,ORGANIZATION = SEQ | REL | INX | PGM | MDS ]
   [ ,TYPE = < [   WORK | NOT-WORK ]
               [ , AUTO | NOT-AUTO ]
               [ , COMPRESSED | NOT-COMPRESSED ]
               [ , SPOOL | NOT-SPOOL ]
               [ , SINGLE | NOT-SINGLE ] > ]


/SDUMP
   [ ,DEVICE = name | DSO1 ]
     ,SECTOR = int
   [ ,NUMBER = int | 1 ]


/SEND
      ,LINK LOGICAL NAME = lname
   [ ,LOCAL NAME = < lname | acnm [, lname | acnm ]... > ]
   [ ,REMOTE NAME = lname | acnm ]
   [ ,MODE = CODED | IMAGE ]
```

```
/SETCOND
   [ ,PARTITION = int ]
   [ ,VALUE = string ]


/SHOW
      ,NAME = lname ¦ acnm


/SMODIFY
   [ ,DEVICE = name ¦ DSO1 ]
     ,SECTOR = int
   [ ,BYTE = hex ¦ 0 ]
   [ ,VERIFY = < hex [ ,hex ]... > ]
     ,PATCH = < hex [ ,hex ]... >


/SORT
      ,INPUT LOGICAL NAME = < lname [ ,lname ]... >
      ,OUTPUT LOGICAL NAME = lname
      ,KEY START = < int [ ,int ]... >
      ,KEY LENGTH = < int [ ,int ]... >
   [ ,ASCENDING = < YES ¦ NO [ ,YES ¦ NO ]... > ]
   [ ,KEY TYPE = < name ¦ GRP [ ,name ]... > ]
   [ ,RECORD SIZE = int ]
   [ ,MERGE ONLY = YES ¦ NO ]


/STATUS
   [ ,PARTITION = int ]
   [ ,DEVICE = name ]
   [ ,LOGICAL NAME = lname ]


/SWITCH
   [ ,STATUS = ON ¦ OFF ]
   [ ,SWITCH = < int [ ,int ]... > ]
   [ ,PARTITION = int ]


/SYNONYM
      ,SYNONYM = lname
   [ ,VALUE = string ]


/SYSTEM-FILE
      ,NAME = acnm
      ,FILE TYPE = NEWSFILE ¦ USRDEFIL


/SYSTEM-SHUTDOWN
```

```
/TAPE-ASSIGN
      ,LOGICAL NAME = lname
   [  ,DEVICE = name ]
   [  ,VOLUME = < string [ ,string ]... > ]
   [  ,SCRATCH = < string [ ,string ]... > ]
   [  ,PREMOUNT = string ]
   [  ,SET NAME = string ]
   [  ,POSITION = int ]
   [  ,BLOCK SIZE = int ]
   [  ,RECORD SIZE = int ]
   [  ,FORMAT = FIXED ¦ DECIMAL ¦ SPANNED ¦ UNDEFINED ¦
                VARIABLE ¦ < VARIABLE, SPANNED > ]
   [  ,PADDING = hex ]
   [  ,OFFSET = int ]
   [  ,OPTIONS = BACKUP ¦ NONE  ]


/TEST-SYSDEFIL
      ,VOLUME = name
      ,DEVICE = name
   [  ,TEST SYSDEFIL FILE NAME = acnm ]


/TIME
      ,YEAR = int
      ,MONTH = int
      ,DAY = int
      ,HOUR = int
      ,MINUTE = int
   [  ,SECOND = int ¦ 0 ]


/UNCOUPLE
   [  ,PARTITION = int ]


/UNLOAD
      ,VOLUME = name


/VARY
      ,DEVICE = name
   [  ,STATUS = ON ¦ OFF ]


/VCOPY
      ,SOURCE VOLUME NAME = name
      ,SOURCE DEVICE = name
      ,DESTINATION VOLUME NAME = name
      ,DESTINATION DEVICE = name
```

APPENDIX G

GLOSSARY

INTRODUCTION

The terms in this appendix are defined in accordance with their meaning as used in this document describing RM/COS and may not have the same meaning for other operating systems.

These definitions are also intended to be either reference material or introductory material to be reviewed prior to reading the detailed specifications given in this document. For this reason, these definitions are, in most instances, brief and do not include detailed specifications.

DEFINITIONS

Access name. (See Pathname)

Access type. One of the four following types of access allowed to a device or disk file when the file is assigned to a logical name:

RO - Read Only
SH - Shared
EW - Exclusive Write
EA - Exclusive All

Acknowledge Key. A system key used to inform the system that the operator has seen a system error message displayed on the error line, or that the display of information which fills more than one screen may continue. The specific key on the station keyboard used as the Acknowledge Key is defined in the Terminal Guide.

ADU. (See Allocatable Disk Unit)

Allocatable Disk Unit (ADU). A unit of storage on a disk volume consisting of an integral multiple of physical disk sectors. The system allocates space on disk volumes to files as integral multiples of ADUs. The ADUs allocated to a particular file are not necessarily contiguous on the disk volume.

ASCII. American National Standard Code for Information Interchange, X3.4-1977. The standard code, using a coded character set consisting of 7-bit coded characters (8-bits including parity check), used for information interchange among data processing systems, communications systems, and associated equipment. The ASCII set consists of control characters and graphic characters. Synonymous with USASCII.

Auto file. A file having the AUTO attribute. An AUTO file is intended for use in situations where a file: (1) is open I-O or OUTPUT for long periods of time, (2) requires a high level of data integrity, and/or (3) is so large that a RECLOSE operation is highly undesirable. The file management system simulates an open and close operation for each output operation on an AUTO file, thus reducing the window during which a RECLOSE is necessary in case of system stoppage. The additional physical disk operations required for AUTO files causes output operations to require more time than for equivalent non-AUTO files.

Batch mode. The JDL command mode in which a series of JDL commands are obtained from a file and acted upon sequentially. The batch mode is explicitly initiated by the BATCH command which specifies the file containing the JDL commands. Batch mode is implicitly entered during the log-in process if the user ID has an associated initial batch command file.

Batch stream. A series of JDL commands stored in a file for use in batch mode.

Binary synchronous communications (BISYNC). A uniform discipline, using a defined set of control characters and control character sequences, for synchronized transmission of binary coded data between units in a data communications system. Also abbreviated as BSC.

BISYNC. (See Binary synchronous communications)

BISYNC link. A device for transmission of data across a communications link using binary synchronous communications protocol. BISYNC link devices have a device name with a device prefix of 'BL'.

Bit. A single binary digit which may represent the value 0 or 1.

Block. A collection of contiguous records or partial records recorded as a single unit within a disk or tape file. When used without qualification, block is synonymous with logical block.

Block oriented device. A device which requires that the logical records from a program be combined or divided into physical units for transfer to and from the device. Examples of block oriented devices are disk and tape drives and a BISYNC link. The opposite of a block oriented device is a character oriented device.

Boot. To load the operating system from the system disk volume into the computer memory at initial program load (IPL) time. Boot is short for bootstrap.

Boot device. The disk device which contained the system disk at the time the system was initially loaded into the computer (IPL time). This disk device is the system disk device until the next IPL.

Buffer. An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. As used in this manual, a buffer is usually a block buffer for a disk or tape file; block buffers are allocated only while the file is open. A block buffer is allocated from the user's partition.

Byte. A sequence of eight adjacent binary digits (bits) that are operated on as a unit and that constitute the smallest addressable unit in the system storage. A byte may contain a single character or an integer value in the range 0 to 255, inclusive.

Ceiling. A function defined as: ceiling (x) is the least integer greater than or equal to the real number x.

Character oriented device. A device such that the characters from a program's record area are transferred directly to the device. A terminal or a printer are examples of character oriented devices. The opposite of a character oriented device is a block oriented device.

COBOL. COmmon Business-Oriented Language. A business data processing language. As used in this manual, COBOL refers particularly to that language defined in ANSI X3.23-1974, American National Standard programming language COBOL, 1974.

COBOL file organizations. Any of the three file organizations available to COBOL programs as data files: sequential, relative, and indexed.

Coded mode. A transmission mode for data to be transmitted across a communications link when the data consist only of text; that is, the data contain only ASCII graphics characters and ASCII noncommunication control characters; ASCII carriage control characters such as carriage return, line feed, and form feed are allowed. The prohibited control characters have the following hexadecimal values: 01-06, 0E-10, 15-17, 19, and 1C-1F; if the data contains these character values then image mode must be used to prevent confusion with link control characters. The coded mode requires less overhead than image mode. The table describing the conversion of ASCII characters to EBCDIC in coded mode is found in Appendix J.

Command Key. A system key used to: (1) terminate the log-in process prior to entering a valid passcode; or (2) return to the command prompt after parameter entry has begun while in the interactive or interrupt mode. At all other times the Command Key is treated as any other function key. See the Terminal Guide for the specific terminal key used for the Command Key.

Command prompt. A prompt in interactive or interrupt mode indicating that the system is waiting for a JDL command to be entered by the user. An asterisk in front of the prompt indicates interrupt mode. The command prompt is normally a pair of brackets ([ ]) but may be different on international terminals or in cases where a different command prompt has been specified in the System Definition File.

Compiler. A program which prepares an executable object program from a computer source program written in a programming language which is not native to the computer upon which it is to be executed.

Compressed file. A disk file with the COMPRESSED file type attribute. A compressed file has two or more consecutive blanks or binary zeroes replaced by a single character and three or more other consecutive repetitive characters replaced by two characters when the logical records are written to the file. The logical records are decompressed to their original form when the file is input. Compression reduces the use of disk storage by the file and also reduces transfer time between disk storage and system storage since fewer characters need be transferred.

Condition code. A single character associated with each partition which can be used to control the execution of a batch stream executing in the partition. Batch JDL commands can be conditionally executed based on the current setting of the condition code in the partition in which the batch is executing. The condition code is set to blank when batch processing first begins in a partition. The condition code is unchanged by a JDL command process which terminates normally but is set to 'Z' by a JDL command process which terminates because of an error. The condition code may be explicitly set by the JDL commands SETCOND and MESSAGE (with REPLY) or by calling the C$SCC subroutine from a COBOL program.

Configuration. (See System configuration)

Console terminal. Either the terminal which receives messages during the system IPL procedure, or the lowest numbered station (STO1). Usually these are both the same terminal.

Control characters. Characters from a character code set used to specify control information as opposed to characters which specify text (see graphic characters). Examples of control characters are line feed, form feed, carriage return, and negative acknowledge.

Controller. A hardware board which interfaces the CPU to one or more devices.

CPU. Central Processing Unit; that portion of the computer which executes instructions.

Cross support. A set of programs and procedures which execute under an operating system other than RM/COS, but which provide certain RM/COS functions while using that operating system. The Operator Guide describes the cross support available, if any.

Data item. A character or a set of contiguous characters defined as a unit of data by a COBOL program.

Data set. A term used by IBM which is synonymous with an RM/COS file. Data set is also used in communications as synonymous with modem, but is not so used in this document except within terms such as 'data set ready'.

Data Set Ready. A line defined in EIA RS-232-C which when on indicates that the device, usually a modem, has its power on, is not in voice mode or test mode, and is ready to transfer data.

Data Terminal Ready. A line defined in EIA RS-232-C which when on indicates that the device, usually a modem, is to maintain the connection established by external means (e.g., manual call origination, manual answering, or automatic call origination). The off condition causes the data communication equipment to be removed from the communication channel following the completion of any in process transmission.

Device. An input, output, or input/output hardware unit attached to the computer. Examples of devices are disks, line printers, stations, and communications links.

Device name. A unique four character identifier comprised of a device prefix and a device suffix which identifies a particular device connected to the computer. Device names are bound to specific devices by the system configuration process (see System Definition File).

Device prefix. The two character prefix of a device name which identifies a device class:

AH - (ad hoc) special device
BL - BISYNC link device
DS - disk device
LP - line printer device
MT - magnetic tape device
ST - station device
UL - user link device

Device Service Routine (DSR). A program which handles a particular hardware device for the system by providing a standardized interface for making I/O requests to the device.

Device suffix. A suffix of a device name consisting of two decimal digits. The suffix indicates a particular device within the class of devices indicated by the associated device prefix. In addition, for station devices, the device suffix indicates the partition number of the partition to be used by the station.

Directory. (1) An index which is used by the system to find a particular file among the set of files contained on a disk volume; (2) an index which is used by the system to find a particular program among the set of programs contained in a program file; or (3) an index which is used by the system to find a particular member among the set of members contained in an MDS file.

Directory file. A disk file with directory organization. Directory files are not available as data files to COBOL programs; only JDL commands may manipulate directory files.

Directory organization. The permanent logical file structure which groups a set of files by file name and contains FDEs which describe the named files. Directory organization is established by the DIRECTORY command.

Disk. A mass storage device. Disk device names have a device prefix 'DS'.

Disk volume. The storage area available on a single disk device. Some disk volumes are removable from the device and therefore the same device may contain different volumes at different times. Removable disk volumes are often called disk packs or cartridges if rigid and diskettes if flexible.

Dope. A data item descriptor which provides information about the data item needed during execution of a program. Such information includes the type, length, location, and editing requirements of the data item.

DSR. (See Device Service Routine)

DUMY. Device name of the dummy device. The dummy device is always provided by the system; it is not configured by the user. Sequential I/O operations are allowed on DUMY. The system returns an end of file status on reads and discards data written to DUMY. Any number of logical names may be assigned to DUMY, with any access.

EBCDIC. Extended Binary Coded Decimal Interchange Code. EBCDIC is the preferred code for IBM and IBM compatible machines. EBCDIC is an eight-bit code containing both control characters and graphic characters.

EIA. (See Electronic Industries Association)

EIA RS-232-C. The standard for Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Interchange. Copies of this standard may be obtained from EIA Engineering Department, Electronic Industries Association, 2001 Eye Street, N.W., Washington, D.C. 20006.

Editor. A utility program which performs operator specified modifications to data files. Within this document, an editor is specifically a text editor for modifiying files containing text such as COBOL source programs and JDL batch streams.

Electronic Industries Association (EIA). A standards organization specializing in the electrical and functional characteristics of interface equipment.

End of volume mark. A specially recorded sequence on tape provided by some tape controllers to indicate the end of a file section that is not the end of the file. When an end of volume mark is encountered on input, the input operation continues with the next logical track or volume.

Error line. The rightmost columns of the bottom line of the terminal screen. The error line is used to display system error messages (e.g., command syntax) and system request messages (e.g., volume removal). When a system message is displayed on the error line, the user must acknowledge receipt of the message by pressing the Acknowledge Key.

Expandable file. A disk file which has a nonzero secondary allocation value and which can therefore expand beyond its current allocation of disk space when that space is exhausted as the result of output to the file.

Expert mode. A variation of JDL command input allowed in interactive and interrupt mode which allows specification of parameter values in conjuction with the entering of the JDL command. The syntax and semantics for expert mode are the same as for batch mode executing from a terminal partition except that the leading slash character is not entered.

Extent. A contiguous area of storage on a disk volume allocated to a particular disk file. On RM/COS format disk volumes, a disk file may have up to 47 discontiguous extents. An extent's area of storage is contiguous in that the extent consists of consecutive sector addresses. Interleaving of sectors on a disk track may cause the physical space occupied by an extent to be discontiguous.

External name. The name by which a file is known external to a COBOL program as opposed to the file-name used to reference the file in COBOL statements. Under RM/COS the external name is the logical name assigned to the file.

FDE. (See File Description Entry)

FIFO. (See First-in/first-out)

File. A collection of related records treated as a unit. In this document, a file refers to a disk or tape file, but the system allows other devices to be treated as a sequential file consistent with the device's capabilities (i.e., a line printer is output only).

File count. The count of the number of file sections on a logical track of tape. On most hardware, when reading unlabelled tape, the file count of each logical track on each volume must be specified when the tape device is assigned. For a particular tape controller, the Operator Guide will indicate whether a file count is required or illgeal.

File Description Entry (FDE). A data structure maintained within directory files which describes the disk file indicated by the associated file name in the directory. The description includes: the file organization, the file type attributes, the file's privilege level, the file's logical record and block sizes, the file's creation and last update dates, file organization dependent attributes, and a map of the ADU's allocated to the file.

File name. A one to eight character identifier for a disk file which begins with a letter and is comprised of only letter, digit, hyphen, and dollar sign characters. File names are contained within directory files along with an associated data structure (FDE) which describes the named file. File names are used in pathnames to access specific files.

File organization. The permanent logical structure established at the time a file is created. The system supports the three COBOL file organizations: sequential (SEQ), relative (REL), and indexed (INX). The multivolume disk (MVD) organization, used for file backup and restore operations, can also be used as a COBOL sequential file. Additionally, the system supports directory (DIR), program (PGM), and multipartite direct secondary (MDS) organization files for system use. Directory files are created by the DIRECTORY command; all the other file organizations are created by the CREATE or SCRATCH commands.

File section. The portion of a multivolume disk organization file which resides on a single volume. The portion of a tape file which resides on a single volume or logical track. A file which does not span volumes or logical tracks consists of only one file section.

File set. A collection of one or more files recorded consecutively on a sequence of one or more tape volumes. For all but the last tape volume of the sequence, the last file on one volume is continued as the first file section on the next volume.

File type. An attribute of a disk file specified when the file is created. A disk file may have more than one type attribute and some of the attributes may be changed after the file is created. The attributes are: COMPRESSED, SCRATCH, SINGLE, SPOOL, AUTO and WORK.

First-in/first-out (FIFO). A method of queue management in which the first item placed in the queue is the first item removed from the queue. FIFO queues are sometimes called circular stores.

Floor. A function defined as: floor (x) is the greatest integer less than or equal to the real number x.

Fragmentation. The division of storage, either memory or mass storage, into many small available areas which occurs when multiple processes are reserving and releasing portions of the storage area in a random fashion. In severe cases, the storage may become so fragmented that a request to reserve storage cannot be satisfied because there is no contiguous area large enough to satisfy the request even though a large amount of storage is available.

Function key. Any of the several keys on a terminal keyboard which are used to indicate actions to be taken rather than graphic characters to be entered as data.

Graphic characters. Characters from a character code set used to specify text as opposed to characters which specify control information (see control characters). Examples of graphic characters are the displayable letter, digit, and punctuation characters. Graphic characters should not be confused with block graphics available on some display terminals.

I-O. The COBOL open mode which is established by the execution of an OPEN I-O statement. The I-O open mode is the only mode in which records may be deleted or rewritten. Its use therefore usually indicates that file updating activity will occur.

I/O. An abbreviation for input/output.

Image mode. A transmission mode for data to be transmitted across a communications link when the data consist of nontext; that is, the data contain characters which are used for communication link control. Data files which contain binary or packed decimal data (COBOL usages COMP-1 or COMP-3) and program files are examples of nontext files. The image mode requires more overhead than coded mode to prevent nontext characters in the data from being interpreted as link control characters. Image mode must be used when the data contain any of the hexadecimal character values: 01-06, 0E-10, 15-17, 19, and 1C-1F.

Indexed file. A disk file with indexed organization.

Indexed organization. The permanent logical file structure in which each record is identified by the value of one or more keys within that record.

Index sector. A reserved sector on a disk volume used to record information regarding the volume when it is initialized to RM/COS format. The index sector is always sector four.

Index track. A set of reserved sectors which, depending on the disk type, may be a partial track or multiple tracks. The index track contains the system boot program, the index sector, the bad ADU map, and the allocated ADU map.

Initial disk allocation. The number of allocatable disk units (ADUs) allocated to a disk file when it is initially created. The user specifies a value in terms of logical records in the CREATE command and the system converts this number to the number of ADUs required. The system attempts to allocate the initial disk allocation contiguously, but if this attempt fails the system attempts to allocate two or more extents which satisfy the initial disk allocation requirement.

Initial Program Load. The process of loading the operating system into the computer. It consists of mounting the system disk in the system disk drive and executing the built-in loader program. Successful IPL automatically results in the system disk volume being a loaded volume. The initial loading procedure is often referred to as "booting the system". Terminating the operating system is called "shutdown".

Input procedure. A COBOL procedure, to which control is given during the execution of a SORT statement, for the purpose of controlling the release of specified records to be sorted.

Interactive mode. The JDL command mode in which commands are entered from a station keyboard by the user. A terminal partition enters interactive mode following successful log-in unless the user ID has an associated initial batch stream. A terminal partition also returns to interactive mode whenever a batch processing completes, unless the batch processing executes a QUIT command specifying LOGOFF=YES. Interactive mode exists only for terminal partitions and is also called terminal mode.

Interleaving. The assigning of consecutive sector addresses to nonconsecutive sectors within a disk track. The purpose of interleaving is to minimize rotational delay caused when processing time required between sectors would cause missing a physically consecutive sector until its next rotation under the disk head.

Interrupt Key. A system key used to interrupt a batch stream, a JDL processor, or an executing COBOL program. The Interrupt Key actually requires the simultaneous depression of at least two keys in order to make inadvertent interruption unlikely. See the Terminal Guide for the specific keys used for the Interrupt Key.

Interrupt mode. The JDL command mode in which an executing JDL command or COBOL program has been suspended so that JDL commands may be entered by the user. The suspended command or program may be CONTINUEd or EXITed depending on the commands entered during interrupt mode. Only a subset of JDL commands are allowed during interrupt mode. An asterisk preceding the command prompt characters indicates that the command mode is interrupt mode. The interrupt mode is entered by use of the Interrupt Key.

IO. An abbreviation for input/output.

IPL. (See Initial Program Load)

JDL. (See Job Description Language)

JDL command mode. Any one of the modes for entering JDL commands and command parameter values: interactive, interrupt, or batch. For experienced users of RM/COS, the expert mode of JDL command input may be used as a variation of interactive or interrupt mode. The JDL Definition File controls which JDL commands are available for use in each JDL command mode. Each command description indicates the modes in which the command is normally allowed.

JDL Definition File (.JDLDEFIL). A system file used during IPL which defines the characteristics of JDL commands for the system. Characteristics of JDL commands controlled by this file are: (1) the minimum privilege level required to use each command; (2) the availability of each command in the various JDL command modes; and (3) which commands may be used before the time has been initialized. A user with a privilege level greater than or equal to the JDL Definition File's privilege level may edit the file to make changes which will take effect at the next IPL.

JDL Image File. The system file with MDS organization whose members are the JDL command processor images. The file is provided on the standard system disk. The INSTALL-SYSTEM command is used to declare the pathname of the file to be used as the JDL Image File; the default pathname is .JDLFILE. The JDL command processor images are loaded from this file into a partition depending on the commands entered from the partition.

JDL prompt. Any one of the prompts for specific parameter values required by a JDL command. JDL prompts are given as English language text displayed on the system line during interactive or interrupt JDL command mode. The same prompts are used as keywords in batch JDL mode to indicate the parameter value being supplied.

Job Description Language (JDL). A language which provides a means of communicating with and controlling the actions of the operating system. JDL consists of a set of commands and each command has a set of zero or more parameters for which values are provided by the user.

Key. (1) A data item which identifies the location of a logical record in a relative or indexed file; (2) a set of data items which serve to identify the ordering of data for sorting or merging; or (3) a depressible switch on a terminal keyboard used to enter a data character or cause a function.

Key Information Table (KIT). An area of storage reserved whenever an indexed organization file is assigned by a logical name. The area is used by the system to describe keys defined for the indexed file. The KIT is allocated in the same partition as the PFDT for the indexed file.

KIT. (See Key Information Table)

Labelled tape. Tape reels which contain tape labels. At this time, labelled tape may be read only by treating the tape as unlabelled. The user must specify the characteristics of the file to the TAPE-ASSIGN command, and must specify the unlabelled tape position. The first file section of a labelled tape is unlabelled position 2, the second file section is unlabelled position 5, and the nth file section is unlabelled position (3 * n - 1). Each file section of a multireel labelled file must be read separately.

LFDT. (See Logical File Description Table)

Line editor. A text editor for modifying lines (logical records) of a file of text. A line editor uses a concept of a line pointer which points to the current line being edited. A set of editor commands provide for moving the line pointer and making edits to the current line. When used in this document line editor refers specifically to the editor described in Chapter 5, Line Editor. An advantage of the line editor is that commands may be entered from a command file when one set of changes apply to several files; the line editor also accepts commands interactively from a terminal.

Line printer. A device for printing character graphics as lines on a form with vertical forms control. Line printer device names have the device prefix 'LP'.

Loaded volume. A RM/COS format disk volume which has been mounted in a disk device known to the system and the presence of the volume has been successfully declared to the system by the LOAD command. The successful execution of an UNLOAD command returns a volume to the mounted, but not loaded, state. The system disk is automatically loaded during IPL and cannot be unloaded.

Log-in. The process by which a user activates a station for use of the system. The log-in process identifies the user and establishes his privilege level. After successfully logging-in, the command prompt is displayed or batch processing is initiated on the initial batch stream associated with the user ID.

Log-in key. A system key used to initiate the log-in process. On most terminals, the Return Key is the Log-in Key. See your Terminal Guide if the Return Key does not initiate the log-in process.

Log-off. (See Log-out)

Log-out. The process of inactivating a station which has previously been activated by the log-in process. The LOGOUT or QUIT command is used to log-out.

Logical File Description Table (LFDT). An area of storage reserved whenever a logical name is assigned to a device or file. The area is used in controlling logical I/O operations. The LFDT is allocated from the user's partition.

Logical block. A collection of contiguous records or partial records recorded as a single unit within a disk file.

Logical name. A one to eight character identifier which begins with a letter and is comprised of letter, digit, hyphen, and dollar sign characters. A logical name is assigned to a device or file to provide access to that device or file for logical I/O operations. The CONNECT command is used to assign a logical name to a communication link device; the TAPE-ASSIGN command is used to assign a logical name to a tape device; and the ASSIGN command is used to assign a logical name to a disk file or other devices. A logical name is the external name used by a COBOL program to access RM/COS devices or files.

Logical record. A collection of contiguous data items treated as a unit for I/O operations on a file.

Logical track. One or four tracks of a tape cartridge, depending on the tape controller hardware. If a tape mark must be written at the end of each track, the tape cartridge is said to have four logical tracks. If the controller will switch from one track to the next without a tape mark being written, then the tape cartridge is treated as one logical track. A nine track tape reel is considered to have one logical track.

Mass storage. A storage medium on which data may be organized and maintained in both a sequential and nonsequential (random) manner. The primary example of mass storage devices are disk devices.

MDS file. A disk file with MDS organization.

MDS organization. The permanent logical file structure in which a set of sequentially organized members are collected into one file along with a directory which associates member names with the locations of the members within the file.

ME. An identifier which may be used as a device name parameter to a JDL command whenever the user wishes to refer to the terminal device name associated with the partition in which the process is executing.

Member. A subdivision of an MDS file which is named. A member is equivalent to a sequential file, except that a logical name may not be assigned to it.

Member name. A one to eight character identifier which begins with a letter and consists of letter, digit, hyphen, and dollar sign characters and is optionally followed by a slash (/) or period (.) and a modifier value. A modifier value is specified as a hexadecimal integer with or without a leading greater than sign (>). A member name identifies a member of an MDS file.

Merge. The process of combining two or more identically sequenced files on a set of specified keys into an output file such that the output file is sequenced on the same set of keys. The keys must be contained within the logical records of the file.

Modem. A modulator-demodulator device that allows data to be transmitted over communications circuits. A modem is sometimes called a data set.

Mounted volume. A disk volume which is mounted on a disk device but which has not necessarily been declared to the system by a LOAD command or by virtue of its being the system disk.

Multipartite direct secondary. (See MDS organization)

Multivolume sequential disk. (See MVD organization)

MVD file. A disk file with MVD organization.

MVD organization. The permanent logical file structure, similar to sequential organization, in which a record is identified by a predecessor-successor relationship established when the record is placed into the file. Unlike sequential files, MVD files may consist of multiple file sections (q.v.), each on a different disk volume.

MYLP. An identifier which may be used as a device name parameter to a JDL command whenever the user wishes to refer to the printer device connected to the terminal indicated by the name ME.

News File (.NEWSFILE). A system file displayed on the screen as the initial display during log-in. The file may contain zero records to indicate a blank display, but the file must be present on the system disk. Only the first twenty-three records are displayed and any additional records in the file are ignored.

Non-mass storage. A storage medium on which data may be organized and maintained only in a sequential manner. Examples of such devices are card readers, line printers, magnetic tape drives, and terminals.

Nondirectory file. A disk file which is not directory organization and which therefore does not contain file names and pointers to other files. When a nondirectory file name occurs in a pathname, it must be the rightmost name. Nondirectory files may be program files, indexed files, relative files, sequential files, or MDS files.

Nonterminal partition. A partition which is not associated with a terminal. Nonterminal partitions may be used to execute batch streams which do not need to interact with a user.

Output procedure. A COBOL procedure, to which control is given during execution of a SORT statement after the sort function is completed or during the execution of a MERGE statement after the merge function has selected the next record in merged order, for the purpose of returning sorted or merged records to the program for disposition.

Partition. A segment of the available user memory allocated for use by a single process. RM/COS supports multiple processes, each executing in a separate partition. There are four partition types: terminal, nonterminal, shared file, and unused.

Partition number. A number, assigned to partitions by their order of definition in the System Definition File, which is used throughout the system to reference a particular partition. The partition number of a terminal partition is always the same as the device suffix of the associated terminal. Terminal partitions are numbered beginning with 1; nonterminal partitions are numbered beginning with 101; the shared file partition is numbered 999. Most JDL commands interpret partition number zero as identifying the partition in which the command is executing.

Passcode. An alphanumeric string of zero to eight characters associated with each user ID. The passcode must be correctly entered during log-in to validate the user ID and allow log-in to continue. The passcode is associated with the user ID by the User Definition File.

Pathname. An identifier of an actual device or disk file which consists of: (1) a nondisk device name; (2) a disk device name and a series of one or more file names, all separated by periods; (3) a volume name and a series of one or more file names, all separated by periods; or (4) a period followed by a series of one or more file names, each separated by periods (implicit reference to the system disk volume). When a series of file names occurs in a pathname, all but the rightmost name in the series must be names of directory files.

PCB. (See Process Control Block)

PFDR. (See Program File Descriptor Record)

PFDT. (See Physical File Description Table)

Physical block. A collection of characters written to or read from tape as a single unit. A physical block may contain one or more logical records. A block may contain segments of one or more spanned records. In general, tape blocks are separated by an interblock gap.

Physical File Description Table (PFDT). An area of storage reserved whenever a file or device is assigned to one or more logical names. This area of storage is used in controlling access to the file or device. If the shared file partition size is nonzero, the PFDTs for disk files not assigned with Exclusive All access are allocated from the shared file partition; in all other cases the PFDT is allocated from the user's partition. If there is insufficient space in the partition in which the PFDT is allocated, the file assignment fails, even if sufficient space exists in the user's partition when allocation is in the shared file partition.

Physical record. The smallest addressable unit of storage on a disk volume. A physical record is equivalent to a sector. Logical blocks must begin on a physical record boundary, but logical block sizes may be larger than physical record sizes.

Priority. A number in the range 1 to 65535 inclusive associated with each partition which determines what fraction of the available CPU time a process executing in that partition receives. The initial priority associated with each partition is established by the System Definition File during IPL, but may be changed after IPL by the PARTITION command.

Privilege level. A number in the range 0 to 65535 inclusive associated with each user ID which determines the privileges available to that user ID. Each JDL command has a privilege level required for its use as established by the JDL Definition File during IPL. Each disk file also has a privilege level associated with it when the file is CREATEd or CHANGEd; a user cannot assign a file which has a greater privilege level than that of the user ID and therefore cannot access the file.

Process Control Block (PCB). An area of storage reserved whenever a terminal or nonterminal partition is activated. The area is used as a register save area and a stack area for processes executing in the partition. The PCB is allocated from the user's partition.

Program file. A disk file with program organization. Program files are used to contain COBOL object programs, but are not available as data files to COBOL programs; only JDL commands may manipulate program files.

Program organization. The permanent logical file structure in which one or more executable object programs are stored.

Program File Descriptor Record (PFDR). An area of storage reserved whenever a program file is assigned to a logical name with other than Exclusive All access. The area is used in controlling loading of the program or programs contained in the program file. The PFDR is allocated from the user's partition.

Record. A unit of data involved in an input or output operation. When used without qualification, record is synonymous with logical record.

Record lock. A system provided lock for logical records of a shared file when the file is open in the I-O mode. Record locking allows multiple processes executing at the same time to update a shared file in an orderly fashion. When one process reads a record from the shared file, the system locks the record such that other processes may not obtain the same record. Thus, the first process to read the record can inspect the data items in the record, modify some, and rewrite the record to the file. When the record is rewritten the lock is released and other processes may then obtain the record.

Relative file. A disk file with relative organization.

Relative organization. The permanent logical file structure in which each record is uniquely identified by an integer value greater than zero, which specifies the record's ordinal position in the file.

Reqest to send. A line defined in EIA RS-232-C which is used to condition the local data communications equipment for data tranmission and, on a half-duplex channel, to control the direction of data transmission of the local data communication equipment.

On one way only channels or duplex channels, the on condition maintains the data communication equipment in the transmit mode. The off condition maintains the data communication equipment in a non-transmit mode.

On a half-duplex channel, the on condition maintains the data communication equipment in the transmit mode and inhibits the receive mode. The off condition maintains the data communication equipment in the receive mode.

Roll. (1) A stack data structure such as is used by the COBOL compiler and the JDL command interpreter; or (2) the action of repositioning the display window in the screen editor by a fixed number of lines from its current position, either toward the beginning or end of file (synonymous with scroll).

Roll memory. An area of storage used to contain rolls.

Roll pointer. A pointer into a table maintained as a roll.

Roll value. The number of lines by which the screen editor display window is repositioned.

Scratch file. A disk file with the SCRATCH file type attribute. A scratch file is a temporary file which exists only while it is assigned to a logical name. The file is automatically deleted by the system when the logical name is released from the file. A scratch file is always treated as if it also had the WORK file type attribute.

Screen editor. An interactive text editor which supports text file modifications by displaying current file lines and then allowing overtyping of modifications, line deletion, and line insertion. When used in this document, screen editor refers specifically to the editor described in Chapter 4, Screen Editor.

Scroll. The action of repositioning the display window in the screen editor by a fixed number of lines from its current position, either toward the beginning or end of file.

Secondary disk allocation. The number of allocatable disk units (ADUs) to be allocated to a file each time the disk space currently allocated to the file is exceeded. The user specifies this number in terms of logical records when the file is CREATEd and the system converts this number to the number of ADUs required. When a file is expanded, the secondary disk allocation is not necessarily obtained contiguously as a single extent nor is it necessarily contiguous with previous allocations to the file, but the system first attempts to do so.

Sequential file. A file with sequential organization.

Sequential organization. The permanent logical file structure in which a record is identified by a predecessor-successor relationship established when the record is placed into the file.

Shared file partition. The partition reserved for allocating data structures of disk files which are not assigned exclusively to a single user, i.e., that are assigned with Read Only, Shared, or Exclusive Write access.

Shutdown. The orderly process of terminating the operating system. See the SYSTEM-SHUTDOWN command.

Single file. A disk file with the SINGLE file type attribute. A single file is allocated contiguous storage on the disk volume (i.e., the file has only one extent).

Sort. The process of sequencing one or more files on a set of specified keys into an output file such that the output file is sequenced on the specified set of keys. The keys must be contained in the logical records of the file.

Sort work file. A sequential organization file required for intermediate storage when using the SORT JDL command or the SORT COBOL statement. Sort work files are recognized by the system as having logical names of the form SW$n, where n is a single decimal digit. At least three such files must be assigned prior to initiating a sort; a batch stream is provided with the system for assigning sort work file logical names to scratch files (see Appendix E, 'Vendor Supplied Batch Streams).

Spool file. A disk file with the SPOOL file type attribute. A spool file preserves vertical forms control information specified at the time logical records are written into the file. When a spool file is copied to a line printer device, the vertical forms control information is used to control the printer accordingly. If logical records are written to a nonspool disk file using vertical forms control, such as the WRITE...ADVANCING statement in COBOL, the vertical forms control information is lost and when the nonspool file is copied to a line printer, only single spacing of logical records is provided. JDL commands which produce listings use vertical forms control.

Station. (See Terminal)

Storage. A device which can store and retrieve data. When used without qualification, storage refers to the computer's main memory. Storage is also used to refer to devices such as disk (see mass storage).

Subdirectory. A directory which is not the volume directory and is therefore pointed to by an entry in another directory (either the volume directory or another subdirectory).

Switch. Any one of eight software switches associated with each partition. The switches may set to ON or OFF by use of the SWITCH command. Switches may be tested for an ON or OFF status by COBOL programs.

Synonym. A string of characters, usually short, which can be used in place of another string of characters, usually longer. The system provides substitution of synonym values for the synonym under rules explained in this manual. Synonyms may be used to parameterize generic batch streams based on synonyms set to specific values by the user prior to initiating the batch. Synonyms are particularly useful as the leading portion of a pathname.

Synonym file. A sequential organization disk file used to preserve the values of synonyms, defined by a user, from the time the user logs-off until he next logs-in. When a user logs-in, his synonym table area is initialized from the synonym file, restoring the definitions in effect when he last logged-off. The logical record size of a synonym file must be 78 characters. The pathname of the synonym file is established by the user ID entry in the User Definition File.

System configuration. A process of tailoring the factory delivered system to the requirements of a specific installation. System configuration allows: (1) definition of the number and size of partitions; (2) the selection of a subset of the standard JDL commands and the privilege level required for use of each command; (3) specification of devices (6) definition of the log-in initial display; and (7) definition of the valid user IDs, user privilege levels, and other user characteristics.

System Definition File. A system file used during IPL to configure the system. The file specifies the number and size of partitions, defines the devices, and selects certain system functions. A user with a privilege level greater than or equal to the System Definition File's privilege level may edit the file to make system configuration changes which will take effect at the next IPL. The pathname of the System Definition File is declared by use of the INSTALL-SYSTEM command; the default pathname is .SYSDEFIL.

System disk. A disk which contains the set of system files and on which a system has been established by the use of the INSTALL-SYSTEM command. In the context of the usage of the RM/COS operating system the system disk is a specific system disk volume used during the IPL procedure. The system disk is automatically loaded during IPL and cannot be unloaded; however, some JDL processors allow the system disk to be temporarily dismounted and another disk mounted in the system disk device during operation of the command. RM/COS allows other system disk volumes to be loaded as data disks in any available disk device, but such system disk volumes do not become the system disk until they are used at IPL time.

System file. One of several disk files provided by the factory on a system disk which are required by the system during its operation. Some of the system files are marked with the system file attribute to prevent their invalid manipulation by the use of JDL commands, but the use of the term system file is not restricted to files which are marked with the system file attribute. Examples of system files are: System Image File, JDL Image File, System Definition File, JDL Definition File, User Definition File, and News File.

System Image File. The system file with MDS organization whose members are the system boot image and the operating system image. The file is provided on the standard system disk. The INSTALL-SYSTEM command is used to declare the pathname of the file to be used as the System Image File; the default pathname is .SYSFILE. The images from this file are loaded into memory at IPL time.

System key. Any of the several special keys used to interact with the system: Acknowledge Key, Command Key, Interrupt Key, and Log-in Key. See the Operator Guide for the specific keys which cause these functions.

System line. The bottom row of the terminal screen used for entering JDL commands and JDL command parameter values in the interactive and interrupt modes of JDL input. Messages sent to the terminal by the MESSAGE command are also displayed on the system line.

System synonym. Any of the special synonyms set by the system to values dependent on information managed by the system. The system synonyms are

$PART = three digit partition number

$TYPE = T if terminal partition
N if nonterminal partition

The value of system synonyms may not be modified. System synonyms are defined only for partitions with a system synonym table area.

System synonym table area. An area of storage reserved for the recording of system synonyms and their associated values. The system synonym table is allocated from the user's partition when the user logs-in if the user ID is enabled for system synonyms by the User Definition File.

Tape cartridge. A physical cartridge of 4-track .25 inch tape. The 4-track tape cartridge is standardized in ANSI X3.55-1977. Depending on the hardware controller, RM/COS treats a 4-track cartridge as containing one or four logical tracks.

Tape labels. Records written at the front of a tape volume and before and after each file section which contain information pertaining to the volume or file. Tape label information includes the name of the volume, and the characteristics of each file. ANSI labels are standardized in ANSI X3.27-1978, Magnetic Tape Labels and File Structure for Information Interchange. IBM labels are described in OS/VS Tape Labels, GC26-3795.

Tape mark. A specially recorded sequence on tape used to indicate the end of a file or file section.

Tape reel. A physical reel of 9-track .50 inch tape. The 9-track tape reel is standardized in ANSI X3.40-1976.

Tape volume. The removable tape reel or cartridge. A tape volume identifier consists of one to six graphic characters excluding quotation mark, slash, comma, angle brackets, and square brackets (" / , < > [ ]). A tape volume may contain one or more files, and a single file may span more than one tape volume.

Temporary file. (See Scratch file)

Terminal. A device consisting of a video display for displaying output and a keyboard for accepting input. Terminals are also called stations and have device names with a device prefix 'ST'.

Terminal mode. (See Interactive mode)

Terminal partition. A partition associated with a terminal device. Each terminal device has its own associated partition reserved for its use alone. A terminal partiton becomes active when a user logs-in at the terminal device associated with the partition and becomes idle when the user QUITs.

Text editor. (See Editor)

Text-name. A COBOL user-defined word which identifies library text which is to be copied into the source program during compilation. For RM/COS, a text-name is the logical name assigned to a sequential file containing the library text to be copied.

Time slice. The unit of CPU time allocated to a process executing in a partition when that partition is scheduled for execution. The process is allowed the full unit of CPU time for execution before the operating system repeats its scheduling algorithm to select a partition, possibly the same one, for execution. A partition gives up its time slice if the executing process requests I/O, a time delay, or time slice termination (see C$TTSL subroutine description).

Type-ahead. The feature that allows the user to enter characters at a terminal prior to the input request being sent to the terminal.

Type-ahead buffer. A first-in/first-out queue storage area allocated for each terminal to allow the user to enter characters prior to the request for input.

Undefined file. A disk file which has been created by the CREATE command but which has never been opened OUTPUT, I-O, or EXTEND by a JDL command processor or a COBOL program. Therefore the file has no data records and certain characteristics of the file, such as key position and key length for an indexed file, are still undefined. An undefined file may not be opened INPUT.

Unit. A single device from a set of one or more devices attached to a controller.

Unlabelled tape. Tape reels which do not contain tape labels.

Unused partition. A partition which has been assigned no memory in the System Definition File. An unused partition cannot be assigned memory by use of the PARTITION command. The only purpose of an unused partition is as a partition number place holder.

Unit number. The number which selects a specific device from a set of devices attached to a controller.

Use procedure. A COBOL procedure for input/output error handling which is in addition to the standard procedures provided by the system.

User Definition File (.USRDEFIL). A system file which defines the valid user IDs and associated attributes. The file is used by the system during log-in to validate the user ID and obtain the user's attributes. A batch stream and a program for making modifications to the User Definition File are provided with the system.

User ID. A one to thirty character alphanumeric string which identifies a valid user of the system. Each user ID has associated with it the following items

Passcode
Privilege level
Initial batch pathname
System synonym availability
User synonym table size
Synonym file pathname

The user ID is entered by the operator to log-in to the system. The system verifies the user ID in the User Definition File and obtains the above information for establishing the log-in.

User link. A device for transmission of data across a communications link using a user-defined protocol. User link devices have a device name with a device prefix 'UL'.

User synonym table. An area of storage reserved for the recording of user synonyms and their associated values. The user synonym table is allocated from the user's partition when the user logs-in. The size of the table is controlled by the User Definition File entry for the user ID.

Volume. (See disk volume or tape volume)

Volume directory. The root directory of the tree structure of directories on a disk volume.

Volume name. A one to eight character identifier associated with a disk volume at the time the volume is initialized. Once a volume is declared to the system by a LOAD command, the volume may be referred to by its name without identifying which disk device contains the volume.

Word. A sequence of two contiguous bytes of storage. A word in the CPU memory is always at an even byte address.

Work file. A disk file with the WORK file type attribute. On output to a work file, the logical records are not written immediately to the disk, but are deferred until (a) the block buffer is full, (b) the block buffer is needed for an input operation, or (c) the file is closed. Output to work files will be faster than to nonwork files, but several logical records may be lost if a power failure occurs.

APPENDIX H

COBOL RUNTIME DEBUG

INTRODUCTION

The COBOL runtime debug monitor allows the user to control the execution of his COBOL program at the source language level. The user can cause a single statement to be executed, cause execution to continue until a preset breakpoint location is reached, and display or modify the value of any data item defined in the program. The debug monitor is interactive and is independent of the debug compilation option. The source listing and allocation map are needed by the user to operate the debugger.

The debug monitor is given control at execution time if the debug option is selected when executing a COBOL program. Since the debug monitor is interactive, the COBOL program must be executed in a terminal partition. The debug monitor requires approximately 3K bytes of memory in addition to that normally required for execution of the program; the debug monitor is allocated in the user's partition.

DISPLAYS

The station display will be in the following format on the first time entry, single step, breakpoint stop or when a valid command is completed:

```
nnnnnnnn xx/yyyy zz D?
```

where:

nnnnnnnn = first eight or less characters from program-name specified in PROGRAM-ID paragraph of currently executing program

xx = hexadecimal overlay number (blank if fixed peramanent segment)

yyyy = hexadecimal byte location within the overlay of the next instruction to be executed

zz = stop indicator:
- BP for breakpoint
- ER for error in COBOL program
- PS for object instruction step
- SS for source statement single step and first entry

The xx and yyyy are from the DEBUG column of the source listing produced by compilation. The program-name is the name specified in the PROGRAM-ID paragraph of the IDENTIFICATION DIVISION.

ERRORS

Errors encountered by the debug monitor are displayed on the system error message line. After the message is acknowledged by pressing the Acknowledge Key appropriate to the terminal being used, the command prompt is redisplayed. Error messages are as follows:

4301 BREAKPNT?	Unrecognized breakpoint reference.
4302 OVERFLOW!	Breakpoint table overflow.
4303 DUPLICATE!	Duplicate breakpoint setting.
4304 COMMAND?	Unrecognized command.
4305 LINE!	Invalid screen line number.
4306 LOCATION!	Invalid program location values.
4307 SYNTAX!	Invalid command syntax.
4308 TRUNCATION	Modify data too long for size given.
4309 TYPE?	Invalid data type.
4310 VALUE?	Invalid numeric parameter syntax or value
4311 WAIT	Waiting for data display acknowledgement.

COMMANDS

All commands are specified by a single character possibly followed by one or more parameters. All numeric parameters are in hexadecimal notation unless otherwise noted. Generally addresses are expresed in hexadecimal and lengths are expressed in decimal. The numeric notation is a function of the parameter type; the user has no control over the notation and must enter a valid entry for each parameter (i.e., the character ">" cannot be entered for either decimal or hexadecimal parameters). All parameters of the form "[xx/]yyyy" indicate a program location as shown in the DEBUG column of the source listing. The "xx" is an overlay number; its omission indicates that "yyyy" is in a fixed segment.

B[xx/]yyyy[,nnnnnnnn]

Set breakpoint. A breakpoint is set in overlay number "xx" and object instruction location "yyyy" in program-name "nnnnnnnn". If "nnnnnnnn" is omitted, it is assumed to be the name of the currently executing program. When execution reaches a breakpoint, the debug monitor prompts for another command. Zero to three breakpoints may be in effect at one time. A program or an overlay segment need not be loaded at the time a breakpoint is set within that program or overlay segment.

C[[xx/]yyyy[,nnnnnnnn]]

Clear breakpoint or all breakpoints. The breakpoint specified by overlay "xx" at location "yyyy" in program "nnnnnnnn" is cleared. C with no operand clears all breakpoints. Omission of "nnnnnnnn" implies the name of the currently executing program.

Dxxxx[Un],yyyy[,tttt]

Display data value. Display the value of a data item starting at hexadecimal byte offset "xxxx" with decimal size "yyyy" and data type "tttt". "Un", where "n" is the using argument number in decimal, may only be specified for linkage data items; such data items must be listed in the USING phrase of the Procedure Division header or be defined subordinate to such a data item. The data type indicator "tttt" may be chosen from the following abbreviations:

Abbreviation	Data Type
ABS	Alphabetic String
ANS	Alphanumeric String
ANSE	Alphanumeric String Edited
GRP	Group
HEX	Hexadecimal
NBS	Numeric Signed (COMP-1)
NCS	Numeric Signed (COMP)
NCU	Numeric Unsigned (COMP)
NLC	Numeric Signed (DISPLAY)
NLS	Numeric Signed (DISPLAY)
NPS	Numeric Signed (COMP-3)
NPU	Numeric Unsigned (COMP-6)
NSE	Numeric Edited
NSS	Numeric Signed (DISPLAY)
NSU	Numeric Unsigned (DISPLAY)
NTC	Numeric Signed (DISPLAY)
NTS	Numeric Signed (DISPLAY)

If "tttt" is not specified, HEX is defaulted. The HEX dump format displays two hexadecimal digits for each character dumped.

Dump display is in the format:

```
xxxx tttt dddd...
```

where "dddd..." = the value of the data item in the specified format. If the display fills the debug command line, the wait message 4311 is displayed on the error line. After the wait message is acknowledged, the debug monitor will display the next line of data, if any. After all the requested data has been displayed and acknowledged, the command prompt is redisplayed.

Only items defined in the currently executing program can be displayed, but this includes data items which have been passed as arguments to linkage data items defined in the currently executing program. The values for "xxxx", "yyyy", and "tttt" can be obtained for each data item directly from the first three fields of the allocation map at the end of the program compilation listing. For a linkage data item, the value of "n" is the ordinal position of the data-name in the USING phrase of the Procedure Division header, with 1 corresponding to the first, 2 to the second, and so on. The value of "n" is indicated in the allocation map preceding the data item in the Linkage Section portion which defines or contains the data item of interest.

A value error 4310 will occur if "xxxx" to "xxxx" + "yyyy" is not fully contained within the data area allocated to the currently executing program. A value error will also occur if argument "n" was not passed to the currently executing program or if "xxxxUn" to "xxxxUn" + "yyyy" is not fully contained within the actual argument "n" passed to the currently executing program.

I[xx/]yyyy[,nnnnnnnn]

Initialize location counter. The object instruction location counter will be set to "yyyy" in overlay "xx". The user should exercise extreme care in using this command. The paragraph control of the program may be incorrect after use of this command. The value of "xx" must correspond to the currently loaded overlay. The name "nnnnnnnn", if specified, must be the name of the currently executing program.

K

Kill program execution. Force an immediate STOP RUN in the currently executing program.

Ln

Set debug line number. The debugger will begin using line "n" for prompts after this command. Initially the debugger will use line twenty-four.

Mxxxx[Un],yyyy,[tttt],dddd...

Modify data value. The data item beginning at hexadecimal byte offset "xxxx" for a decimal length of "yyyy" is set to the value "dddd...". "Un", where "n" is the using argument number in decimal, may only be specified for linkage data items. The value "dddd..." is always entered in DISPLAY format and is converted to type "tttt" by the debugger. The operands other than "dddd..." are as described for the Display (D) command. For type HEX destinations, "dddd..." is taken as two hex digits per byte stored left justified with zero fill or truncation on the right. Nonnumeric destinations (ANS, ANSE, GRP, ABS, NSE) are left justified with blank fill or truncation on the right. Edited items must be presented in edited form; no editing is performed. For numeric destinations (NSU, NSS, NCS, NCU, NPS, NBS) the following rules apply.

(1) Only digit characters '0', '1',....'9' and sign character '-' from the string "dddd..." are used.
(2) After deleting unused characters from the string "dddd...", the resultant string is treated as a right justified integer multiplied by the scale of the target item. Zero fill or truncation is provided on the left.
(3) The size "yyyy" is limited to the range 0 to 255.

N

No debug mode. Continue execution without debugger monitoring.

R[[xx/]yyyy[,nnnnnnnn]]

Resume execution after optionally setting a breakpoint. The operands are as described for the Breakpoint (B) command.

S[,m]

Single step statements. Execute "m" (decimal) statements and return to debug command input. If "m" is omitted, then one COBOL statement is executed.

EXAMPLES

The following COBOL compilation listing is annotated to show which values are used as parameters in the debug monitor commands.

```
COS68000 XCOBOL VERSION 2.2 COMPILED: 82/07/27  12:51:02  OPT=****

LINE  DEBUG PG/LN  A...B.......2.........3.........4.........5....

   1                IDENTIFICATION DIVISION.
   2                PROGRAM-ID.
   3                    DEBUG-EXAMPLE.
   4
   5                ENVIRONMENT DIVISION.
   6                CONFIGURATION SECTION.
   7                SOURCE-COMPUTER.  WI-150.
   8                OBJECT-COMPUTER.  WI-150.
   9
  10                DATA DIVISION.
  11                WORKING-STORAGE SECTION.
  12                01 WS-DATA.
  13                   02 WS-DATA-1    PIC  X(5) VALUE "XXXXX".
  14                   02 WS-DATA-2    PIC S9(3) COMP-3 VALUE -103.
  15                LINKAGE SECTION.
  16                01 LK-DATA.
  17                   02 LK-DATA-1    PIC  X(5).
  18                   02 LK-DATA-2    PIC S9(3) COMP-3.
  19
  20                PROCEDURE DIVISION USING LK-DATA.
  21  >0002         DRIVER SECTION 0.
  22  >0002         P-1.
       yyyy
          \
       ----------------------------------------------------------
       :                                                        :
       : B2,DEBUG-EX   Set breakpoint at P-1 OF DRIVER in       :
       :               DEBUG-EXAMPLE program.  This command     :
       :               would be entered while in the calling    :
       :               program before DEBUG-EXAMPLE is called.  :
       :                                                        :
       ----------------------------------------------------------
  23  >0002             PERFORM P-1 OF INITIALIZE.
  24  >0006         P-2.
  25  >0006            EXIT PROGRAM.
  26>01000E         INITIALIZE SECTION 50.
  27>01000E         P-1.
     xxyyyy
          \
       ----------------------------------------------------------
       :                                                        :
       : R1/E         Set breakpoint at P-1 OF INITIALIZE in    :
       :              DEBUG-EXAMPLE program and resume exection.:
       :              This command might be entered when above  :
       :              breakpoint was reached.                   :
       :                                                        :
       ----------------------------------------------------------
  28>01000E             MOVE LK-DATA TO WS-DATA.
```

```
COS68000 XCOBOL VERSION 2.2 COMPILED: 82/07/27  12:51:02  OPT=****

 ADDRESS  SIZE DEBUG ORDER TYPE                    NAME

WORKING-STORAGE SECTION

  >0034      7  GRP     0   GROUP                  WS-DATA
  >0034      5  ANS     0   ALPHANUMERIC            WS-DATA-1
   xxxx   yyyy
          \
        ┌──────────────────────────────────────────────────────┐
        ¦  xx y                                                ¦
        ¦ D34,5          Display WS-DATA-1 in hexadecimal format.¦
        ¦                                                      ¦
        └──────────────────────────────────────────────────────┘

  >0039      2  NPS     0   PACKED SIGNED          WS-DATA-2
   xxxx   yyyy  tttt
          \
        ┌──────────────────────────────────────────────────────┐
        ¦  xx y ttt                                            ¦
        ¦ D39,2,NPS      Display WS-DATA-2 converted from type NPS¦
        ¦                to display format.                    ¦
        ¦                                                      ¦
        └──────────────────────────────────────────────────────┘

LINKAGE SECTION

    n
  U1+                                USING ARGUMENT 1:
  >0000      7  GRP     0   GROUP                  LK-DATA
  >0000      5  ANS     0   ALPHANUMERIC            LK-DATA-1
  >0005      2  NPS     0   PACKED SIGNED          LK-DATA-2
   xxxx   yyyy  tttt
          \
        ┌──────────────────────────────────────────────────────┐
        ¦  x n y ttt                                           ¦
        ¦ M5U1,2,NPS,-5   Modify value of LK-DATA-2 to -5.     ¦
        ¦                                                      ¦
        └──────────────────────────────────────────────────────┘

READ ONLY BYTE SIZE =            40     ( >0028 )

READ/WRITE BYTE SIZE =          122     ( >007A )

OVERLAY SEGMENT BYTE SIZE =      36     ( >0024 )

TOTAL BYTE SIZE =               198     ( >00C6 )

    0 ERRORS

    0 WARNINGS
```

APPENDIX I

STATION CHARACTERISTICS

PHYSICAL AND LOGICAL ATTRIBUTES

An RM/COS visual display station provides 24 lines of 80 characters each for program input/output. Lines are numbered from top to bottom as 1 through 24. Character columns are numbered left to right as 1 through 80.

The display screen is logically divided into three overlapping areas: the user display, the system line, and the error line. The user display area is the entire screen. The system line consists of the leftmost characters of line 24 and is used to prompt and input JDL commands and parameters; user messages generated by the MESSAGE command are also displayed on the system line. The error line consists of the rightmost characters of line 24 and is used to display system error and control messages to the operator. All messages to the error line must be acknowledged by the operator pressing the Acknowledge Key.

A character sized block, called a cursor, is visible to cue the operator when the system is waiting for input (on some terminals the cursor is an underscore or other indicator and in some cases is always visible even when input is not expected; see your Terminal Guide). Keyboard input is buffered other times but is not displayed until input is requested by the system or user program. This type-ahead buffer is flushed when an error message is displayed in the system message area.

Cursor Positioning

In the following discussion, the term "cursor position" is used to describe the place on the display screen where the next character will be output or echoed from keyboard input. The term "cursor" is used to describe the visible indicator that physically marks the cursor position for the operator in most cases. An exception here is when the system is waiting for the operator to acknowledge an error message or full display condition; the cursor is visible in the bottom right corner of the display (and is blinking on some terminals) but the cursor position for ongoing input/output remains at the point the input/output was interrupted.

Cursor Progression

The cursor position on the display screen determines where the next character will be displayed, whether it is the result of a WRITE or DISPLAY statement or of keystrokes entered on the keyboard and echoed to the screen in response to a READ or ACCEPT statement. As each character is written to the display screen, the cursor advances right one character position for the next character. After advancing to the right screen boundary, it is repositioned to the left screen boundary on the next line down. At the end of the

bottom line of the display screen, the cursor advances to the left screen boundary on the top line of the screen. This natural progression of the cursor during input/output is limited by the restriction that any given input/output cannot logically extend beyond the bottom right screen boundary. An example of this being a DISPLAY of 80 characters starting in position 20 of line 24 which would be truncated to 60 characters and the cursor position would be set to position one, line one after the completion of the DISPLAY.

Field Definition

In the following discussion the term "field" refers to an area of the display screen bounded on the left by the starting cursor position, whether explicitly set or allowed to default, and extending to the right for the number of characters to be entered, whether limited by the size of the operand involved or the remaining screen from the cursor to the lower right boundary.

FILE I/O

The OPEN, CLOSE, WRITE, and READ file I/O statements may be used on a station device assigned as a file. These statements treat the screen as a teletype or printer. The READ and WRITE statements cause I/O on the 24th line of the screen. The top 23 lines of the screen provide a history of the last 23 operations.

OPEN Statement

The OPEN statement erases the screen and the cursor position is set to the lower left corner of the screen.

WRITE Statement

The WRITE statement performs four steps: if present, the AFTER ADVANCING clause is processed, the cursor position is set to the lower left corner of the screen, the record area is displayed on the screen, and if present, the BEFORE ADVANCING clause is processed.

ADVANCING Clause

The ADVANCING clause causes the display to be scrolled up by the number of lines specified. A display "scroll up" means moving all the lines of the display up one line to create a blank line at the bottom of the display and removal of what was the top line. After the display is scrolled, the cursor position is set to the bottom left corner of the screen, at the start of the last blank line created. When no ADVANCING clause is specified, the default is equivalent to an AFTER ADVANCING 1 clause. Termination of input by the operator in response to a READ statement also causes the display to be scrolled up one line.

For every 24 lines that are advanced without intervening input, OPEN, or CLOSE functions, the cursor appears at the bottom right corner of the screen and the beep tone is sounded. This signals that the operator must press the Acknowledge Key to continue the current WRITE function.

When an ADVANCING clause specifies a page advance the display is erased. If one or more lines have been output to the station since it was opened or a page was advanced the operator must acknowledge his readiness for the current display to be erased, as described previously. After the display is erased to advance the page, the cursor position is set to the lower left corner of the display screen.

READ Statement

If the cursor position is not on the bottom line, the READ statement causes the screen to be scrolled up one line setting the cursor position to the lower left corner of the screen. The READ statement allows input within the current field. Filling the input field or entry of one of the input termination keys described later terminates station input. Any characters displayed in the bounds of the input field are returned in the input buffer as if they had been entered from the keyboard. Upon input termination, the screen is scrolled up one line and the cursor position is set to the lower left corner of the screen.

SCREEN I/O

The DISPLAY and ACCEPT statements allow a COBOL program to treat the station as a video terminal by providing for random positioning of the cursor and greater access to the keyboard.

LINE and POSITION Clauses

The ACCEPT and DISPLAY statements provide for explicit cursor positioning through the LINE and POSITION clauses. The values specified are used modulo of the associated screen dimensions. For example LINE 25 POSITION 88 sets the cursor position to line 1 column 8.

Specifying a value of zero or omitting either or both clauses implies default positioning. The default for LINE is the next line below the current cursor position unless POSITION is also defaulted, in which case the current cursor position is unchanged. If POSITION is not specified, a value of one is assumed for the first operand of the ACCEPT/DISPLAY and zero for each subsequent operand. If POSITION 0 is specified the current character position is unchanged regardless of any line repositioning. Table I-1 presents these positioning rules in a simplified form.

Specification	Line	Column
(none)	next	1 *
LINE 0	next	1 *
LINE x	x	1 *
POSITION 0	current	current
LINE 0 POSITION 0	current	current
LINE X POSITION 0	x	current
POSITION y	next	y
LINE 0 POSITION y	next	y
LINE x POSITION y	x	y

* one for first operand only; current for each subsequent operand.

Table I-1

LINE/POSITION Defaults

If default positioning is requested and the last operation set the cursor position past line 24 position 80, then the screen is erased and the cursor position is set to the upper left corner of the screen before performing the ACCEPT or DISPLAY operation.

ERASE Clause

The ERASE clause on an ACCEPT or DISPLAY statement causes the entire display to be erased before the ACCEPT or DISPLAY is performed. The cursor position is set to the lower left corner of the display screen after the display is erased.

ACCEPT Statement

An ACCEPT statement with no modifiers other than positioning clauses allows input within the field determined by the positioning clauses. Filling the input field or pressing one of the input termination keys described later terminates station input. Any characters displayed in the bounds of the input field are returned in the input buffer as if they had been entered from the keyboard. The cursor position is set to the next position after the input field regardless of where the cursor was in the input field when the input was terminated.

ACCEPT with PROMPT Clause

The PROMPT clause modifying an ACCEPT statemet causes the input field to be initialized with the character operand of the PROMPT clause. If no operand is specified the default prompt character is an underscore ("_"). Except when the prompt character is a blank, it cannot be entered as input from the keyboard nor can the cursor be spaced over a non-blank prompt character without overwriting it. A non-blank prompt also acts as a field terminator in determining the size of the field returned (in assembly language input) and the final cursor position. That is, only characters entered up to the leftmost prompt character are returned and the cursor position is set at the leftmost prompt character position.

ACCEPT with TAB Clause

The TAB clause enables the task edit keys as input termination keys and requires a termination key to be entered to terminate the input. The task edit keys and their functions are detailed below in Key Classifications. Normally when an input field is filled the input from the station is terminated.

When a TAB clause is present and the input field is filled, the cursor position remains beyond the last character of the field and subsequent keystrokes are ignored while a warning beep is given; a termination key must be entered to terminate the input. If the field ends at the right screen boundary the cursor remains in the last character position of the field and subsequent keystrokes overwrite that character until a termination key is entered.

ACCEPT with ON EXCEPTION Clause

The ON EXCEPTION clause enables the special event keys and task edit keys as input termination keys as well as enabling the return of exception codes. Special event keys, task edit keys, and exception codes are detailed below in Key Classifications. The value of a two digit numeric data item specified by the ON EXCEPTION clause is set according to which input termination key is entered. A program may use the exception code value returned to modify the program flow.

KEY CLASSIFICATIONS

RM/COS station handling divides keyboard characters into four groups: data keys, system interaction keys, system edit keys and input termination keys. Input termination keys include the sub-groups task edit keys and special event keys which are enabled only under certain conditions. Some system edit keys are also task edit keys.

The following discussion describes the function keys with names corresponding to the function to be performed. These names are not in general identical to the names on the keycaps of the various terminals supported by RM/COS. The function names and the terminal keycaps for each terminal are detailed in the Terminal Guide.

Data Keys

These keys produce the 95 displayable ASCII characters in columns two through seven of the ASCII character table, including shift key modifications for lower and upper case generation. These keys make up the main (center) keypad as well as the right (numeric) keypad. Most data keys have the character they generate embossed on the keytops. The lower case alphabet is produced by releasing the SHIFT and SHIFT LOCK keys and using the embossed alphabetic keys. Other characters produced by SHIFT key modification are shown as the upper embossed keytop character and the characters produced when the SHIFT key is released are shown as the lower embossed keytop character.

System Interaction Keys

RM/COS defines four keys for interaction with the system:

Acknowledge Key - Used to respond to messages on the error line and display paging conditions.

Command Key - Used to return to the JDL command prompt once parameter entry has begun (i.e., to abort parameter entry). The Command Key is also an input termination key (see below).

Interrupt Key - Used to interrupt JDL command processors and user programs. The Interrupt Key may also be used to return to the JDL command prompt once parameter entry has begun.

Log-in Key - Used to initiate log-in at an inactive terminal. On most keyboards the Log-in Key is the Return Key.

The Command Key is also a Special Event Key. The Command Key and Acknowledge Key are recognized in the type-ahead buffer.

System Edit Keys

The system edit keys perform local functions on the current input field. In cases where these keys are also task edit keys, the system edit function is performed prior to acting as an input termination key.

Erase Field - resets the input field to blanks or prompt characters and sets the cursor position at the start of the field.

Delete Line - resets the input field to blanks or prompt characters, sets the cursor position to the start of the field, and acts as a task edit key.

Home - sets the cursor position to the start of the field, and acts as a task edit key.

Left Arrow - moves the cursor position back (left) one character position. If the cursor position is already at the start of the field, a warning beep sounds and the cursor position remains at the start of the field.

Right Arrow - moves the cursor position forward (right) one character position. The character displayed in that position is accepted as if it were entered instead of the right arrow. If that character is a nonblank prompt character, a warning beep sounds and no cursor movement occurs.

Insert Character - sets the station in insert mode so that subsequent data keys entered will be inserted immediately under the cursor by moving all characters from the cursor to the end of the field right one position and removing one trailing blank or prompt character to make room for the inserted character. The station is taken out of insert mode by the entry of key which is not a data key. If, while inserting data characters, all trailing blanks or prompts are removed, i.e., the field is filled, subsequent data key entries will sound a warning beep and be ignored until the station is removed from insert mode. Note that the cursor must exit the last character position of the field to cause input termination if appropriate for the input mode.

Delete Character - deletes the character under the cursor and moves all characters to the right of the cursor one character position left, inserting a blank or prompt character at the end of the field. If the cursor is positioned to a nonblank prompt character, a warning beep is given and the keystroke is ignored.

Tab Left - resets the cursor position to the start of the field, and acts as a task edit key.

Erase Right - resets the input field from the cursor to the end of the field to blanks or prompt characters and terminates input. The cursor position is unchanged.

System edit keys are affected by the OFF clause of an ACCEPT statement. The OFF clause prevents the characters entered from being displayed in the input field. The cursor position advances with each character entered but cannot be advanced using Right Arrow if nonblank prompt characters were specified. Likewise Insert Character and Delete Character will only sound the warning beep if nonblank prompt characters are present. When the OFF clause is present, no characters logically exist to the right of the cursor. If the cursor is moved to the left using the Left Arrow or set to the start of the field with Home or Tab Left, the characters entered which are now to the right of the cursor, including the character under the cursor, are lost. Also note that any data visible under and to the right of the cursor in the input field of an ACCEPT statement with the OFF clause specified and the PROMPT clause not specified will be returned to the calling program following the characters actually entered and to the left of the cursor.

Input Termination Keys

Input termination keys are used by the station operator to signal to the system that the current data input is complete and should be returned to the calling program. Task edit keys and special event keys are conditionally enabled input terminators which perform other operations and may return self-identification fields, called exception codes, depending on the type of input request made. The following keys are always input termination keys:

Return - terminate input and return the entire input field up to the first nonblank prompt character, if present. The cursor position is set to the next character position after the input field or to the first nonblank prompt character, if present.

Tab Right - functions the same as Return, but sets a different exception code if an ON EXCEPTION clause is present.

Erase Right - resets the input field from the cursor to the end of the field to blanks or prompt characters and terminates input. The cursor position is unchanged.

Send - functions the same as Return for ACCEPT operations, but sets a different exception code if an ON EXCEPTION clause is present. For READ statements the input field is set to blanks, and an AT END condition is signaled.

Task Edit Keys

These keys are enabled for ACCEPT statements when the TAB or ON EXCEPTION clause is present. Some task edit keys also perform system edit functions prior to input termination. Exception codes returned when the ON EXCEPTION clause is specified are detailed in Table I-6.

Delete Line	*
Insert Line	
Up Arrow	
Home	*
Down Arrow	
Tab Left	*
Erase Right	*
Tab Right	

* These keys also perform system edit functions described previously.

Table I-3

Task Edit Keys

Special Event Keys

These keys are enabled for ACCEPT statements when the ON EXCEPTION clause is present and for interactive JDL command input. All function as simple input termination keys but return a unique exception code (see Table I-6) to identify the key entered. Table I-4 lists the special event keys.

New Line	Function 9
Print	Function 10
Command	Function 11
Attention	Function 12
Function 1	Function 13
Function 2	Function 14
Function 3	Function 15
Function 4	Function 16
Function 5	Function 17
Function 6	Function 18
Function 7	Function 19
Function 8	Function 20

Table I-4

Special Event Keys

Function Sequences

Each nondata key on a terminal keyboard is assigned to its most reasonable function based on its keycap mnemonic and these assignments are provided in your Terminal Guide. Some simpler keyboards may not have a reasonable key to assign to a specific function supported by RM/COS. In order to assure that all functions are available on all terminals, the various functions can be accessed with a two key sequence called a function sequence. This sequence consists of a Function Sequence Introducer character followed by a Function Sequence Code. The Function Sequence codes and the Exception codes returned for the task edit keys are detailed in Table I-6. The Function Sequence Introducer character for each terminal may be found in the Terminal Guide.

Control Characters

Certain terminals use control characters as special functions. The characters Escape, CONTROL A, and CONTROL B introduce terminal functions and will cause some number of succeeding characters to be ignored. CONTROL S will stop RM/COS from sending any characters to the terminal. CONTROL Q will allow RM/COS to resume sending characters to the terminal. Some terminals may have internal errors which cause the terminal to send CONTROL S and never send a CONTROL Q. If it appears that the system no longer responds to terminal keystrokes, pressing CONTROL Q may cause correct operation to resume.

Function	Function Sequence Code	Exception Code
Return	M m	0
Function 1	1	1
Function 2	2	2
Function 3	3	3
Function 4	4	4
Function 5	5	5
Function 6	6	6
Function 7	7	7
Function 8	8	8
Function 9	9	9
Function 10	0 (number)	10
Function 11	! (exclamation)	11
Function 12	@ (at sign)	12
Function 13	# (pound)	13
Function 14	$ (dollar)	14
Function 15	% (per cent)	15
Function 16	^ (carat)	16
Function 17	& (ampersand)	17
Function 18	* (asterisk)	18
Function 19	((left paren)	19
Function 20	) (right paren)	20
Command	Q q	40
Attention	E e	41
Print	P p	49
Up Arrow	K k	52
Down Arrow	J j	53
Home	G g	54
New Line	F f	55
Tab Left	B b	56
Erase Right	R r	57
Tab Right	T t	58
Insert Line	N n	59
Delete Line	Z z	61
Send	O o	64
Erase Field	C c	*
Left Arrow	H h	*
Right Arrow	L l	*
Insert Character	I i	*
Delete Character	D d	*
Acknowledge	A a	*

* These keys do not act as input terminators and thus do not return an exception code.

Table I-6

Function Sequence and Exception Codes

APPENDIX J

CHARACTER SET CONVERSIONS

ASCII TO EBCDIC CONVERSION TABLE

	ASCII Code	U.S. Character	EBCDIC Code	U.S. Character
	00	NUL	00	NUL
(1)	01	SOH	01	SOH
(1)	02	STX	02	STX
(1)	03	ETX	03	ETX
(1)	04	EOT	37	EOT
(1)	05	ENQ	2D	ENQ
(1)	06	ACK	2E	ACK
	07	BEL	2F	BEL
	08	BS	16	BS
(2)	09	HT	05	HT
	0A	LF	25	LF
(2)	0B	VT	0B	VT
(2)	0C	FF	0C	FF
	0D	CR	0D	CR
(1)	0E	SO	0E	SO
(1)	0F	SI	0F	SI
(1)	10	DLE	10	DLE
	11	DC1	14	ENP
	12	DC2	24	INP
	13	DC3	04	SEL
	14	DC4	15	NL
(1)	15	NAK	3D	NAK
(1)	16	SYN	32	SYN
(1)	17	ETB	26	ETB
	18	CAN	18	CAN
(1)	19	EM	19	EM
	1A	SUB	3F	SUB
(3)	1B	ESC	27	ESC
(1)	1C	FS	1C	IFS
(1)	1D	GS	1D	IGS
(1)	1E	RS	1E	IRS
(1)	1F	US	1F	IUS
	20	Space	40	Space
	21	Exclamation Mark	4F	Long Vertical Line
	22	Quotation Marks	7F	Quotation Marks
	23	Number Sign	7B	Number Sign
	24	Dollar Sign	5B	Dollar Sign
	25	%	6C	%
	26	&	50	&
	27	'	7D	'
	28	(	4D	(
	29	)	5D	)
	2A	*	5C	*
	2B	+	4E	+
	2C	,	6B	,
	2D	-	60	-
	2E	.	4B	.
	2F	/	61	/
	30	0	F0	0

31	1	F1	1
32	2	F2	2
33	3	F3	3
34	4	F4	4
35	5	F5	5
36	6	F6	6
37	7	F7	7
38	8	F8	8
39	9	F9	9
3A	:	7A	:
3B	;	5E	;
3C	<	4C	<
3D	=	7E	=
3E	>	6E	>
3F	?	6F	?
40	Commercial At	7C	Commercial At
41	A	C1	A
42	B	C2	B
43	C	C3	C
44	D	C4	D
45	E	C5	E
46	F	C6	F
47	G	C7	G
48	H	C8	H
49	I	C9	I
4A	J	D1	J
4B	K	D2	K
4C	L	D3	L
4D	M	D4	M
4E	N	D5	N
4F	O	D6	O
50	P	D7	P
51	Q	D8	Q
52	R	D9	R
53	S	E2	S
54	T	E3	T
55	U	E4	U
56	V	E5	V
57	W	E6	W
58	X	E7	X
59	Y	E8	Y
5A	Z	E9	Z
5B	Opening Bracket	4A	Cent Sign
5C	Reverse Slant	E0	Reverse Slant
5D	Closing Bracket	5A	Exclamation Mark
5E	Circumflex	5F	Logical NOT
5F	_	6D	_
60	Grave Accent	79	Grave Accent
61	a	81	a
62	b	82	b
63	c	83	c
64	d	84	d
65	e	85	e
66	f	86	f

67	g	87	g
68	h	88	h
69	i	89	i
6A	j	91	j
6B	k	92	k
6C	l	93	l
6D	m	94	m
6E	n	95	n
6F	o	96	o
70	p	97	p
71	q	98	q
72	r	99	r
73	s	A2	s
74	t	A3	t
75	u	A4	u
76	v	A5	v
77	w	A6	w
78	x	A7	x
79	y	A8	y
7A	z	A9	z
7B	Opening Brace	C0	Opening Brace
7C	Split Vertical Line	6A	Split Vertical Line
7D	Closing Brace	D0	Closing Brace
7E	Tilde	A1	Tilde
7F	DEL	07	DEL

(1) These ASCII character codes are communication control characters. If these characters are encountered in user data when sending a file with the 2780/3780 Emulator in coded mode, they will be replaced with a Substitute character (ASCII >1A or EBCDIC >3F).

(2) These ASCII character codes are format control characters. They are allowed when sending a file with the 2780/3780 Emulator in coded mode. The requested formatting operation should be performed correctly. However, if the destination is a data file, this file will not be identical to the file being sent.

(3) In 2780/3780 communications, when sending a coded file the Escape character introduces a vertical positioning sequence or a horizontal tab stop definition sequence.

ASCII character codes with the most significant bit set are treated as illegal characters. These characters will be replaced with a Substitute character (ASCII >1A or EBCDIC >3F).

EBCDIC TO ASCII CONVERSION TABLE

	EBCDIC Code	U.S. Character	ASCII Code	U.S. Character
	00	NUL	00	NUL
(1)	01	SOH	01	SOH
(1)	02	STX	02	STX
(1)	03	ETX	03	ETX
	04	SEL	13	DC3
(2)	05	HT	09	HT
	07	DEL	7F	DEL
(2)	0B	VT	0B	VT
(2)	0C	FF	0C	FF
	0D	CR	0D	CR
(1)	0E	SO	0E	SO
(1)	0F	SI	0F	SI
(1)	10	DLE	10	DLE
	14	ENP	11	DC1
	15	NL	14	DC4
	16	BS	08	BS
	18	CAN	18	CAN
(1)	19	EM	19	EM
(1)	1C	IFS	1C	FS
(1)	1D	IGS	1D	GS
(1)	1E	IRS	1E	RS
(1)	1F	IUS	1F	US
	24	INP	12	DC2
	25	LF	0A	LF
(1)	26	ETB	17	ETB
(3)	27	ESC	1B	ESC
(1)	2D	ENQ	05	ENQ
(1)	2E	ACK	06	ACK
	2F	BEL	07	BEL
(1)	32	SYN	16	SYN
(1)	37	EOT	04	EOT
(1)	3D	NAK	15	NAK
	3F	SUB	1A	SUB
	40	Space	20	Space
	4A	Cent Sign	5B	Opening Bracket
	4B	.	2E	.
	4C	<	3C	<
	4D	(	28	(
	4E	+	2B	+
	4F	Long Vertical Line	21	Exclamation Mark
	50	&	26	&
	5A	Exclamation Mark	5D	Closing Bracket
	5B	Dollar Sign	24	Dollar Sign
	5C	*	2A	*
	5D	)	29	)
	5E	;	3B	;
	5F	Logical NOT	5E	Circumflex
	60	-	2D	-
	61	/	2F	/

6A	Split Vertical Line	7C	Split Vertical Line
6B	,	2C	,
6C	%	25	%
6D	_	5F	_
6E	>	3E	>
6F	?	3F	?
79	Grave Accent	60	Grave Accent
7A	:	3A	:
7B	Number Sign	23	Number Sign
7C	Commercial At	40	Commercial At
7D	'	27	'
7E	=	3D	=
7F	Quotation Marks	22	Quotation Marks
81	a	61	a
82	b	62	b
83	c	63	c
84	d	64	d
85	e	65	e
86	f	66	f
87	g	67	g
88	h	68	h
89	i	69	i
91	j	6A	j
92	k	6B	k
93	l	6C	l
94	m	6D	m
95	n	6E	n
96	o	6F	o
97	p	70	p
98	q	71	q
99	r	72	r
A1	Tilde	7E	Tilde
A2	s	73	s
A3	t	74	t
A4	u	75	u
A5	v	76	v
A6	w	77	w
A7	x	78	x
A8	y	79	y
A9	z	7A	z
C0	Opening Brace	7B	Opening Brace
C1	A	41	A
C2	B	42	B
C3	C	43	C
C4	D	44	D
C5	E	45	E
C6	F	46	F
C7	G	47	G
C8	H	48	H
C9	I	49	I
D0	Closing Brace	7D	Closing Brace
D1	J	4A	J
D2	K	4B	K
D3	L	4C	L

D4	M	4D	M
D5	N	4E	N
D6	O	4F	O
D7	P	50	P
D8	Q	51	Q
D9	R	52	R
E0	Reverse Slant	5C	Reverse Slant
E2	S	53	S
E3	T	54	T
E4	U	55	U
E5	V	56	V
E6	W	57	W
E7	X	58	X
E8	Y	59	Y
E9	Z	5A	Z
F0	0	30	0
F1	1	31	1
F2	2	32	2
F3	3	33	3
F4	4	34	4
F5	5	35	5
F6	6	36	6
F7	7	37	7
F8	8	38	8
F9	9	39	9

(1) These EBCDIC character codes are communication control characters and cannot be received as user data characters by the 2780/3780 Emulator unless the file is sent in transparent (image) mode. If received where user data is expected, these characters will be replaced with an ASCII Substitute character (>1A).

(2) These EBCDIC character codes are format control characters. They are allowed when receiving a coded file with the 2780/3780 Emulator, and control the insertion of spaces in a line or the slewing control information of a printer or SPOOL file.

(3) In 2780/3780 communications, when receiving a coded file the Escape character introduces a vertical positioning sequence or a horizontal tab stop definition sequence.

EBCDIC character codes not appearing in this table are treated as illegal characters. These characters will be replaced with an ASCII Substitute character (>1A).

CHARACTER ABBREVIATIONS

ACK	Acknowledgement
BEL	Bell
BS	Backspace
CAN	Cancel
CR	Carriage Return
DC1	Device Control 1
DC2	Device Control 2
DC3	Device Control 3
DC4	Device Control 4
DEL	Delete
DLE	Data Link Escape
EM	End of Medium
ENP	Enable Presentation
ENQ	Enquiry
EOT	End of Transmission
ESC	Escape
ETB	End of Transmission Block
ETX	End of Text
FF	Form Feed
FS	File Separator
GS	Group Separator
HT	Horizontal Tab
IFS	Interchange File Separator
IGS	Interchange Group Separator
INP	Inhibit Presentation
IRS	Interchange Record Separator
IUS	Interchange Unit Separator
LF	Line Feed
NAK	Negative Acknowledgement
NL	New Line
NUL	Null
RS	Record Separator
SEL	Select
SI	Shift In
SO	Shift Out
SOH	Start of Heading
STX	Start of Text
SUB	Substitute
SYN	Synchronous Idle
US	Unit Separator
VT	Vertical Tab