ADMINISTRATOR'S GUIDE TO THE ALPHA MICROSYSTEM AM-100® AND AMOS®

by Leo J. Rotenberg

Table of Contents

PREFACE

The purpose of this administrator's guide is to present the concepts and procedures necessary for the smooth operation of the Alpha Microsystem AM-100 computer and its time-sharing system AMOS.

We suggest the following plan for using this guide: you should have read, or be reading, the AMOS manual from Alpha Microsystems. This is the basic reference work giving facts about AMOS, except for the fact that as AMOS is changed, new and changed features are documented only in a release notice. Therefore you need to keep some release notices with your AMOS manual, and update your copy of the manual as releases arrive.

The AMOS manual will familiarize you with Alpha Micro vocabulary and acquaint you with how things look. You should read about the EXEC, which accepts commands from the terminal and loads and runs programs for you. The EXEC prompts you to enter a command by displaying the character "." on your terminal. Commands have the form "PROGRAM PARAM1 ... PARAMn", which you can think of as the same as "VERB OBJECT1 ... OBJECTn", and the meaning of the command is to do VERB to the named OBJECTs, or to run PROGRAM with certain actual objects as arguments, the latter being named by the parameters PARAM1 ... PARAMn. Often the object that a program is to work on is a file of data, and when this is the case, the associated parameter is a file name, like "DSK0:MESS.DAT[7,4]".

If there is something in this administrator's guide that you don't understand, it might be explained in the AMOS manual, and vice versa. The bulk of this guide is devoted to explaining the web of administrative arrangements required in using time-sharing, but three sections at the end, about EDIT, the text formatter, and command files, are tutorial introductions intended to supplement the information presented in the AMOS manual.

When your organization or household acquires an Alpha Microsystem AM-100, someone will have to be responsible for certain administrative functions which lead to the smooth functioning of the system. These administrative duties are described briefly in this section. Several of these duties are necessary because the Alpha Microsystem AM-100 comes with a time-sharing operating system (called AMOS). The power that time-sharing brings to your computing workload is well worth the somewhat increased complexity of the computer system itself and the duties required of the computer system administrator.

The duties and responsibilities of the computer system administrator are:

1. To guide programmers and users into compatible modes of sharing the machine. The Alpha Microsystem AM-100 can be used by more than one person "at the same time," because it has time-sharing. But, because the resources of the machine are limited, programmers and users must be guided into compatible sharing modes. To assist the computer system administrator in doing this, this manual contains information on space allocation theory, configuration management, and compatible computer usage modes.

2. To introduce programmers and other users to the computer and its terminals all users need to know about prompting characters emitted by system software and control characters (e.g., CTRL-C) available to terminal users. Non-programmer users need to know about the software they will be working with. Programmer users need to know the EDITor, at least one programming language system, existing file formats, local programming standards, etc.

3. To teach programmers to back up their files. Several procedures for file backup are included in this manual. Programmers should be responsible for backing up their programs, test data, etc.

4. To organize and monitor backup procedures for application data. The "live" data that relates to your organization *must* be backed up *at least* once per day. The procedures for file backup in this manual should be applicable.

5. To assign project numbers and programmer numbers (PPNs) to programmers and other users. PPNs are used to name areas that organize the storage of information on disk, and to define the scope of password protection. Assignment criteria are discussed in this manual.

6. To maintain passwords and system security. Passwords are used to restrict access of users to computing facilities. Procedures are included in this manual.

7. To maintain the inventory of disks. Disks must be prepared for use. Procedures are included in this manual.

8. To install new versions of system software from Alpha Microsystems. A general procedure, which can be tailored to your particular needs, is included in this manual.

9. To maintain backups of system software received from Alpha Microsystems.

10. To monitor computer usage and computer performance/availability.

ORGANIZING DISK STORAGE

Project numbers and programmer numbers are used to organize disk storage and establish protection boundaries in the Alpha Micro Operating System (AMOS). Together, a project number and a programmer number (or PPN) name an area of disk storage where files may be stored.

For example, suppose project number 20 designates your accounts receivable project and programmer number 4 designates John Doe, a programmer working on accounts receivable programs. Then John Doe's PPN would be [20,4], and [20,4] would name the area on each of the accounts receivable project's disks where John Doe might store the files that he develops. John Doe would LOG in to [20,4] to get access to his files.

The computer system administrator will assign project numbers to programming projects and operating departments that use the computer in your organization. The following considerations are important in assigning project numbers and programmer numbers:

1. Project numbers and programmer numbers must be octal numbers between 1 and 377.

2. Project numbers 1 through 7 are reserved by Alpha Microsystems.

3. Project number 10 is reserved by the author of this manual, for use by programmers in your organization.

4. Users having the same project number can write into and erase each other's files. Users having different project numbers and secret passwords cannot write into and erase each other's files. "Good fences make good neighbors."

5. Each different programming project probably ought to have its own project number.

6. For each computerized function in your organization, you might use two different project numbers: one to be used in connection with "live" production, i.e., the "real" operation of that function; and the other to be used by programmers while they develop new or modified programs to perform that function. When the responsible person decides that a program is ready to go into production, it is then moved from the "test" area to the "real" area.

7. Each sensitive and restricted computerized function can have its own PPN, protected by its own password, which can protect files stored in that PPN's area from unauthorized writing or erasing. However, the degree of protection provided by this arrangement depends on the memory protection feature which might or might not be part of your AM-100. A program that performs a sensitive or restricted function can demand another password. The degree of protection provided by this technique depends on how cleverly jumbled the program's copy of the password is, and on how securely the secret of the program is kept. All source copies of the program must be locked up.

   An example of a sensitive function which is usually restricted in use is alterations to accounts receivable records. Usually only a few people in an organization are authorized to alter such records, which are records of assets of the organization.

SPACE ALLOCATION THEORY

There are two important kinds of space (i.e., space for information to be stored) in the Alpha Microsystem AM-100: the main storage and the disk storage. In this manual, we assume that you have a sufficient, but not excessive, amount of main storage (semiconductor memory), and two or more disk drives for floppy (soft) disks.

The main storage is used for storing active programs, including the time-sharing operating system that implements the sharing of the machine among several users, plus the users' programs which the users cause to be loaded from disk by typing commands at their terminals. More main storage is required for blocks that define jobs in the computer, interface and terminal driver programs, buffers that hold characters going to and from terminals, and for bit maps read in from file structured disks. (The system uses a bit map for each disk.) Finally, certain system programs such as the BASIC compiler and/or the BASIC run-time package can be made resident in main storage and thus shareable by all the users without requiring multiple copies of the same program to be in main storage.

The space that is available for users' programs is the space that is left over after the system monitor is expanded by job blocks, drivers, buffers, bit maps, and resident programs. Therefore, there will be more space for a big user program if the system is configured for less jobs and terminals or if none of the sharable system programs are made resident.

All of the activities in your organization that use your Alpha Micro computer will generate requirements for main storage space. This includes both the activities of your organization (e.g., accounts receivable) and the added activity of computer programming, program testing and checkout, etc., that naturally comes along with the acquisition of a computer. The more you want to do with your computer "at the same time," the more main storage you will require.

The Alpha Micro Operating System has one very nice feature which will help you to reduce your total requirement for main storage. This is the capability of making shareable program modules resident in main storage with the system monitor. Such system-resident shareable program modules can be used by any number of jobs at the same time, even though only one copy of the module is in main storage.

For example, suppose your Alpha Microsystem is running an inventory query program that gives inventory information to any of five different terminals. Each of these terminals has a job in the computer attached to it, and each of these jobs uses the same program to respond to inventory queries. If this inventory query program is shareable and system-resident, only one copy of the program needs to be in main storage. If the inventory query program is not shareable and system-resident, five copies of the program will have to be in main storage - one per job.

The following programs are shareable and can be made system-resident:

1. BASIC, the BASIC compiler
2. RUN, the BASIC run-time package
3. ISAM, the Indexed Sequential Access Method package
4. All BASIC-compiled object programs
5. All Assembly language programs which adhere to the discipline of *reentrant* coding.

Disk Storage Allocation Theory

The Alpha Microsystem AM-100 can have one or more disk drives. One of these (drive 0, also called DSK0:) is reserved for a system disk containing a system monitor, initialization files, and all the system programs (e.g., BASIC, RUN, EDIT, MACRO, LINK, MOUNT, etc.). This system disk is mounted in DSK0: most of the time. It is almost full, and as Alpha Microsystems continues to expand the system, this disk will get more and more full. The other disk drives (DSK1:, DSK2:, etc.) will be used to hold disks containing your files. The disks that will be inserted in these other disk drives do not need to contain a system monitor or system programs, because these will be available from DSK0:. Such disks will contain user files only, plus passwords for user PPNs that protect those files.

The needs of your organization, and the services which you will require from your Alpha Micro computer, will dictate which files will have to be on-line at which times. (A file is *on-line* if it is on a disk which is mounted in a disk drive.) For example, the inventory query service mentioned above will require on-line access to a file telling the status of each inventoried item.

When you first start to use your Alpha Micro computer, you are likely to find that all your programs and data will fit on one disk. However, this wonderful situation will not last forever. You will notice that you are running out of free space on your disk(s).

A floppy disk in the IBM format, which is addressed as "DSKn" holds 500 blocks of 512 bytes each. This is sufficient space for about 120 typewritten pages at 2100 characters or bytes per page, which would be typical for double-spaced pages. Or, if you need to store names and addresses, one disk in the IBM format will hold about 2,721 of them at 93 bytes each (4 lines of 22 bytes each plus 5 bytes for a zip code).

If your system includes the AM-200 disk controller, you can reformat your disks into the denser AMS format. Such disks are addressed as "AMSn", and each one contains 616 blocks. However, you should not attempt this conversion until you thoroughly understand the three pages of instructions from Alpha Microsystems entitled, "Converting To The AMS Diskette Format."

In order to help you work with limited disk storage space, this manual contains procedures for programmers to use to share the available space. These procedures are contained in the section, "Compatible Computer Usage Modes." These procedures assume that some space is available on DSK0: for temporary use, and another disk drive (DSK1:) is available for temporary use in copying information to and from the space on DSK0:. The disk drive (DSK1:) which is used temporarily by your programmers in the course of program development, should not be used for storage of files that need to be on-line, because your programmers will be inserting their own disks in DSK1: whenever they need to.

You should keep the following rules of thumb in mind: (1) The more you want the computer to do "at the same time," the more disk storage you need. (2) You can't do much of anything with only one disk drive, because of space shortage and possible lack of backup. (3) You probably can't have programmers doing program development and keep large files on-line (both at the same time) with only two drives. (4) You will probably need at least four disk drives if you will be keeping large files on-line and having programmers developing large programs at the same time. Don't forget that disk drives contain moving parts and will fail from time to time.

The rules of thumb given above contain some fuzzy words like "programmers" (number unspecified) and "large." You will have to count bytes, records, and blocks to figure out how much disk storage space you need to do the job you have in mind. An example of such a computation is given above for the case of storing names and addresses. Programmers will be able to estimate how many blocks of disk storage space they require to do their assigned work.

It is possible for AMOS, the Alpha Micro Operating System, to configure itself to run different numbers of jobs and terminals, and to have different program modules resident with the system. This section describes how to make use of this feature of AMOS.

System Initialization

When the computer's RESET button is pressed, a fresh copy of the system monitor (i.e., the Alpha Micro Operating System) is brought into main storage from the file DSK0:SYSTEM.MON[1,4], and this monitor proceeds to configure itself according to instructions in the file DSK0:SYSTEM.INI[1,4]. The decisions about how many jobs, how many and what kind of terminals, how big the terminals' buffers should be, how many bit map buffers, what programs should be resident with the monitor, etc.; are all encoded in the initialization file DSK0:SYSTEM.INI [1,4], which is implicitly associated with the RESET button.

The contents of system initialization command files like DSK0:SYSTEM.INI[1,4] are described in a document which is provided by Alpha Microsystems, titled "System Initialization Command File (Version X.X)." Be sure to read the warning printed at the end of this document, because this warning is meant to be taken seriously!

Each job in the system being configured must be named in the JOBS command in the system initialization file. Each job must have a unique name.

Each terminal in the system being configured must be named and defined in a call on the TRMDEF command. There will be one TRMDEF command in the system initialization file for each terminal being defined. The parameters of the TRMDEF command specify the interface type, terminal type, and buffer sizes in addition to the terminal name. The interface type and the terminal type are both names of device driver programs which must be stored in disk area DSK0:[1,6]. Drivers not needed may be stored on another disk and eliminated from DSK0: to save space.

The ATTACH command is used to link a named terminal to a named job so that the job is thereafter controlled by that terminal. However, the first named job is always automatically ATTACHed to the first named terminal.

The SYSTEM command is used to load shareable program modules which are to reside with the system monitor. Also, when called without a file name argument, the SYSTEM command is used to signal the end of the process of initializing the system monitor.

The FORCE command is useful in an initialization file as a means of automatically FORCEing simulated terminal input (commands or data) into a job. For example, if a terminal is supposed to be used to make inventory queries 100 per cent of the time, this can be set up by FORCEing commands into the job attached to that terminal to LOG in, grab a MEMORY partition, and enter the inventory query program. Since this is done at system initialization time, the terminal will be ready to do inventory query as soon as the system is initialized.

SYSTEM RECONFIGURATION

Once the system configures itself and is up and running, it is possible to change to a different configuration through use of the MONTST command. For example, suppose the file DSK0:OTHER.INI [1,4] exists and is also a properly coded initialization file, like SYSTEM.INI. Then it is possible to get a monitor configure according to this alternate initialization file by typing the command

MONTST SYSTEM[1,4],OTHER.INI[1,4]

The first argument of the MONTST command names the file SYSTEM.MON ("MON" is the default extension for the first argument), so the same system monitor is brought in fresh from DSK0:. The second argument specifies an initialization file to be used in place of SYSTEM.INI in configuring this fresh monitor. There is no default extension for the second argument.

The MONTST command must not be used indiscriminately because it abruptly interrupts all the work being done by all the users of the computer! All programmer users should SAVE their BASIC programs, finish their EDITing, unmount their disk(s), and generally clean up as if they were about to leave the computer before the configuration is modified with MONTST. You should define procedures to be followed by non-programmer users to clean up their work to make the computer ready to deal with the MONTST command; the purpose of such procedures is to store all important information in disk files because the contents of main storage is discarded by MONTST.

Your Standard Configurations

In the natural course of events you will evolve a set of useful configurations for your organization. You should describe these configurations in a document which tells the purpose of each configuration, the name of the initialization file which brings up each configuration, the numbers of jobs and terminals in the configuration, the names of the resident shareable programs, etc. This list of configurations is referred to in the following procedure, "How To Start Up A Differently Configured Version Of AMOS."

HOW TO START UP A DIFFERENTLY CONFIGURED VERSION OF AMOS

This procedure should be used when you want to modify the number of working terminals on the computer, or when you want to modify the selection of resident system programs.

1. Obtain consent to modify the configuration from all the users on the computer.

2. *Wait* until all the users on the computer have arrived at a safe stopping point, e.g., BASIC users should SAVE their programs, users of EDIT should finish editing, all crucial data should be stored in disk files, etc.

3. Press the computer's RESET button. This causes a fresh copy of AMOS to be brought into main storage from the file SYSTEM.MON, and AMOS proceeds to initialize itself according to the commands in SYSTEM.INI. After this command file is executed, the first declared terminal will enter EXEC mode and prompt for a command to be entered by displaying a period (.).

4. Determine the name of the system initialization file which will bring up the desired configuration of AMOS. Use the table of configurations showing the standard configurations for your organization (not in this manual). If you choose the configuration associated with SYSTEM.INI, skip the remainder of this procedure. (SYSTEM.INI is permanently associated with the RESET button.)

5. LOG into any PPN on DSK0 if SYSTEM.INI has not already logged you in.

6. Type the command "MONTST SYSTEM [1,4], *file*.INI [1,4]" followed by CR, where *file*.INI is the file you chose in step 4.

7. If nothing goes wrong, a fresh copy of AMOS will be loaded into main storage and configured according to the instructions in the initialization file you chose. If you get one of the error messages, "?MONTST?" or "SYSTEM.MON NOT FOUND", make sure that you are using the correct system disk and get most of the system's main storage into your job's partition and try again.

METHODS OF FILE AND DISK BACKUP

The purpose of file and disk backup is to have duplicate copies of all files at all times. The reason for this is that disks sometimes fail, resulting in a loss of data. Murphy's Law suggests that disks will go bad at the most inopportune time. To prevent this from being catastrophic, it is necessary to regenerate the lost data. The easiest way to do this is to have an exact copy available on another disk.

The way to get a copy of an entire disk is to use the command DSKCPY. (See the procedure, "How to copy disks.") DSKCPY takes several minutes to copy your disk and then verify the copy by reading and comparing the disks. Use of DSKCPY will probably be regarded as an inconvenience by other users, if these other users' disk(s) will have to be removed to accommodate disk copying. Your organization should have a policy regarding when DSKCPY may be used, and by whom. DSKCPY is recommended for backing up system disks and disks holding very large files.

Another way to get duplicate copies of your files is to use the COPY command. This method is recommended to programmers. A programmer who has a file stored in the shared space on the system disk DSK0: would copy that file onto her file disk which is inserted in DSK1:. To get duplicate copies, she will have *two file disks,* and each of these will be inserted in turn to have files copied onto it by COPY. See the procedure, "How To Share Disk Storage Space." This procedure should be used by programmers in your organization as they develop and maintain software.

We suggest that you establish a schedule for making backup copies of disks and files. Twice a day is not too often. A written log of backup activity is a helpful record.

COMPATIBLE COMPUTER USAGE MODES

The purpose of this section is to describe how a group of users, including programmers, can conveniently and comfortably share the Alpha Microsystem AM-100. Basically, a group of users will share the machine comfortably if their combined requirements for information storage and access do not exceed the capacity of the machine. The machine's information storage capacity, as previously described, consists of main storage and disk storage.

The means for guiding users into compatible usage patterns are user education and a computer time sign-up sheet, which users will use to reserve rights of access to terminals, main storage, and disk storage.

In many typical organizations, the computer will be providing some kind of *on-line service* during normal working hours, such as the inventory query application mentioned above. If your organization decides to use such a service, there will be an absolute commitment of main storage and disk storage to provide that service during certain hours.

The remainder of the services provided by the computer can be characterized as *occasional services* which are continual rather than continuous (on-line) in nature. For example, printing mailing lists and updating accounts receivable information would be occasional services. These services might keep one terminal and one job busy for a few hours, perhaps not every day. The activity of programming the computer is also supported by occasional computer services.

In your organization, you will want to insure that a compatible pattern of computer usage is both possible and normal. What this means is that the occasional services must be compatible with the on-line services and with each other. You will need to insure that certain services are compatible when you are establishing schedles, and you will need to guage the degree of incompatibility when mediating scheduling conflicts that might arise. The main things to consider when assessing compatibility are main storage requirements, disk storage requirements, language requirements, and other parts of your computer system that might be required, such as printing terminals. These things are affected by the system configuration you choose: the numbers of jobs and terminals, etc.

Sharing Main Storage

If your organization requires an on-line service, the first priority in allocating main storage is that on-line service. If such an on-line service serves more than one terminal, then hopefully it includes shareable program modules which can be made resident with the system monitor (e.g., BASIC programs and RUN, the BASIC run-time package). Also, if the on-line service includes shareable modules which are likely to be useful in providing occasional services (e.g., RUN, the BASIC run-time package), then those shareable modules should be made resident with the system monitor even if the on-line service serves only one terminal. Don't forget that even if all of a job's programs are resident with the system monitor, the job will still require a small memory partition to hold program variables and the like.

In addition to shareable program modules which you make resident with the system monitor to support on-line services, you might want to make some program modules resident to support occasional services (e.g., BASIC (the compiler) to support programming). You should choose program modules in this category by getting a feel for what the users need and when they need it, and then matching the supply to the demand in a way that maximizes happiness, income, or both.

Scheduling Main Storage

The computer time sign-up sheet must state explicitly which systems programs will be resident during which hours or other blocks of time into which the days are divided. Users should be encouraged to sign up for time in those blocks that offer resident programs which they can use. The users who sign up are the occasional users, since the on-line services are scheduled onto the sign-up sheet before the occasional users get a chance to sign up.

Once a collection (possibly empty) of resident programs has been chosen for a given block in the schedule, and an amount (possibly zero) of main storage allocated to partitions of jobs providing on-line services, it can be known how much main storage remains for use by occasional users. This number (of bytes of main storage) will be noted in that block of the sign-up sheet. Then, when users come to the sign-up sheet to reserve computer time/resources, they will know how much main storage will be available during that block of time.

When a user signs up for a terminal in a block of the sign-up sheet, he will fill in his name and a statement of how much main storage he will be using, e.g., "2K." Thus, when another user signs up in that block, she will know how much main storage remains unreserved (by subtracting). Thus the sign-up sheet will prevent users' false expectations regarding how much main storage they can get.

Users of BASIC should not expect to get much done with less than 2000 bytes of main storage in their partition, assuming that BASIC and RUN are resident with the system. Users of EDIT should take at least 14,000 bytes.

Allocating and Estimating Main Storage

The mechanism for implementing the allocation of main storage space among users is the MEMORY command. This command establishes the size of the main storage partition belonging to a user's job. For detailed information regarding the use of the MEMORY command, see the procedure, "How To Share Main Storage Space."

There are three methods of getting an indication of how big a program is, i.e., how much main storage space it requires. The surest technique is to run the program with less and less space available, until you find the smallest partition size that the program will run in. This procedure can be dull and time-consuming, but it is the most accurate because you are working with the running program.

The SIZE command will tell you how many bytes a program occupies in its disk file, but this is usually less than the number of bytes required to run the program in main storage. The MAP command will tell you how much space a program occupies after it is loaded into your partition by the LOAD command, but as with the SIZE command, the number reported is usually less than the number of bytes required to run the program. This is because most programs allocate buffers for input/output after they receive control, and this requires additional main storage. You can LOAD several programs before running MAP to find their pre-execution sizes. Then you should type the command "MEMORY 0" which will remove the things you just LOADed into your partition.

Sharing Disk Storage

In the following discussion of means for sharing disk storage, we assume that there will be two "parts" of the disk storage system that will be shared: some space on DSK0:, the system disk, and disk drive 1. These assumptions are justified as follows:

First, we assume that you have one free disk drive for your occasional users to use to move their programs and data in and out of the computer. If you didn't have that much, your occasional users wouldn't get much done in any case. We assume that it is drive 1 that is available in order to make our discussion relevant to organizations having an Alpha Microsystem AM-100 with only two disk drives. If you have on-line files, please refrain from putting them on drive 1 unless you won't have occasional users while providing on-line services. If you have a third or fourth disk drive, put your on-line files there.

Second, we assume that there is some free space on DSK0:, the system disk. This free space is your responsibility to create, which you must do by ERASEing files from the AMOS system disk you receive from Alpha Microsystems. Of course, you will not actually throw away the files that you ERASE to free up space, because you will keep a backup copy of the unmodified software release disk that you receive. See the manual section, "Installation of System Software."

One group of files that you can ERASE is unused input/output driver programs stored in area [1,6]. If you will not be using Assembly Language, you can ERASE MACRO and its overlays, LINK, and SYM from area [1,4]; and also the assembly system libraries in [7,7]. Generally, you should examine all the areas with project numbers less than 10, because these are the areas reserved by Alpha Microsystems for its software.

There are several ways that might be used to organize the free space on DSK0:. One way to do this would be to give each user space on DSK0: in the area named by that user's PPN. This would be logical and straightforward, but wasteful of space on DSK0: because of the resulting proliferation of file directories on DSK0:. Another way to do this is to allocate a unique PPN on DSK0: for each terminal in the system, and then require each user to use the area associated with his/her terminal if he/she wants to use space on DSK0:. This technique avoids creation of masses of file directories on DSK0:, since the only file directories required are the ones associated with terminals. Also, this technique is efficient in that many users will share the file directories on DSK0: associated with terminals. The PPNs that name areas associated with terminals cannot have passwords since they would be shared by everyone in your organization.

We recommend the second way described above. We recommend that project number 10 be used in all the PPNs associated with terminals. PPN [10,n] shall be associated with terminal n. The terminal numbering can be assigned arbitrarily, taken from port numbers on the AM-300 serial interface board, or taken from the sequence of TRMDEF commands that define the terminals in your system initialization file(s). We recommend that each terminal be labeled with its terminal number, or its associated PPN, to remind the occasional user at that terminal to use that PPN on DSK0:. When a group of users is sharing the machine, they will have, collectively, more than one file disk containing files that they need access to. These file disks can't all be in DSK1: at the same time. This is when the shared space on DSK0: becomes important. Each user will insert his disk in DSK1: only long enough to COPY files from it into his terminal's area on DSK0: (when starting to use the computer); or long enough to COPY files from DSK0: back to DSK1: (when preparing to leave the computer). In other words, in order to share the machine, users will minimize their use of DSK1:, using it only to get files to and from their terminal's area on DSK0:. Thus each user will be leaving DSK1: free most of the time for use by other users. This technique is described in detail in the procedure, "How To Share Disk Storage Space."

The technique for sharing DSK1: outlined in the paragraph above will work perfectly well provided there is enough space on DSK0: for all the files of all the users on the machine. But inevitably, there will be a user requiring so much space that this technique will fail. The only solution for such a user is to give him primary access to DSK1:. In other words, that user will leave his disk inserted in DSK1: most of the time, removing it only to allow other users to move files to / from DSK0:. This user will have to be reasonably interruptible to respond to other users' requests for access to DSK1:. Also, two such users cannot share the machine at the same time.

Scheduling Disk Storage

As with the sharing of main storage, the sharing of disk storage will be mediated by the computer time sign-up sheet. When a user signs up for a terminal in a block of the sign-up sheet, the user will make a statement of how much DSK0: disk storage space (in blocks) he/she will be using, or whether he/she wants to take primary access to DSK1:. The sign-up sheet will help users avoid exceeding the capacity of the machine (which can only lead to unhappiness when it happens).

If you have several users who generally will be using the computer at the same time, you should consider the possibility of those users sharing a single disk. This would reduce the number of disks to be mounted at the shared disk drive.

HOW TO SHARE MAIN STORAGE SPACE

This procedure describes the use of the MEMORY command. If the storage requested by the MEMORY command is not available, it might be because some other user has grabbed additional storage, or it might be because the free storage is fragmented into pieces smaller than the size requested.

1. Arrive at the terminal; LOG in; and type the commands "MEMORY 0" and "MEMORY x" where x is the amount of main storage that you reserved. The command "MEMORY 0" causes your job's main storage partition to be reduced to size 0. The second MEMORY command will then establish your job's main storage partition size at x bytes. If you got what you reserved, skip the remainder of this procedure. Don't forget you need at least 2000 bytes.

2. Type the command "SYSTAT". SYSTAT will tell you how much main storage each job has grabbed.

3. If there is one user who has grabbed extra storage, that guilty user must come to a safe stopping point and then type the commands "MEMORY 0" and "MEMORY z" where z is the amount of main storage which that user actually reserved. Then you should return to step 1 of this procedure.

4. If the problem is caused by fragmentation (assume this to be the case if step 2 does not implicate any user), all users must come to safe stopping points and type the command "MEMORY 0". After *all* users have done this, each user should use the MEMORY command to grab his/her reserved share of main storage space.

HOW TO SHARE DISK STORAGE SPACE

This procedure describes how users will use the available space on DSK0: for temporary storage of their files, while maintaining two copies of each file by maintaining duplicate file disks. Each of the user's file disks is a backup for the other.

In this procedure (as in others), capitalized words (e.g., ERASE, COPY) refer to commands. You should refer to the descriptions of these commands in the AMOS manual to aid understanding of this procedure.

1. Arrive at the terminal you signed up for and get access to it from its previous user. Make sure the system is configured to run a job at that terminal. In the following steps (of both parts of this procedure), your terminal will be referred to as terminal n. We assume that n is an octal number associated with your terminal.

2. LOG in to DSK0:[10,n] or some other PPN that exists. No password is required to log in to PPNs in project 10.

3. Use MEMORY to get your share of main storage to run programs in. (See the procedure, "How To Share Main Storage Space.")

4. ERASE everything in DSK0:[10,n]. You can use the command "ERASE ". You can free up more space on DSK0: by doing the same thing for unused terminals: that is, if k is the number of an unused terminal, you may LOG in to DSK0:[10,k] and type the command "ERASE *" to ERASE all the temporary files in DSK0:[10,k].

5. Obtain access to DSK1: from the other users on the computer, insert one of your duplicate file disks in DSK1: and type the command "MOUNT DSK1" followed by CR (carriage return). Now COPY files you want to work with to DSK0:[10,n] from DSK1:[your PPN]. As you do this, you will be using up the available space on DSK0:. You should use SYSTAT to keep track of how much space remains on DSK0:, and you should use only the amount of space that you reserved, or obtain a new agreement with the other users on the computer regarding sharing of space on DSK0:. After you finish COPYing files, remove your file disk from DSK1: and type the command "MOUNT DSK1/U" followed by CR.

6. Do the things you want to do with the files you copied in to DSK0:[10,n]. This might involve changing some of those files and/or making new ones.

Getting Off The Computer

1. When you are ready to leave the computer, or if you need to free up space on DSK0: for yourself or other users, you must move some or all of your files from DSK0: back to your duplicate file disks. So obtain access to DSK1: from the other users on the computer, and perform the following steps for *each* of your duplicate file disks:

2. Insert your file disk in DSK1: and type the command "MOUNT DSK1" followed by a carriage return.

3. LOG in to DSK1:[your PPN]. This step will require you to use your password. (Skip this step if you are already logged in there.)

4. ERASE files from DSK1:[your PPN] that you have new versions of on DSK0:[10,n] that you want to move to DSK1:. Also, ERASE files that you don't need anymore.

5. COPY your new and revised files that you want to move to DSK1:[your PPN] from DSK0:[10,n].

6. If you have updated only one of your file disks, return to step 2 and update your duplicate file disk. Each of the duplicate file disks is the other's backup.

7. Otherwise, you have made your duplicate copies of the files you want to save. Remove your disk from DSK1: and type the command "MOUNT DSK1:/U followed by a carriage return to unmount the disk.

8. LOG back in to DSK0:[10,n] and ERASE the files that you just copied onto your file disks. This step frees up space on DSK0:

9. If you intend to work on some other files, return to step 5 under "Getting On The Computer".

10. If you are leaving the computer, type the commands "LOGOFF" and "MEMORY 0" before you go.

SYSTEM SECURITY FEATURES AND MAINTENANCE

The Alpha Micro time-sharing system AMOS provides a system of password protection to prevent unauthorized writing or erasing of files. Passwords may be associated with PPNs by the computer system administrator. Project numbers define the scope of protection of passwords. The following excerpt from the AMOS manual describes these protection features of AMOS.

> In order to gain access to the disk, the user must log into a PPN, and issue the password associated with that PPN if it has one. Otherwise the user has no system access. Once logged into a PPN area he may read any file in any area, but he may only erase or write into files in his own area or any other area that has the same project number as his area.

You will need to decide which groups of people in your organization should share the same project number. You will also have to decide to use passwords at all, since the null (length zero) password is a standard default. We recommend the use of passwords; we know of one computer department without security that is plagued by strange troubles whenever a certain programmer goes on a trip. When he comes back from his trip, he fixes the problem and gets the credit. His management thinks he's valuable but his brother thinks he's becoming a thief. If his management wakes up and catches him, they'll probably be too embarrassed to go to the police.

The computer system administrator will be responsible for explaining the system's protection features to users, and for maintaining users' passwords. Users can choose their own passwords, and communicate these to the computer system administrator by writing them on little slips of paper which they give to the administrator. The administrator will write the user's PPN on the slip of paper to avoid confusion about which password is whose. Then, using a private terminal, with no one looking over his/her shoulder, the administrator will enter the passwords onto the appropriate disks. Then the administrator will *destroy* the slips of paper (by burning, eating, or flushing away).

The computer system administrator will use the program SYSACT to enter and/or modif, passwords. To use this program, the administrator will have to LOG in to [1,2] and use his/her administrator's password. SYSACT will list the passwords currently on a disk on the administrator's terminal, so it is very important that the administrator use a private terminal to do this. This means being alone with the terminal: e.g., alone in a room with the door(s) closed and the windows covered. The terminal's screen should be cleared before the doors are opened. (Use of a video terminal for dealing with passwords is preferred.) The point of all this is to protect the secrecy of users' passwords.

Passwords are stored on disks. Occasional users will have file disks where they store their files, and these are the disks where users' passwords must be stored. Project number 10 will not have any passwords because it would be impossible to keep them secret. For detailed instructions on entering user passwords, see the procedure, "How To Set Up Disks For Computer Users."

MAINTENANCE OF DISK INVENTORY

The accompanying disk flow diagram indicates how to maintain an inventory of disks. The diagram shows the important states that a disk can be in, and how it gets from one state to another. The following paragraphs give a general overview of the disk flow depicted.

The purpose of disk inventory maintenance is to have on hand at all times a supply of file structured disks, which can be configured quickly for use by computer users, by simply adding a user's PPN to a disk. This configuration step is notated in the disk flow diagram by means of the boxes denoting sets of disks labeled "file structured disks" and "disks in use," connected by the box labeled "SYSACT." SYSACT is the program which adds PPNs to disks. In other words, the boxes andd arrows in the diagram say that the way to get a disk to be used is to start with a file structured disk, and do something to it with SYSACT. The exact something is specified in the procedure, "How to set up disks for computer users."

New disks are received from a vendor and must first be formatted (top of diagram). A formatted disk can be used immediately to receive a copy of data from another disk and thus become a backup disk, by using the program DSKCPY. This is described in the procedure, "How To Copy Disks." Also, a formatted disk can be initialized with file structure data by SYSACT. This is described in the procedure, "How To Put File Structure On Disks." See also the procedure, "How To Format New Disks."

A file structured disk is put in use by adding a user's PPN to it with SYSACT, as described above, and giving it or selling it to the user.

Disks in use must be tested periodically for defects, which is done with the program REDALL. This is described in the procedure, "How To Test Disks For Physical Defects." Users will be responsible for doing this with their own disks. Since disk defects are localized (usually), it is sometimes possible to copy good files off defective disks with the program COPY.

When a computer user no longer needs a disk, it can be used by another user. However, before such usage occurs, the disk must be tested for defects to protect its new user. As above, this is done with the program REDALL. Also, used disks should be labeled "USED," so the new user knows that the disk isn't new; and the disk should be reformatted to erase the previous user's data. (This protects privacy.)

Defective disks can be returned to the manufacturer in exchange for good new disks if the defective disk remains covered by warranty. In order to judge the different manufacturers' disks, it would be helpful to label each disk with the date it is put into service, but this isn't absolutely necessary.

The following pages contain the detailed procedures referred to in the above paragraphs.

HOW TO FORMAT NEW DISKS

This procedure assumes that two disk drives are working: drive 0 for the system disk and drive 1 for the disk to be formatted. Disk formatting is a necessary step, although some disks come from the manufacturer with IBM format data already written.

You can use AMS format or IBM format if you use the AM-200 disk controller. The format that gets written onto the disk depends on which formatting program you call up.

1. Obtain access to drive 1 from the other users on the system, remove the disk from drive 1 (if one is there), and type the command "MOUNT DSK1:/U " followed by a carriage return to tell the system that DSK1: is unmounted.

2. If you use the AM-200 disk controller or some other disk controller with a FORMAT ALLOW switch, turn it on. (The AM-200 disk controller printed circuit board can be found in the main computer box.)

3. Type one of the commands "FORMAT" or "AMSFMT" followed by a carriage return.

4. FORMAT or AMSFMT will request a drive number. Insert the disk to be formatted into drive 1 and type "1". Be *sure* you have typed "1"! (If you type "0" you will destroy the system disk!) Then type a carriage return.

5. FORMAT will write IBM format data onto the disk, or AMSFMT will write AMS format data onto the disk. This *erases* the data on the disk. When FORMAT or AMSFMT is finished, it returns to the EXEC.

6. If you have more disks to format, go to step 3.

7. Otherwise, turn off the FORMAT ALLOW switch that you turned on in step 2, if you have one. Do not skip this step!!

HOW TO COPY DISKS

Disk copying is required to make backup copies of entire disks.

1. Obtain access to DSK0: and DSK1: from the other users on the computer. Since you will be removing the system disk from DSK0:, other users will not be able to access system programs or the files they have stored on the system disk while you are using DSKCPY.

2. Remove the disk from DSK1: (if one is there) and insert the disk which is to receive the copy. This disk must be formatted.

3. Type the command "MOUNT DSK1:/U" followed by a carriage return.

4. Be sure that you have plenty of main storage in your job's partition or you'll crash when DSKCPY gets to the verification step. Then type the command "DSKCPY" followed by a carriage return.

5. DSKCPY will prompt for the input drive number. Remove the system disk from DSK0: and insert the disk to be copied. Type "0" followed by a carriage return.

6. DSKCPY will prompt for the output drive number. Type "1" followed by a carriage return.

7. DSKCPY will copy data from DSK0: to DSK1:, and then verify the copy by reading from both drives and comparing the data. This process requires several minutes. If a memory allocation failure message appears, verification cannot be attempted.

8. When your terminal returns to EXEC mode (indicated by the "." prompt), remove the disk from DSK0: and insert the system disk there.

9. Remove the disk which received the copy from DSK1: and label it, preferably with a sticky label. (A piece of paper kept in the disk's slip cover can also be used.) Then type the command "MOUNT DSK1/U" followed by a carriage return.

A disk containing files must also contain certain structural information (bit map, directories). This can be obtained by copying it from a disk that already has it, or it can be written on the disk by SYSACT. Note that a disk must be foratted before it receives file structure.

1. Obtain access to DSK1: from the other users on the computer.

2. LOG in to [1,2]. You will have to use your system administrator's password.

3. Insert the disk which is to receive file structure in DSK1:, and type the command "MOUNT DSK1" followed by a carriage return.

4. Type the command "SYSACT DSK1:" followed by a carriage return.

5. SYSACT will prompt you with an asterisk. Type "I" (for "initialize") followed by a carriage return.

6. SYSACT will ask if you really want to do that. Make sure that you specified "DSK1:" in step 4 above (you don't want to wipe out the system disk!).

7. Type "Y" followed by a carriage return. SYSACT will initialize the disk to contain no accounts, no passwords, no files, and a proper bitmap.

8. If you want to put PPNs onto this disk, go to step 6 of the next procedure, "How To Set Up Disks For Computer Users." Otherwise, type "E" followed by a carriage return to exit from SYSACT.

9. If you have more disks to initialize, go to step 3.

10. Otherwise, remove the disk from DSK1: and type the command "MOUNT DSK1:/U" followed by a carriage return.

11. Type "LOGOFF" followed by a carriage return, or LOG in under some other PPN, to prevent other users from using area [1,2].

HOW TO SET UP DISKS FOR COMPUTER USERS

The inputs to this procedure are project numbers, programmer numbers, and passwords. As described previously, project numbers are assigned by you, the computer system administrator, and passwords can be chosen by users.

Disks must be formatted and have file structure before they can be used in this procedure. (However, they don't have to be empty of files.)

1. Obtain access to a private terminal to do this work. A video terminal is preferable. Close the door(s) and cover the window(s) of the room. Passwords *should be kept secret*.

2. Obtain access to DSK1: from the other users of the computer.

3. LOG in to [1,2]. You will have to use your system administrator's password.

4. Examine the disk that you want to set up to be sure that it is properly labeled. If it isn't, label it with the user's name. Then insert it in DSK1:. Type the command "MOUNT DSK1" followed by a carriage return.

5. Type the command "SYSACT DSK1:" followed by a carriage return.

6. SYSACT will prompt you with an asterisk. Type an "H" to list SYSACT's functions.

7. Use the "A" function of SYSACT to add one PPN to the disk.

8. SYSACT will prompt you for a password. Type the password for the PPN you just entered.

9. If you have more PPNs to enter, go to step 7.

10. Otherwise, use the "L" function to list all the PPNs and passwords on the disk. Make sure this list is correct. You can use SYSACT to *Add* to, *Change*, or *Delete* from this list. If you make any further changes, repeat this step.

11. Type "E" followed by a carriage return to exit from SYSACT. Then *clear the screen* on your terminal.

12. If you have more file structured disks to set up, go to step 4 of this procedure. If you have more formatted disks without file structure to set up, go to step 3 of the previous procedure, "How To Put File Structure On Disks."

13. Otherwise, remove the disk from DSK1: and type the command "MOUNT DSK 1:/U " followed by a carriage return.

14. Type the command "LOGOFF" followed by a carriage return, or LOG into some other PPN, to prevent other users from using area [1,2].

15. *Destroy* the slips of paper telling user passwords; e.g., shred them into a toilet and flush them away.

HOW TO TEST DISKS FOR PHYSICAL DEFECTS

Used disks which are to be recycled to other users must be tested for defects before they are reused. Only zero-defect disks should be in use.

System disks should be tested for defects with this procedure every two weeks, and all disks should be tested at least every two months.

1. Obtain access to DSK1: from the other users on the computer.

2. Remove the disk from DSK1: (if one is there) and insert the disk which is to be tested in DSK1:. This disk must be formatted.

3. Type the command "MOUNT DSK1" followed by a carriage return.

4. Type the command "REDALL DSK1:" followed by a carriage return.

5. REDALL will read the entire disk and report any read errors encountered on your terminal.

6. If REDALL reports any read errors, interrupt it with CTRL-C. Label the disk as defective. If you want to recover files from the defective disk, use the COPY command to COPY good files onto a good disk.

7. If REDALL does not find any read errors, REDALL will return to the EXEC, which will then prompt you for another command by displaying a period (.).

8. Type the command "MOUNT DSK1:/U followed by a carriage return. Remove the disk from DSK1:.

9. Take defective disks to the computer system administrator for possible return to the vendor under warranty.

SYSTEM CRASHES (WHAT TO DO)

There are three known symptoms of system crashes: (1) one previously working terminal will stop working or get into an infinite output loop that can't be interrupted with CTRL-C, or (2) all terminals will stop working, or (3) the disk drive will go into a loop that involves repeated rhythmic clicking sounds (a read/write head is repeatedly loaded and unloaded).

CASE I

In the case of a one-terminal crash, the users at the other terminals should attempt to reach safe stopping points (e.g., finish EDITing, SAVE BASIC programs, etc.) Then the common crash recovery procedure (below) should be followed.

CASE II

In the case of an all-terminal crash, the common crash recovery procedure should be followed.

CASE III

When the disk starts doing rhythmic clicking, the disks that are inserted at that moment should be removed. Then turn off power to the disk drive, and turn off the computer. Then turn on the computer power, and then turn on the disk drive. Finally, perform the common crash recovery procedure.

COMMON CRASH RECOVERY PROCEDURE

1. Press the computer's RESET button.

2. If you removed any disks (CASE III), re-insert them.

3. Use MONTST to bring up the configuration you want to use.

4. Use DSKANA to recompute the bit maps and look for file errors on all disks that were mounted at the time of the crash. This can be done for two disks at the same time by two jobs (at two terminals).

5. *After* DSKANA has finished fixing up the disks involved in the crash, users may begin to get back on the computer.

6. Users should check to see if their files are intact. Lost files should be recovered from backup disks.

EDITor

This section of the manual is a tutorial introduction to EDIT, the system editor. This introduction does not attempt to duplicate the essential information about EDIT found in the AMOS manual; you should read that information and this information together to gain full insight, and to learn how to interpret the EDIT command descriptions in the AMOS manual.

EDIT always expects to be working with a pre-existing file. When you want to create a new file, you have to MAKE an empty file first (with the MAKE command) and then use EDIT to put data into it.

EDIT uses a temporary file and leaves a backup file in your file area. These files have the same name as your file, except for the extension part. These files, their names, and the manipulations of them by EDIT are fully described in the AMOS manual.

The files edited by EDIT are ASCII text files which are broken into *lines* of text by the ASCII characters CR (carriage return) followed by LF (line feed). In other words, each line of text is terminated with the characters CR and LF. ("CR" and "LF" are each one ASCII character.) In the world of EDIT, the *end* of a line is the LF that terminates the line, and the *beginning* of a line is the first character after an LF, or the first character in the file.

EDIT maintains a text buffer containing text from your file, and EDIT keeps one pointer called DOT that points into this buffer. You will have to be aware of the position and movements of DOT as you edit. DOT does not move randomly; its excursions are controlled by commands which you give to EDIT. If you do lose mental track of the position of DOT, your editing efforts will surely lead to frustration. While you are learning to use EDIT, you will have an easier time keeping track of DOT if you keep DOT at the beginnings of lines as much as possible. The EDIT command "0C" will move DOT to the beginning of the line it is currently in.

EDIT commands are cryptic looking strings of characters like "0C". EDIT prompts for a command to be entered by displaying an asterisk (*). EDIT puts the commands you enter into a buffer, in a process which is fully described in the AMOS manual. The command or commands in the buffer are carried out when you enter a double altmode, i.e., two presses of the ALT or ESC key on your terminal. The altmode echoes on your terminal as "$", and we use "$" in this manual section to represent altmode, as does the write-up of EDIT in the AMOS manual.

There are two EDIT commands which you will need to use to browse through a text file: "L" and "T". The command "L" or "nL" or "-nL" will move DOT forwards or backwards in the text buffer (see AMOS manual for details), and will leave DOT at the beginning of a line. The command "T" or "nT" or "-nT" will display the contents of the text buffer on your terminal, with one end of the display (beginning or end) being at DOT, and the other end of the display (end or beginning) being the end or beginning of a line. If DOT is at the beginning of a line, as recommended above, the "T" command (or any form thereof) will always display an integral number of whole lines. This helps avoid confusion. For example, suppose DOT is at the beginning of the text buffer and the text buffer contains the following:

```
FIRST LINE
START MUDDLE FINISH
THIRD LINE
```

Then the command string "T$$" will cause the following display:

```
FIRST LINE
```

and the command string "2T$$" will cause this display:

```
FIRST LINE
START MUDDLE FINISH
```

Multiple commands can be keyed into the command buffer before the double altmode that starts the execution of the commands. So the sequence of commands in the command string "2LT-LT$$" will cause the following display:

```
THIRD LINE
START MUDDLE FINISH
```

and leave DOT at the beginning of the second line. The first line displayed was displayed by the first "T" command; the second line was displayed by the second "T" command. From this point, the command string "2T$$" will cause this display:

```
START MUDDLE FINISH
THIRD LINE
```

and will not move DOT.

Text may be inserted into the text buffer with the "I" command. The text is inserted at DOT. When an entire line is being inserted, DOT should be positioned at the beginning of the line after the insertion point, and the inserted line should be terminated with CR (carriage return) followed by altmode (represented by "$"). The CR that you key at the end of the line will cause EDIT to terminate the line in the buffer with CR and LF. The altmode ("$") marks the end of the text to be inserted. If you omit keying the CR, the inserted material will become the new beginning of the line that DOT was at the beginning of, and DOT will be left in the middle of that line. For example, continuing the example above (recall that DOT was at the beginning of the second line), the command string "IANOTHER LINE /$$", where "/ CR /" stands for CR, will insert the line "ANOTHER LINE" into the text buffer, resulting in the following
text buffer contents:

```
FIRST LINE
ANOTHER LINE
START MUDDLE FINISH
THIRD LINE
```

with DOT positioned at the beginning of the line, "START MUDDLE FINISH".

Multiple lines may be inserted with one insert command, since the text to be inserted may contain any number of CRs. Insertion of partial lines will be described below, after the description of positioning commands.

DOT may be positioned into the middle of a line with the "C" command. The form "nC" moves DOT forward and the form "-nC" moves DOT backwards. Before using the "C" command, you should get DOT to be at the beginning of the line that you want to work in the middle of. For example, continuing the example above, the command string "7CT$$" will move DOT forward seven characters and then display the remainder of the line that DOT is in, as follows:

UDDLE FINISH

The "T" command did not display the beginning of the line that DOT is in, because DOT is not at the beginning of the line.

Text may be inserted in the middle of a line once DOT is positioned at the insertion point. Continuing our example, the command string "II$0CT$$" will insert the character "I" at DOT, move DOT back to the beginning of the line, and display the line, as follows:

START MIUDDLE FINISH

As a special case of insertion in the middle of a line, text can be inserted at the end of a line once DOT is positioned after the last printed character on the line, just at the CR and LF that terminate the line. When DOT is positioned thusly, the command "T" will cause the display of a blank line.

Text may be deleted from the text buffer with the commands "D" and "K". The "D" command deletes characters, while the "K" command kills (deletes) lines. Continuing our example, the command string "-K$$" will delete the line "ANOTHER LINE", and the command string "8CD0C$$" will delete the character "U" from "MIUDDLE" and return DOT to the beginning of the line. Now the text buffer looks like this:

FIRST LINE
START MIDDLE FINISH
THIRD LINE

You need to be very aware of the CR and LF characters at the end of every line when you are using the "D" delete command. If you want the line to end, you need both of them; whereas if you want to get rid of the "end of line" condition, you have to delete both of them. The "K" command always deletes both of them properly.

You can cause EDIT to exit to the system monitor with the "E" command. As for all other commands, two altmodes ("$$") must terminate the command string.

This completes the description of elementary EDIT commands. You can do any small EDITing job with just the commands listed above. If this is the first time you are reading this section, you should stop here and do some practicing. The following paragraphs introduce some more advanced EDITing features.

You can cause EDIT to search for a specified text string with the "S" command. This is very handy because it relieves you of counting characters. But you should not assume that it will always find what you intended! Also note that it almost always leaves DOT in the middle of a line. When learning to use the "S" command, use this command string: "Stext$0CTS$T$$". This will search for your text, display the whole line containing it, reposition DOT at the end of it (with the "S$"), and display that part of the line following it. These displays will help you catch possible errors.

You can move blocks of text around in the text buffer, or replicate blocks of text, using the commands "X", "V", and "G". These commands operate with auxilliary buffers (26 of them).

When you are editing a file which is so large that it won't fit into EDIT's text buffer in the memory partition available to you, you will need to use the "N" command to output your current buffer and bring in the text that follows. This is called "turning over buffers" and consists of the following: the current buffer is written out to EDIT's output file, the buffer is cleared, a new buffer full of data is read from the input file, DOT is jammed to the beginning of the buffer, and processing continues.

You should use the "N" command after you finish all EDITing you want to do to the current buffer, because the contents of the current buffer will be inaccessible to EDIT after the "N" command. The "N" command is a search command, like the "S" command. When using it to turn over buffers, be sure to make it search for something *it will find* in the next section of input file to be read.

The following EDIT commands are undocumented in our current AMOS manual:

Command	**Function**
FSold text$new text$	FINDS AND SUBSTITUTES the new text for the old text. The search for the old text begins at DOT. When the command completes execution, DOT is just after the substituted new text. If the search for old text is not successful, an error message occurs and the remainder of the command buffer is aborted.
FNold text$new text$	WHOLE FILE FIND AND SUBSTITUTE. The action of this command is similar to the action of command "FS". The new text is substituted for one occurrance of the old text. If the search for the old text does not succeed in the current buffer, and the current buffer is not the entire input file, then EDIT turns over buffers as described above, and the search for the old text continues.

TEXT FORMATTER

AMOS includes a very useful program named TXTFMT which can help you produce high-quality printed or typed documents with your Alpha Microsystem AM-100. All you need to use this program is a line printer or a printing terminal with a good quality type font to produce the finished document (or offset master, or the like).

The most useful feature of TXTFMT is text filling. This feature allows you to prepare input to TXTFMT in which you ignore the fact that you want the finished document to have regular left and right margins. TXTFMT will fill your text into the space between margins that you specify. For example, suppose the input to TXTFMT is our example from the section on the EDITor:

```
FIRST LINE
START MIDDLE FINISH
THIRD LINE
```

Then the output from TXTFMT would be this same text, filled into the space between TXTFMT's standard margins:

```
FIRST LINE START MIDDLE FINISH THIRD LINE
```

But we can specify any margins that we want to. If our input is:

```
/LINESIZE 20
/MARGIN 10
FIRST LINE
START MIDDLE FINISH
THIRD LINE
```

then the output will have to be contained in the space between a right margin at position 20 and a left margin at position 10:

```
          FIRST LINE
          START
          MIDDLE
          FINISH
          THIRD LINE
```

The text formatter has many features which aid the production of quality documentation. For details about TXTFMT, and a good example of what TXTFMT can do, see the TXTFMT User's Manual which is published by Alpha Microsystems.

COMMAND FILES

Command files are files of commands and (optionally) program input lines that you can specify to organize fixed, or partially fixed, sequences of computer operations. You have already seen one example of command files, i.e. the system initialization command file.

You should use command files to organize recurring jobs like report generation. You can save time and avoid recurring operator errors in this way.

A command file tells the sequence of operations to be performed by the computer. Each line is a command to be performed by the EXEC or input data for some program. Special commands control what appears at your terminal as the command file is executed, and permit input from the terminal.

Here is an example of a command file that specifies a classic "strip-sort-print" report generation:

```
:T
STOCK1
SORT STOCK.DAT
88
10
2
A

STOCK 2
ERASE STOCK.DAT
```

This command file calls the program STOCK1 to strip some data out of an existing fille (whose name is built into STOCK1), and write the intermediate file STOCK.DAT. Then the command file calls SORT to sort the file STOCK.DAT according to the parameters (88,10,2,A,blank line) which are the "canned" "keyboard input" to SORT to answer the questions about (record size, key size, key position, key order, next key size) which SORT requires to be answered before SORT will go to work. After SORT completes, the command file calls STOCK2 to print the report from the sorted data file, and then the command file ERASEs the sorted data file.

The command file processor in the EXEC of AMOS will process several special commands that are not available from a terminal. These are the commands that begin with a colon (":") which are described on page 7 of the AMOS manual. These special commands allow the command file to contain messages to be displayed at the operator's terminal, control output to the terminal, and permit selective operator entry of either monitor commands or program input. It is this last feature that allows partially fixed sequences of operations to be organized by command files: the varying input data can be keyed by the operator.

In effect, command files are programs in a high level language without if-then-else or "goto" statements. They have to be tested like any other programs.

Special-purpose command files may be stored in any PPN's area, and DSK0:[2,2] is the public command file storage area. Command files always have a file name extension of "CMD". Note that DSK0:[2,2] is searched before the area the user is logged in under when EXEC is trying to locate a command file called by the user (see page 6 of the AMOS manual).

INSTALLATION OF SYSTEM SOFTWARE

The purpose of this section of the manual is to tell how to take a disk containing a new version of AMOS from Alpha Microsystems, and transform it into a standard system disk for your organization. This task involves some judgment regarding the differences between the old and the new versions of AMOS, especially in the area of system initialization files.

1. Use DSKCPY to make a copy of the disk received from Alpha Micro. If lightning should strike while you are carrying out the rest of this procedure, or if some mundane failure occurs, you will need this backup.

2. Study the documentation of the new system release, especially the sections entitled "Update information for AMOS version X.X", and "System initialization command file (version X.X)." The former section should always be provided, while the latter section will be provided when there has been any change in the features or functions of commands that appear in the system initialization files.

3. Determine whether any changes are required in your standard system initialization command files. If changes are required, you will have to EDIT these files in step 13.

4. Obtain access to a private terminal, because some passwords must be entered in the following steps.

5. Obtain access to DSK1: from the other users of the computer. Insert the software release disk in DSK1: and type the command "MOUNT DSK1" followed by CR.

6. LOG in to [1,2].

7. Type the command "SYSACT DSK0:" followed by CR (carriage return). Use the "L" feature of SYSACT to list the PPNs and passwords on the system disk. Then type "E" to exit SYSACT. Leave the listing on your terminal's screen for the moment.

8. Type the command "SYSACT DSK1:" followed by CR. Use the "L" feature again to list the PPNs on the software release disk. Use the "C" feature to install passwords on the software release disk. You should duplicate the passwords on the current system disk unless you think they have been compromised. Use the "A" feature of SYSACT to add the shared area PPNs [10,1], [10,2], etc. These should not have passwords. Use the "L" feature again to check your work.

9. Type "E" to exit SYSACT, type the command "SYSACT DSK0:" followed by CR, and use the "L" feature to get another listing of the system disk. Compare these listings from the two disks. If you need to make any further changes, return to step 7. Otherwise, type "E" to exit SYSACT and then clear your terminal's screen. From this point forward, you don't need a private terminal.

10. LOG in to DSK1:[1,4]. (Not DSK0:!!)

11. Type the command "ERASE SYSTEM.INI" followed by CR.

12. Use the COPY command to move all your system initialization command files to DSK1:[1,4] from DSK0:[1,4]. This must include SYSTEM.INI.

13. If changes in the features or functions of system initialization commands require it, EDIT the files that you copied in step 12. Then type the command "ERASE *.BAK" followed by CR to eliminate the editor's backup files.

14. Type the command "SYSTAT" followed by CR. This will tell you how many free blocks remain on DSK1:.

15. If there isn't enough space on DSK1: to serve as the users' shared file area, find some rarely used files to ERASE from DSK1:. You have copies of these files on the backup disk you made in step 1. You ought to make a list of things you ERASE, both for the purpose of informing the users in your organization, and to help you perform this step quickly when the next release comes.

16. Use COPY to move program and data files that you keep on your system disk from DSK0: to DSK1:. If some of these data files are being updated by on-line services, this step must be delayed until the on-line services shut down, and they must stay shut down until you get to step 19. Maybe you should delay completing this procedure until night or next weekend, to avoid conflict with on-line services.

17. Use DSKCPY to make *two* backup copies of the disk that you have been working on in DSK1:. This disk is your new system disk.

18. Obtain consent from the users on the computer to change system disks. *Wait* for all users to arrive at a safe stopping point (as when a configuration change is to be made).

19. Insert the new system disk in DSK0: and press the computer's RESET button. Now check it out! Use your judgment to correct any problems that occur.

20. Install the new documentation (printed materials) sent with the new version of AMOS into your manuals, note changes there, and remove obsolete documentation.

APPENDIX A

DON'Ts

1. Don't use EDIT, or use BASIC to edit your BASIC programs, for more than a half hour without exiting the editor or SAVEing your BASIC program. This is because the system might crash and lose the fruit of your efforts. "SAVE early, SAVE often!"

2. Don't forget to use the MOUNT command. You might destroy your files if you forget.

3. When you get the error message "OUT OF MEMORY" from BASIC, don't do a RUN because it will crash the system. SAVE your program, exit from BASIC, and get a larger memory partition.

4. Don't try to EDIT with a memory partition smaller than 14K bytes.

5. Don't leave the FORMAT ALLOW switch on the AM-200 disk controller board turned on.

6. Don't give your variables in BASIC programs names that begin with BASIC reserved words.

7. Don't execute a RETURN in BASIC outside the context of a prior GOSUB.

8. Don't ALLOCATE a random file with a size of 0.

9. Don't type a CTRL-3 at your terminal. If you do, type CTRL-Q to recover. Evidently CTRL-3 and CTRL-S do the same thing.

10. Don't forget to read Appendix A of the AMOS manual. The do's and don'ts there are very useful. Read it once a month for four months.

APPENDIX B

CRYPTIC ERROR MESSAGES EXPLAINED

When MONTST says it can't find SYSTEM.MON, it doesn't really mean that. The problem is that either MONTST is running in a partition that is too small or SYSTEM.MON is not on the disk you are logged onto. Use MEMORY to get a larger partition and call MONTST again, or log into DSK0 and call MONTST.

"ERROR 10" means "there was one or more CRC (cyclic redundancy checksum) errors encountered on an unsuccessful track verification operation." It is possible to correct the error condition with DSKDDT. For example, if the machine reports, "ERROR 10 DSK1:RECORD 24" then you should type the command "DSKDDT DSK1: 24" followed by CR. DSKDDT will try to read the record eight times and then stop. You should then type "E" and DSKDDT will write the record back onto the disk with the correct cyclic redundancy checksum.

"ERROR 20" means "seek error" or "the desired track was not verified." Two different fixes for this problem have been reported for the Persci disk drive: (1) adjust resistor R-13, the 200Kohm frequency adjust trimpot that points down from the phase locked double-f data separator board, and (2) adjust R-3, the trimpot on the lamp amplifier board.

APPENDIX C

AM-100 PERFORMANCE DATA

The purpose of this section is to present the procedures and results of some performance measurement studies carried out on the Alpha Microsystem AM-100. The speeds of arithmetic, sequential I/O, and random I/O were measured.

The performance measurement of arithmetic was obtained with compute-bound programs doing floating point adds, subtracts, multiplies, and divides. Also, the performance of a program doing "nothing" was measured as a control. All of the arithmetic operands which these programs used were in the range between 1 and 10,000.

The performance measurement of sequential I/O was obtained with programs doing input alone and output alone, in the environment of an otherwise quiet machine. This test probably should not be used to infer performance of programs doing input from one file and output to another when both files are on the same disk, and it probably cannot be used to infer performance of similar programs running on an otherwise busy machine.

The performance measurement of random I/O was obtained with programs doing input alone and output alone, accessing the random file sequentially in the environments of a quiet machine and a busy machine. The data obtained in the environment of a quiet machine can be interpreted as a lower bound on the performance of random I/O; while the data obtained in the environment of a busy machine might represent the average performance of random I/O. The "busy machine" which was set up for these experiments had one compute-bound job and one I/O-bound job in addition to the job running the performance measurement experiments.

We hope you will not be extremely distressed if you find the results reported here tantalizingly close to something you really wanted to know, but sufficiently different to be utterly useless to your purpose. This is the fate of many performance measurement studies: the "wrong thing" turns out to have been painstakingly measured. The results might be very pretty but the practical outcome is nil. Just so you won't blame us if you arrive at such a conclusion, as you very likely will, this caveat to defuse your expectations is included here. We urge you to devise and carry out your own performance measurements, because you are the best judge of what you should be measuring.

The experiments reported here were carried out using an Alpha Microsystem AM-100 with an S-100 bus and an Icom FD3712-58 floppy disk drive, which spins the disk at 360 rpm and takes 10 ms. to move the access arm from one track to another (700 ms. for 70 tracks), plus 20 ms. mechanical head settling time, plus 40 ms. to load the head if it's not already loaded. The disk used was in the "DSK" format (not the "AMS" format). The execution times were reported by BASIC and derived from the AM-100's line time (60 Hz) clock. All of our results are plotted on graph paper with logarithmic-logarithmic axes.

Performance of Arithmetic Operations

The following program was used to measure the performance of the arithmetic operations. The program shown here represents a family of programs which was used to carry out the experiment.

```
10 PRINT "STARTING"
20 FOR A = 1 TO/LIMIT/
30 LET B = A/OP/ 100.751
40 NEXT A
50 PRINT "DONE"
```

This program was used to measure the speeds of addition, subtraction, multiplication, and division by making "/OP/" be "+", "-", " * ", and "/", respectively. The body of the loop, statement 30, was executed 100, 500, 1000, 5000, and 10,000 times by substituting these constants for "/LIMIT/". Also, this program was run without any statement 30 as a control, to measure the overhead incurred by the other program statements.

The times required to run these programs are shown in graph 1. The line labeled "control" shows the timings for the program without statement 30. The other lines show the timings for addition, subtraction, multiplication, and division.

Calculating from the results for 10,000 executions, we conclude that the average speed of addition is 1.39 ms.; of subtraction, 1.40 ms.; of multiplication, 1.65 ms.; and of division, 1.82 ms. However, because the variable A evidently is implemented as fixed point to control the FOR loop, these times probably include time for conversion of A from fixed point to floating point. Also, this test did not employ any very large or very small numbers, which take slightly more time to process when combined with numbers of widely differing magnitude. For a comparison of other computers' capabilities and listings of seven BASIC benchmark programs, refer to KILOBAUD Magazine's "BASIC Timing Comparisons", ISSUE Number 10, October, 1977, pp. 20-25.

Performance of Sequential Input/Output

The following program was used to measure the performance of sequential output operations.

```
20 OPEN #1, "DATA", OUTPUT
25 RING$ = " THIS IS A RECORD"
30 FOR A = 1 TO/LIMIT/
40 PRINT #1,A,A,A,A,A,RING$
50 NEXT A
60 CLOSE #1
```

The size of the output record was varied by including the variable A once, five times, or ten times in the list in statement 40. The number of records output was 5, 50, 250, 500, or 1000; with at least three of these figures being tested for each output record size. The file DATA.DAT was KILLed before each test of the program, so the times reported do not include time required to KILL DATA.DAT.

The times required to run these programs are shown in graph 2. The bottom line indicates the speed of the programs outputting short records; the middle line plots speeds for medium length records; the top line plots speeds for the long records.

We conclude that output of our short records requires 31-33 ms. per record; output of medium length records requires 96-99 ms. per record; and output of our long records requires 174-180 ms. per record. These times per record are marginal times, that is, times for each additional record. This includes time to convert the value of A to print format, plus time to write the record to disk.

The following program was used to measure the performance of sequential input operations.

```
20 OPEN #1,"DATA",INPUT
30 FOR A = 1 TO/LIMIT/
40 INPUT #1,B,C,D,E,F,RING$
50 NEXT A
60 CLOSE #1
```

The size of the input record was varied by including one, five, or ten variables in the list in statement 40. The number of records input was 5, 50, 250, 500, or 1,000; with at least three of these figures being tested for each input record size. For each test of this sequential input test program, the file DATA.DAT was produced by a corresponding version of the sequential output test program. By this means the file DATA.DAT was guaranteed to have the correct number of records and the correct number of variables per record for each test. The times required to run these programs are shown in graph 3.

We conclude that input of our short records requires 24-27 ms. per record; input of medium length records requires about 74 ms. per record; and input of our long records requires 131-133 ms. per record. As above, these times per record are marginal times. This includes time to read records from disk, plus time to convert the values in the records to internal floating point format.

Performance of Random Input/Output

The following program was used to measure the performance of random output operations in the environment of a quiet machine:

```
20 OPEN #1,"DATA",RANDOM,/ LENGTH/,REC
30 LET RING$ = " THIS IS A RECORD"
40 FOR REC = 1 TO/LIMIT/
50 WRITE #1,REC,REC,REC,REC,REC,RING$
60 NEXT REC
70 CLOSE #1
```

The size of the output record was made 24, 48, and 72 bytes by substituting these numbers for "/LENGTH/" in statement 20, and including the variable REC once, five times, or nine times, respectively, in the list of variables in statement 50. The number of records output was 5, 50, or 500. The file DATA.DAT was created with the statement

```
ALLOCATE "DATA",72
```

for use throughout the experiments on random I/O.

The times required to run these programs are shown in graph 4. The bottom line plots the speed of the program outputting short records; the middle line plots speeds for medium length records; the top line shows speeds for the long records.

We conclude that output of our short records requires 25-28 ms. per record; output of medium length records requires 54-56 ms. per record; and output of our long records requires 75-90 ms. per record. These times per record are marginal times; that is, times for each additional record. Because the random file is being accessed sequentially and the machine is otherwise quiet, these times are the best times that can be expected from random output using this particular floppy disk hardware.

The following program was used to measure the performance of random input operations in the environment of a quiet machine:

```
20 OPEN #1,"DATA",RANDOM,/ LENGTH/,REC
40 FOR REC = 1 TO/LIMIT/
50 READ #1,A,RING$
60 NEXT REC
70 CLOSE #1
```

The size of the input record was made 24, 48, or 72 bytes by substituting these numbers for "/LENGTH/" in statement 20, and including one, five, or nine variables, respectively, in the list of variables preceding RING$ in statement 50. The number of records input was 5, 50, or 500. For each test of this random input program, the file DATA.DAT was first initialized by a corresponding version of the random output test program. By this means the file DATA.DAT was guaranteed to have the correct number of records and the correct record length. The times required to run these programs are shown in graph 5.

We conclude that input of our short records requires 14-15 ms. per record; input of medium length records requires 31-32 ms. per record; and input of our long records requires 44-48 ms. per record. These times per record are marginal times. Because the random file is being accessed sequentially and the machine is otherwise quiet, these times are the best times that can be expected from random input using this disk drive.

The following two programs were used to create the environment of a "busy machine." Each of these programs had its own job and terminal. The first program is compute-bound:

```
10 LET I = 1
20 FOR A = 1 TO 500
30 LET B = A * 100.751
40 NEXT A
50 PRINT "DONE",I
60 LET I = I + 1
70 GOTO 20
```

The second program is I/O-bound:

```
5 STRSIZ 20
10 LET X = 0
20 OPEN #1,"SEQFIL",OUTPUT
25 RING$ = " THIS IS A RECORD"
30 FOR A = 1 TO 100
40 PRINT #1,A,A,A,A,A,RING$
50 NEXT A
60 CLOSE #1
70 OPEN #1,"SEQFIL",INPUT
80 FOR A = 1 TO 100
90 INPUT #1,B,C,D,E,F,RING$
100 NEXT A
110 CLOSE #1
120 KILL "SEQFIL"
130 LET X = X + 1
140 PRINT "DONE",X
150 GOTO 20
```

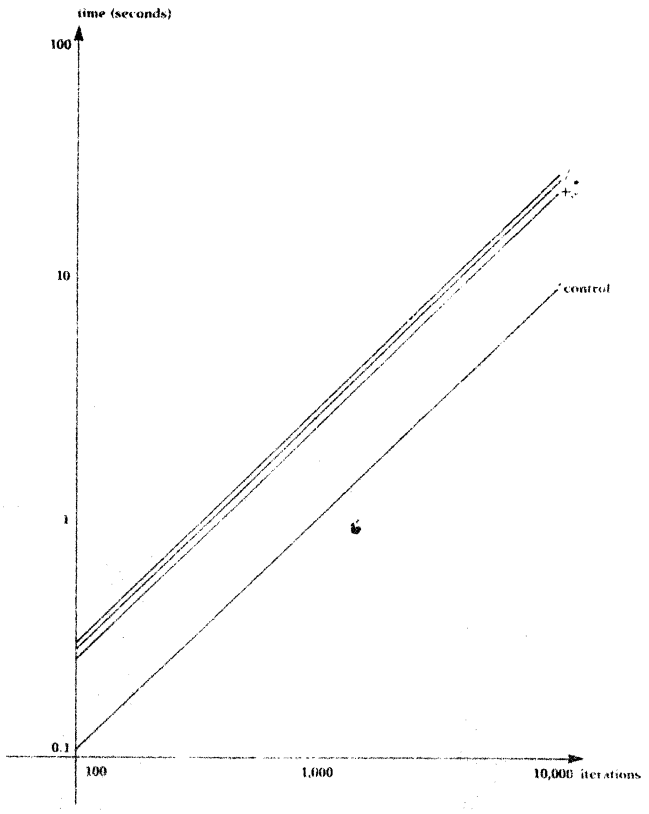
In the context of this "busy machine" environment, the random I/O programs were run again. These programs are the same as the random I/O programs just described, except for the addition of the statement

at the beginnings of both of them. The programs were run with the same record sizes and the same numbers of records as in the experiments just described.

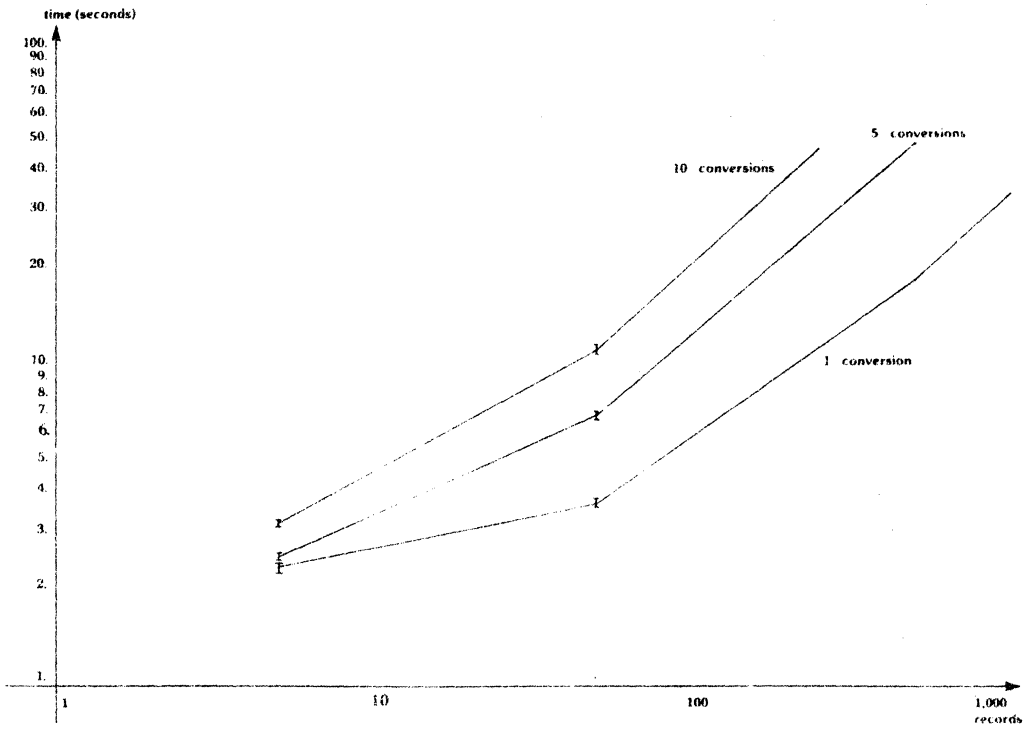
The times required to run the random output program in the "busy machine" environment are shown in graph 6. We conclude that in the "busy machine" environment, output of our short records requires 48-52 ms. per record; output of medium length records requires 79-99 ms. per record; and output of our long records requires 141-156 ms. per record. As before, these are marginal times.

The times required to run the random input program in the "busy machine" environment are shown in graph 7. We conclude that in the "busy machine" environment, input of our short records requires 18-34 ms. per record; input of medium length records requires 64-73 ms. per record; and input of our long records requires 84-104 ms. per record. As before, these are marginal times.

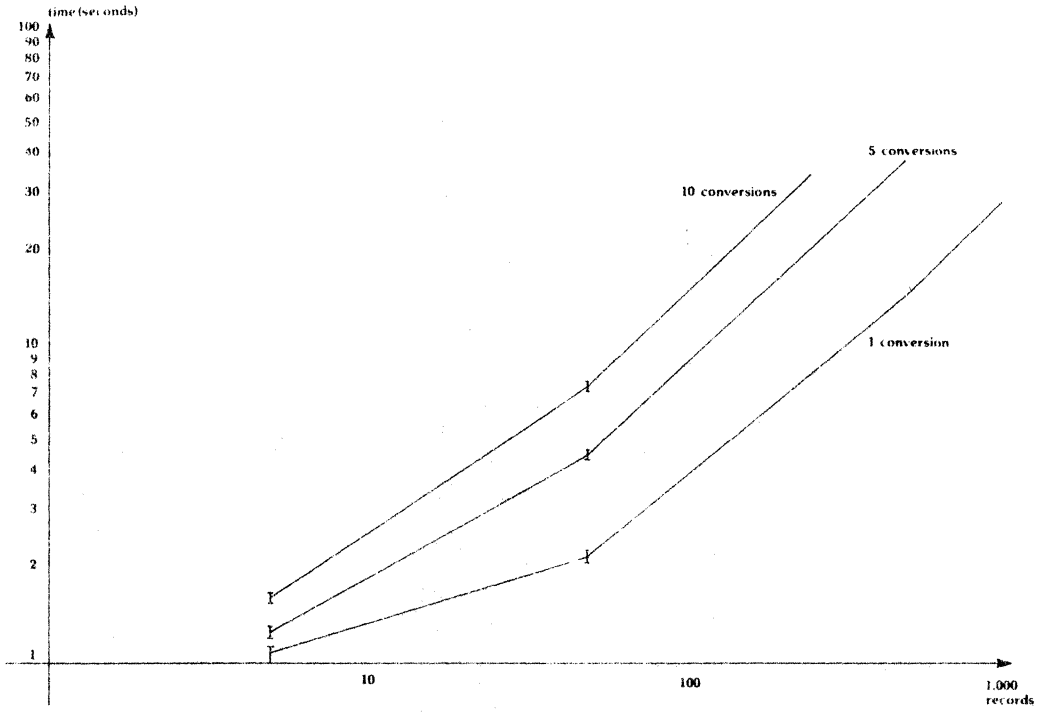
Comparing the "busy" environment with the quiet environment, you should note that things take longer in a busy environment. This effect is inherent to time-sharing. Also, note that the range of observed times plotted in the graphs is much larger in the "busy" environment. In other words, the speed of an individual transaction in a busy environment is less predictable.



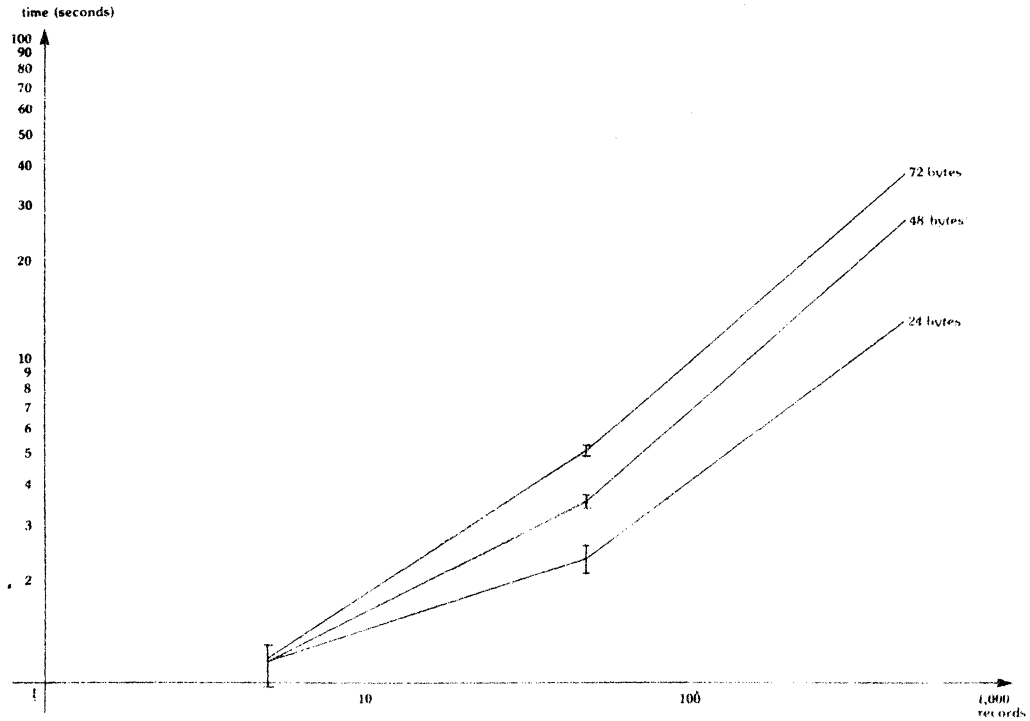
Graph 1



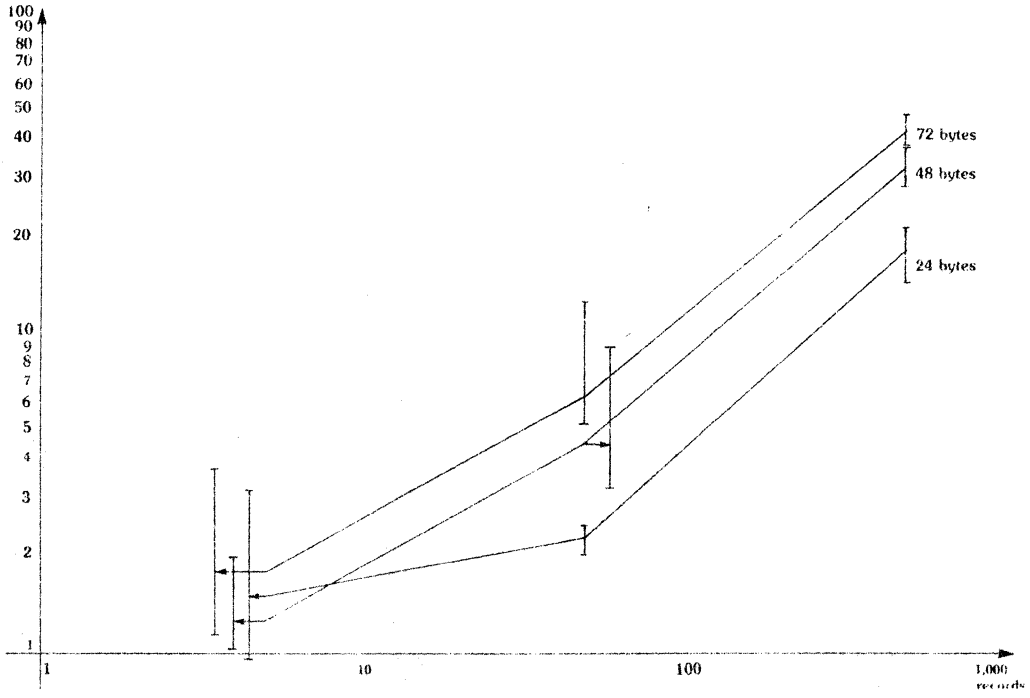
Graph 2



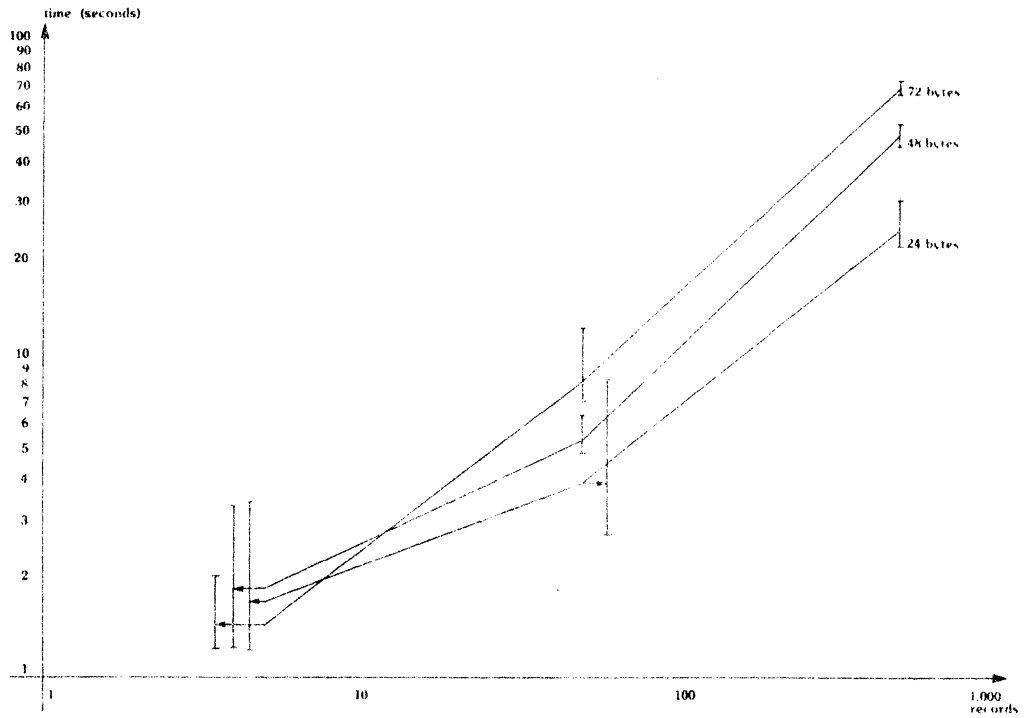
Graph 3



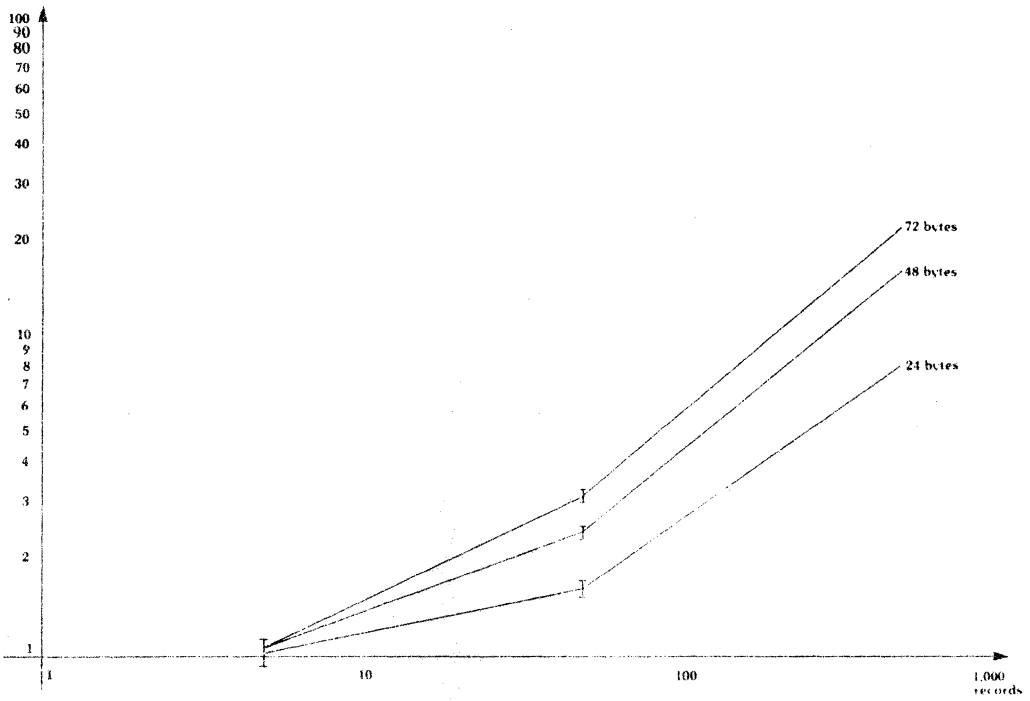
Graph 4



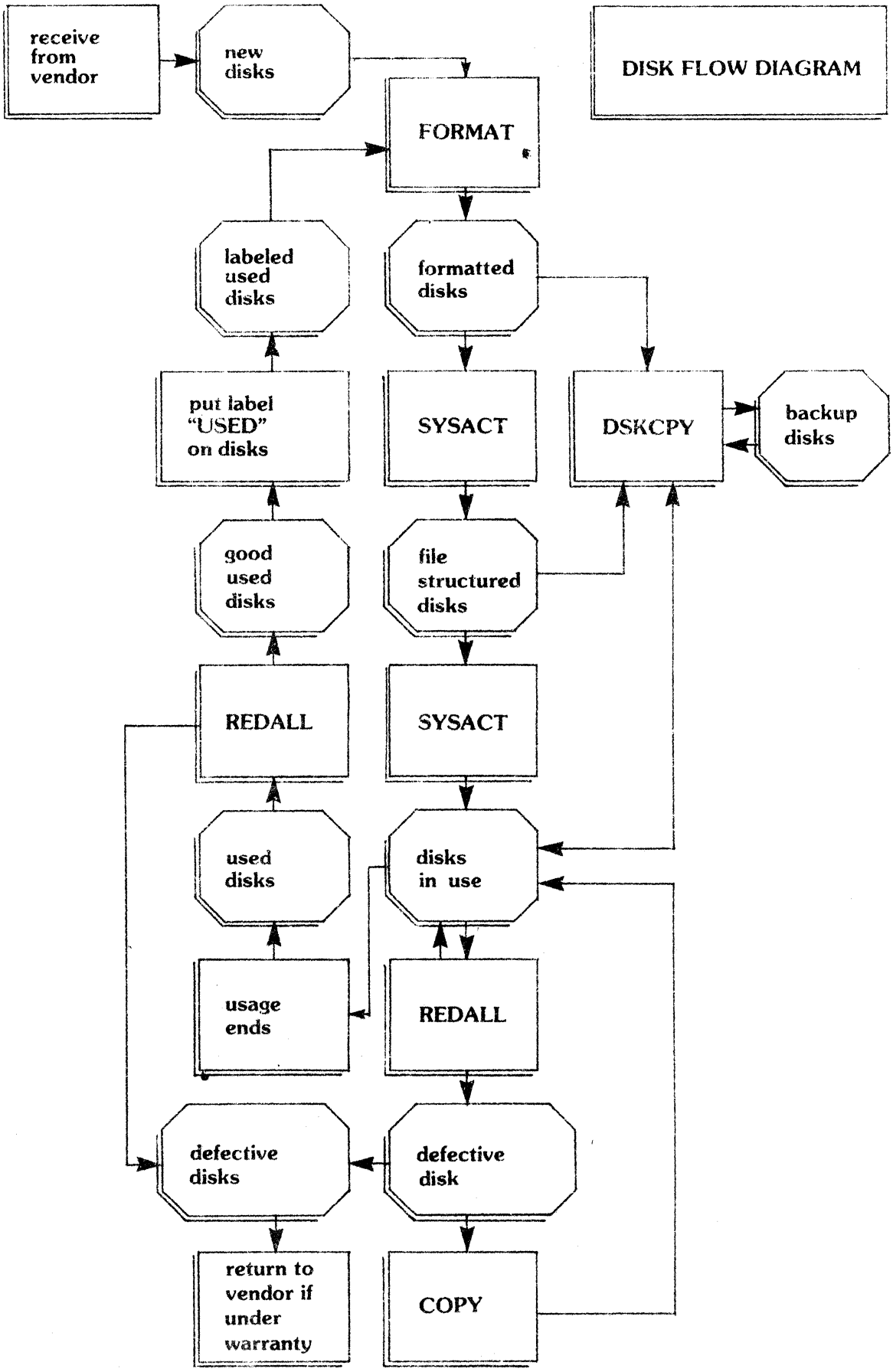
Graph 5



Graph 6



Graph 7


DISK FLOW DIAGRAM
receive from vendor
new disks
FORMAT
labeled used disks
formatted disks
put label "USED" on disks
SYSACT
DSKCPY
backup disks
good used disks
file structured disks
REDALL
SYSACT
used disks
disks in use
usage ends
REDALL
defective disks
defective disk
return to vendor if under warranty
COPY