10-000000 3-4

# MONROE UTILITY PROGRAMS
# PROGRAMMER'S REFERENCE MANUAL

September 1981

MONROE SYSTEMS FOR BUSINESS
The American Rd.
Morris Plains, N.J. 07950



2107-S

## PURPOSE OF THIS DOCUMENT

This document is a Programmer's Reference Manual. It is to be used by experienced programmers as a reference tool. It is not intended for use as a learning aid by non-programmers.

RECORD OF CHANGES

| Change No. | Date | Pages Affected | Description of Changes |
|---|---|---|---|
| | | | |
| 1-3 | 6/81 | All | Reviewer's Changes |
| -4 | 10/81 | All | Preliminary Edition |

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

# SECTION 1

# INTRODUCTION

# SECTION 1
# INTRODUCTION

## 1.1 OVERVIEW

This manual contains descriptions of the utility programs which can be used on Monroe's 8800 series Computer Systems. The term "utility program" as it is used in this manual means any Monroe supplied program which is resident on a disk or which controls communication to a disk and is not part of the operating system software proper. Therefore COPYLIB, which is an assembly language program that allows you to copy files between disks, and DIRECTORY LIST, which allows you to open files, are both utility programs.

The utility programs allow you to perform maintenance and file functions directly without having to write specialized systems.

## 1.2 TYPES OF UTILITIES

In this document a distinction is made between two types of utilities, hence the manual is divided into two parts. Part I deals with Disk Maintenance Utilities and Part II deals with Task Maintenance Utilities.

### Disk Maintenance Utilities

Disk Maintenance Utilities perform functions related to the creation, maintenance, and overall housekeeping of ordinary disk files. They consists of those programs and commands that will be most commonly used by both Applications and System Programmers alike.

### Task Maintenance Utilities

Task Maintenance Utilities perform functions related to the creation, maintenance, and overall housekeeping of task files. Task Maintenance Utilities will be most frequently used by System Level Programmers who wish to gain a deeper understanding of how the operating system functions and to exploit its capabilities.

## 1.3 DOCUMENT CONTENTS

This manual is divided into an Introduction, Part I Disk Maintenance Utilities, Part II Task Maintenance Utilities, three appendices, a glossary, and an index. Utilities are listed in alphabetical order in each part. All commands together with their format and function are listed, in Appendix A. Appendix B contains error codes and meanings. A helpful cross-reference in finding pertinent examples in this text can be found in Appendix C. This appendix provides a concise list of the examples presented for each utility. The glossary is a quick reference for new or unfamiliar concepts.

## 1.4 HOW TO USE THIS MANUAL

Although the utilities are listed in alphabetical order this is not to suggest that there is no common level of importance or sequence of use. Clearly, some utilities will be more important than others in the sense that they will be used most frequently (OPEN, CLOSE, DIRECTORY LIST). Others will be more important in the sense that they provide extended file handling capabilities (COPYA, COPYI, COPYLIB, and COPYT).

Moreover, there are utilities which almost always appear in sequence with other utilities. FORMAT for example, must be run before DISKINIT is used. DIRECTORY LIST can be used after ALLOCATE or any of the COPY routines to verify whether file space has in fact been allocated or that specific files have indeed been copied. Often particular sequences of use are suggested in the manual and examples are given; but once again, these are only suggestions.

No doubt you, the user, will expand upon these, find new applications of your own, and establish levels of importance in terms of what your own priorities are. It must be emphasized that this is a PROGRAMMER'S REFERENCE MANUAL and not a Tutorial. Hence it is designed to be used primarily as a reference tool by experienced programmers. Examples, however, are provided for all utilities to facilitate their use and enhance your understanding of the operating system.

To this end a format has been chosen for all utility descriptions in this manual. First, each utility program is briefly introduced and its basic function is summarized. Then each command is reviewed with respect to its Function, Mode, Format, Arguments, and Use. The Function of a command or program is the specific task it is designed to accomplish. The Mode of a command refers to the way (remote or interactive) it is used during actual programming. The Format of a Command refers to the structure of its syntax. The Arguments of a Command refers to the variables or descriptors in the Commands format. The Use of a Command describes the way in which the Command is applied including available options that may be relevant to a particular situation. Examples are then provided of each command together with possible variations or other commands that may be used in conjunction with it.

In the initial examples for each utility, the Signon from the Utility program is included, however, this is dropped from later examples in order to avoid repetition.

## 1.5 TEXT SYMBOLS AND CONVENTIONS

Throughout this manual specific documentation conventions are used to describe formats for writing UTILITY commands, statements, and functions. The following conventions are in effect:

| Symbol | Description and Use |
|---|---|
| 1. CAPITAL LETTERS | In the format under which each utility is discussed, capital letters are used for all keywords, commands, functions, and statements that are to be explicitly typed by the user. The abbreviated form of each command is used in each of the examples which illustrate a utility.<br>Examples: COPYLIB MONT:,PASC:<br>ST A<br>R COPYA |
| 2. Lower case letters | Lower case letters represent variables which can be supplied by the user according to the rules explained below and in this text. They are optional and may or may not be included in typing a command.<br>Examples: Lib,F PASC:,BIGFILE<br>OPEn FPY0 |
| 3. < > | Angle brackets indicate that the fields they enclose are required for valid syntax. They are not to be typed.<br>Example: COPYLIB <fd1>,<fd2> |

4. []

Brackets enclose optional elements of a command or indicate an optional choice of elements.
Example: L,F [fd]
COPYI[,switch],[buffsize]

5. ¶

The symbol "¶" indicates the depression of the RETURN key.
Example: L <fd>¶

## 1.6 COMMAND SYNTAX

The general command structure, that is used in this manual, is as follows:

MNEMONIC[,[[switches][,buffsize]][parameter1],[parameter2],...]

| Symbol | Description and Use |
|---|---|
| 1. MNEMONICS | Mnemonics are shown in this manual in Uppercase letters. A mnemonic is a command name and may be entered into the system in its entirety or in an abbreviated form. The minimum abbreviations are indicated in this manual by Uppercase letters. Any number of characters between the minimum and the entire command may be entered.<br>Examples:<br>The command TASK is given in the manual as TASK. The following forms of this command may be entered:<br>TA<br>TAS<br>TASK<br>Illegal forms of this command are:<br>T too short.<br>TAK misspelled.<br>TASKS too long |
| 2. Switches | Switches follow the mnemonic immediately, and are separated from the mnemonic by a comma. The |

| Symbol | Description and Use |
|---|---|
| 2. Switches (Cont.) | switches provide the user with ability to specify certain options in the form of unseuenced alphabetic characters related to the particular mnemonic. These switches are normally passed to the CPU registers of the task.<br><br>Examples:<br><br>COPYLIB,G<br>COPYLIB,DG |
| 3. Buffsize | Buffsize specifies the amount of extra memory, in bytes, to be added to a program. This memory expands the working area, which normally increases the execution speed of a program. Specifying too much results in a load error.<br><br>Example:<br>COPYLIB,G,14000 MONT:BASC,PASC:BASC |
| 4. Parameters | Parameters are separated from the mnemonic and the switches by one or more spaces. These parameters are transferred to the CPU stack of the task.<br><br>Example:<br>COPYA,A MONT:ASCI,PASC:ASC2 |

## 1.7 FILE-VOLUME-DEVICE-NAMING CONVENTIONS

The file, volume, and device naming conventions that are used throughout this manual are defined as follows:

A) A file is a program or a collection of data stored on a disk-type storage medium. Files stay in the system permanently unless they are explicitly removed.

B) A volume name is a name given by the user to a disk (e.g., MONT:,PASC:,FIX:). The system volume is the volume from which the operating system is booted.

C) A device name is a name given to a physical device (e.g., CON for the console, PR: for the printer, FPY0: for disk 0, FPY1: for disk 1). These names cannot be changed by the user.

D) The file/device descriptor, referred to in this manual by <fd>, may refer to any of the above (A, B or C) depending upon the content of the utility command being discussed.

E) File/descriptors can be composed of four fields: vol, filename, directory, and type, where vol can be either a volume or device name. Device descriptors are composed of the device mnemonic only.

F) The format can be expressed in four ways:

1. <vol:>
2. [vol:]<filename>[/type]
3. [vol:]<directory>
4. [vol:]<directory.filename>[/type]

where:

vol — Name of the volume on which the file resides if the file descriptor refers to a file, or the name of a device if the file descriptor refers to a device. It may be from one to four characters. The first character must be alphabetic and the remaining alphanumeric. If the volume is not specified, the default volume is the SYSTEM volume.

filename — Name of the user's file directory (UFD). It may be from one to twelve alphanumeric characters. If not Specified, the directory defaults to the master file directory.

directory — Name of the user's directory file. It may be from one to twelve alphanumeric characters. If not specified, the directory defaults to the master directory.

type — Type of file, i.e., ASCII, Binary, etc.

G) Often task identifiers <tid> appear in a command. Task identifiers are names of programs or utilities which are used by other utility commands. For example:

```
-SETAUTO TA
```

Example: Examples of legal file/device descriptors are:

| | |
|---|---|
| OPEN MONT: | Opens the volume named MONT. |
| AL,C MONT:PASTOR/A | Allocates a Contiguous ASCII file named PASTOR on the volume named MONT. |
| L PACK:DIRC | Lists the files in the directory DIRC on the volume named PACK. |
| COPYLIB,D PACK:DIRC.LOOP/B | Deletes the Binary file LOOP from the directory DIRC on the volume named PACK. |
| COPYA CON:,PR: | Copy ASCII data from the console to the printer. |

## 1.8 KINDS OF FILES

With each file there is a type specification that describes, for the system and the user, what kind of data is in the file. These appear next to the filename for your files in your Master File Directory. Table 1-1 lists these specifications and their meanings. The type of the file is normally implied by the program, and does not need to be specified. If the file type is not implied by the program, it must be specified! Note you can also have file type combinations like Bin Pas as well as abbreviations like /BP for Bin Pas.

Table 1-1. Type Specifications

| Symbol | Description and Use |
|---|---|
| Asm | ASSEMBLER source code. |
| Bas | BASIC source code, or data produced by BASIC |
| Und | Undefined data, which verifies to any other type. |
| Asc | ASCII data readable without any special handling. |
| Lst | List file, ASCII data together with position information. |
| Obj | Object code, readable by the Task Establisher. Cannot be loaded and executed. |
| Bin | Binary data, which is unspecified. |
| Tsk | Task file, either relocatable or absolute. Can be loaded and executed. |
| Ism | ISAM index file. |
| Pas | PASCAL source code, or data produced by PASCAL. |
| Ufd | User File Directory. |
| Mfd | Master File Directory. |

## 1.9 WILD-CARD

Some utility commands may handle generalized File Descriptors. These are used to specify a searching key when scanning for one or several files.

The following special formats are available:

| Symbol | Description and Use |
|---|---|
| * | An asterisk * in a position indicates that a character in that position is to be ignored. It may be placed anywhere except after a dash. |
| - | A dash indicates that all further characters are to be ignored. |

Examples:

| | |
|---|---|
| ABC** | Any file that starts with ABC and has five characters in its name. |
| *BC*- | Any file that starts with any character, followed by BC, and is at least four characters long. The slash - indicates that remaining characters are ignored. |

## 1.10 TERMINAL USAGE

The Terminal Management System is controlled by the user through a terminal device. The name of the logical terminal device is always CON for every user, and may be assigned to a task for ordinary I/O operations, just as any other device.

Utility Commands which reside on diskette are invoked by typing the command followed by a carriage return. The system is ready to accept the command when the CRT responds with a dash -.

### Prompting

When the terminal operator is expected to enter data at the terminal, a prompt is output. This prompt takes one of the following forms:

| Prompt | Type of Request |
|---|---|
| - | Command Request |
| no prompt | Data Request |

The command request prompt is output whenever the system is ready to accept another command.

The data request prompt is output whenever a task is attempting to perform a read request to the terminal device. This request should be satisfied as soon as practical, since messages are held in abeyance until the data request is satisfied.

### Control Characters

The control character conventions in effect for terminal devices are described below.

| Function | Operation |
|---|---|
| Deleting a Line | Depress and hold the CTRL-key while striking the X-key (CTRL-X). |
| Deleting a Character | Depress and hold the CTRL-key while striking the H-key (CTRL-H). |

| Function: | Operation |
|---|---|
| Ending an Input Line | If the input line is complete and ready to be processed, depress the carriage return key. |
| End-Of-File Function | Simultaneously depress the CTRL-key and the equals (=) key. |
| Attention Function | If the data request prompt appears and the user wishes to communicate with the Command System rather than a task, the CTRL-key should be depressed and held while striking. The system responds with the command request prompt, and is ready to accept a command. To return to the data request, just hit the Return-key.<br><br>If an input or output to the terminal is in progress, use of the CTRL-key and the A-key will interrupt the process. For example, if the TASK command has been entered and the output is in progress, CTRL-A halts the output in progress. The system is then ready to accept a command. |
| Abort | Some commands and programs recognize an abort function, which aborts the terminal transfer and cancels the task execution. The abort function is generated by depressing and holding the CTRL-key while striking the C-key. |

Command Handling

The command is the basic unit of conversation between a terminal user and the System. A command directs the MTM System to take a specific action. In general, a single command results in a single action being taken by the System.

A command consists of a mnemonic which normally describes the action the user wishes to take place, and arguments which provide the details necessary to perform the action.

Commands are accepted on line (i.e 80 characters) at a time. A command may not spread over two or more lines. A command line is terminated by a carriage return.

Unknown Commands

If an unknown command is entered, the System tries to load a program with the same name. If found, it will be started as a primary task, and the rest of the command line will be transferred as parameters to that task.

## 1.11 RELATED MANUALS

Every attempt has been made to make this manual as self contained as possible. Nevertheless, there are many sections where an understanding of material from other reference manuals as either implied or strongly recommended. Thus, an understanding of the DEVICES and TASK, utilities discussed in Part II is greatly enhanced by reviewing the material on Terminal Management in the MONROE OPERATING SYSTEM PROGRAMMER'S REFERENCE MANUAL.

While a thorough understanding of the LOAD and RUN commands (also in Part II) is greatly enhanced by a detailed understanding of task establishment. All of these in turn imply a basic understanding of the way programs are assembled and executed; hence beyond a certain level of detail and understanding of assembly language and system concepts is not only helpful but necessary.

On the other hand, the utilities discussed in Part I are almost completely self contained and no knowledge of assembly language or system concepts is presupposed.

# PART I

# DISK MAINTENANCE UTILITIES

# SECTION 2

# ALLOCATE: FILE ALLOCATION UTILITY

# SECTION 2
# ALLOCATE: FILE ALLOCATION UTILITY

## 2.1 INTRODUCTION

The ALLOCATE Command is used to create a direct-access file. It must be emphasized that on an empty mini-floppy diskette there are 1280 free sectors. There are 256 bytes in each sector. Therefore, if you want to allocate continuous sectors you must be certain that you in fact have continuous sectors available to allocate; otherwise you cannot create the file.

## 2.2 ALLOCATE COMMAND

Function: Allocate either a contiguous or indexed direct access file on a diskette.

Mode: Remote

Format:
(a) ALlocate [I] <fd> [,rec length][,sectors] [sectors/blk]
(b) ALlocate,C <fd> <sectors>

Arguments: Switch I indicates that an indexed file is going to be allocated. Switch C indicates that a contiguous file is being allocated. If I is not included in the Command the system assumes an indexed file is being allocated.

File descriptor fd indentifies the file to be allocated where the file type must be specified if the I option is chosen. If the file being allocated is on disk drive FPY0, then the volume name must be included in the fd; otherwise, the file being allocated will default to the system volume.

Rec length is optional and specifies the logical record length in bytes. It cannot exceed 65535 bytes. The default is 0 bytes (variable record length). The file size operand <sectors> specifies the total allocation size in sectors.

Sectors/blk specifies the principal physical block size in sectors being allocated. If the switch C is chosen, the file size operand <sectors> is required, which specifies the total allocation size in 256-byte sectors. This may be any value up to 65,535* CLUSIZE (c.f. The DISKINIT Utility) or the total number of contiguous free sectors existing on the specified volume at the time the command is entered. The size is specified as a decimal number.

Use: Allocate can be used to create either BINARY or ASCII files. After a file has been allocated, the LIB Command can be used to check that the desired file with the indicated characteristics has in fact been created.

Example:

Ex. 1

Allocate, on the system volume, a binary indexed file named THISFILE with a logical record length of 126 Bytes, and default preallocated.

```
-AL THISFILE/B,126¶
-
```

Ex. 2

Allocate, on the Volume MTM a contiguous ASCII file named BIGFILE/Asc whose size is 100 sectors.

```
-AL,C MTM:BIGFILE/A,100¶
-
```

# SECTION 3

# BOOTGEN: DISK BOOTSTRAP GENERATOR

# SECTION 3
# BOOTGEN: DISK BOOTSTRAP GENERATOR

## 3.1 INTRODUCTION

In order for the computer to "boot up" from a disk it must know where on that disk the operating system resides and how to load it into memory. When you are creating a new disk this information must be placed on it. The function of the BOOTGEN Command is precisely to locate the operating system.

## 3.2 BOOTGEN COMMAND

**Function:** Write a loader onto a disk.

**Mode:** The disk device to be written must be opened NON-FILE-STRUCTURED before any write.

**Format:** BOOTGEN,B <fd>

**Arguments:** The device descriptor <fd> is the name of the disk drive together with the name of the file or program to be located (normally the operating system). It is not a volume name.

The switch B indicates that it is a mini floppy disk that is to be booted up.

**Use:** BOOTGEN is generally used with the utilities FORMAT and DISKINIT. As such it is part of the disk initialization sequence. Note the disk to be "bootgened" must reside on disk drive FPY0.

**Example:**

The following procedure is recommended to write the loader onto the disk:

Step 1: Open the disk.

Step 2: Start the program with the Command:
BOOTGEN,B <fd>

# SECTION 4

# CLOSE: CLOSE DEVICE

# SECTION 4
# CLOSE: CLOSE DEVICE

## 4.1 INTRODUCTION

There are many instances when it is necessary to take a device off-line. For example, when dismounting a disk. This is the purpose of the CLOSE command.

## 4.2 CLOSE COMMAND

**Function:** Take a device off-line.

**Mode:** It is assumed that the device is on-line before being closed.

**Format:** CLose <fd>

**Arguments:** The device descriptor fd is the mnemonic name of the device.

**Use:** The CLOSE Command most commonly is used when the mode under which a device is being operated is changed or you are changing disks. For example, in going from file-structured to non-file-structured mode a device is first closed so it can be reopened in non-file-structured mode. The Close Command is rejected if it is directed to a device which is currently assigned by a task. If the device being closed is a direct-access device, then fd is not the volume identifier but the actual device mnemonic.

**Example:** Close disk device 1.

```
-CL FPY1:¶
```

This action removes the volume MS8: from the system. The disk may now be removed or changed. If a direct-access device is dismounted without being CLOSEd, it may only be OPENed on line in the write protect mode.

# SECTION 5

# COMMAND FILE

# SECTION 5
# COMMAND FILE UTILITY

## 5.1 INTRODUCTION

The COMMAND FILE UTILITY allows you to execute a special file, called a Command File. This file's entries can consist of utility commands, task files, and/or entire programs which you wish to run in a specific sequence. For example, formatting, initializing, and bootgening a disk involves executing the utility programs FORMAT, DISKINIT, and BOOTGEN in that order. You can either execute these programs separately or create and execute a command file to execute the three programs in sequence. Obviously this latter procedure would be more efficient when you have a large number of disks to create.

More generally, it is not difficult to imagine situations where you have a series of programs together with utilities involving tasks of a repetitive nature that must be run frequently during specific time of the day, week, or month. A weekly payroll, a daily check on inventory, etc. all involve procedures that must be invoked frequently and repetitively. If these programs need only minor modification or maintenance, it makes much more sense to execute them as a command file rather than tying up an operators time as well as your computer.

Command Files can be created either with the editor or by using the COPYA command (copying from the console to the Command File.fd). They can consist of separate commands, tasks, and programs; or they can invoke other Command Files.

After execution, Command Files are not automatically closed, hence you must run a DISKCHECK before attempting to edit them.

## 5.2 COMMAND FILE PROCEDURE

**Function:** Execute one or more programs, tasks, and/or commands which exist in either a single file or a group of files.

**Mode:** Remote or interactive.

**Format:** !<fd>

**Arguments:** ! indicates that the file <fd> is to be executed.

fd is the file descriptor for the Command File and consists of the Command File's name.

**Examples:** Ex. 1

Create and execute a Command File which consists of a FORMAT, DISKINIT, and BOOTGEN.

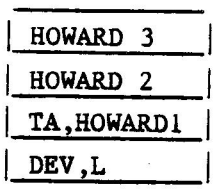
```
-COPYA CON:,CMD$COPY¶
 ASCII Copy Rx-yz¶
 New File¶
-CLOSE FPY1:¶
-FORMAT DEV=M4,DR=FPY1:¶
-OPEN,N FPY1:¶
-DISKINIT DEV=M4,DR=FPY1:,VOL=MOS1,CLEAR¶
-CLOSE FPY1:¶
-OPEN FPY1:¶
-COPYLIB,G,24000 ALEX:,MOS1:¶
-CLOSE FPY1:¶
-OPEN,N FPY1:¶
-BOOTGEN,B FPY1:MS8¶
-CLOSE FPY1¶
-CNTRL=¶
 12 record copied
 End of task
-!CMD$COPY¶
```

Ex. 2

Execute a sequence of nested Command Files.

| Message/Command | Description |
|---|---|
| -COPYA CON:,HOWARD1¶ | Creates the ASCII file named Howard1 whose contents consist of the DEVICES utility program and the DIRECTORY LIST utility program. |
| ASCII copy Rx-yz | |
| New file | |
| -DEV¶ | |
| -L¶ | |
| -CNTRL=¶ | |
| 2 records copied | |
| End of task 0 | |
| -COPYA CON:,HOWARD2¶ | Creates the ASCII file named Howard2 whose contents consist of the TASKS utility program and the command file program !HOWARD1. |
| ASCII copy Rx-yz | |
| New file | |
| -TA¶ | |
| -!HOWARD1¶ | |
| -CNTRL=¶ | |
| 2 records copied | |
| End of task 0 | |
| -COPYA CON:,HOWARD3¶ | Creates the ASCII file HOWARD3 whose contents consist of the command file program !HOWARD2. |
| ASCII Copy Rx-yz | |
| New file | |
| -!HOWARD2¶ | |
| -CNTRL =¶ | |
| 1 record copied | |
| End of task 0 | |

The nesting of the above sequence of programs appears as follows:




If you enter the command -! HOWARD 3¶ this will cause the command file utility to execute the commands in HOWARD 3 which in turn will cause the execution of the commands in HOWARD2. Hence, the TASKS utility will list the status of each task to the console and then the utility programs DEVICEs and DIRECTORY LIST in HOWARD1 will output the devices supported by the operating system, and the Master File Directory on the System Volume, to the terminal. Note, all messages that appear as a result of executing the Command File Program are just the individual messages output to the console by the specific utilities in the file.

Ex. 3

Execute a Monroe BASIC program as part of a Command File.

```
-COPYA,CON:,NEWPGM
 ASCII COPY Rx-yz
 New file
-BASIC SEARCHFILE
-COPYLIB,G,24000 ALEX:SEARCHFILE,MOS1:SEARCHFILE
-L,F MOS1:,SEARCHFILE
-CNTRL=
 3 records copied
 End of task 0
 Upon entering the command
-! NEWPGM
```

The BASIC program SEARCHFILE will execute, after which COPYLIB will copy SEARCHFILE from the system volume named ALEX to the volume MOS1. Then DIRECTORY LIST will output the relevent information about SEARCHFILE to the console. Note when executing SEARCHFILE as a Command File the termination command "bye" must be included as the last line of the BASIC program before the END statement. This is generally true for all BASIC programs which are to be executed as part of Command Files.

# SECTION 6

# COPYA: ASCII COPY UTILITY

# SECTION 6
# COPYA: ASCII COPY UTILITY

## 6.1 INTRODUCTION

Suppose you wish to transfer ASCII data between two devices and/or files. This data can be either formatted (as in the case when you create a file on the EDITOR which automatically compresses any spaces between characters), or unformatted (as might be the case with ASCII data obtained from another operating system). You may want to create a new data file for the second device containing the data from the first, or append that data to an already existing file. The COPY ASCII UTILITY (COPYA) allows you to perform the above. If you have a file with fixed length records, COPYLIB can also be used, however, COPYA has the additiona capability of being able to copy between devices as well as files. COPYLIB, although it can copy between all files, cannot copy between devices. Lastly, COPYA can be used to convert a variable record length file to a fixed record length file provided that the output is preallocated with the AL Command.

## 6.2 COPYA COMMAND

**Function:** Copy ASCII date between two files.

**Mode:** Remote

**Format:** COPYA[,switch] <fd1>,<fd2>

**Arguments:** Switch is optional and can take on the values A, and I.

Switch A is used to append the source file to the destination.

Switch I indicates that data should be transferred as ASCII image data. It is used to copy a variable record length file to a fixed record length file.

File/device descriptor fd1 specifies the source file or the device.

File/device descriptor fd2 specifies the destination file or the destination device (if it does not already exist). If fd2 already exists, the previous content will be destroyed if switch A is not used.

Examples: Ex. 1

Consider the following file directories on the disk devices whose volume names are MONT and PASC.

```
-LIB MONT:
 Directory:  MONT:MFDIR                              42 of 180 entries used.
 CMD$ALDERE     Tsk      CMD$LIB      Tsk      CMD$SPACE    Tsk
 CMD$DEVICES    Tsk      CMD$TASK     Tsk      CMD$TIME     Tsk
 CMD$VOLUME     Tsk      CMD$LDST     Tsk      CMD$POS      Tsk
 BOOTGEN        Tsk      DISKDUMP     Tsk      DISKINIT     Tsk
     .           .          .          .          .          .
     .           .          .          .          .          .
     .           .          .          .          .          .
     .           .          .          .      MFDIR        Mfd
 FIX10          Asc

-LIB PASC
 Directory:  PASC:MFDIR                              43 of 180 entries used.
 BOOTGEN        Tsk      CMD$ALDERE   Tsk      CMD$LIB      Tsk
 CMD$SPACE      Tsk      CMD$DEVICES  Tsk      CMD$TASK     Tsk
 CMD$TIME       Tsk      CMD$VOLUME   Tsk      CMD$LDST     Tsk
 CMD$POS        Tsk      DISKDUMP     Tsk      DISKCHECK    Tsk
 COPYLIB        Tsk      COPYA        Tsk      EDIT         Tsk
     .           .          .          .          .          .
     .           .          .          .          .          .
     .           .          .          .          .          .
 TEMP           Asc      S1           Asc      FIXLEN       Bac
```

Note that FIX10 is an ASCII file on MONT: When you type the command.

```
-COPYA MONT:FIX10,PASC:FIXXTEN¶
```

The following message is displayed on the console:

```
ASCII copy Rx-yz
New file
40 records copied
End of task 0
```

This command has the effect of copying the contents of FIX10 From MONT TO PASC and storing those contents on PASC under the new file name FIXXTEN.

Examing the directory for PASC shows that FIXXTEN has been created as an ASCII file as seen below.

```
-LIB PASC:
Directory:  PASC:MFDIR                          44 of 180 entries used.
BOOTGEN       Tsk       CMD$ALDERE    Tsk       CMD$LIB       Tsk
CMD$SPACE     Tsk       CMD$DEVICES   Tsk       CMD$TASK      Tsk
CMD$TIME      Tsk       CMD$VOLUME    Tsk       CMD$LDST      Tsk
CMD$POS       Tsk       DISKDUMP      Tsk       DISKCHECK     Tsk
COPYLIB       Tsk       COPYA         Tsk       EDIT          Tsk
   .          .            .          .            .          .
   .          .            .          .            .          .
   .          .            .          .            .          .
FIXXTEN       Asc          .          .            .          .
```

Obviously, it is not necessary to change the filename every time you wish to copy data, that is, the filenames fd1 and fd2 can both be the same.

Ex. 2 -
Copy the ASCII file ASC1 from the volume MONT to PASC and append it to ASC2 on PASC.

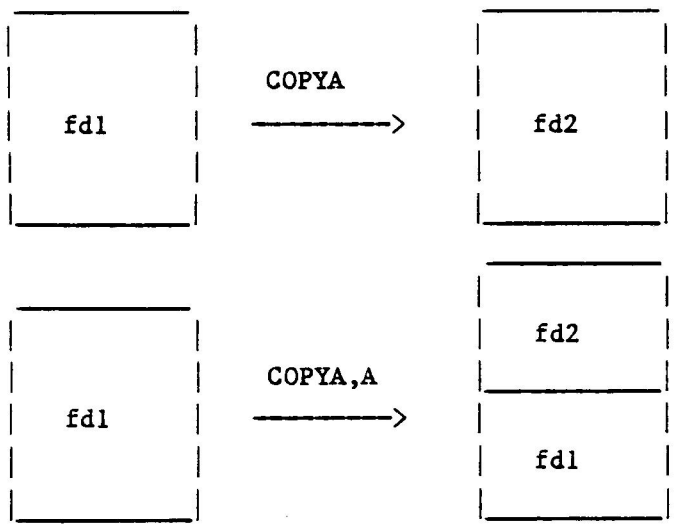
```
-COPYA,A MONT:ASC1,PASC:ASC2¶
 ASCII copy Rx-yz
 14800 Records copied
 End of task 0
```

Ex. 3 -
Copy ASCII data -"END OF JOB2" from the console to the printer.

```
-COPYA CON:,PR:¶
-END OF JOB2¶
-CTRL¶.
 ASCII copy Rx-yz
20000 Records copied
End of task 0
```

Ex. 4 -

The following figures illustrate the COPYA Command both without and with the A option.




Ex. 5 -

Consider the following file, named INPFILE, created by the editor, and whose contents consists of the following three statements:

```
RECORD 1
RECORD  2
RECORD   3
```

Then INPFILE has variable record length. Now use the ALLOCATE Command.

```
-AL OUTFILE/A,10¶
```

To create an output file OUTFILE with fixed record length, enter the Command:

```
-COPYA,I INPFILE,OUTFILE¶
 ASCII copy Rx-yz
New file
End of task 0.
```

Ex. 6

Use COPYA to create a new file called NEWFILE.

| Command/Message | Meanings |
|---|---|
| -COPYA CON:,NEW FILE | Copy the following text from the console to the system volume under the filename NEWFILE. |
| ASCII copy Rx-yz<br>New file | Signon from the program. |
| -.....<br>-..... | Text for the file. |
| -CNTRL =<br>x records copied<br>End of task 0 | Exit from the program. |

## 6.3 MESSAGES AND DIAGNOSTICS

The following messages will be output to the screen:

| | Message | Definition |
|---|---|---|
| a) | ASCII copy Rx-yz | Signon by the program, where the revision level is x, and the update level is yz. |
| b) | New file | If the destination was created. |
| c) | End of tasks | Where s is the SVC error status. Refer to Appendix B. |

Depending upon the error, the following diagnostics will be output to the screen:

| | | |
|---|---|---|
| d) | Errors on input LU 0 | Failed to assign to, or read from, the source. The error status is s. |
| e) | Errors on output LU 1 | Failed to create, or assign/ write to the destination. The error status is s. |

# SECTION 7

# COPYI: IMAGE COPY UTILITY

# SECTION 7
# COPYI: IMAGE COPY UTILITY

## 7.1 INTRODUCTION

The COPYI utility is used to copy and/or verify data between devices and/or files. There are some switches to control the action of the program. It can be used to copy both physical and logical files. It can also be used to verify the integrity of a disk file by doing a bit by bit comparison of that file with an identical file on a backup disk. Lastly it is an excellent way to make back-up copies of master and data disks.

## 7.2 COPYI COMMAND

Function: Performs an image copy and/or verifies data between devices and/or files.

Mode: Remote

Format:
1) COPYI <fd1>,<fd2>
2) COPYI[,switch][,buffsize] <fd1>,<fd2>

Arguments: Switch is optional and can take on the values V or VO. Switch V is used to verify the source file with the destination during a copy. Switch VO is used when no copy shall be done; only a verify of data between the source and destination.

Buffsize is optional and is used to add additional memory to the copy buffer. The memory is specified in bytes and serves to speed up the copy or verification process. Note: If buffsize is used without any switches, two commas are needed before you enter the size of the buffer.

File/device descriptor fd1 indicates the source file or device (e.g., FPY0:).

File/device descriptor fd2 indicates the destination file or device (e.g., PR:). If the output file has not already been allocated before the COPY routine, it is created at the time of the COPY.

Note: If fd2 already exists as a file, its previous contents will be destroyed by the copy.

Examples:

Ex. 1 -

Performs an imabe copy of the file named ERRGEN on the Volume MTM to a file named ERRGEN on the Volume PASC.

```
-COPYI MTM:ERRGEN,PASC:ERRGEN¶

Image copy Rx-yz
New file.
Copy requested!
20 records copied.
End of task 0.
```

(The previous information displayed on the screen indicates that 20 records were copied, and that the file copied was a new file on PASC:).

Ex. 2 -

Performs an image copy of the file named PASTOR on the Volume MTM to a file named PASTOR on the Volume named PASC and then verify the two files.

```
-COPYI,V MTM:PASTOR,PASC:PASTOR¶
```

The following messages is displayed on the screen:

```
Image copy Rx-yz
New file.
Copy requested!
20 records copied.
Verify requested.
Verify o.k!
End of task 0.
```

Ex. 3 -
Perform verification of the file ERRGEN on MTM with the file ERRGEN on PASC.

```
-COPYI,VO MTM:ERRGEN,PASC:ERRGEN
```

The following messages are displayed on the screen:

```
Image copy Rx-yz.
Verify requested!
Verify o.k!
End of task 0.
```

## 7.3 MESSAGE AND DIAGNOSTICS

The program may output the following messages:

| Message | Definition |
|---|---|
| a) Image copy Rx-yz | Signon by the program, where the revision level is x, and the update level is yz. |
| b) New file | If the destination has to be created. |

The following diagnostics will be output to the screen if an error has occurred.

| | |
|---|---|
| c) Errors on input LU 0 | Failed to assign to, or read from, the source. The SVC error status is s. Refer to Appendix B. |
| d) Errors on output LU 1 | Failed to create, or assign/write to the destination. The SVC error status is s. |
| e) Copy requested | Indicates start of the copy phase. |
| f) <nnnn> Records copied | Number of 256 bytes records copied. |
| g) Verify requested | Indicates start of the verify phase. |
| h) Bytes s not equal to c in record r at a | Verify mismatch between source byte s and verified byte c in record number r at relative address a. Each record is 256 bytes, and the address is within a record. |
| i) Verify ok | When no mismatch was found between source and destination. |
| j) End of task 0 | The program terminates. |

# SECTION 8

# COPYLIB: FILE COPY UTILITY

# SECTION 8
# COPYLIB: FILE COPY UTILITY

## 8.1 INTRODUCTION

COPYA allows you to transfer data between files when the records have variable length. Obviously, there are many instances where you would like to transfer ASCII data with fixed record length or unformatted ASCII data. Furthermore, you may also want to copy or delete files whose filenames share certain common characters. COPYLIB is a utility that allows you to transfer data between all files, formatted or unformatted, with fixed length or variable length records. It is a fast method of copying files. In addition, the COPYLIB program can be used to delete files under directory control, and depending on the start switches, may be executed in either interactive or remote mode. The interactive mode works in two phases. The first is a query in which the user is asked what he wishes done with each file. The second phase initiates the actual data transfer. The transfer takes place without any regard to the type of the file, or the content in the file, except transfer of Ascii files to a printer.

## 8.2 COPYLIB COMMAND

Function: Copies data between all files.

Mode: Interactive Remote

Format:
1) COPYLIB[,switch][,buffsize] <fd1>[,,select file]
2) COPYLIB[,switch][,buffsize] <fd1>, <fd2> [,select file]

Arguments: In format 1 switch can take on the values D or DG. D indicates a delete in the interactive mode. DG indicates a delete in the remote mode. To delete a file with COPYLIB the file must be closed. Otherwise, an error occurs.

Select file is the name of a file or device which contains the names of the files to be deleted. It is an ASCII-file containing either a filename or a wild-card specification on each line. When a select file is specified the program enters the remote mode.

The file descriptor fd1 specifies the source delete volume where ordinary filename specifications or wild-card file-name specifications may be used.

In format 2 switch can take on the value G which indicates a copy in the remote mode. When left out the copy will be in interactive mode.

Buffsize is optional and specifies the amount of extra memory (in bytes) needed to speed up the copy operation.

The file descriptor fd1 specifies the source copy volume where ordinary filename specification may be used.

The file/device descriptor fd2 is the name of the volume or device, or perhaps theUser-File-Directory to which the input file or files will be copied.

Select file is the name of a file or device which contains the name of the file to be copied. As in format 1 it is an ASCII-file containing either a filename or a wild-card specification on each line. When a select file is specified the program enters remote mode.

Note: If fd2 already exists as a file, its previous contents will be destroyed by the copy. Ifbuff-size is used without any switches, two commas are needed before you enter the size of the buffer.

Use: The interactive mode has two phases. In the first phase all files which are to be copied or deleted are collected. In the second the actual copy or delete takes place. When a program is entered in the interactive mode, a series of options is displayed. They are:

| Option | Meaning |
|---|---|
| C | copy file. |
| C =file(.element) | Copy, with using of a new name. |
| D | delete file. |
| A | abort. |
| I | ignore rest of library. |
| P | pause. |

A series of question marks (???) is presented after the name of each file in the directory asking you which of the above options you wish to choose. If you wish to copy the file, you type a C after the question marks and the file will be copied. If you wish to ignore the file, you simply press RETURN. If you wish to delete the file, you simply type D and the file will be deleted. If you wish to ignore the presently queried file and all the subsequent files in the directory, type "I".

Lastly, an "A" typed in response to a query will abort the entire session, in which case no files are copied or deleted.

Example: Ex. 1 -

Consider once again, the directories for the devices whose volume names are PASC and MONT.

```
-LIB PASC:
 Directory:  PASC:MFDIR                          43 of 180 entries used.
 BOOTGEN       Tsk       CMD$ALDERE     Tsk      CMD$LIB        Tsk
 CMD$SPACE     Tsk       CMD$DEVICES    Tsk      CMD$TASK       Tsk
 CMD$TIME      Tsk       CMD$VOLUME     Tsk      CMD$LDST       Tsk
 CMD$POS       Tsk       DISKDUMP       Tsk      DISKCHECK      Tsk
    .          .            .           .           .           .
    .          .            .           .           .           .
    .          .            .           .           .           .
 FIXXTEN       Asc       MFDIR          Mfd

 -LIB MONT:
 Directory:  MONT:MFDIR                          42 of 180 entries used.
 CMD$ALDERE    Tsk       CMD$LIB        Tsk      CMD$SPACE      Tsk
 CMD$DEVICES   Tsk       CMD$TASK       Tsk      CMD$TIME       Tsk
 CMD$VOLUME    Tsk       CMD$LDST       Tsk      CMD$POS        Tsk
 BOOTGEN       Tsk       DISKDUMP       Tsk      DISKINIT       Tsk
    .          .            .           .           .           .
    .          .            .           .           .           .
    .          .            .           .           .           .
 DATA10        Bac       PASC           Efd      MFDIR          Mfd
```

If you issue the Command:

-COPYLIB PASC:,MONT:¶

you can interactively query all of the files on PASC and copy those that you want onto MONT. At the beginning of the interactive sessions the following query appears on the screen:

BOOTGEN TSK ???

BOOTEN is the first file in the PASC directory.

Respond to this query. The list of PASC's directory will appear sequentially as queried (CMD$SPACE, CMD$TIME, etc). In response to each query type in one of the commands previously outlined. For example, suppose you want to delete the file FIXXTEN from PASC. Then in responce to the query:

FIXXTEN ASC???D¶

you type in a D. After you have finished the program your file directory will look like this:

```
Directory:  PASC:MFDIR                          43 of 180 entries used.
BOOTGEN      Tsk      CMD$ALDERE    Tsk      CMD$LIB       Tsk
CMD$SPACE    Tsk      CMD$DEVICES   Tsk      CMD$TASK      Tsk
CMD$TIME     Tsk      CMD$VOLUME    Tsk      CMD$LDST      Tsk
CMD$POS      Tsk      DISKDUMP      Tsk      DISKCHECK     Tsk
   .          .           .          .           .          .
   .          .
TEMP         Asc      S1            Asc      FIXLEN        Bac
MFDIR        Mfd
```

with FIXXTEN deleted and MFDIR moved to its place in the list.

Ex. 2

Suppose you want to copy a file from one disk to another under a new file name. For example, copy the file DATA10 on MONT to PASC under the filename HOWARD. Type:

```
- COPYLIB MONT:,PASC:¶

CMD$ALDERE    TSK    ???
```

Each file on MONTS directory will be listed for the user to interrogate. Eventually the user must respond to a query concerning the file DATA10.

In response to the query-

```
   DATA10     BAC     ???
enter:
        C = PASC:HOWARD¶
```

which has the effect of copying the contents of DATA10 to PASC under the filename HOWARD. The new file directory for PASC now looks like:

```
-LIB PASC:
 Directory:  PASC:MFDIR                          43 of 180 entries used.
 BOOTGEN       Tsk       CMD$ALDERE    Tsk      CMD$LIB       Tsk
 CMD$SPACE     Tsk       CMD$DEVICES   Tsk      CMD$TASK      Tsk
 CMD$TIME      Tsk       CMD$VOLUME    Tsk      CMD$LDST      Tsk
 CMD$POS       Tsk       DISKDUMP      Tsk      DISKCHECK     Tsk
    .          .            .          .           .          .
    .          .            .          .           .          .
 DATA10        Bac       HOWARD        Bac      MFDIR         Mfd
```

Ex. 3 - Remote Copy from PASC to MONT.

Type in Command:

```
-COPYLIB,G,14000PASC:,MONT:¶
```

Then all files from PASC will be copied to MONT remotely.

The file directory for MONT now looks like:

```
-LIB MONT:
 Directory:  MONT:MFDIR                          46 of 180 entries used.
 BOOTGEN       Tsk      CMD$ALDERE     Tsk      CMD$LIB        Tsk
 CMD$SPACE     Tsk      CMD$DEVICES    Tsk      CMD$TASK       Tsk
 CMD$TIME      Tsk      CMD$VOLUME     Tsk      CMD$LDST       Tsk
 CMD$POS       Tsk      DISKDUMP       Tsk      DISKCHECK      Tsk
    .          .           .           .           .           .
    .          .           .           .           .           .
 TEMP          Asc      S1             Asc      FIXLEN         Bac
 MFDIR         Mfd      PASSYS         Tsk      PASTRL         Obj
```

Compare this to the directory for MONT in the previous example. Note that the task file, PASSYS and object file PASTRL among others, have been added.

Ex. 4 -
Delete all file three characters long and having the characters A and T in the second and third positions, (* indicates a missing character).

```
-COPYLIB,DG MONT:*AT¶
 COPYLIB Rx-yz
 CAT Asc Deleted
 End of task 0
```

Ex. 5 -

Delete all files having filenames of 3 characters in length and the character A in the second position, (* indicates a missing character).

```
-COPYLIB,DG MONT:*A*¶
 COPYLIB Rx-yz
 RAY Asc Deleted
 End of task 0
```

Ex. 6 -

Delete all files having the characters M and A in the first and second positions, (- indicates all remaining characters are ignored), using the wild-card-specifications.

```
-COPYLIB,DG MONT:MA-¶
 COPYLIB Rx-yz
 MATT Asc Deleted
 End of task 0
```

Ex. 7 -

Delete all files with filename of five characters and having the characters I and D in the fourth and fifth positions, using the wild-card specifications.

```
-COPYLIB,DG MONT:*AT¶
 COPYLIB Rx-yz
 DAVID Bac Deleted
 End of task 0
```

Ex. 8 -

Consider the following directory for MONT:

```
Directory:  MONT:MFDIR                          45 of 100 entries used.
BOOTGEN       Tsk       CMD$ALDERE     Tsk      CMD$LIB        Tsk
CMD$SPACE     Tsk       CMD$DEVICES    Tsk      CMD$TASK       Tsk
CMD$TIME      Tsk       CMD$VOLUME     Tsk      CMD$LDST       Tsk
MAT           Asc       FIXLEN         Boc      GAT            Asc
   .          .            .           .        BAMAT          Asc
   .          .            .           .           .           .
MATT1         Asc       FIX10          Asc      MFDIR          Mfd
```

Note MAT and GAT are the two filenames having three characters in the directory for MONT. Note A and T in the second and third positions in both cases. After typing in

```
-COPYLIB,DG MONT:*AT¶
```

the response

```
MAT          Asc          Deleted.
GAT          Asc          Deleted.
-
```

appear on the screen indicating that the above files in fact have been deleted.

The resulting directory now looks like:

```
-LIB MONT:
 Directory:  MONT:MFDIR                          43 of 100 entries used.
 BOOTGEN       Tsk       CMD$ALDERE    Tsk      CMD$LIB       Tsk
 CMD$SPACE     Tsk       CMD$DEVICES   Tsk      CMD$TASK      Tsk
 CMD$TIME      Tsk       CMD$VOLUME    Tsk      CMD$LDST      Tsk
 CMD$POS       Tsk       DISKDUMP      Tsk      DISKCHECK     Tsk
    .          .            .          .           .          .
    .          .            .          .           .          .
 GAMAT         Asc       MATT1         Asc      FIX10         Asc
 MFDIR         Mfd
```

Note also that although the file named GAMAT ends in A and T it has not been deleted. This is because the A and the T appear in the fourth and fifth positions rather than the second and third, furthermore GAMAT has five characters in its filename, not three.

If you wanted to delete GAMAT you would type:

```
-COPYLIB,DG MONT:*A*A*¶
```

which will delete all five character filenames having A in the second and fourth character positions. The response:

```
GAMAT          Asc          Deleted.
```

indicates that the file has been deleted.

Ex. 9 -
Copy all files on the volume named MONT having first three characters BAS onto the volume named PASC.

```
-COPYLIB,G,14000 MONT:BAS-,PASC:¶
```

The response at the terminal is:

```
BASIC          Tsk          33792 Bytes Copied
BASICERR       Tsk           8192 Bytes Copied
```

The resulting file directory for PASC is:

```
-Lib PASC:
 Directory:  PASC:MFDIR                          41 of 180 entries used.
 BOOTGEN       Tsk      CMD$ALDERE     Tsk      CMD$LIB        Tsk
 CMD$SPACE     Tsk      CMD$DEVICES    Tsk      CMD$TASK       Tsk
 CMD$TIME      Tsk      CMD$VOLUME     Tsk      CMD$LDST       Tsk
 CMD$POS       Tsk      DISKDUMP       Tsk      DISKCHECK      Tsk
   .            .          .            .          .            .
   .            .          .            .          .            .
 HOWARD        Asc      MFDIR          Mfd      BASIC          Tsk
 BASICERR      Asc
```

Ex. 10 -
Copy all User files from the User-File-Directory ASC on the volume named MONT to the volume named PASC, and expand these to ordinary files.

```
-COPYLIB,G,14000 MONT:ASC,PASC:¶
```

Although these files will be copied to PASC they will not appear on PASC as User files. Only as ordinary files in PASC's directory. In order for them to appear as User files in PASC's UFD you have to type:

```
-COPYLIB,G,14000 MONT:ASC,PASC:ASC¶
```

Ex. 11 -
Copy each user file beginning with FIX in the User File Directory ASC from the volume named MONT to the volume named PASC.

```
-COPYLIB,G,14000 MONT:ASC.FIX-,PASC:¶
```

Ex. 12 -
Copy the files from the select file SELECTFILE on MONT to PASC: Note SELECTFILE must be created as a select file before the copy can be done. To do this type

```
- EDIT SELECTFILE
> RE
> IL
    1. # GAMAT
    2. # MATT1
    3. # FIX10
    4. ##
> EN
```

This creates SELECTFILE as a select file. Then type in the command:

```
-COPYLIB MONT:,PASC:,PASTOR¶
```

## 8.3 MESSAGES AND DIAGNOSTICS

This program may display any of the following messages:

| Message | Meaning |
|---|---|
| a) Copylib Rx-yz | Signon by the program, where the revision level is x, and the update level is yz. |
| b) End of task s | Where s is the SVC error status. Refer to Appendix B. |

Depending upon the error, the following diagnostic will be output to the screen.

| | |
|---|---|
| c) Not restartable, reload program | It is impossible to restart the program without reloading it first. |
| d) Assign error | Failed to assign input and output, or end of media, or not enough contiguous space. |
| e) Input and output not directory oriented | Invoked device of wrong type. |
| f) Table full | The interactive response table is full. |
| g) Error at read | When failed to read. |
| h) Error at write | When failed to write. |

# SECTION 9

# COPYT: COPY TASK UTILITY

# SECTION 9
# COPYT: COPY TASK UTILITY

## 9.1 INTRODUCTION

Every system disk must contain at least one task file, namely, the operating system; otherwise, your computer would be useless as a problem solving device. Note that a task file can have one of two designations, either as an absolute or relocatable task file. The memory address of an absolute file is specified during the file creation stage, hence task files can be assigned specific memory locations.

A relocatable task file, on the other hand, can be loaded (relocated) anywhere in memory (where there is space available). The COPY TASK UTILITY, COPYT is used to copy both absolute or relocatable task files between devices and/or files.

## 9.2 COPYT COMMAND

**Function:** Copies task files between devices and/or files.

**Mode:** Remote

**Format:** COPYT <fd1>,<fd2>

**Arguments:** fd1 is the file descriptor for the source file to be copied.

fd2 is the file descriptor of the file which is to contain the copied data.

**Note:** If fd2 already exits, its previous contents will be destroyed by the copy.

**Examples:** Ex. 1 -
Consider the following file directory for the Volume named MONT.

```
-LIB MONT:
 Directory:  MONT:MFDIR                          38 of 180 entries used.
 BOOTGEN       Tsk      CMD$ALDERE     Tsk      CMD$LIB        Tsk
 CMD$SPACE     Tsk      CMD$DEVICES    Tsk      CMD$TASK       Tsk
 CMD$TIME      Tsk      CMD$VOLUME     Tsk      CMD$LDST       Tsk
 CMD$POS       Tsk      DISKDUMP       Tsk      DISKCHECK      Tsk
    .          .           .           .           .           .
    .          .           .           .           .           .
 FIX10         Asc      MFDIR          Mfd
```

The Command:

```
-COPYT PASC:PASCAL,MONT:PASCAL¶
```

has the effect of transferring the task file named PASCAL from the Volume named PASC to the Volume named MONT under the filename PASCAL. Note: Once you press the return key the following message appears on the console:

```
Copy task Rx-yz
    68 Records
End of task 0
                -
```

This indicates that 68 records have been copied. Note: You do not need to copy a file under the same filename. If you enter:

```
-COPYT PASC:PASSYS,MONT:HENRY¶
```

the task file PASSYS on PASC is copied under the filename HENRY to MONT. The directory for MONT now looks like this:

```
-LIB MONT:
 Directory:  MONT:MFDIR                          40 of 180 entries used.
 BOOTGEN       Tsk       CMD$ALDERE    Tsk      CMD$LIB       Tsk
 CMD$SPACE     Tsk       CMD$DEVICES   Tsk      CMD$TASK      Tsk
 CMD$TIME      Tsk       CMD$VOLUME    Tsk      CMD$LDST      Tsk
 CMD$POS       Tsk       DISKDUMP      Tsk      DISKCHECK     Tsk
 DISKINIT      Tsk       COPYLIB       Tsk      FORMAT        Tsk
    .          .            .          .           .          .
    .          .            .          .        HENRY         Tsk
 COPYI         Tsk       PASCAL        Tsk      BASIC         Tsk
```

with PASCAL and HENRY both resident as task files.

## 9.3 MESSAGES AND DIAGNOSTICS

The COPYT program may display the following messages:

| Messages | Meaning |
|---|---|
| a) Copy task Rx-yz | Signon by the program, where the revision level is x, and the update level is yz. |
| b) End of tasks | Where s is the SVC error status. Refer to Appendix B. |

The following diagnostics will appear if an error has occurred:

| | |
|---|---|
| c) Bad input file descriptor | Syntax error in source name. |
| d) Bad output file descriptor | Syntax error in destination name. |
| e) Input assign error s | Failed to assign the input file, where s is the SVC error status. Refer to Appendix B. |
| f) Input format error | The input is not a task file. |
| g) Output assign error s | Failed to create and assign the output file, where s is the SVC error status. Refer to Appendix C in the MONROE OPERATNG SYSTEM PROGRAMMERS REFERENCE MANUAL. |
| h) Output error | Failed to write to the output. |
| i) Input error | Expected more data, didn't find end-of-file or timeout occurred. |
| j) Record size error | Input data not module 256 bytes. |
| k) (nnnnn) Records copied | Number of records copied. |

# SECTION 10

# CREINDEX: CREATE INDEX UTILITY

# SECTION 10
# CREINDEX: CREATE INDEX UTILITY

## 10.1 INTRODUCTION

Utility program CREINDEX is used to allocate and create an ISAM index file and its associated data file. ISAM, Indexed Sequential Access Method, is a technique used for indexed access to large data files. It can be used for random access using a key string as the search argument, or sequential access using the index.

The data in the data file is divided into RECORDS. The records have a fixed, user definable length, and they are stored in a fixed record length file, the DATA file. Each data file has an ISAM file associated to it.

The ISAM file may contain up to ten indices into the data file. Each index has a symbolic name. It contains one KEY for each data record. The key consists of a key string, which also is a part of a data record, and a pointer to that record. The keys are ordered within the index to form a B-tree structure.

All record pointers are logical and file reference is symbolic. This means that the data and ISAM files may be copied and utilized on any random-access device supported by the operating system.

The ISAM file is initialized by this utility program. After initialization, the ISAM and data files are built by the user using ISAM-write operations. Since the index trees are built in a well-structured way, there is no need for time consuming reorganizations once the indices are established. The access times will always be at an optimum.

## 10.2 CREINDEX COMMAND

**Function:** Allocates ISAM index and data files.

**Mode:** Interactive.

**Format:** CREINDEX¶

**Arguments:** None.

**Use:** To create an ISAM Indexed File the user must type in commands in response to queries from the console. These queries concern the defining parameters of the file being created.

Five different formats are defined for the key strings. The formats are:

B-Binary
This is a string of bytes of selectable length. The string is interpreted as an unsigned binary integer, most significant byte first.

A-ASCII
This is a string of bytes of selectable length. The bytes are interpreted as 7-bit ASCII characters. Upper and lower case characters have the save value.

I-Integer
This is a string of two bytes, representing a signed integer, least significant byte first. Compatible with BASIC and PASCAL formats.

F-Floating Point

This is a string of four bytes, representing a single precision floating point number. Compatible with BASIC and PASCAL formats.

D-Double Precision Floating Point

As above, but string length is eight bytes. Compatible with BASIC format.

The ISAM file format is built on the B-tree concept. This concept makes it possible to maintain the search path through the tree at an optimum through insertions and deletions of key items.

The first record of an ISAM file is a header record. It contains information about the ISAM-file and the data file it indexes.

An ISAM file may contain up to ten separate indices with symbolic names. All information about the indices e.g. symbolic name, key type, key positions, key length and the B-tree root pointer is stored in the ISAM-file header. The ISAM-file contains one B-tree for each index.

The Multi-Task ISAM facility makes it possible for several users to use the same data base. Each user may also use more than one data base. The facility is added to the multi-task operating system by loading and starting an ISAM task. Each user may then assign to the ISAM task, and the ISAM task will assign the selected ISAM/data files. All input/output operations will then take place through the ISAM task which will coordinate the different users and their requests.

When the CREINDEX Utility is run the user is prompted as follows.

```
Enter name of index file?_____
Enter name of data file?_____
Enter record length?_____
Enter key start position?_____
Enter key type (B,A,I,F or D)? _____
Ascending or Descending sequence (A/D)?_____
Are duplicate key values allowed? (Y or N)_____
Are there any more indices? (Y or N)?_____
Is information correct (Y or N)?_____
```

If there are any more, indices the user is returned to the first query inputting the name of the index file, the name of the data file, and so on, until all indices have been entered. Then a table is output to the console summarizing all of the information entered during the session.

Examples:

Ex. 1

Create an ISAM index file called FILE with one index, a key start position at byte 10, and a key length of 10 bytes.

```
-CREINDEX¶
Enter name of index file? FILE¶
Enter name of data file? FILE¶
Enter record length? 80¶
Enter name of index? NAME¶
Enter key start position? 10¶
Enter key length? 10¶
Enter key type (B,A,I,F or D)? A¶
Ascending or descending sequence (A/D)?A¶
Are duplicate key values allowed? (Y or N)?Y¶
Are there any more indices? (Y or N)? N¶
```

The following information appears on the screen:

```
Create ISAM Files Ver. P-3.00  1981-07-15/00.26.12

Data and Index File Information.

Index File name: file
Data File name : file
Record size    : 80
```

| Index No. | Index Name | Key Type | Sort order | Dupl. | Key Start/Length |
|---|---|---|---|---|---|
| 1 | name | Ascii | Descending | Yes | 10/10 |

```
Is information correct (Y or N)? Y¶
Would you like a copy on the printer (Y or N)? N¶
```

The program ends with the message

```
Index file created!
Data file created!
End of task 0.
```

Ex. 2

Use the illustration in Ex. 1 to define an ISAM index file with three indices.

```
Enter name of index file? INFILE¶
Enter name of data file? DATFILE¶
Enter record length? 80¶
Enter name of index? NAME1¶
Enter key start position? 10¶
Enter key length? 10¶
Enter key type (B,A,I,F or D)? A¶
Ascending or Descending sequence (A/D)? A¶
Are duplicate key values allowed?  (Y or N)?  Y¶
```

```
Are there any more indices? (Y or N)? Y¶
Enter name of index? NAME2¶
Enter key start position? 20¶
Enter key length? 10¶
Enter key type (B,A,I,F or D)? A¶
Ascending or Descending sequence (A/D)? D¶
Are duplicate key values allowed? (Y or N)? Y¶
Are there any more indices? (Y or N)? Y¶
Enter name of index? NAME3¶
Enter key start position? 30¶
Enter key length? 10¶
Enter key type (B,A,I,F or D)? I¶
Ascending or Descending sequence (A/D)? A¶
Are duplicate key values allowed? (Y or N)? N¶
Are there any more indices? (Y or N)? N¶
```

The following output appears on the console:

```
Create Isam Files  Ver. P-3.00  1981-97-15/00.04.20

     Data and Index File Information.

     Index File name:  infile
     Data File name :  datfile
     Record size    :  80
```

| Index No. | Index Name | Key Type | Sort order | Dupl. | Key Start/Length |
|---|---|---|---|---|---|
| 1 | name1 | Binary | Ascending | Yes | 10/10 |
| 2 | name2 | Ascii | Descending | No | 20/10 |
| 3 | name3 | Integer | Ascending | Yes | 30/10 |

```
     Information correct (Y or N)? Y¶
     Would you like a copy on the printer (Y or N)? N¶
```

Note: Suppose that you make a mistake during the interactive session. You would then respond to the query:

"Information correct (Y or N)?"

with an "N". A series of prompts will ask if you want to change any of the information in the table, i.e, the Index No., Index Name, Key Type, etc.

If you want to abort the entire session, type "A" in response to the query:

```
Abort, modify file or index information
(A,F,I)?-A¶
```

Type "A" and the session will be aborted.

# SECTION 11

# DELETE FILES COMMAND

# SECTION 11
# DELETE FILES COMMAND

## 11.1 INTRODUCTION

The Delete Command is used to delete a file from a specified volume.

## 11.2 DELETE COMMAND

**Function:** Deletes a file.

**Mode:** Remote

**Format:** DELete <fd1>[,fd2]....[,fdn]

**Arguments:** fd1, fd2, ..., fdn are the filenames of the files to be deleted.

**Examples:** Ex. 1 -
Delete the ASCII file BIGFILE.

```
-DEL BIGFILE/A¶
```

Ex. 2 -
Delete Binary files XRAY and ZRAY.

```
-DEL XRAY/B,ZRAY/B¶
```

## 11.3 MESSAGES AND DIAGNOSTICS

| Message | Meanings |
|---|---|
| Name Error | Failed to supply a file type in the fd. |
| I/O Error | File is open. |

# SECTION 12

# DISKCHECK: DISK INTEGRITY CHECK

# SECTION 12
# DISKCHECK: DISK INTEGRITY CHECK

## 12.1 INTRODUCTION

The Disk Integrity Check, DISKCHECK, provides a means of recovering open disk files following an operating system crash. The program closes all files found to be assigned, and validates some control information on the disk.

## 12.2 DISKCHECK COMMAND

**Function:** Checks the integrity of an open disk file.

**Mode:** Remote.

**Format:** DISKCHECK <fd1>[,fd2]

**Arguments:** The device descriptor fd1 is the device name of the disc drive on which the disk resides.

The device descriptor fd2 is an optional mnemonic for one or more output devices (PRINTER, or TERMINAL, etc.) to which your output is the result of the check. If the above options are not used, the results of the DISKCHECK default to the console.

**Use:** In order to use DISKCHECK you have to OPEN the device you want to check. This is normally done in non-file-structured mode.

**Examples:** Ex. 1 - Diskcheck In Non-file-structured Mode

```
-CLOSE FPY1:¶          Close drive FPY1.
-OPEN,N FPY1:¶         OPEN drive FPY1 in
                       non-file-structured
                       mode.
-DISKCHECK FPY1:,PR:¶
-CLOSE FPY1:¶
-OPEN FPY1:¶
```

You will get a return message on the Printer (for example)-

```
COPYI        was assigned for Read
BASIC        was assigned for Read
FIX10        was assigned for Read
-
```

which means that all of the above files had previously been assigned to be Read.

Note: After the DISKCHECK you must CLOSE and OPEN the drive FPY1 again for normal use before continuing processing. This insures that FPY1 has the structure of a file-structured device so you can continue processing.

## 12.3 MESSAGES AND DIAGNOSTICS

This program may output the following messages:

| Messages | Meaning |
|---|---|
| a) Diskcheck Rx-yz | Signon by the program, where the revision level is x, and the update level is yz. |
| b) Please reload program, not restartable | The program must be loaded prior to each new session. |
| c) End of task s | Where s is SVC the error status. Refer to Appendix B. |
| d) No param. | Start parameter missing. |
| e) Inv. name | Syntax error in file-descriptor or device name missing. |
| f) Disk-asgn | Failed to assign the device s. |
| g) Disk-Re/Wr | Directory read/write error. |
| h) Error in hash key sector in tFD directory | Invalid data in a type t directory. |
| i) Re/Wr error in tFD | Read/write error in type t directory. |
| j) (filename) has no first index sector, is deleted | A file only existing as a name in the directory is deleted. |
| k) (filename) was assigned for read/write | An open file is closed. |

# SECTION 13

# DISKINIT: DISK INITIALIZER

# SECTION 13
# DISKINIT: DISK INITIALIZER

## 13.1 INTRODUCTION

The Disk Initializer, DISKINIT, initializes a disk for use with the Monroe Operating System. Initialization includes placing the Volume Name as well as a pointer to the Bit Map and Master File Directory on the Volume Descriptor, which is located on the first sector of the disc.

The Volume Name consists of one to four characters, the first of which must be alphabetic. This name identifies the disk to the system. The DISKINIT allows a disk to be named or renamed.

The Master File Directory describes all files on the disk, while filenames and the starting sector address identify each file.

DISKINIT allows the user to clear the Directory and Bit Map in order to delete all files. DISKINIT also provides a facility for clearing a new disk.

Note: The new disk must be formatted (FORMAT) before it is initialized.

## 13.2 DISKINIT COMMAND

**Function:** Initializes a new disk.

**Mode:** Interactive and Remote.

**Format:** DISKINIT.

**Arguments:** None.

**Use:** Seven options are available with the DISKINIT program: CLEAR, NOREADCHECK and READCHECK, CLUSIZE, BLOCKSIZE, DEFAULT and DIRECTORY.

Whenever command CLEAR is specified, a READCHECK operation checks the disk for bad sectors, unless the command NOREADCHECK is given. These sectors are marked as unavailable in the Bit Map. If the first sector on the disk fails the read check operation, the disk cannot be used and a message to that effect is printed. Because media degradation may occur at any time, there are instances in which sectors not flagged at format time are flagged as bad sectors during initialization. In such instances, it is recommeded that the disk be backed-up, re-formatted and re-initialized. The data can then be restored.

The command NOREADCHECK could be given to speed up initialization, and the disk will not be checked for bad sectors. This command should not be used if the disk is not known to be in "top shape", and is primarily used for test purposes.

The command READCHECK tests the integrity of each sector on the disk and flags off all bad sectors so they cannot be used. If the CLEAR option is used, READCHECK is automatically invoked.

The CLUSIZE command is used to specify the smallest allocatable element on the disc, and is given in sectors. The value must be a power of 2. e.g. 1, 2, 4, 8, 16 etc. Maximum value is 128.

The BLOCKSIZE command is used to specify the default blocksize in sectors, at SVC7 allocate, (refer to SVC7 FILE HANDLING in the MONROE OPERATING SYSTEM PROGRAMMER'S REFERENCE MANUAL). The value given is rounded off to a multiple of the CLUSIZE. The value may range from 1 to 255.

The DEFAULT command is used to specify the number of blocks specified in BLOCK SIZE to be allocated as default, if not given in the SVC7 Block. The value given may range from 1 to 65535.

The DIRECTORY command may be used to locate the Master File Directory (MFD) at some location other than the default location on the disk. A second parameter is used to specify the directory size in blocks of 16 sectors. Currently only a directory size of 1 is supported.

All information required by the DISKINIT is specified within the DISKINIT command or if no start parameters are given in Interactive Mode. The following information appears on the screen:

```
Diskinit     Rx-yz
Enter nondefault parameters
Devtype = M4

Drive = Dev     Device name:
Volume = Name   Volume name:
```

The following keywords which are required must be entered at the terminal in the specified order.

1) DEVtyp = t
   The type of device, which is presented in a menu by the program.

2) DRive = fd
   Where fd is the name of the disk device.

3) Volume = xxxx
   Where xxxx is the volume name to be given to the disk pack.

The following commands are optional, and may be entered in any order:

-PARameters
Will display the actual setting of initialization parameters. This command will not affect any parameters.

-STart
Will start the initialization.

-ENd

Terminates the program execution.

-CLEar

Specifies a clear disk and read check operation. When entered, all files are deleted from the disk. A read check of the entire disk is performed and bad sectors are flagged in the Bit Map. All flagged sectors are identified in a message by their decimal sector addresses on the disk.

-NOreadcheck

Specifies whether the disk should be checked for bad sectors or not. If given, no read check will be performed. Should be used with care.

-REadcheck

Similar to NOreadcheck, but will turn readcheck on.

In addition there are three more interactive commands:

CLUsize=n

Specifies clustersize in sectors where n must be a power of 2. This command will affect the size of the Bit Map.

Blocksize=n

This parameter specifies the default block size in a file, and is given in sectors. The number of sectors is, if not a multiple of clustersize, rounded off at SVC7 allocate. N may be in the range from 1 o 255.

DEFault=n

The parameter given will specify the number of blocks to be preallocated at allocation time. The value n is the number of blocks ranging from 1 to 65535.

DIRectory=lsa/n

This command is used to change the default location/size of the Master File Directory. This command should be used when any of the system sectors is found to be bad. The starting sector should be chosen so it will not interfere with any bad starting sector. The size in sectors is calculated as (n*16+1+bitmapsize) divided by (CLUSIZE*CLUSIZE+CLUSIZE).

Ex. 1

This is an example of an initialization of a disk.

| Command/Message | Meanings |
|---|---|
| -CLOSE FPY0:¶ | Close the drive if it is opened, then mount the disk to be initialized. |
| -OPEN,N FPY0:¶ | Open the drive non-file-structured. |
| -DISKINIT¶ | Load and start the DISKINIT program. |
| Diskinit Rx-yz | Signon by the program, followed by a menu of the possible device types. |
| -DEV M4¶ | Device type is Mini-Floppy. |
| -DR FPY0:¶ | The drive where the disk is. |
| -V=FIXX¶ | Gives the disk the name. |
| -CLEAR¶ | Clear and readcheck of the disk. |
| -STart¶ | Start the initialization. |
| No sectors flagged | When finished, the program presents a table of disk characteristics. |
| -END¶ | Terminate the program. |
| -CLOSE FPY0:¶ | Close the drive. |
| -OPEN FPY0:¶ | Open the drive file-structured if you want to use the disk immediately. |

-

After the initialization the following information will be output to the screen:

```
No sectors flagged.
Diskinit completed.
==INITIALIZATION PARAMETERS==
Volume name = FIXX:
Master file directory = 1 Segments.
Directory start loc. =      32
Allocation table size = 1 Sectors.
Cluster size =          1 Sectors.
Default blocksize =     4 Sectors.
Default allocation =    1 Blocks.
```

Ex. 2

Initialize a Disk doing a CLEAR but not a READCHECK.

```
-CLOSE FPY0:¶
-OPEN,N FPY0:¶
-DISKINIT¶
 Diskinit Rx-yz
-DEV M4¶
-DR FPY0:¶
-V=MOS2¶
-CLEAR¶
-NOREADCHECK
-ST
 No sectors flagged
-END¶
-CLOSE FPY0:¶
-OPEN FPY0:¶
-
```

Ex. 3

Initialize a disk using the PARAMETERS option and specifying a cluster size of 2 (e.g. initialize the 1280 sectors 2 at a time).

```
-CLOSE FPY0:¶
-OPEN FPY0:¶
-DISKINIT¶
 Diskinit Rx-yz
-DEV M4¶
-DR FPY0:¶
-V=DEN2¶
-CLEAR¶
-PAR¶
-CLU=2¶
-ST¶
 No sectors flagged
-END¶
-CLOSE FPY0:¶
-OPEN FPY0:¶
-
```

## 13.3 MESSAGES AND DIAGNOSTICS

DISKINIT may display the following messages:

| Message | Meaning |
|---|---|
| a) Diskinit Rx-yz | Signon by the program, where the revision level is x, and the update level is yz. |
| b) More than 65535 sectors flagged, format disk | Readchecks finds an extremely bad disk. The disk should be formatted prior to use. |
| c) Missing nondefault parameter | The START command was given and any of the commands DEVICE, DRIVE or VOLUME was not given. |
| d) Sequence | The non-default commands are not given in proper order. |
| e) No sectors flagged | No sectors on the disk were found bad. |
| f) Directory write error | The program has failed to write on the disk. Write protect switch should be checked. |
| g) System sector flagged in readcheck | The program should be rerun, and the DIRectory command should be used to locate the directory at another starting position. |
| h) Readcheck error! local sector nnn(10) flagged off | One sector is found to be bad and is marked busy in the Bit Map. nnn is the logical sector address on the disk. 10 indicates that nnn is a decimal value. |

| Message | Meaning |
|---|---|
| i) Diskerr er=ffQ return | An unexpected error has occurred at disk read or write. FF is the SVC function code in octal; Q is the SVC error status in octal. See Appendix B. |
| j) Cmd-err, type=tttt | A command was not recognized or was in bad format. |

tttt is the error specification as:

| | |
|---|---|
| k) Dev-type | Device type not found in the default table. |
| l) Invalid parameter | Bad format or can't be accepted due to range. |
| m) Invalid fd | Bad format or device not found. |
| n) Unknown command | Command not recognized. |
| o) Vol-name | Bad format on volume name. |
| p) Too many arguments | Expected comma or end, found another parameter. |
| q) Missing nondefaulted parameter | The START command was given and any of the commands DEVICE, DRIVE or VOLUME was not given. |
| r) Cat out of disc | The start position given in DIRECTORY command will not give sufficient space for the Bit Map sectors. |
| s) Sequence | The nondefault commands are not given in proper order. |

# SECTION 14

# FORMAT: DISK FORMATTER

# SECTION 14
# FORMAT: DISK FORMATTER

## 14.1 INTRODUCTION

In order to read from or write on a disk it must be formatted, that is, appropriate sectors and tracks must be designated using magnetic codes. The Disk Formatter (FORMAT) generates the magnetic structure on the disk, such as the pre-amble with the address mark, a data section with dummy data, the post-amble containing the checksum and the inter-record gap.

## 14.2 FORMAT COMMAND

**Function:** Prepares a disk for initialization.

**Mode:** Interactive and remote.

**Format:** FORMAT

**Arguments:** None

**Use:** To format a disk, first set the disk off-line using the CLOSE command, then start the program by the command:

-FORMAT

When started, the program displays a menu of available prompts, which appear on the screen. They are:

| | |
|---|---|
| DEvtype | M4-Minifloppy quad. |
| DRive | Device name. |
| Fill n | Fill number. |
| Interval x-(y) | Format cyl. x to y. |
| Parameter | Types the parameters |
| Start | Starts the formatting. |
| Help | Types this text. |
| End | Exits. |

In the above menu of prompts some require responses and some are optional when doing a format.

Required Responses

| Prompt | Meaning |
|---|---|
| DEVtype | The type of disk-device; in this case a Minifloppy quad, Enter:M4 |
| DRive | The name of the disk drive on which the disk being formatted resides (Enter: FPY0 or FPY1). |

After entering FPY0 or FPY1, the user must start the program. Enter: ST¶

The program is terminated by entering: End¶

Optional User Entries:

| Command | Meaning |
|---|---|
| Fill n | Fill the decimal value n into all sectors. The default is 299 (E5H). |
| Interval x-[y] | Specifies the interval to be formatted on the disk. Formatting begins on the track containing sector x and ends on the track containing sector y (if included). On a minifloppy you cannot format just one sector. For example, if you enter interval 10, the program will not just format sector 10 but rather will format the entire track containing sector 10. |
| Help | Will display the available commands. |
| Parameter | Will display the actual setting of the formatting parameters. |

Ex. 1 -
The following commands will FORMAT a disk.

```
-CLOSE FPY0:¶
```

| | |
|---|---|
| -FORMAT¶ | Load and start the program. |
| Disk Formatter | Signon by the program, then the command menu is displayed on the terminal. |
| -DEV M4¶ | Specify device. |
| -DR FPY0:¶ | Specify drive. |
| -ST¶ | Start formatting all the tracks. |
| -END¶ | Terminate the task. |
| End of task 0 | The program terminates. |

```
-
```

Ex. 2 -
Format a disk filling in all sectors with the decimal value 10 (OAH).

```
-CLOSE FPY0¶
-FORMAT¶
 Disk Formatter
-DEV m4¶
-DR FPY0¶
-FILL=10¶
-ST¶
-END
 End of task 0
```

Ex. 3 -
Format all tracks containing sectors 10 through 20 on a disk.

```
-CLOSE FPY0¶
-FORMAT¶
 Disk Formatter¶
-DEV M4¶
-DR FPY0¶
-Interval 10-20¶
-ST¶
-END¶
 End of task 0
```

## 14.3 MESSAGES AND DIAGNOSTICS

The FORMAT program may display the following messages:

| Message | Meaning |
|---|---|
| a) Disk formatter | Signon by the program. |
| b) Command error | When an unknown command is entered. |
| c) Assign error | Failed to assign the device specified in the DRIVE command. |
| d) Not implemented in the version | The device type specified in the DEVTYPE command is not supported. |
| e) Interval out of disk | The interval specified within command INTERVAL command is outside the disk. |
| f) Undefined drive | The required command DRIVE has not been entered. |
| g) Undefined device | The required command DEVTYPE has not been entered. |
| h) Disk not ready | Time-out on the disk drive. |
| i) I/O error | When an error is detected during disk read or disk write, the return status is s. The Return Status Code is the same as found under SVC1. |
| j) End of task s | The program terminates where s is the SVC error status. Refer to Appendix B. |

# SECTION 15

# LIB: DIRECTORY LIST

# SECTION 15
# LIB: DIRECTORY LIST

## 15.1 INTRODUCTION

The LIB command is used to display the contents of a volume. Suppose you create a file and you want to examine its size, record length, the time it was created, the last time it was updated, or the last time it was used. You can use the LIB Command to display this (and other) information to the console or printer.

## 15.2 LIB COMMAND

Function: Displays the contents of a volume to the terminal or printer.

Mode: Remote

Format: L[,F][[fd1],[fd2][,fd3]]

Arguments: Switch F is used to display all information about each file, such as its length, creation date, and so on.

The file descriptor fd1 is an optional file descriptor which contains the name of the volume and or the directory from which the display will be done. If omitted, the system volume will be used.

The file descriptor fd2 is the filename of a unique file, or a wild card specification of a group of files to be displayed. If omitted, all files will be displayed.

The file/device descriptor fd3 is the name of the device or the file where the listing should be directed. If omitted, the terminal device is assumed.

Note: If fd1, fd2 and fd3 are omitted in the above format, information on all files on the system volume will be displayed on the console. If you want to stop the output of Lib to the console press CNTRL-A. To resume the output again press RETURN.

Examples: Ex. 1

List all file names on "MSTM" to the printer.

```
-L MSTM:,,PR:¶
```

Ex. 2

Display the name, file modifier, type, record length, size, and other relevent information about BIGFILE on volume PASC on the console.

```
-L,F PASC:,BIGFILE¶
```

```
Directory:  PASC:MFDIR  48 of 180 entries used.  1981-04-15 00.00.54
  Name       Mod    RW  Type  Recl  Size     Time Created
BIGFILE      Asc          co      0  2560    81-03-05  00.09.00

     Time Last updated       Time Last used
   00-00-00  00.00.00      00-00-00  00.00.00
```

Ex. 3

Print out the same information as above.

```
-L,F PASC:, BIGFILE,PR:¶
```

Ex. 4

Use the wild-card option to display the above information for all files having first three characters CMD (where the remaining characters are ignored).

```
-L,F PASC:,CMD-¶
```

Ex. 5

Do the same for all ASCII files having A and C in the second and third positions (the remaining files being ignored).

```
-L,F PASC:,*AC-/A¶
```

# SECTION 16

# OPEN: OPEN DEVICE

# SECTION 16
# OPEN: OPEN DEVICE

## 16.1 INTRODUCTION

The OPEN channel is used to bring on-line a device that was previously off-line.

## 16.2 OPEN COMMAND

**Function:** Brings on-line a device that was previously off line.

**Mode:** Remote

**Format:**

1. OPEn <fd>:
2. OPEn[,N] <fd>:
3. OPEn[,P] <fd>:

**Arguments:** The device descriptor fd is the mnemonic name of the physical device. After opening a direct-access device, the volume name associated with it is output to the console device.

If the optional switch P is specified, the device is opened as write protected. If the device is hardware write protected, it will be automatically opened as protected.

The optional switch N is used to open a direct-access device non-file-structured. This means that no directory is present and that no volume name will be established at open.

While a device is off-line, it cannot be assigned to any task.

Examples: If the device being OPENed is a direct-access device, the <fd> used in the command is not the volume identifier, but the actual device mnemonic. For example, to make the new volume known to the system, the operator enters:

```
-OPEN FPY1:¶
-FPY1 MOS1¶
```

To open the device write protected the operator enters:

```
-OPEN,P FPY1:¶
-FPY1 ALEX¶
```

# SECTION 17

# OPTION: OPTION UTILITY

# SECTION 17
# OPTION: OPTION UTILITY

## 17.1 INTRODUCTION

Every task file has associated with it certain options which will determine its status after it has completed execution. For example, a task can be an exclusive resource; it can be resident in memory, nonresident in memory, abortable, or protected. When a task is loaded into memory it is given the status of an abortable task, i.e., it is cancellable from other tasks, unless that status is changed. The OPTION Utility is used to change the options of a specified task.

## 17.2 OPTION COMMAND

**Function:** Change the options of a task.

**Mode:** Remote.

**Format:** OPTION, opt1[opt2...]<tid>

**Arguments:** opt1, opt2.,...can be any of the following options:

R = resident in memory; the task remains in memory after execution.

N = nonresident in memory; the task is removed after end of task.

A = abortable; the task can be cancelled by other tasks.

P = protected; the task cannot be cancelled by other tasks.

**Use:** The OPTION command can be used in conjunction with the TASK command to check on a task's status with regards to the options that are in effect.

Examples:

Ex. 1 - LOAD COPYLIB and give it the option P.

```
-LOAD COPYLIB¶
-OPT, P COPY¶
-TA¶
```

which results in the following output to the console:

| Task | Nr | Stat | Type | Prio |
|---|---|---|---|---|
| MTCM | 1 | W | ERN | 20 |
| COMO | 2 | W | EN | 90 |
| COPY | 3 | Dorm | N | 128 |
| UTLO | 4 | | E | 90 |

Note that in the above task block COPY has the type designated by N which means nonabortable (c.f. the TASK utility in Part II). This is equivalent to an option of P.

Ex. 2 - In Ex. 1 change the P option for COPYLIB to A.

```
-OPT, A COPY¶
-TA¶
```

The task block for COPY now looks like

| Task | Nr | Stat | Type | Prio |
|---|---|---|---|---|
| MTCM | 1 | N | ERN | 20 |
| COMO | 2 | W | EN | 90 |
| COPY | 3 | Dorn | | 128 |
| UTLO | 4 | | E | 90 |

# SECTION 18

# PRIORITY: PRIORITY UTILITY

# SECTION 18
# PRIORITY: PRIORITY UTILITY

## 18.1 INTRODUCTION

The PRIORITY Utility is used to change the priority of a specified task.

## 18.2 PRIORITY COMMAND

**Function:** Change the priority of a task.

**Mode:** Remote.

**Format:** PRIority <tid>,n

**Arguments:** The task identifier tid is the name of the task whose priority is being changed. The number n is the new priority which is to be assigned to the task identifier. It is a decimal number from 10 to 255 inclusive.

**Use:** PRIORITY can be used in conjunction with the TASK utility to examine the priority of a given task once it has been set.

**Example:** Ex. 1 - Load COPYA into memory - check its priority, then change its priority to 30.

```
-LOAD COPYA¶
-TA¶
```

The task block for COPYA looks like:

| Task | Nr | Stat | Type | Prio |
|---|---|---|---|---|
| MTCM | 1 | W | ERN | 20 |
| COMO | 2 | W | EN | 20 |
| COPY | 3 | Dorm | | 128 |
| UTLO | 4 | | E | 90 |

Then enter:

```
-PRI COPY,30¶
-TA¶
```

The task block for COPY now looks like

| Task | Nr | Stat | Type | Prio |
|---|---|---|---|---|
| MTCM | 1 | W | ERN | 20 |
| COMO | 2 | W | EN | 20 |
| COPY | 3 | Dorm | | 30 |
| UTLO | 4 | | E | 90 |

and COPYA's priority has been lowered to 30.

# SECTION 19

# RENAME: THE RENAME FILES COMMAND

SECTION 19

# RENAME: THE RENAME FILES COMMAND

## 19.1 INTRODUCTION

The RENAME Command is used to change the name of a file.

## 19.2 RENAME COMMAND

**Function:** Changes the name of a file. This command cannot be used to rename a volume. DISKINIT must be used for that function. If the volume is the system volume you can also use the Volume Utility.

**Mode:** Remote

**Format:** REName <fd1>,<fd2>

**Arguments:** fd1 is the file descriptor of the file to be renamed, as well as the type.

fd2 is the file descriptor of the renamed file. Here only the filename is required. If type on the destination is omitted, the destination file type will be the same as the source file type.

**Examples:**

Ex. 1

Rename the file Henry to file George.

```
-REN HENRY/A, GEORGE¶
```

Ex. 2

Rename the ASCII file MASTERID in directory DIRECTORYA to NEWMAST.

```
-REN DIRECTORYA: MASTERID/A,NEWMAST¶
```

Ex. 3

Rename the old directory DIRECTID to the new directory named NEWDIRECT.

```
-REN DIRECTID/D,NFWDIRECT¶
```

# SECTION 20

# SPACE: SPACE UTILITY

# SECTION 20
# SPACE: SPACE UTILITY

## 20.1 INTRODUCTION

The SPACE Utility is used to examine the space available on a volume.

## 20.2 SPACE COMMAND

**Function:** Examines the space available on a direct-access volume.

**Mode:** Remote

**Format:** SPace <fd>

**Arguments:** fd is a file descriptor containing the volume name of the disk. If omitted, the System Volume is assumed.

**Example:** Examine the space available on the volume MONT.

```
-SP MONT ¶
```

# SECTION 21

# SET: SET AUTO UTILITY

# SECTION 21
# SET: SET AUTO UTILITY

## 21.1 INTRODUCTION

The SET AUTO Utility is used to automatically execute task files. The SET AUTO program maps any task file from its address at the sector of the disk it occupies to byte 178 of sector 0. If you then take out the disk and boot it up again the task file will execute automatically. Obviously this is an efficient way of executing programs which must run repetitively and at certain fixed intervals of time. SETAUTO can be used to load and execute individual programs, or command and select files consisting of many programs and/or commands.

## 21.2 SETAUTO COMMAND

**Function:** Loads a task file into sector 0 of a disk and execute it automatically.

**Mode:** Interactive and remote.

**Format:**

1) SETAUTO [fdl]
2) SETAUTO <fdl> <tid> [switch] [Parameters]

**Arguments:** In format 1, fdl is an optional volume or device descriptor which represents the volume or device on which will reside the program (or programs) to be auto set. If omitted the program defaults to interactive mode. If included the program still defaults to interactive mode but without the volume or devices query during the interactive session.

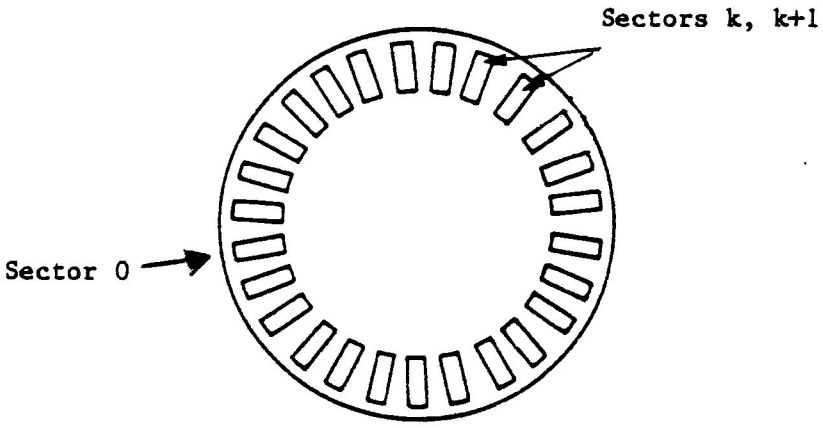
If format 2 the program executes in remote mode. fdl is a volume or device descriptor which represents the volume or device on which will reside the program (or programs) to be auto set.

tid is the task identification name of the program or utility (i.e. Monroe BASIC or COPYLIB for example) which is to be loaded into sector 0 of fdl.

Switch and parameters represent any optional switches and parameters associated with the tid the user wishes to include. Note that the syntax of the tid must be consistent with the program or utility being SETAUTO'd (See examples).

Use:

Suppose you have a task file which occupies sector k of a disk. If this is a file that you would like to have executed automatically SETAUTO will load that file into byte 178 of sector 9 (c.f. the figure below).




Note that if the task file consists of a Monroe BASIC program SETAUTO must first load the Monroe BASIC system followed by a space and the program name.

SETAUTO is executed in interactive mode. The user enters the command.

-SETAUTO¶

to which the system responds with the message

Set Auto 1.xy
Device?-

asking for the name of the disk drive FPY0 or FPY1 which contains the disk that is to be auto set.

If, for example, the drive is FPY0, and no prior tasks have been auto set to sector 0, entering

```
Device?-FPY0¶
```

produces the response

```
No Auto line
New line (y/n)?
```

which asks if you would like to auto set a new task file (yes or no). If you enter

```
New line (y/n)?-N¶
```

the system responds with the message

```
End of task 0.
```

If you enter

```
New line (y/n)?-Y¶
```

it responds with the massage

```
Enter newline:-
```

Suppose you want to auto set the utility program DIRECTORY LIST. You type in

```
Enter new line:-L¶
```

and the system responds with the message

```
End of task 0.
```

Then you take out the disk, reenter it into drive FPY0, boot up the system, and the utility program LIB will automatically list the files contained in the Master File Directory for the volume on FPY0.

If you do not enter a device name but hit a carriage return instead the program will terminate.

If you do not enter a new line but hit a carriage return instead the previous line will be erased and you will get the "SETAUTO Removed" message.

If you hit the carriage return and enter a new line prompt and if there is no previous line the "no entry" message will appear.

If you want to execute a series of programs and/or commands you can enter into a command file and type in the format SETAUTO! <fd> where fd is the filename. Of the command file, and the programs or commands run automatically when you boot up the disk.

Example:

Ex. 1 - Auto set the utility program TASKS.

```
-SETAUTO¶
 Set auto 1.xy
 Devices?-FPY0¶
 No Auto line
 New line (y/n)-Y¶
 Enter new line:-TA
 End of task 0
```

Ex. 2 - Auto set the Monroe BASIC program SEARCH FILE

```
-SETAUTO¶
 Set auto 1.xy
 Device?-FPY0¶
 No auto line
 New line (y/n)?-Y¶
 Enter new line:BASIC SEARCHFILE¶
 End of task 0
```

Ex. 3 - Auto set COPYLIB in remote mode with a buffer size of 14,000.

```
-SETAUTO FPY0:COPYLIB,G,14000 MONT:,PASC¶
```

# SECTION 22

# SORT: SORT UTILITY

# SECTION 22
# SORT: SORT UTILITY

## 22.1 INTRODUCTION

The SORT Utility is used to sort the contents of a file. The input file can be either of fixed record length or variable record length, depending upon the key which is input to the command. Temporary files can also be allocated by this utility on the system volume.

## 22.2 SORT COMMAND

**Function:** Sorts the contents of a file.

**Mode:** Remote.

**Format:** SORT [,,buffsize] <fd1>,[<fd2>,<key1>,<key2>,...,
<keyn>]

**Arguments:** Buffsize is an optional parameter which specifies additional memory to speed up the SORT when this is desired. If no extra memory is needed the SORT buffer is fixed at approximately 4KB.

The file descriptor fd1 specifies the input file which is to be sorted. This may be either of fixed record length (which is mandatory if a fixed length key is used) or variable record length.

The file descriptor fd2 specifies the output file containing the sorted data. This is created if it does not already exist. The file attributes of the created file will be the same as those of the input file.

The key variables (key1, key2, ...) indicate the types of data that are to be sorted. Each key variable has a fixed format which is more fully described below.

Use:

The key format is ss[-ee]/t/s where ss is the start position of the key, ee is an optional parameter giving the end position of the key, T is the type of key, and s is the collating sequence (A or D for ascending or descending). If the end position is not specified on a variable length key, it is set to the same value as the start position. The end position is ignored on a fixed length key. Keys may be of the following types:

A=ASCII Format key.

All control characters are treated as spaces, and upper/lower case letters have the same value. Space-compress information is recognized. Record length of the input file may be fixed or variable.

B=Binary Format key.

This key is evaluated as an unsigned binary integer. The record length of the input file must be fixed.

I = signed integer.

This key is evaluated as a signed two-byte integer with the least significant byte first, compatible with Monroe BASIC and PASCAL formats. The key length is always two bytes.

F=single-precision floating-point number.

This key is evaluated as a floating point number, compatible with Monroe BASIC and PASCAL formats. The key length is always two bytes.

D=Double-precision floating-point number.

This key is evaluated as a floating point number compatible with the Monroe BASIC format. The key length is always eight bytes.

The A and B type keys are of variable length and may be specified left to right or right to left. If the end position of the key is not specified the key length will be one byte. The I,F, and D type of keys are of fixed length, and any end position given for the key is ignored. Input file record length must be fixed.

The collating sequence may be

A-Ascending or
D-Descending

for each key. Multiple keys of different types may be used in the same SORT session.

Examples:

Ex. 1 - SORT an input file named INFILE using a single character ASCII key located in the first position of each record. Deposit the sorted data on an output file named OUTFILE. The input file may be of fixed or variable record length.

```
-SORT INFILE, OUTFILE, 1/A/A¶
```

Ex. 2 - SORT an input file named INFILE using an additional buffer of 10,000 bytes and two keys. The first is an ASCII key which uses position 10-15 of each record in backward order and a DESCENDING collating sequence. The second is a double-precision floating-point key located in position 20 of each record using an ASCENDING collating sequence. Deposit the sorted data on

an output file named OUTFILE. The input file must be of fixed record length.

```
-SORT,,10000 INPUT,OUTFILE,15-10/A/D,20/D/A¶
```

Ex. 3 - Consider the first 20 entries in the following file, called RANDOM 1

| | | |
|---|---|---|
| 1158.22 | abcde | 100 |
| 1088.34 | abcde | 101 |
| 2530.53 | abcde | 102 |
| 2955.14 | abcde | 103 |
| 1415.95 | abcde | 104 |
| 1724.97 | abcde | 105 |
| 1147.6 | abcde | 106 |
| 2232.23 | abcde | 107 |
| 1421.78 | abcde | 108 |
| 2775.36 | abcde | 109 |
| 1176.29 | abcde | 110 |
| 1253.93 | abcde | 111 |
| 1917.36 | abcde | 112 |
| 2987.35 | abcde | 113 |
| 1508.7 | abcde | 114 |
| 2468.7 | abcde | 115 |
| 2583.69 | abcde | 116 |
| 2767.87 | abcde | 117 |
| 1839.83 | abcde | 118 |
| 2999.16 | abcde | 119 |
| 1071.19 | abcde | 120 |
| . | . | . |
| . | . | . |
| . | . | . |

which consists of 400 decimal numbers which are randomly listed, the characters abcde, and the indexes 100-500 which appear in their natural arithmetic order. If you enter the command

```
-SORT RANDOM1,SEQ,1/F/D¶
```

the entries in the left-most column of RANDOM1 will be sorted in descending order and output to a file named SEQ. The first 20 entries of the sequenced file now look like

| | | |
|---|---|---|
| 2999.75 | abcde | 195 |
| 2999.57 | abcde | 264 |
| 2999.16 | abcde | 119 |
| 2990.92 | abcde | 417 |
| 2990.15 | abcde | 234 |
| 2987.35 | abcde | 113 |
| 2985.26 | abcde | 130 |
| 2980.64 | abcde | 384 |
| 2977.27 | abcde | 386 |
| 2966.36 | abcde | 406 |
| 2964.25 | abcde | 316 |
| 2959.08 | abcde | 320 |
| 2955.14 | abcde | 103 |
| 2949.8 | abcde | 303 |
| 2942.25 | abcde | 222 |
| 2935.98 | abcde | 498 |
| 2930.74 | abcde | 436 |
| 2921.84 | abcde | 212 |
| 2915.02 | abcde | 321 |
| 2908.76 | abcde | 433 |
| . | . | . |
| . | . | . |
| . | . | . |

where the indexes in the right-most column are now randomized as a result of the SORT procedure. Note we could have easily sorted the decimal numbers in ascending order by typing an A instead of a D in the SORT command:

```
-SORT RANDOM1, SEQ, 1/F/A¶
```

# SECTION 23

# TIME: TIME UTILITY

# SECTION 23
# TIME: TIME UTILITY

## 23.1 INTRODUCTION

The TIME Utility should be entered when the system is booted. It may be entered at any other time that the system clock is incorrect. The day, month and year are automatically updated by the system, even during leap years.

## 23.2 TIME COMMAND

**Function:** Enters the day, month, year, and time.

**Mode:** Remote

**Format:** Time <yyyy-mm-dd,hh.mm.ss>

**Arguments:**

yyyy = year
mm = month
dd = day
hh = hours, 24-hour clock
mm = minutes
ss = seconds

If yyyy-mm-dd, hh.mm.ss is omitted, the current time will be displayed.

**Example:**

```
-TI 1981-05-04, 08.30.00¶
```

# SECTION 24

# VOL: VOLUME UTILITY

# SECTION 24
# VOL: VOLUME UTILITY

## 24.1 INTRODUCTION

The VOLUME Utility is used to set or change the name of the system volume. Alternatively, it is used to interrogate the system for the current name associated with the system volume.

## 24.2 VOLUME COMMAND

**Function:** Sets or changes the name of the system volume.

**Mode:** Remote

**Format:** Volume [fd]

**Arguments:** The file descriptor fd is optional and specifies the new system volume identifier.

**Use:** No test is made to ensure that the volume is actually on-line at the time the command is entered. If fd is not specified, the name of the currently default system volume is output to the console.

**Examples:**

Ex. 1

Interrogate the system for the current volume name.

```
-V¶
(The following information appears on the
console:)

SYSTEM      (SYSTEM Indicates that system volume
MSTM        MSTM is the current name.)
-
```

Ex. 2

Change the above system volume name to ACCT.

```
-VOL ACCT:¶
 SYSTEM
 ACCT
-
```

# PART II

# TASK MAINTENANCE UTILITIES

# SECTION 25

# CANCEL: CANCEL TASK UTILITY

# SECTION 25
# CANCEL: CANCEL TASK UTILITY

## 25.1 INTRODUCTION

The CANCEL Utility is used to terminate a task. The cancel takes place exactly as if the task had executed an SVC-6 with cancel as function code and 255 as cancel code (c.f. MONROE OPERATING SYSTEM PROGRAMMER'S REFERENCE MANUAL). CANCEL may be used to cancel a current task or cancel a task that is dormant, i.e., one that has been loaded but not executed. If the task is nonresident, it is removed from system memory. When a task is cancelled all outstanding I/O requests are terminated with the termination of the task and all of the task's logical units are closed. This command may be entered even when the specified task is dormant. It has no effect on a resident task that has already gone to an End of task.

## 25.2 CANCEL COMMAND

**Function:** Terminates a task.

**Mode:** Remote

**Format:** Cancel [tid]

**Arguments:** tid is an optional task identifier (e.g. Monroe BASIC, PASCAL, etc.)

**Example:**

Ex. 1 -

Cancel COPYLIB

```
-COPYLIB MONT:,PASC¶        Copy MONT to PASC
- ...¶                      Additional Key Ins
CTRL A                      Return Curser to Screen
-CA¶                        Cancel COPYLIB
```

Ex. 2 -

Cancel LIB

```
-LIB,F MONT:¶
CTRL A
-CA¶
```

# SECTION 26

## CONTINUE: CONTINUE TASK UTILITY

# SECTION 26
# CONTINUE: CONTINUE TASK UTILITY

## 26.1 INTRODUCTION

Often you must pause in the middle of execution of some task in order to make a decision or review new information that may effect the outcome of the task. For example, one of the terminal options allowed by COPYLIB is a pause in response to the terminal query about a particular file. The CONTINUE utility allows a task which has been paused by an SVC or the operator to continue operation.

## 26.2 CONTINUE COMMAND

**Function:** Resumes operation of a task.

**Mode:** Remote

**Format:** COntinue [tid]

**Arguments:** tid is the task identifier (e.g. Monroe BASIC, PASCAL, etc.) which is optional.

**Examples:** Ex. 1 - Continue COPYLIB

```
-COPYLIB MONT:,PASC¶        Copy files from MONT
         .                  to PASC.
         .                  Additional Key Ins.
         .
CTRL A                      Return Curser to Screen.
-PA¶                        Pause.
-CO¶                        Continue.
-
```

Ex. 2 - Continue LIB

```
-LIB,F PASC:¶               List files on PASC.
CTRL A                      Return Curser to Screen.
-PA¶                        Pause.
-CO¶                        Continue.
-CA¶                        Cancel LIB.
-
```

# SECTION 27

# DEVICES: DEVICES UTILITY

# SECTION 27
# DEVICES: DEVICES UTILITY

## 27.1 INTRODUCTION

The DEVICE Utility allows the user to obtain the name, number, status, type, volume name, current request, channel number, and the address of the Device Control Block (DCB) of all devices in the operating of system. (See the MONROE OPERATING SYSTEM REFERENCE MANUAL for a complete discussion of DCB's.)

## 27.2 DEVICES COMMAND

Function: Lists the devices in the operating system.

Mode: Remote

Format: DEVices [fd]

Arguments: The device descriptor fd is optional and is the device name of an output device (terminal or printer). If no output device is specified, the display of device in the system will go to the terminal.

Use: The DEVICES command will display the following information:

Mnem - Contains the symbolic name of the device.

Nr - Contains the system device number.

Stat - Contains information about the status of the device. This can be:
- OFFL - Off-Line
- PROTL - On-Line (Write Protected)

Type - Contains information about the type of device, i.e, is it:
- DIR - Directory Oriented
- TASK - Task Oriented

Voln - Contains the Volume Name of a Directory Oriented Device or the name of the Symbiont task that owns the device.

Dcb-addr - Contains the address of the DCB for each device.

Req - Contains the name of the task currently accessing the device.

Svc-blk - Contains the address to the involved parameter blocks.

NULL - Dummy Device

PR - Printer

FPY0 - Disk Drive 0

FPY1 - Disk Drive 1

TRM0 - Terminal

CON - Console

Example: Ex. 1 - List the devices on the console.

-DEV¶ Display the devices to the console.

(The following is displayed on the console:)

```
-DEV
Mnem  Nr  Stat  Type  Voln Dcb-Addr Req  Svc-blk
NULL   0
PR     6                 275:034
FPY0   8        DIR  DOS8 267:030
FPY1   9                 266:266
TRM0  64  OFFL           265:342
CON   65                 275:162
-
```

Ex. 2 - List the devices on the printer.

```
-DEV PR:¶
-
```

Will display the same information as Ex. 1 to the printer.

# SECTION 28

# LOAD: LOAD UTILITY

# SECTION 28
# LOAD: LOAD UTILITY

## 28.1 INTRODUCTION

In order for a task to be executed its subprograms must first be "established" by the Assembly Language Utility Program ESTAB. Once a task has been established it can then be loaded into memory using the LOAD Command. One of the unique features of the Monroe Operating System is that is allows you to load and execute more than one program. This is called "multi-programming" and is described in more detail in the MONROE OPERATING SYSTEM PROGRAMMER'S REFERENCE MANUAL. All utility programs can be used in either a single programming or multi-programming mode. Note: programs can be loaded and executed either under their own name or under a temporary name used during a specific programming sequence. It must be emphasized that LOAD can only be used with task files. A task is loaded into the first memory segment large enough to accommodate it.

## 28.2 LOAD COMMAND

Function: Loads one or more tasks into memory.

Mode: Interactive

Format:
1) LOad <fd1>
2) LOad <fd1>,[fd2],[size]

Arguments: The file descriptor fd1 is the name of the file or utility program to be loaded.

The file descriptor fd2 is the name under which the file or program is to be known during program execution. If fd2 is omitted, it defaults to the first four characters in fd1. The size is the amount of area in the tasks impure memory segment (c.f., MONROE OPERATING SYSTEM PROGRAMMER'S REFERENCE MANUAL). It is specified as the decimal number if the decimal number is bytes, and if omitted, the default if 0. If LOAD is performed from a device, the logical record length must be 256.

Use: LOAD is most useful in a multi-programming setting when you have several programs to execute simultaneously. You load your programs in sequence into memory and then use the START Command to execute them. The procedure is interactive in so far as you can use the various utilities to key in or out of programs from the terminal and output data to one or more devices, even while a program is running. For example, you may use one COPY routine to output ASCII data to the printer while using another (or even the same) COPY routine to copy files to a disk device. Hence the PRINTER

will operate in "background mode" printing the ASCII file at the same time that the console is operating in "foreground mode", i.e, transferring disk files from one disk device to another disk device. In using LOAD, the error message output to the screen are those of the task file or utility being loaded. Hence you should consult the appropriate manuals before loading a particular program or utility. If a task file has been improperly loaded, a LOAD ERROR will appear on the screen.

Examples:

Ex. 1 Copy an ASCII file to the console.

| | |
|---|---|
| -LO COPYA,A¶ | Load COPYA under the task identification name A. |
| -ST A ASCI,CON:¶¶ | Copy the ASCII file ASCI to the console. |

Ex. 2 Copy an ASCII file to the Printer.

| | |
|---|---|
| -LO COPYA, B¶ | Load COPYA under the task identification name B. |
| -ST B ASC2, PR:¶¶ | Copy the ASCII file ASC2 to the printer. |

Ex. 3 - Load and execute a Monroe BASIC, COPYLIB and COPYA programs.

| | |
|---|---|
| -LO BASIC,A¶ | Load Monroe BASIC under the task identification move A. |
| -LO COPYLIB,B¶ | Load COPYLIB under the task identifier move B. |
| -LO COPYA,C¶ | Load COPYA under the task identification name c. |
| -ST A¶¶ (¶¶ = Two returns) | Start Monroe BASIC. |
| -...¶ | Execute a Monroe BASIC Program. |
| -PAUSE¶<br>CTRL A | Pause Monroe BASIC.<br>Return to console. |
| -ST B MONT:,ALEX:¶¶<br>(¶¶ = Two returns) | Start COPYLIB; copy from MONT to ALEX. |
| - ...¶<br>CTRL A | Execute COPYLIB. |
| -ST C PMCMD, CON:¶¶ | Start COPYA; copy the ASCII file PMCD to console. |
| -CTRL A | Return to console. |
| -CO A¶ | Continue Monroe BASIC. |
| -BYE¶ | End Monroe BASIC. |
| -CNTRL A | Return to console. |
| -LO COPYA,A¶ | Load COPYA under the task identification name. |
| -ST A ASC7, CON:¶¶ | Copy the ASCII FILE ASCI to the console. |

Note: You can either hit 2 returns (PP) after each ST statement (to execute the LOADS in sequence), or you can hit two returns after the last ST statement (in which case the three programs are executed in reverse order). After the last program has been executed, CNTRL A returns the curser to the console.

Ex. 4 - Copy an ASCII file to the Console, another ASCII file to the printer, and a third file from one disk device to another.

| | |
|---|---|
| -LO COPYA,A¶ | Load COPYA under the task identification name A. |
| -LO COPYA,B¶ | Load COPYA under the task identification name B. |
| -LO COPYLIB,C¶ | Load COPYLIB under the task identification name C. |
| -ST A RKDISKDUMP,CON:¶¶ | Copy the ASCII file RKDISKDUMP to the console. |
| -ST B CEDIRREA,PR¶¶ | Copy the ASCII file CEDIRREA to the printer. |
| -ST C MONT:PMCMD,ALEX:PMCMD¶¶ | Copy the file PMCMD on the volume MONT to the volume ALEX under the name PMCMD. |

If you wish to cancel any (or all) of the above programs you simply type:

-CA A¶
-CA B¶
-CA C¶

# SECTION 29

# PAUSE: PAUSE TASK UTILITY

# SECTION 29
# PAUSE: PAUSE TASK UTILITY

## 29.1 INTRODUCTION

The PAUSE Utility will cause a specified task to pause. Any ongoing I/O is allowed to complete itself at the time a task is paused. If a task is in a Wait state at the time it is paused, then all external wait conditions have already been satisfied. The PAUSE Utility is rejected if a task is dormant or paused at the same time it is entered. Note that a number of utilities (e.g. COPYLIB) have their own pause built into the command while for others CNTRL A has the effect of interrupting the task. In such cases typing PA after CNTRL A will cause a sequence error since the task is already interrupted. The PAUSE Utility is most useful when pausing a program which is outputting data to an I/O device other than the console.

## 29.2 PAUSE COMMAND

**Function:** Pauses a task file.

**Mode:** Interactive

**Format:** PAuse <tid>

**Arguments:** None

**Example:** Consider a Monroe BASIC program XBAS which outputs data to the printer. The following commands will cause the program to pause, and the output to the printer will stop until the program is continued.

```
-BASIC¶
*BASIC*
-LOAD XBAS¶
-RUN¶
 CNTRL A
-PA¶¶
PAUSED
```

# SECTION 30

# RUN: RUN TASK UTILITY

# SECTION 30
# RUN: RUN TASK UTILITY

## 30.1 INTRODUCTION

One problem with the LOAD Utility is that the LOAD and START Commands must be entered independently and that each task file must be loaded and started separately. The RUN Utility allows the user to LOAD and START a program with the single command of RUN. After the task has been loaded it is given a system name consisting of the first four entries of its filename. (See the section on Terminal Management in the MONROE OPERATING SYSTEM PROGRAMMER'S REFERENCE MANUAL). Because the system uses the first four letters of a task's filename as its system name you cannot run more than one COPY Utility in a Multi-Program.

## 30.2 RUN COMMAND

**Function:** Loads and starts a program.

**Mode:** Interactive

**Format:** RUn [switches] <fd>[,parameters]

**Arguments:** Switches are optional and consist of any required or optional switches that you wish to use in the program you are running. For example, the F switch in LIB, or the A switch in COPYA.

The file descriptor fd is the name of the program being run.

Parameters which is also optional consists of any additional parameters in the programs syntax that must be included in the RUN statement. For example, the volume name of the source and destination files.

**Examples:** Ex. 1 - Load and start Monroe BASIC.

```
-RU BASIC¶                  Loads and starts Monroe
*BASIC*                     BASIC.
```

Ex. 2 - Run Monroe BASIC and a Monroe BASIC program.

```
-RU BASIC XBAS¶             Loads and starts Monroe
*BASIC*                     BASIC: then executes the
                            program XBAS.
```

Ex. 3 - Do a RUN, PAUSE, and another RUN.

```
-RU COPYLIB,G MONT:,ALEX¶        Load and start COPYLIB.
-......¶
-CNTRL A¶                        Pause COPYLIB.

-RU BASIC¶                       Load and start Monroe
*BASIC*                          BASIC.
-
-......¶
-CNTRL A
-RU PASCAL¶                      Load and start PASCAL.
(Pascal sign-on)
```

Note: From the last example in a multi-programming mode, you do not need to pause after each RUN before you load and start a new task file, but you do need to press CNTRL A. The program being RUN is loaded from the file descriptor fd and the task is given the first four characters in fd as its name, In example 1, the system will give the name BASI to Monroe BASIC, in Example 3 the names COPY, BASI, and PASC will be given to COPYLIB, Monroe BASIC and PASCAL. Then the task is started and the switches and parameters are passed to the task.

# SECTION 31

# SLICE: SLICE TASK UTILITY

# SECTION 31
# SLICE: SLICE TASK UTILITY

## 31.1 INTRODUCTION

The Monroe Operating System allows tasks to be scheduled in two ways. A task can be scheduled on a strict priority basis (c.f., MONROE OPERATING SYSTEM PROGRAMMER'S REFERENCE MANUAL), or a task can be "time sliced" within a given priority.

In the first case each task has a fixed priority with respect to every other task. This in turn dictates how long a task must wait before it gains control of the processor. If two tasks are in queue, and one has a higher priority than the other, the task with highest priority will get all of the processor time it needs before control is relinquished to the second task. If the two tasks have equal priority, the task which is first in queue will remain active until it relinquishes control to the second. This can occur in one of three ways. Either the task is paused or cancelled by the console operator or some other task, or a higher priority task suddenly becomes ready because of some external event, or the active task executes an SVC that places it in a Wait, Paused, or Dormant state. Therefore, tasks which do not frequently give up control of the processor can lock out other tasks which do.

With time slicing, two tasks having equal priority both will receive equal shares of processor time. It must be emphasized that time slicing does not change the priority basis of the scheduling queue. It does not reassign priorities or give a task of lower priority higher priority. In order for two tasks to be time sliced they must have the same priority within a particular queue. When they are sliced, each will receive the same amount of processor time. The time-slice utility is initiated when the operating system is generated.

## 31.2 SLICE COMMAND

**Function:** Time-slices two tasks.

**Mode:** Remote

**Format:** SLice [t]

**Arguments:** t is optional. It specifies the time slice in milliseconds as a decimal number. If t is omitted the current slice value is displayed. If <t> is 0 the time-slice mode of scheduling is disabled. The time slice represents the maximum time, in milliseconds, any task can remain active if another task of equal priority is ready.

**Example:** Specify 100 milliseconds as the maximum time any task can remain active if another task of equal priority is ready.

```
-SL 100¶
Current slice is 100 milliseconds
```

# SECTION 32

# START: START TASK UTILITY

# SECTION 32
# START: START TASK UTILITY

## 32.1 INTRODUCTION

The START Utility initiates task execution and is used after the task has been loaded. A task can be started only if it is dormant.

## 32.2 START COMMAND

**Function:** Starts a task.

**Mode:** Remote

**Format:**
1) STart <fd1>
2) STart <fd1>,[switches] [parameters]

**Arguments:** The file descriptor fd1 is the task identification name of the task file and may consist of from one to four alphanumeric characters.

Switches are optional and consist of any required or optional switches or variables that you wish to use in the utility you are starting. For example, D and G in COPYLIB, or F in LIB, or BUFFSIZE in COPYI.

Parameters are also optional and consists of any additional parameters in a utility syntax that are necessary to START that utility. For example, the volume name of the SOURCE and destination files etc.

**Example:** Start COPYLIB under the G option.

```
-ST COPYLIB,G, 14000 FIX:,NULL:¶
```

Start task with task identifier A and display file RKDISKDUMP on the printer.

```
-ST A RKDISKDUMP,¶
-PR:¶
```

SECTION 33

TASK: TASK UTILITY

# SECTION 33
# TASK: TASK UTILITY

## 33.1 INTRODUCTION

The TASK Utility causes a listing of the status of each task to be output to either the console or the printer.

## 33.2 TASK COMMAND

**Function:** Lists the status of each task.

**Mode:** Remote

**Format:** TAsk [,F] [fd]

**Arguments:** F is an optional switch. When included, a list of the TCB-ADR fields for each task will be output. These contain the address of the TCB's for each task (c.f. MONROE OPERATING SYSTEM PROGRAMMER'S REFERENCE MANUAL). Also, the memory size of each task, and each task default address (ENTRY) will be output as well.

The device descriptor fd is a device, either the console or printer to which the information is to be output.

**Use:** The TASK Command is commonly used during multi-programming when loading and executing a number of programs. The information output is in a table whose entries have the following meaning:

Task - Contains the symbolic name of the task.
Nr - System task number.
Stat - Contains the current status of the task.
Type - Indicates the type of task.
Prio - Indicates the tasks current priority.

A task can have any of the following as a status:

| | |
|---|---|
| DORM | Dormant, not started. |
| C | Cancel pending, on its way to terminate. |
| P | Paused. |
| S | Suspended. |
| W | Waiting. |

It can be of any one of the following types:

| | |
|---|---|
| E | Exclusive task |
| N | Non-abortable |
| P | Pure code (fuller reentrant) |
| R | Resident-task |

Examples:

Ex. 1. Load three programs and use TASK without the F switch.

```
-LOAD COPYA,A¶
-LOAD COPYA,B¶
-LOAD COPYLIB,C¶
-TA¶
```

The following information is output to the terminal:

| Task | Nr | Stat. | Type | Prio |
|---|---|---|---|---|
| MTCM | 1 | N | ERN | 20 |
| COM 0 | 2 | N | E | 90 |
| A | 4 | DORM | | 120 |
| B | 5 | DORM | | 120 |
| C | 6 | DORM | | 128 |
| UTLO | 7 | | | 90 |

The tasks MTCM,COMO, and UTLO are the Multi-Terminal Console Motor, Command Handler, and Utility Program, respectively (c.f. the section on Terminal Management in the MONROE OPERATING SYSTEM PROGRAMMER'S REFERENCE MANUAL).

Note that A,B,C COMO and UTLO do not have the same priority (c.f. Section 28). Note also that the TCB-Addr fields do not appear in the output.

Ex. 2.

-TA,F PR:¶

-

(This command will print the following table to the printer:)

| Task | Nr | Stat | Type | Prio | Tcb-Addr | Size | Entry |
|---|---|---|---|---|---|---|---|
| MTCM | 1 | W | RN | 20 | 2844 | 0kb | 4000 |
| COMO | 2 | W | | 30 | 1035 | 0kb | 426D |
| UTLO | 3 | | | 128 | 1B4t | 8kb | E000 |

Note that by including "F", the TCB-Addr, Size and Entry fields will also be output to the printer.

# APPENDIX A

## COMMAND SUMMARY

# APPENDIX A
# COMMAND SUMMARY

| Command | Format | Function | Page |
|---|---|---|---|
| ALLOCATE | ALlocate,[switch] <fd> [,record length][,size] [blk] | Allocates either a contiguous or indexed direct access file on a diskette. | 2-1 |
| BOOTGEN | BOOTGEN,B <fd> | Writes a loader onto a diskette. | 3-1 |
| CANCEL | CAncel | Terminates a task. | 25-1 |
| CLOSE | CLose <fd> | Takes a device off line. | 4-1 |
| COMMAND FILE | ! <fd> | Executes one or more programs, tasks, and/or commands which exist in either a single file or a group of files. | 5-1 |
| CONTINUE | COntinue | Resumes operation of a task. | 26-1 |
| COPYA | COPYA[,switch] <fd1>,<fd2> | Copies ASCII date between two files. | 6-1 |
| COPYI | COPYI <fd1>,<fd2> or COPYI[,switch][,buffsize] <fd1>,<fd2> | Performs an image copy and/or verifies data between devices and/or files. | 7-1 |
| COPYLIB | COPYLIB[,switch][,buffsize] <fd1>[,select file] or COPYLIB[,switch][,buffsize] <fd1>, <fd2> [,select file] | Copies data between all files. | 8-1 |

## APPENDIX A - COMMAND SUMMARY

| Command | Format | Function | Page |
|---|---|---|---|
| COPYT | COPYT <fd1>,<fd2> | Copies task files between devices and/or files. | 9-1 |
| CREATE INDEX | CREINDEX | Allocates ISAM Index and data files. | 10-1 |
| DELETE | DELete <fd>[,fd2]....[,fdn] | Deletes a direct-access file. | 11-1 |
| DEVICES | DEVices [fd] | Lists the devices in the operating system. | 27-1 |
| DISKCHECK | DISKCHECK <fd1>[,fd2] | Checks the integrity of an open disc file. | 12-1 |
| DISKINIT | DISKINIT | Initializes a new disk. | 13-1 |
| FORMAT | FORMAT | Prepares a disk for initialization. | 14-1 |
| LIB | Lib [,F] [[fd1],[fd2][,fd3]] | Displays the contents of a volume to the terminal or printer. | 15-1 |
| LOAD | LOad <fd1><br>LOad <fd1>,[fd2] [size] | Loads one or more task into memory. | 28-1 |
| OPEN | OPEn <fd>:<br>OPEn[,N] <fd>:<br>OPEn[,P] <fd>: | Brings on-line a device that was previously off line. | 16-1 |
| OPTION | OPTION,opt1[opt2...]<tid> | Changes the options of a task. | 17-1 |

| Command | Format | Function | Page |
|---|---|---|---|
| PAUSE | PAuse | Pauses a task file. | 29-1 |
| PRIORITY | PRIority <tid>,n | Changes the priority of a task. | 18-1 |
| RENAME | REName [fd] <fd1>,<fd2> | Changes the name of an unassigned direct - access file. This command cannot be used to rename a direct-access Volume. DISKINIT must be used for that function. If the direct-access volume is the system volume you can also use Volume. | 19-1 |
| RUN | RUn [switches] <fd> [,parameters] | Loads and starts a program. | 30-1 |
| SET AUTO | SETAUTO | Load a task file into section 0 of a disk and execute it automatically. | 21-1 |
| SLICE | SLice [t] | Time-slices two tasks. | 31-1 |
| SORT | SORT [,,buffsize] <fd1> [<fd2>,<key1>,<key2>,...,<keyn>] | Sorts the contents of a file. | 22-1 |
| SPACE | SPace <fd> | Examines the space available on a direct-access volume. | 20-1 |

| Command | Format | Function | Page |
|---|---|---|---|
| START | STart <fd1><br>STart <fd1>,[switches]<br>[parameters] | Starts a task. | 32-1 |
| TASK | TAsk [F] [fd] | Lists the status of each task. | 33-1 |
| TIME | TIme <yyyy-mm-dd,hh.mm.ss> | Enters the day, month, year, and time. | 23-1 |
| VOLUME | Volume <fd> | Sets or changes the name of the system volume. | 24-1 |

# APPENDIX B

# ERROR CODES

# APPENDIX B
# ERROR CODES

## COMMON ERRORS

| s | SYMBOLIC | ERROR TEXT |
|---|---|---|
| 0 | SOS.OK | No error. |
| 1 | SOS.EON | End of nodes. |
| 2 | SOS.IFC | Invalid function code. |
| 3 | SOS.PRO | Can't connect at unconditional proceed. |
| 4 | SOS.OFFL | Off line. |
| 5 | SOS.PRES | Not present in this system. |
| 6 | SOS.NYET | Not yet implemented function. |
| 7 | SOS.CAN | Request is cancelled. |
| 8 | SOS.SVC | Invalid SVC function. |

## SVC-1 I/O ERROR CODES

| s | SYMBOLIC | ERROR TEXT |
|---|---|---|
| 10 | S1S.LU | Illegal LU, LU not assigned. |
| 11 | S1S.AM | Invalid access modes. |
| 12 | S1S.TOUT | Time-out. |
| 13 | S1S.DWN | Device down. |
| 14 | S1S.EOF | End-of-file. |
| 15 | S1S.EOM | End-of-media. |
| 16 | S1S.RER | Recoverable error. |
| 17 | S1S.UNR | Unrecoverable error. |
| 18 | S1S.RND | Invalid random address. |
| 19 | S1S.NRND | Non-existent random address. |

## SVC-2 SUBFUNCTION ERRORS

| s | SYMBOLIC | ERROR TEXT |
|---|---|---|
| 20 | S2S.ISB | Illegal sub-function number. |

SVC-3 TIMER ERRORS

| s | SYMBOLIC | ERROR TEXT |
|---|---|---|
| 30 | S3S.PAR | Invalid timer parameter. |

SVC-4 DEVICE ERRORS

| s | SYMBOLIC | ERROR TEXT |
|---|---|---|
| 40 | S4S.ASGN | Not assigned. |
| 41 | S4S.TYPE | Invalid device type. |

SVC-5 LOADER ERRORS

| s | SYMBOLIC | ERROR TEXT |
|---|---|---|
| 50 | S5S.TID | Illegal task-id. |
| 51 | S5S.PRES | Task present. |
| 52 | S5S.PRIO | Illegal priority. |
| 53 | S5S.OPT | Illegal option. |
| 54 | S5S.CODE | Illegal code/item at load. |
| 55 | S5S.SIZE | Overlay doesn't fit. |

SVC-6 TASK ERRORS

| s | SYMBOLIC | ERROR TEXT |
|---|---|---|
| 60 | S6S.TID | Illegal task-id. |
| 61 | S6S.PRES | Task present. |
| 62 | S6S.PRIO | Illegal priority. |
| 63 | S6S.OPT | Illegal option. |
| 64 | S6S.EQUE | Event queue disabled. |
| 65 | S6S.STAT | Invalid task status. |

SVC-7 FILE ERRORS

| s | SYMBOLIC | ERROR TEXT |
|---|---|---|
| 70 | S7S.ASGN | Assignment error, double assign. |
| 71 | S7S.AM | Illegal access modes. |
| 72 | S7S.SIZE | Size error. |
| 73 | S7S.TYPE | Type error. |
| 74 | S7S.FD | Illegal file descriptor. |
| 75 | S7S.NAME | Name error. |
| 76 | S7S.KEY | Invalid key. |
| 77 | S7S.FEX | File exist error. |

SVC-8 RESOURCE ERRORS

| s | SYMBOLIC | ERROR TEXT |
|---|---|---|
| 80 | S8S.ID | Illegal resource-id. |
| 81 | S8S.CLAS | Invalid resource class. |
| 82 | S8S.PRES | Resource already present. |
| 83 | S8S.PRNT | Parent not present. |
| 84 | S8S.DUAL | Dual DCB not present. |
| 85 | S8S.RCB | Invalid RCB-type. |
| 86 | S8S.EOM | End-of-memory. |

# APPENDIX C

## LIST OF UTILITY EXAMPLES

# APPENDIX C
## LIST OF UTILITY EXAMPLES

## APPENDIX C - LIST OF UTILITY EXAMPLES

## APPENDIX C - LIST OF UTILITY EXAMPLES

## APPENDIX C - LIST OF UTILITY EXAMPLES

# GLOSSARY OF TERMS

# GLOSSARY OF TERMS

| | |
|---|---|
| Abortable Task | An abortable task is a task that can be cancelled from another task. A task that cannot be aborted from another task is nonabortable. |
| Absolute Task File | A Task File which has information consisting of absolute machine code. |
| Bit - Map - Directory | A directory of information concerning what parts of the disk are occupied. |
| Command-File | A file which consists of individual programs and/or commands. |
| Contiguous - File | A file which is stored on contiguous sectors of a diskette. |
| User-File-Directory | A sub-directory of the Master-File-Directory. |
| Elementary - File | An element or file in the Element-File-Directory. |
| Exclusive Resource | An exclusive resource cannot be used by any other resource. A resource that is not exclusive is called nonexclusive. |
| FILE-STRUCTURED-DEVICE | Any device which contains a directory. |
| Fixed Record Length File | A file, all of whose records have the same length. |
| Formatted - Data | Data that must be manipulated by either a program or by the operating system to be used. |

| | |
|---|---|
| Indexed - File | A file which is distributed over many non-contiguous sectors of a diskette. There is a pointer to each sector occupied by the file which indexes the file. |
| NON-FILE-STRUCTURED Device | Any device that does not contain a directory. |
| Non-resident in memory | A task is non-resident in memory when it is removed from memory after execution. |
| Logical - File | A file consisting of logical records. |
| Master-File-Directory | A directory consisting of information concerning all disk files. |
| Physical - File | A file consisting of physical records. |
| Resident in Memory | A task is resident in memory when it remains in memory after execution. |
| Relocated-Task-File | Any Task File file that must be relocated by the Relocatable - Loader before it can be used. A Relocatable-File has information consisting of relocatable machine code. |
| Select - File | A file which consists of individual commands. |
| Variable Record Length File | Any file consisting of records whose lengths vary. |

# INDEX

READER COMMENT FORM DATE _____

Your comments and suggestions help to improve this publication.
Please complete the questionaire. Fold, staple, and mail it to Monroe.

Name_____ Title_____
Organization_____
Street_____ State_____ Zip_____

Publication Title_____
Publication No._____ Revision Letter_____ Date_____

CIRCLE YOUR RESPONSES TO THE STATEMENTS BELOW. IF YOU RESPOND "NO" TO A STATEMENT, ENTER THE STATEMENT NUMBER AND THE PAGE AND PARAGRAPH IN THE PUBLICATION THAT PROMPTED YOUR RESPONSE.

1. The publication was used for
   Learning Installing
   Reference Maintaining
   Sales Programming

2. The user/reader was
   High-level Programmer
   Occasional Programmer
   Student Programmer
   Data Entry Operator
   Other (specify)_____

3. The material is accurate. YES NO
4. The material is clear. YES NO
5. The material is complete. YES NO
6. The material is well organized. YES NO

ENTER DETAILED INFORMATION FOR STATEMENTS 3-6.

| Statement No. | Page No. | Paragraph No. | Comments |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

7. The overall rating for this publication is
   Very Good Good Fair Poor Very Poor

   Briefly explain your rating._____

8. Additional comments_____

FOLD

Software Publications Dept.

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 731 MORRISTOWN, N. J.

POSTAGE WILL BE PAID BY ADDRESSEE

MONROE SYSTEMS FOR BUSINESS, a Div. of
Litton Business Systems, Inc.
Box 9000R
Morristown, N.J. 07960

CUT ALONG LINE

FOLD

STAPLE

STAPLE

This is the normal sequence to bootgen a disk:

| | |
|---|---|
| -CLOSE FPY0¶ | Close the drive if it is opened file-structured. Then mount the disk upon which you wish to write the loader. |
| -OPEN, FPY0¶ | Open the drive non-file-structured. |
| -BOOTGEN,B FPY0:MS8¶ | Load and start the program, and pass start parameters to it. |
| -Bootgen Rx-yz | Signon from the program. |
| -Doing one-<br>Board floppy<br>boot! | Tells the type of boot written down. |
| -End of task 0 | Good task termination. |
| -CLOSE FPY0¶ | Finish up by closing the disk drive. It is now possible to boot from the disk. |

## 3.3 MESSAGES AND DIAGNOSTICS

The following BOOTGEN messages may be displayed on the screen:

| Message | Description |
|---|---|
| a) Bootgen, Rx-yz | Is the sign-on at the start of your program. Note x is called the revision level and yz is called the update level. |
| b) End of task s | An error has occurred The SVC error status is s, where s appears in Appendix B. Depending upon the error, the following diagnostics will be output to the screen: |
| c) Please reload program, not restartable | It is impossible to restart the program without reloading it first. |
| d) No parm | The start parameter is missing. |
| e) Inv. name | You have a Syntax error, or the device/filename is missing. |
| f) Disk - asgn | You failed to assign the disk device. |
| g) Disk - read | There is a READ error either on sector 0 or in the directory. |
| h) Not found | Your filename has not been found in the directory. |
| I) Disk - write | There is a WRITE error on one of the sectors 0-4. |