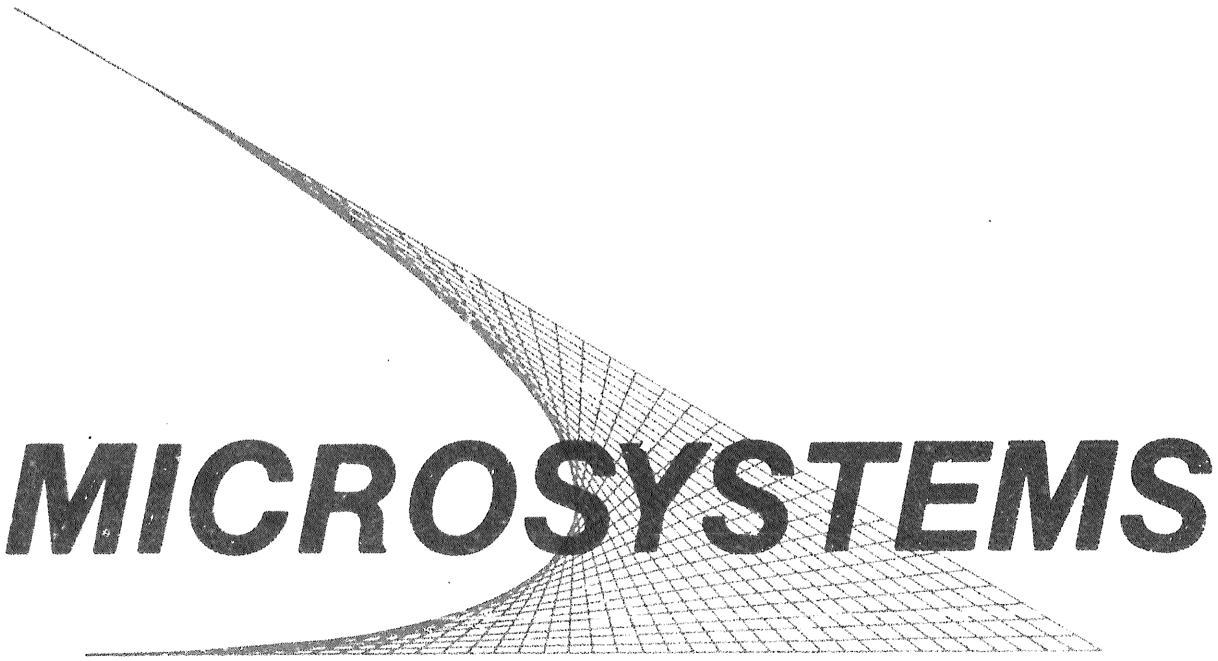
MOTOROLA

M68KVSF/D4

M68000 Family VERSAdos System Facilities Reference Manual



QUALITY • PEOPLE • PERFORMANCE

MOTOROLA

M68KVSF/A1

OCTOBER 1983

ADDENDUM TO M68000 FAMILY VERSAdos SYSTEM FACILITIES REFERENCE MANUAL M68KVSF/D4

This addendum transmits a replacement page for page 4-123/4-124. It reflects a change in the name of the chain file used to link TRANSFER, from LINKTRAN.CF to TRANSFER.LF.

Insert the new page into the manual and discard the original page of the same number. It is suggested that you insert this title page into the manual behind the front cover, as a record of the change.

MICROSYSTEMS

QUALITY • PEOPLE • PERFORMANCE

16886 PRINTED IN USA (10/83) MPS 3M

M68KVSF/D4

SEPTEMBER 1983

M68000 FAMILY
VERSAdos SYSTEM FACILITIES
REFERENCE MANUAL

Fourth Edition

Copyright 1983 by Motorola Inc.

Third Edition June 1983

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

TABLE OF CONTENTS (cont'd)

TABLE OF CONTENTS (cont'd)

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1
GENERAL INFORMATION

CHAPTER 1

GENERAL INFORMATION

1.1 OVERVIEW - INTRODUCTION TO VERSAdos

The VERSAdos operating system presents two faces to the environment. Seen from within the environment, VERSAdos can buffer an application program from the demanding direct interface with peripherals, yet does not deny such contact. Seen from without, it frees the user from the exacting requirements of direct interface with the machine and other system resources.

VERSAdos achieves its purpose by means of a modular design. Two of the major modules are:

a. A real-time multitasking executive kernel which includes a task controller, an inter-task communications facility, an optional memory management facility, and an initialization facility. A full description of this software is provided in the M68000 Family Real-Time Multitasking Software User's Manual, M68KRMS68K.

b. An Input/Output (I/O) module comprised of two facilities: Input/Output Services (IOS) and File Handling Services (FHS) which between them offer file and device access, file and device protection, and file management. A full description of this software is provided in the VERSAdos Data Management Services and Program Loader User's Manual, RMS68KIO.

A third major module, Session Control, supplies the means by which the user can interactively communicate his processing needs to the system. It is the purpose of this manual to describe these means and the VERSAdos command format, the disk and file structure, the system utilities, and various other functions provided by the VERSAdos operating system.

As used above and, unless otherwise specified, as used in the rest of this manual, the term "VERSAdos" refers to the VERSAdos operating system supplied for use on one of the following M68000 family-based systems:

. EXORmacs Development System

. VMC 68/2 Microcomputer System

. VME/10 Microcomputer System

. VERSAmodule 01 or 02 Monoboard Microcomputer (hereafter referred to as the VM01/VM02)

. VMEmodule Monoboard Microcomputer (hereafter referred to as the MVME110)

The standard utilities described in this manual are available to all the above systems, with the following exceptions. The MBLM utility is functional only with EXORmacs and VME/10 systems, and the SNAPSHOT utility can be used only with EXORterm 155 terminals. The DISMOUNT and MOUNT utilities are not functional on EXORmacs.

CONVEN-
TIONS

Most of the preceding systems use the firmware-resident debug monitor to boot the furnished initial program load file, IPL.SY, which in turn boots VERSAdos. VM01-based systems are initialized differently, because no IPL.SY file is supplied. Paragraph 7.3.1 describes modifications required for VM01-based systems to boot VERSAdos without the IPL.SY program.

The following lists the debug monitor provided with each system.

System Type	Debug Monitor
EXORmacs	MACSbug
VMC 68/2	VMCbug
VME/10	TENbug
VM01-based	VERSAbug
VM02-based	VERSAbug 2.n
MVME110-based	VMEbug

The SYMbug symbolic debugger and the debugging software DEbug are furnished with all VERSAdos systems.

1.2 CONVENTIONS USED IN THIS MANUAL

In addition to the use of the term VERSAdos just described, several other conventions are utilized in the following material. These include:

a. Unless otherwise noted, numbers are given in decimal representation.

b. Hexadecimal numbers are preceded by a dollar sign or denoted as hexadecimal.

c. Descriptions of command syntax are given in a modified version of the Backus Naur Form (BNF), a brief description of which is provided below:

< >	Angular brackets enclose a symbol, known as a syntactic variable, that is replaced in a command line by one of a class of symbols it represents.
|	This symbol indicates that a choice is to be made. One of several symbols separated by this symbol should be selected.
[ ]	Square brackets enclose a symbol that is optional. The enclosed symbol may occur zero or one time.
[ ]...	Square brackets followed by periods enclose a symbol that is optional/repetitive. It may appear zero or more times.

d. A carriage return is required after all operator entries. It is shown, as (CR), only where necessary for clarity. In some examples, operator inputs are shown underscored; the underscore is not to be typed.

Much of the information in this manual is summarized in the VERSAdos Reference Card (MVDOSCARD), which provides a quick reference guide for users familiar with VERSAdos.

FILE ORGANIZATION

1.3 VERSAdos FILE ORGANIZATION

Three types of files are created on a disk:

Contiguous	These are files within which records are adjacent and all records are 256 bytes long. They are mainly memory loadable modules.
Sequential	These are files within which records need not be adjacent or of the same length. They are mainly object or listing files created by the assembler.
Indexed Sequential	These are files of records to which keys can be attached allowing VERSAdos to access records within a file by key for quick retrieval and greater selectivity. They are usually text files created by the editor.

The type is, in general, transparent to the user -- i.e., he cannot choose the file type. Under some circumstances, however, sequential or indexed sequential file types can be selected when editing.

The VERSAdos I/O Services (IOS) accesses a sector within a file by converting the file's Logical Sector Number (LSN) to a Physical Sector Number (PSN), which gives the actual location of the sector on the disk. Logical sector numbers are assigned sequentially even though the sectors may not be physically contiguous. Thus, although segmented into randomly located blocks of sectors, a file is treated as a unit of logically contiguous sectors. The actual physical accessing of a particular sector is done by the disk controller.

At disk initialization, a Volume Identification Directory (VID) is created which contains volume ID, version and revision of the resident operating system, date of disk generation, user name identification area and user number, pointers to the start of the Sector Allocation Table (SAT) and the first Secondary Directory Block (SDB), and the length of the SAT. The VID must always reside at PSN 0. (See the REPAIR utility description for information on the VID.) The SAT contains a bit map in which cleared bits provide a one-to-one correspondence with sectors available for allocation.

A directory is maintained on each volume which, for each file on the volume, contains: the filename, extension, file start, file length, file type, write and read access codes, and the number and size of records in the file. A secondary directory contains an entry for each unique user number and catalog name on the volume.

So that the entire file space need not be contiguous, the disk sectors for a file are allocated in available groups of physically contiguous sectors called data blocks. This permits th dynamic allocation and deallocation of space without affecting the location of existing files. The File Access Block (FAB) describes the location of data blocks allocated to a file. A FAB can be from one to 20 sectors in length. The length of each data block for a given file is the same, although different files can have different data block sizes. The minimum data block length is currently four sectors.

COMMAND FORMAT

1.4 COMMAND FORMAT

A VERSAdos command line can contain five fields: command mnemonic, input, output, listing, and options. The input and output fields are supported by all utilities; the listing and options fields are supported by particular utilities. Maximum command line length is 160 characters. Multiple commands on the same line are not permitted. The command line is terminated with a (CR). The standard format for a command line is:

```
    <command mnemonic> [<input field>][,<output field>][,<listing field>]
    [;<options>]
```

1.4.1 User Session Default Values

When a user first logs on the system, the session control module initiates a session for the user on that terminal which continues until the user logs off. At the time of logon, default values for the volume ID, user number, and catalog name fields within the VERSAdos file descriptor are established to permit abbreviation of command entry. A particular command may not require an explicit entry in one or more of these fields. In this case, the appropriate session default value(s) is used as the value for the omitted entry.

The nature of the functions provided by some commands also permit command entry abbreviation to include fields for which session defaults do not exist -- i.e., filename, extension, listing, and options. In such cases, the appropriate values are known to the command and are supplied as required.

1.4.2 Input Field

The input or source field may contain device names or file descriptors which are appropriate for the command. File descriptors may be fully specified or abbreviated, in which case default values are assumed as required by the particular command. Concatenation of inputs is done by using the slash (/) character; e.g., file1/file2 or volume ID/file. Space characters are not allowed in the input field, which is terminated by one of the following:

a. The first comma (output field follows), or space.
b. Semicolon (no output field, but options present).
c. (CR) (no output or options field).

1.4.3 Output Field

The output or destination field may contain device names or file descriptors which are appropriate for the command. File descriptors may be fully specified or abbreviated, in which case default values are assumed as required by the particular command. Spaces are not allowed. The output field is terminated by:

a. Comma (listing field follows) or space.
b. Semicolon (options follow).
c. (CR) (no listing field or options).

1.4.4 Listing Field

Where a program or utility has the capability of creating a listing, the listing field may contain device names or file descriptors as appropriate. File descriptors may be fully specified or abbreviated, in which case default values are assumed as required by the particular command. The listing field is terminated by:

a. Semicolon (options follow).
b. (CR) (no options).

1.4.5 Options Field

The options field is opened by a semicolon, consists of alphanumeric and/or special characters, and is terminated with a carriage return. Specific forms of options are found in the descriptions of session control commands, system utility commands, and system languages.

1.4.6 Additional Options

The following three special options provide additional variations in command execution.

1.4.6.1 Command Mnemonic Field Option - At Sign (@). The symbol "@" may optionally be entered preceding a session control command or a system utility call to obtain a variation in the usual processing. If used before a command for which no variation exists, it is ignored. Specific uses and the variation obtained are shown below:

[@]<utility name>	
[@]STAR [<task name>]	Obtains prompt at start of execution (without @,
[@]CONT [<task name>]	task must be completed).
[@]BYE	END SESSION message printed (same as OFF).
[@]CHAI <filename>	Inhibits clearing of conditional processing of
[@]<filename>.CF	pseudo registers (RX, RA, RD).
[@]ASSI	Passes assignment to all successive user tasks loaded or started until a CLOSE is issued. (ASSI without @ passes assignments to next user task only).
[@]END	Terminates chain processing, regardless of nesting depth.

DEVICE NAME FORMAT

1.4.6.2 Input File Descriptor Fields Option - Asterisk (*). The "*", often called "wildcard", represents ANY character or name. When the * is the last character in a field, it means that the remainder of the field is any (or all) characters. An * in one or more character positions in a field means any characters in that/those positions. A field comprised of a single * means all characters. The following examples assume explicit or implicit entries in the preceding volume and user fields and a blank catalog field.

filename.*	All files with the specified filename having any extension.
*.extension	All files with any filename having the specified extension.
.	All files; i.e., those having any filename and extension.
TOM*.SA	All files beginning with TOM and having extension .SA.

1.4.6.3 Output Catalog Field Option - Ampersand (&). An explicit (non-space) catalog name field default value which was established at session 0001 or through subsequent invocation of the USE session control command can be blanked by entering the "&" symbol in the output catalog field of a utility command line -- for example:

```
USE <user>.&
```

Execution of the above command line changes the existing catalog name to eight space characters in the command line buffer which are displayed as a blank catalog field.

1.5 DEVICE NAME FORMAT

A device name can be specified in the input and/or output fields within the VERSAdos command line. Device name is a single field of one to five alphanumeric characters and the pound sign (#). Specific device names for a particular system configuration are established when the operating system is generated. Refer to the System Generation Facility User's Guide, M68KSYSGEN. Device names begin with a pound sign (#), usually have a 2-alphabetic character mnemonic, and usually have a 2-digit number. The "#" used without a descriptor refers to the terminal on which the user logged on to the system. Following are examples:

#FD00	Floppy disk drive 00	#HD00	Hard disk drive 00
.		.	
.		.	
.		.	
#FD03	Floppy disk drive 03	#HDnn	Hard disk drive nn
#CN00	Debug board port 00		
#CN01	Debug board port 01		
#PR	Printer		
#PRn	Printer n		
#	The logon terminal		

FILE DESCRIPTOR FORMAT

1.6 FILE DESCRIPTOR FORMAT

A file descriptor can be specified in the input and/or output fields within the VERSAdos command line. A fully specified file descriptor consists of six fields. The volume ID and user number fields establish location and ownership of the file. The catalog, filename, and extension fields identify the file. The protect key field provides the desired read/write access. The standard format of a fully qualified file descriptor is:

```
<volume ID>:<user number>.<catalog>.<filename>.<extension>(<protect key>)
```

where:

volume ID	A 1- to 4-alphanumeric character string with a leading alphabetic character. Must be terminated with a colon (:).
user number	One to four decimal numeric digits. Terminated with a period if any field follows. Must be preceded by a colon (:) if the file extension is not specified.
catalog	A blank field or a 1- to 8-alphanumeric character string with a leading alphabetic character and terminated by a period. If the catalog field is not preceded by a user number field, the file name and extension must be specified.
filename	A 1- to 8-alphanumeric character string with a leading alphabetic character (except temporary files beginning with "&" and spooling files beginning with "@".)
extension	One or two alphanumeric characters with a leading alphabetic character and preceded by a delimiting period (.).
protect key	A string of two or four alphabetic characters from A through P only; must be enclosed in parentheses.

Brief descriptions of each of the six file descriptor fields follow.

Volume ID

This 4-byte ASCII field specifies the ID of the disk or diskette where a file exists or is to be allocated. The field identifies an online, direct-access volume.

User Number

This field is the number from 0 through 9999 that is entered to represent file ownership.

The values are:

1-9999	Indicates a private user or owner file.
0	Indicates a system file for the administrator, user 0.
*	Asterisk is the user number "wildcard" that transfers a -2 default value to the File Handling System (FHS). Can represent any user number, 0 through 9999.

Catalog

This 8-byte ASCII field is used for a more general level of file identification than that provided by the filename.

Filename

This 8-byte ASCII field is the main identifier for a file.

Extension

This 2-byte ASCII field provides further file identification according to data type. The extension name can be user-specified except for default extensions reserved for use by VERSAdos and its utilities. These extensions are:

AF	Assembly file
AG	Source file assembled at SYSGEN time
AI	Assembly include file
CD	Command code fragments used to build other files
CF	ASCII command file (chain file)
DB	Symbolic debug format
IN	Initialization file
LF	Link file
LG	Link file used at SYSGEN time
LL	Link edit memory map
LO	Load image format
LS	Assembler or Pascal (Phase 2) listing
MX	S-record format
NW	Session management or news file
OV	Overlay image format
OW	Session management
PC	Pascal intermediate code
PF	User profile file (SYMbug)
PL	Pascal (Phase 1) listing
PO	Optimized Pascal code
RO	Relocatable object format
RS	Symbolic debug format
SA	ASCII source
SY	System files; e.g., VERSADOS.SY, ERRORMSG.SY
TF	Temporary file
XX	General instruction file

These extensions may be used for other purposes as long as the user is aware of the meanings VERSAdos assigns to them.

Protect Key

This 2-byte binary field holds a fixed code which limits read and/or write access to a file. Combinations of the characters A through P specified when a file is created are translated to corresponding binary values which are used by the FHS subsystem to control access to the file. A non-owner must then match the assigned protect key to obtain the permitted access.

A key is comprised of two or four characters. Two characters control read access. Four characters control read/write access. Of the possible combinations of the allowed key characters a file creator could specify for conditional access, four are reserved. These provide the access control shown in the following table, and are non-conditional -- that is, a user need not match a key to obtain the assigned access.

SYSTEM DISK DESCRIP-TIONS

RESERVED KEY W	RESERVED KEY R	PERMITTED WRITE ACCESS	PERMITTED READ ACCESS
00	00	Owner	Owner, Administrator
00	PP	Owner	Public
PP	00	Owner, Administrator	Owner, Administrator
PP	PP	Owner, Administrator	Public

(If unspecified, default protect key is PPPP.)

If desired, the RENAME utility can be used by a file owner or the system administrator to change the protect key field of the file descriptor.

1.7 SYSTEM DISK DESCRIPTIONS

This section describes the contents of the system disks or diskettes supplied for use with hard-disk-based or diskette-based systems. Table 1-1 lists the initialization procedure required by each VERSAdos-compatible system and the documentation in which the procedure is described. Modifications to the initialization procedure required for Winchester/SASI drivers and VM01-based systems are described in paragraph 7.3 of this manual.

TABLE 1-1. System Initialization Procedures

SYSTEM TYPE	INITIALIZATION PROCEDURE	REFERENCE DOCUMENT
EXORmacs	via VERSAdos operating system	EXORmacs Development System Operations Manual, M68KMACS
	or	
	with MACSbug BO command	MACSbug Monitor Reference Manual, M68KMACSBG
VMC 68/2	via VERSAdos operating system	VMC 68/2-Series Microcomputer System Manual, MVMCSM
	or	
	with VMCbug BO command	VMCbug Debugging Package User's Manual, MVMCBUG
VME/10	via VERSAdos operating system	VME/10 Microcomputer System Overview Manual, M68KVSOM
	or	
	with TENbug BO command	TENbug Debugging Package User's Manual, M68KTENBG
VM01-based	with VERSAbug BO command	VERSAbug Debugging Package User's Manual, M68KVBUG VERSAdos 4.2 Custommer Letter, M68KSYSLET
VM02-based	with VERSAbug 2.n BO command	VERSAbug 2.n Debugging Package, M68KVBUG2
MVME110-based	with VMEbug BO command	VMEbug Debugging Package User's Manual, MVMEBUG

1.7.1 Diskette-Based System

Diskette-based EXORmacs systems are provided with diskettes having part numbers identified in a letter supplied with VERSAdos. Each diskette initialized for VERSAdos contains a Volume Information Directory (VID). The complement of files on a diskette can be obtained by logging on the system as user 0 and interrogating the VID on that diskette through use of the DIR utility (see Chapter 4).

The diskette inserted in drive 0 is called the system diskette. Because the system is booted and initialized from it, this diskette must contain the initial program loader file, IPL.SY, and the operating system file, VERSADOS.SY. Once the system is brought in, this diskette may be replaced with another containing the files required for a particular activity. Files are grouped on a diskette according to general function. On the diskette with VERSADOS.SY, for example, are the most often-used utilities.

A volume ID is associated with each diskette that has been initialized for VERSAdos. It is essential that at any one time, no two diskettes in the system have the same volume ID if files are to be referenced on both. In the same name case, only files on the first volume installed in a drive could be referenced. The only communication allowed with the second volume would be the physical sector I/O performed by such utilities as BACKUP and DUMP. In spite of this warning, the volume ID of a backup diskette need not be changed from that of the master, since these two are together in the system only at BACKUP time. It is useful to include the file BACKUP.LO on user diskettes so that the BACKUP command can be invoked and the diskette copied to a scratch diskette in another drive. Such minimum system diskettes might be created to provide space on the system diskette for temporary files or for editing, compiling, and assembling with single-sided diskettes.

User diskettes are those that normally reside in drive 1 in a dual-drive system, or in drive 1, 2, or 3 in a four-drive system. Dual-drive system user diskettes should contain, as a minimum, the files VERSADOS.SY and BACKUP.LO for the above reasons. With a four-drive system, it is not necessary to maintain any operating system files on the user diskettes, because the system diskette can be kept in drive 0 while backup is taking place between diskettes in drive 1, 2, or 3.

1.7.2 Hard-Disk Based System

An EXORmacs or VMC 68/2 system based on a hard disk is supplied with a blank fixed disk and a disk in a removable cartridge which contains all required system files. As shipped, the cartridge volume ID and user number are SYST:0. The initial start-up process initializes the fixed disk as SYS:0 and copies files from SYST:0. Included are the chain files STARTUP and RESTORE, the execution of which initializes the fixed disk and performs the file copying.

A VME/10 is supplied with system files on a fixed Winchester disk that has a default volume name of SYS.

Copies of the fixed disk and, when applicable, of the removable disk should be made and held in reserve. The duplicating process is described in the EXORmacs Development System Operations Manual, the VMC 68/2-Series Microcomputer System Manual, and the VME/10 Microcomputer System Overview Manual. For VM01/VM02- and MVME110-based systems, refer to the BACKUP utility, paragraph 4.2.1 of this manual.

1.7.3 System Diagnostics

On diskette-based EXORmacs systems, one diskette (Volume ID = DIAG) contains the diagnostic programs for the system resources. On hard-disk based systems, the diagnostics are included on the system disk.

More information on these files and their use can be found in the EXORmacs Development System Maintenance Manual (M68KDMM). Diagnostics for the VMC 68/2 system are described in the VMC 68/2-Series Microcomputer System Manual. Diagnostics for the VME/10 system are described in the VME/10 Microcomputer System Diagnostics Manual, M68KVSDM.

CHAPTER 2
THE FIRST SESSIONS

CHAPTER 2

THE FIRST SESSIONS

2.1 BASIC CONSIDERATIONS

The purpose of this chapter is to assist the inexperienced user in developing application programs. The facilities most generally used are discussed briefly, leaving many of the less consequential details to be treated in later chapters. It is assumed that the user has access to VERSAdos, including the data management services and utilities options, on one of the systems listed in paragraph 1.1.

2.1.1 Software Development

NOTE

> Unless otherwise specified, the designations "M68000" and "MC68000" will refer to the entire M68000 family of microprocessors.

Source programs are created for assembly or compila[illegible]nd subsequent debugging using the M68000 Family CRT Text Editor because of [illegible] facility it offers for the entry and modification of program text. Program text is comprised of source statements written in assembly language or another high-level language -- such as Pascal, FORTRAN, or COBOL -- for which the user has a compiler that will run on the MC68000-based system. Characters are entered from the EXORterm keyboard or other system console.

The edited source program is assembled or compiled into a relocatable object file. The M68000 Family Linkage Editor is then called to join that file with selected files from the user's library of relocatable object modules to create an absolute load module which can be loaded into memory for execution and/or debugging. These early steps may be repeated many times to form the libraries from which files are drawn for the final linking into the application program. An overall software development scheme is shown in Figure 2-1.

Programs can be developed on a host MC68000 system for use in systems based on MC6800 family microprocessors including the MC6800, MC6801, MC6805 and MC6809. Such development requires a cross assembler or cross compiler program and a cross linkage editor program. To facilitate the transfer of this code, the cross linkage editor transforms relocatable object modules into an output file of easily transportable S-records. Once downloaded into memory, the facilities of the target system are used to transform the code from the output file into a loadable, executable module.

Programs could also be developed on a non-MC68000-based computer system for MC68000 execution. This development would utilize a cross assembler or cross compiler and (on an IBM 370 system) a cross linkage editor which would execute on the host system but produce object code for the MC68000 machine. The resulting program would be downloaded into the MC68000-based system for the final linking and/or debugging.

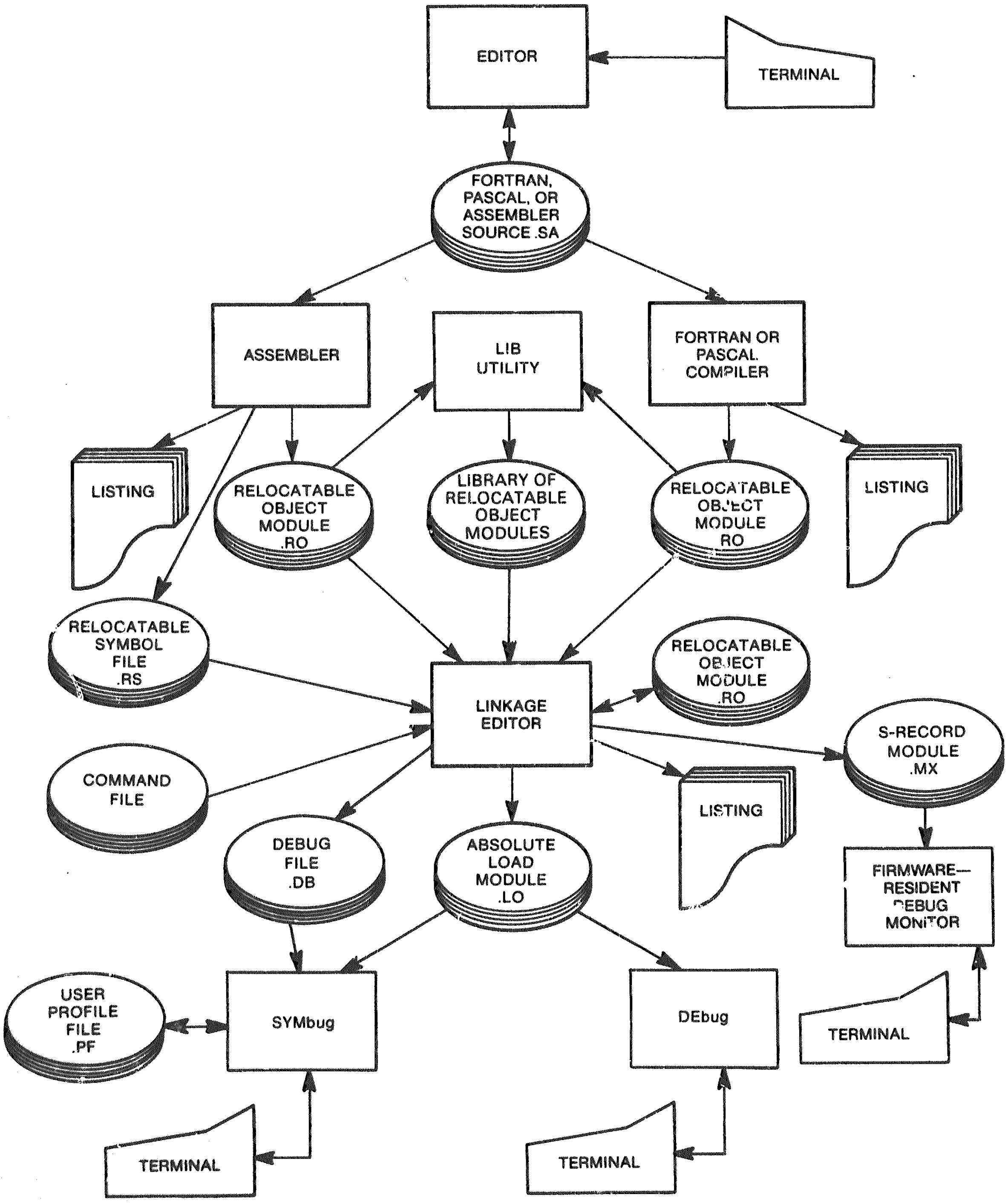

EDITOR
TERMINAL
FORTRAN, PASCAL, OR ASSEMBLER SOURCE .SA
ASSEMBLER
LIB UTILITY
FORTRAN OR PASCAL COMPILER
LISTING
RELOCATABLE OBJECT MODULE .RO
LIBRARY OF RELOCATABLE OBJECT MODULES
RELOCATABLE OBJECT MODULE RO
LISTING
RELOCATABLE SYMBOL FILE .RS
LINKAGE EDITOR
RELOCATABLE OBJECT MODULE .RO
COMMAND FILE
S-RECORD MODULE .MX
LISTING
DEBUG FILE .DB
ABSOLUTE LOAD MODULE .LO
FIRMWARE—RESIDENT DEBUG MONITOR
USER PROFILE FILE .PF
SYMbug
DEbug
TERMINAL
TERMINAL
TERMINAL

FIGURE 2-1. Software Development Process

PROCESSING MODES & OFTEN-USED FACILITIES

Details of the resident and cross assemblers and compilers are provided in the respective manuals for the programs. Descriptions of the debugging programs supplied with VERSAdos -- DEbug and SYMbug -- are provided in the M68000 Family SYMbug/DEbug Monitors Reference Manual, M68KSYMBG. A firmware-resident debugging program is also provided with each VERSAdos system. Table 1-1 lists the appropriate reference manuals for these programs.

2.2 PROCESSING MODES

Several operating modes are provided to support program development from a single terminal or multiple terminals in the interactive online mode and in the batch mode. A user is permitted to initiate multiple, concurrent batch mode tasks, yet remain in the interactive online mode of another task if he chooses.

2.2.1 Batch Processing

In the VERSAdos environment, the user not only can start his own job but can continue operating in the foreground mode while his job is being processed in the background mode. Batch submissions are made possible through a structure of commands; the decision of which batch job to process is handled by a circular job queue. This processing mode is described more fully in paragraph 3.2.1.

2.2.2 Online Processing

Online processing, roughly comparable to foreground processing, includes execution of commands entered from the console and execution of commands read from a chain file. The latter is called chain mode processing and is useful to the application program developer since it can remove the need to reenter every command in an often-used sequence. Chain mode also offers argument substitution, which further reduces the programmer's work since it can eliminate the need for repeatedly typing in a long argument, such as a complete file descriptor, when an argument must be changed. Chain mode processing is described more fully in paragraph 3.2.2.

2.3 OFTEN-USED FACILITIES

The process of developing an application program on a new development system can be broken down into general steps, many of which are the same as would be taken on another system. This section outlines some of the more common steps. It is assumed that a version of VERSAdos is running on the system for the purposes of the examples which follow. Unless otherwise stated, it is also assumed that the user has logged on the system under his own user number (not 0).

SYSTEM DEFAULT VALUES

CREATING OR MODIFYING FILES

2.3.1 Verifying or Setting System Default Values

Before beginning work and until the user is familiar with the system, it is a good idea to review and change, if necessary, the user default values. The DEF and USE session control commands are used to do this. When the user becomes more proficient with the system, the OPT session control command can be used to adjust the control of user session management over the particular session. As one example, the user can have the processor stop execution of his task on receipt of a break code.

When the DEF command is input, a message similar to the following is displayed:

```
    DEF
1)  SYSTEM VOLUME = SYS:  (hard disk-based system)
2)  USE DEFAULT VOLUME = SYS:213.NEW.
3)  USER NUMBER = 213
    USER TASK =
    SESSION = 003A
    TERMINAL = CN22
    OPTION(S) SET
```

Line 1 shows that the system was initialized from volume SYS: -- line 2 shows that the user's default volume is the same with a default catalog name of NEW. The user number in line 2 should be the same as the user number in line 3 (the log on user number); otherwise, access to files on the default volume may be prevented and other difficulties may be encountered. If the user is working with a program on a diskette, for example, the length of entered commands can be shortened by making the diskette his default volume and blanking the default catalog name (eight blanks using the ampersand as the blanking character.) The user would do this by invoking the USE command as follows:

```
    USE VOL1:.&
```

The resulting display would be identical to that shown for DEF but for line 2) which would now be:

```
2)  USE DEFAULT VOLUME = VOL1:213..
```

Hencetorth, user 213 can access files by name on the specified volume without entering the tully specitied tile descriptor. The same technique can be used when the user has files within different catalogs on the same volume.

2.3.2 Creating or Modifying Files

As stated earlier, the CRT Editor is a facility used for creating a source file. When called from a console to create a new file, the editor comes up in the CRT Mode, Page Level, as indicated by the editor prompt (>) in the home position on the screen. A typical call is:

```
    E TESTPROG
```

Source lines can now be entered into the new file TESTPROG.SA. The extension .SA is assumed if another is not specifed. At least one line must be entered betore quitting the editor, or the file is discarded. To exit the editor, depress the F1 key (the > prompt goes to lower left of the screen, indicating that the editor has entered the command mode), and enter QUIT. There now exists an indexed sequential file named TESTPROG.SA with the volume ID, user number, and catalog name previously shown when the DEF or USE command was executed. The process for altering an existing file through use of the editor is similar. Figure 2-2 is the listing of the example assembler source program, TESTPROG.SA.

```
MAINMOD  IDNT     1,1                *MAIN TEST PROGRAM
*                               *THIS PROGRAM TESTS THE FEASIBILITY
*                               *OF CALLING ASSEMBLER ROUTINE
*                               *PRNTLIN2 FROM A PASCAL PROGRAM

         XDEF     START           *PROGRAM ENTRY POINT
         XREF     PRNTLIN2        *ROUTINE TO BE TESTED

         SECTION  1
         DS.B     128             *RESERVE STACK SPACE
STACK    DS.W     0               *
IOSBLK   DC.B     0               *PARAMETER BLOCK START
         DC.B     2
         DC.W     8
         DC.B     0
         DC.B     6
         DC.W     0
         DC.L     0
         DC.L     BEGMSG
         DC.L     ENDMSG
         DC.L     0
         DC.L     0               *PARAMETER BLOCK END
BEGMSG   DC.B     'THIS ASSEMBLER PROGRAM PRINTS LINE ONE'
         DC.W     $0D0A
         DC.B     'THEN CALLS PRNTLIN2 TO PRINT LINE TWO.'
         DC.W     $0D0A
ENDMSG   EQU      *-1

         SECTION  0
START    LEA.L    STACK,A7        *INITIALIZE USER STACK PTR
         LEA.L    IOSBLK,A0       *LOAD IOS PARAM BLK ADDRESS
FRSTLIN  TRAP     #2              *EXECUTE
NEXTLIN  BSR      PRNTLIN2        *CALL EXTERNAL MODULE
         SUB.L    A0,A0           *
         MOVEQ    #15,D0          *TERMINATE TASK
         TRAP     #1              *
         END      START
```

FIGURE 2-2. Assembler Source Program TESTPROG.SA

CREATING A LOAD MODULE

ASM

2.3.3 Creating a Load Module

After a source file has been prepared by means of the CRT Editor, the next step is to assemble the source file if written in M68000 family assembler, or to compile the source file if written in a high-level language. After the relocatable object file is obtained, the linkage editor is called to transform the file into a load module. These processes are discussed briefly in the next three paragraphs.

2.3.3.1 <u>ASM - Transforming an Assembler Source Program into a Relocatable Module.</u> If called and passed the name of a source file, the resident assembler will produce, by default, a relocatable object code file and a listing file having the source file name but another extension -- i.e., relocatable object .RO, listing file .LS. Also by default, the assembler will display warning messages. Any of these default activities and other assembler features can be disabled or enabled by specifying the appropriate characters in the command line options field. These are fully described in the respective manuals for the programs. Figure 2-3 is the listing obtained from assembly of the sample program of Figure 2-2 by executing the command line:

```
ASM TESTPROG;MRD
```

A later example will demonstrate that a Pascal program can be linked to an assembler subroutine as well as to a Pascal subroutine. The example subroutine ASMRTN.SA is shown in Figure 2-4 and the listing obtained from its assembly in Figure 2-5.

```
     MOTOROLA M68000 ASM VERSION x.x FIX : 212        .TESTPROG.SA 05/23/83  11:40:14
 1                             MAINMOD  IDNT     1,1               *MAIN TEST PROGRAM
 2                             *                          *THIS PROGRAM TESTS THE FEASIBILITY
 3                             *                          *OF CALLING ASSEMBLER ROUTINE
 4                             *                          *PRNTLIN2 FROM A PASCAL PROGRAM
 5
 6                                      XDEF     START          *PROGRAM ENTRY POINT
 7                                      XREF     PRNTLIN2       *ROUTINE TO BE TESTED
 8
 9              00000001                SECTION  1
10 1 00000000 00000080                  DS.B     128            *RESERVE STACK SPACE
11 1 00000080 00000000     STACK        DS.W     0              *
12 1 00000080 00          IOSBLK        DC.B     0              *PARAMETER BLOCK START
13 1 00000081 02                        DC.B     2
14 1 00000082 0008                      DC.W     8
15 1 00000084 00                        DC.B     0
16 1 00000085 06                        DC.B     6
17 1 00000086 0000                      DC.W     0
18 1 00000088 00000000                  DC.L     0
19 1 0000008C 0000009C                  DC.L     BEGMSG
20 1 00000090 000000EB                  DC.L     ENDMSG
21 1 00000094 00000000                  DC.L     0
22 1 00000098 00000000                  DC.L     0              *PARAMETER BLOCK END
23 1 0000009C 544849532041 BEGMSG       DC.B     'THIS ASSEMBLER PROGRAM PRINTS LINE ONE'
24 1 000000C2 0D0A                      DC.W     $0D0A
25 1 000000C4 5448454E2043              DC.B     'THEN CALLS PRNTLIN2 TO PRINT LINE TWO.'
26 1 000000EA 0D0A                      DC.W     $0D0A
27 1          000000EB     ENDMSG       EQU      *-1
28
29            00000000                  SECTION  0
30 0 00000000 4FF900000080 START        LEA.L    STACK,A7       *INITIALIZE USER STACK PTR
31 0 00000006 41F900000080              LEA.L    IOSBLK,A0      *LOAD IOS PARAM BLK ADDRESS
32 0 0000000C 4E42         FRSTLIN      TRAP     #2             *EXECUTE
33 0 0000000E 6100FFF0     NEXTLIN      BSR      PRNTLIN2       *CALL EXTERNAL MODULE
34 0 00000012 91C8                      SUB.L    A0,A0          *
35 0 00000014 700F                      MOVEQ    #15,D0         *TERMINATE TASK
36 0 00000016 4E41                      TRAP     #1             *
37 0          00000000                  END      START

**** TOTAL ERRORS       0--
**** TOTAL WARNINGS     0--
SYMBOL TABLE LISTING

SYMBOL NAME       SECT      VALUE       CROSS REF (LINE NUMBERS)

BEGMSG             1       0000009C     -23      19
ENDMSG             1       000000EB     -27      20
FRSTLIN            0       0000000C     -32
IOSBLK             1       00000080     -12      31
NEXTLIN            0       0000000E     -33
PRNTLIN2    XREF   *       00000000      -7      33
STACK              1       00000080     -11      30
START       XDEF   0       00000000     -30      -6       37
```

FIGURE 2-3. Listing from Assembly of TESTPROG.SA

```
CALLMOD  IDNT     0,0            *SUBROUTINE FOR PASCAL CALL
         XDEF     PRNTLIN2
         SECTION  9              *PASCAL REQUIRES ASSEMBLY INTO 9
PRNTLIN2 MOVE.L   (A7)+,A4
         LEA      IOSBLK,A0      *LOAD IOS PARAMETER BLOCK
         TRAP     #2             *EXECUTE TRAP #2
         JMP      (A4)
IOSBLK   DC.B     0              *PARAMETER BLOCK START
         DC.B     2
         DC.W     8
         DC.B     0
         DC.B     6
         DC.W     0
         DC.L     0
         DC.L     MSGBEG
         DC.L     MSGEND
         DC.L     0
         DC.L     0              *PARAMETER BLOCK END
MSGBEG   DC.B     'THIS ASSEMBLER ROUTINE '
         DC.W     $0D0A
         DC.B     'PRINTS LINE TWO.'
         DC.W     $0D0A
MSGEND   EQU      *-1
         END
```

FIGURE 2-4. Assembler Source Routine ASMRTN.SA

```
MOTOROLA M68000 ASM VERSION x.xx FIX : 212.      .ASMRTN  .SA 05/23/83 11:51:21


  1                           CALLMOD  IDNT     0,0              *SUBROUTINE FOR PASCAL CALL
  2                                    XDEF     PRNTLIN2
  3            00000009                SECTION  9                *PASCAL REQUIRES ASSEMBLY INTO 9
  4 9 00000000 285F           PRNTLIN2 MOVE.L   (A7)+,A4
  5 9 00000002 41F90000000C            LEA      IOSBLK,A0        *LOAD IOS PARAMETER BLOCK
  6 9 00000008 4E42                    TRAP     #2               *EXECUTE TRAP #2
  7 9 0000000A 4ED4                    JMP      (A4)
  8 9 0000000C 00             IOSBLK   DC.B     0                *PARAMETER BLOCK START
  9 9 0000000D 02                      DC.B     2
 10 9 0000000E 0008                    DC.W     8
 11 9 00000010 00                      DC.B     0
 12 9 00000011 06                      DC.B     6
 13 9 00000012 0000                    DC.W     0
 14 9 00000014 00000000                DC.L     0
 15 9 00000018 00000028                DC.L     MSGBEG
 16 9 0000001C 00000053                DC.L     MSGEND
 17 9 00000020 00000000                DC.L     0
 18 9 00000024 00000000                DC.L     0                *PARAMETER BLOCK END
 19 9 00000028 544849532041 MSGBEG     DC.B     'THIS ASSEMBLER ROUTINE '
 20 9 00000040 0D0A                    DC.W     $0D0A
 21 9 00000042 5052494E5453            DC.B     'PRINTS LINE TWO.'
 22 9 00000052 0D0A                    DC.W     $0D0A
 23 9          00000053     MSGEND     EQU      *-1
 24                                    END

****** TOTAL ERRORS      0--
****** TOTAL WARNINGS    0--

  SYMBOL TABLE LISTING

  SYMBOL NAME       SECT     VALUE      CROSS-REF (LINE NUMBERS)

  IOSBLK              9     0000000C    -8       5
  MSGBEG              9     00000028    -19      15
  MSGEND              9     00000053    -23      16
  PRNTLIN2    XDEF    9     00000000    -4       -2
```

FIGURE 2-5. Listing from Assembly of ASMRTN.SA

PASCAL

2.3.3.2 <u>PASCAL - Transforming a Pascal Source Program into a Relocatable Module.</u> The output of any of the high-level language compilers available for VERSAdos systems can be a relocatable object code file of the same format as a relocatable object file produced by the assembler. This permits modules to be independently developed by programmers working in various source languages. Libraries can then be created from the debugged modules for use at linkage time, as shown in the software development process diagram.

Although ASM and LINK are supplied with VERSAdos, the Pascal compiler is an optional product which must be purchased separately.

Figure 2-6 is an example Pascal source program; Figure 2-7 is the listing obtained from executing the command lines:

```
a. PASCAL PASCPROG      (first compiler pass)
b. PASCAL2 PASCPROG     (final compiler pass)
```

The optional command POPTIM PASCPROG might have been given after the first compiler pass to generate the most efficient code in the final linked program.

```
PROGRAM PASCPROG (INPUT,OUTPUT);
PROCEDURE PRNTLIN2;  FORWARD;
BEGIN;
WRITELN('THIS PASCAL PROGRAM PRINTS LINE ONE');
WRITELN('THEN CALLS PRNTLIN2 TO PRINT LINE TWO.');
PRNTLIN2;
END.
```

FIGURE 2-6. Pascal Source Program PASCPROG.SA

```
Line   Loc Lev BE Motorola Pascal   x.xx  PASCPROG.SA 08/23/83 14:37:09

   1(   -16) 0)-- PROGRAM PASCPROG (INPUT,OUTPUT);
   2(     0) 1)-- PROCEDURE PRNTLIN2;  FORWARD;
**** PRNTLIN2 Assumed external
   3      1  0)A- BEGIN;
   4      2  0)-- WRITELN('THIS PASCAL PROGRAM PRINTS LINE ONE');
   5      3  0)-- WRITELN('THEN CALLS PRNTLIN2 TO PRINT LINE TWO.');
   6      4  0)-- PRNTLIN2;
   7         0)-A END.

       **** No Error(s) and No Warning(s) detected

       **** 7 Lines 1 Procedures

       **** 73 Pcode instructions
```

FIGURE 2-7. Listing from Compilation of PASCPROG.SA

LINK

2.3.3.3 LINK - Binding Relocatable Modules to Memory. After the relocatable modules required for a function have been wri ten, it is necessary to collect them together into a single relocatable module. All references between modules must be resolved and logical addresses assigned. This is the task of the linkage editor.

Relocatable modules for linking can be drawn from those created by assembly or compilation, or they can be drawn from a library. Libraries of relocatable modules are built through use of the Library utility (LIB). Relocatable modules can also be created by the linkage editor. These are usually large modules built from several smaller modules.

Figure 2-8 is a listing that was obtained by linking the relocatable module TESTPROG.RO (derived from the assembler source program TESTPROG.SA) with the relocatable module ASMRTN.RO (derived from the assembler source subroutine ASMRTN.SA). The listing shows that a load module PRINT2.LO, comprising two memory management unit (MMU) segments, was created. In its default mode (no user commands are issued following invocation), the linkage editor places sections having numbers 0 through 7 in the first segment, SEG0. Other numbered sections, if any, are placed in the second segment, SEG1. In this case, the called module CALLMOD was assembled into section 9, as the Pascal compiler requires, so that the same module can be called from both an assembler and, in the next example, a Pascal program. Consequently, the linkage editor placed section 9 in SEG1.

Figure 2-9 is a listing that was obtained by linking the relocatable module ASMRTN.RO (from the previous example) with the relocatable module PASCPROG.RO (derived from the Pascal source program PASCPROG.SA). Here the load module PRNTLINS.LO also comprises two segments. Now, however, since no sections numbered 7 or below exist, the first segment is SEG1, which the linkage editor identifies as a read-only segment (as it did in the example of Figure 2-8). Note that section 8 contains the Pascal runtime package. Section 8 is not written to and was therefore included in the read-only segment SEG1. Section 15, however, contains the Pascal stack area and was consequently placed in a read/write segment, SEG2.

Figure 2-10 shows the display obtained when PRNTLINS.LO is executed. This simple program demonstrates an assembler routine being called from a Pascal program. A Pascal routine or program could be called from an assembler program; however, the assembler program would have to supply the necessary portions of the Pascal runtime package and initialization procedures.

```
Motorola M68000 Linkage Editor Version x.xx   05/24/83  12:57:21          Page 1

COMMAND LINE

LINK TESTPROG/ASMRTN,PRINT2,PRINT2;HIMDS

Options in Effect:  -A,-B,D,H,I,-L,M,O,P,-Q,-R,S,-U,-W,-X

User Commands: None

Object Module Header Information:

Module      Ver Rev Language Date     Time     Creation File Name

MAINMOD       1   1 Assembly 02/23/82 13:28:47 FIX:212..TESTPROG.SA
            *MAIN TEST PROGRAM

CALLMOD       0   0 Assembly 02/23/82 13:29:12 FIX:212..ASMRTN.SA
            *SUBROUTINE FOR PASCAL CALL
Load Map:

Segment SEG0: 00000000 000001FF 0,1,2,3,4,5,6,7
Module      S   T  Start     End       Externally Defined Symbols

MAINMOD     0      00000000  00000017  START      00000000
MAINMOD     1      00000018  00000103

Segment SEG1(R): 00000200 000002FF 8,9,10,11,12,13,14
Module      S   T  Start     End       Externally Defined Symbols

CALLMOD     9      00000200  00000253  PRNTLIN2   00000200

Unresolved References: None
Multiply Defined Symbols: None

Lengths (in bytes):

     Segment     Hex         Decimal

      SEG0    00000200           512
      SEG1    00000100           256
Total Length  00000300           768

   No Errors

   No Warnings

   Load module has been created
```

FIGURE 2-8. Listing Obtained by Linking TESTPROG.RO with ASMRTN.RO

```
Motorola M68000 Linkage Editor Version x.xx  05/24/83  15:14:48          Page 1

Command Line:

LINK PASCPROG/ASMRTN,PRNTLINS,PRNTLINS;IMD

Options in Effect:  -A,-B,D,-H,I,-L,M,O,P,-Q,-R,-S,-U,-W,-X

User Commands: None

Load Map:

Segment SEG1(R): 00000000 000010FF 8,9,10,11,12,13,14
Module      S   T  Start     End       Externally Defined Symbols

INIT        8      00000000  00000379  .PLJSR     00000372
TRAPS       8      0000037A  00000459  .PADDRER   00000442  .PVBUSER   0000042A
                                       .PVCHKI    00000412  .PVTRAPD   00000412
                                       .PVTRAPE   0000037A  .PVTRAPV   000003FA
                                       .PVZDIV    000003E2
OPTION      8      0000045A  0000053F  .POPTION   0000045A
CLSCOD      8      00000540  000005BF  .PCLSCOD   00000540
ALSTS       8      000005C0  000005E5  .PALSTS    000005C0
CLO         8      000005E6  00000601  .PCLO      000005E6
IFD         8      00000602  000007F3  .PIFD      00000602
RST         8      000007F4  000008A5  .PRST      000007F4
RWT         8      000008A6  00000925  .PRWT      000008A6
ACCPER      8      00000926  00000947  .PACCPER   00000926
CALCLU      8      00000948  00000979  .PCALCLU   00000948
EDTFIL      8      0000097A  00000CFB  .PEDTFIL   0000097A
PRGBUF      8      00000CFC  00000D15  .PPRGBUF   00000CFC
STDFLT      8      00000D16  00000D73  .PSTDFLT   00000D16
DFLT        8      00000D74  00000DCF  .PDFLT     00000D74
WLN         8      00000DD0  00000DDF  .PWLN      00000DD0
WRSWRV      8      00000DE0  00000E29  .PWRS      00000DEA  .PWRV      00000DE0
WRTBUF      8      00000E2A  00000E8D  .PWRTBUF   00000E2A
LBLKS       8      00000E8E  00000E9F  .PLBLKS    00000E8E
IWPTR       8      00000EA0  00000EB3  .PIWPTR    00000EA0
ASGNF       8      00000EB4  00000ED7  .PASGNF    00000EB4
BUFSZ       8      00000ED8  00000EE7  .PBUFSZ    00000ED8
CLOSE       8      00000EE8  00000F19  .PCLOSE    00000EE8  .PCLOSPL    00000F08
CFLDAD      8      00000F1A  00000F41  .PCFLDAD   00000F1A
FLSCN       8      00000F42  00000F6B  .PFLSCN    00000F42
LDC         8      00000F6C  00000F95  .PLDCS     00000F6C  .PLDCV      00000F70
FINIT       8   C  00000F96  00000F97
PASCPROG    9      00000F98  00001057  .PMAIN     00000F98
CALLMOD     9      00001058  000010AB  PRNTLIN2   00001058

Segment SEG2: 00001100 000024FF 15
Module      S   T  Start     End       Externally Defined Symbols
```

FIGURE 2-9. Listing Obtained by Linking PASCPROG.RO with ASMRTN.RO
(Sheet 1 of 2)

```
Motorola M68000 Linkage Editor Version x.xx   05/24/83                  Page 2

 PASCPROG    15     00001100  000024FF  .PZMAIN    000024FE

 Unresolved References: None

 Multiply Defined Symbols: None

 Lengths (in bytes):

      Segment     Hex          Decimal

       SEG1    00001100           4352
       SEG2    00001400           5120
 Total Length  00002500           9472

    No Errors
     2 Warnings

       ** Warning 709 - Unable to include in debug file: FIX:0212.&.PASCPROG.RS
       ** Warning 709 - Unable to include in debug file: FIX:0000.&.PASCALIB.RS


 Load module has been created.
```

NOTE

Pascal symbols (i.e., variables, procedure or function names) can not be accessed by SYMbug; hence, the linkage editor supplies a warning when the D option is specified on the LINK command line.

FIGURE 2-9. Listing Obtained by Linking PASCPROG.RO with ASMRTN.RO (Sheet 2 of 2)

```
THIS PASCAL PROGRAM PRINTS LINE ONE
THEN CALLS PRNTLIN2 TO PRINT LINE TWO.
THIS ASSEMBLER ROUTINE PRINTS LINE TWO.
```

FIGURE 2-10. Display Resulting from Execution of PRNTLINS.LO

DEBUGGING A LOAD MODULE

2.3.4 Debugging a Load Module

A load module often requires debugging to overcome deficiencies which come to light when the program runs in an actual application. Supplied with VERSAdos are two debug monitor programs -- DEbug and SYMbug. In addition to these, a firmware-resident debug monitor program is supplied with each system as follows: EXORmacs systems come with MACSbug, VMC 68/2 systems with VMCbug, and VME/10 systems with TENbug. VMEbug and VERSAbug are available for MVME110-based and VM01/VM02-based systems, respectively, in ROM or as source and object code on disk.

DEbug - The monitor DEbug is a multiuser, multitasking program running under VERSAdos that requires references to the actual memory locations assigned by the linkage editor (for EXORmacs and the VME/10, the absolute locations of module segments are determined by the Memory Management Unit (MMU)).

SYMbug - The SYMbug program is also a multiuser, multitasking program running under VERSAdos, but it permits symbolic references to memory locations. This powerful capability relieves the user from the difficult chore of calculating offsets from a current linkage editor load map, such as that in Figure 2-8.

To utilize the symbolic referencing capability of SYMbug, a relocatable symbol file (.RS extension) is created during assembly by specifying the D option. The .RS file is then changed into a debug file (.DB extension) during linking by specifying the D option. This debug file is in optimized form to increase the symbolic referencing speed of SYMbug.

At present, the Pascal compiler does not provide the option of creating a relocatable symbol (.RS) file for input to the linkage editor. Therefore, the symbolic referencing capabilities of SYMbug cannot be used to access a point in a compiled module represented by a label in the source file. However, provided the D option was set during assembly and linkage, symbolic referencing can be used to access symbolic locations within <u>assembled</u> modules of a load module. Access to relative offsets within compiled modules is also provided.

Referencing the previous assembly and linkage examples, if SYMbug were now called with the command line

```
SYMBUG PRNTLINS
```

the symbols within ASMRTN.RO (see Figure 2-3) could be referenced by name. Refer to the M68000 Family SYMbug/DEbug Monitors Reference Manual, M68KSYMBUG, for more information on SYMbug's symbolic referencing and other capabilities.

MACSbug - The monitor MACSbug does not run under VERSAdos, but runs only in the single-user mode and requires references to absolute memory locations. MACSbug is used for debugging a new or inoperative operating system.

VMCbug - The VMC 68/2 monitor, VMCbug, like MACSbug for EXORmacs, does not run under VERSAdos. It is used for debugging a new or inoperative operating system. It runs in single-user mode and requires references to absolute memory locations.

OFTEN-USED UTILITIES

INIT

TENbug	- TENbug is a firmware-resident debug monitor for the VME/10 system which, like MACSbug and VMCbug, does not run under VERSAdos. It runs in single-user mode only and requires references to absolute memory loctions. It also includes a self-test capability which verifies the integrity of the VME/10.
VMEbug	- VMEbug is a firmware-resident debug monitor for MVME110-based systems which, like the monitors described above, does not run under VERSAdos. It is also available on disk as a package of source and object modules that will run as a stand-alone monitor for specific aplications. It includes an up/downline load command for loading programs from a host, an assembler/disassembler for use in scanning and patching the code being debugged, and a self-test capability which verifies the integrity of the MVME110.
VERSAbug	- VERSAbug is a minimum function firmware-resident debugging monitor for VM01/VM02-based systems. It does not run under VERSAdos but, like VMEbug, VERSAbug is also available in source and object code on disk so that it can be run as a stand-alone monitor.

Documentation for these monitors is listed in paragraph 2.1.1 of this manual.

2.4 OFTEN-USED UTILITIES

This section offers a shorthand look at utilities often used during program development. A brief description is provided and an example shown in which default values are taken for some command line fields. Full descriptions of all utilities are provided in Chapter 4. The following examples assume the session was initiated by a user other than user 0.

2.4.1 INIT - Initializing a Disk

A hard disk or a diskette must be initialized by means of the INIT utility. Hard disks and double-sided diskettes must also be formatted, using INIT, to provide magnetic sector locating marks for use by the disk drive interface electronics. Single-sided diskettes usually do not require formatting.

When called, INIT enters the interactive mode and requests approval to initialize and format. INIT requests a volume ID, a user number, and a 20-character description which will be given to the disk during execution. The latter capability permits INIT to be used to change ownership of a disk. INIT also writes the configuration information on the disk.

If the initializer is user 0, other information is requested according to whether the disk to be initialized is a hard disk or a floppy diskette.

The following command lines will start initialization of a floppy diskette and a hard disk, respectively:

```
    INIT #FD01 (diskette)        INIT #HD01 (hard disk)
```

In the above, #FD01 and #HD01 are the device names of the drives on which the disks are mounted.

DIR

LIST

COPY

2.4.2 DIR - Directory Display

The DIR utility offers a quick means of displaying the volume ID, user number, catalog name, file name, and extension fields for each user-owned file which is stored on the specified volume. The utility uses the default value for each empty file descriptor field on the command line. Thus, specifying DIR alone results in the least selectivity. Increasing selectivity comes with each field entry. Entries in all fields would return a single file descriptor.

When volume ID, catalog name, and user number are not specified on the command line, the utility assumes the user default volume ID, user number, and catalog name, making it possible to obtain directory information with abbreviated entries on the command line. Thus, the command lines

```
    DIR FILNAM.*     and      DIR *.SA
```

will show all extensions of FILNAM and all files of .SA extension, respectively, on the user default volume and within the default catalog.

The DIR utility can also be used to display the volume name, user number, and description of a disk or diskette in a drive.

2.4.3 LIST - ASCII File Display

The LIST utility is used to view the contents of an ASCII file. Scroll of the display can be stopped by simultaneous use of the CTRL and W keys, and can be restarted by pressing any key. Since the function must always find a single file, values are required for all five file descriptor fields. If values are not specified for any of the volume ID, user number, or catalog name fields, user default values are assumed by the utility. If the extension field is omitted, .SA is assumed. Output of the LIST utility can be formatted and can be directed to the system printer; otherwise, the contents of the file will be displayed on the terminal.

Execution of the command line

```
    LIST FILENAM
```

will cause display of the specified file (extension .SA is assumed) if the file belongs to the default user and it exists within the user default catalog on the user default volume.

The DUMP utility must be used to display files of object code such as .RO and .LO files since these are not ASCII files.

2.4.4 COPY - Copying Files

The COPY utility permits files of any extension to be reproduced on the same or another volume. Any of the five file descriptor fields can be changed by specifying the desired form of that field in the command line output field. The COPY utility also allows the asterisk ("wildcard") to be specified to represent a field(s) or a character(s) in the input field and/or the output field. Variations in COPY execution are provided by options which allow verification of the requested copy or permit one file to be appended to another.

```
    COPY OLDNAM.SA,NEWNAM.SA;B
```

DEL

BACKUP

If OLDNAM.SA in the above command line belongs to the default user, execution of the command will cause the file to be copied into the user default catalog and volume and given the same user number but the name and extension NEWNAM.SA. The B option causes comparison of the copied file with the original, and display of differences, if any.

2.4.5 DEL - Deleting Files

The deleting of unwanted files is handled by the DEL utility. DEL removes file entries from a disk directory and frees any space allocated to the corresponding files. Deletion of groups of files having common file descriptor attributes is accomplished by using the asterisk ("wildcard") in the catalog name, file name, or extension fields. Group deletion can proceed file-by-file in response to a prompt, or automatically by specifying the Y option.

```
    DEL CAT1.*.LO;Y
```

The above command will automatically delete all files of .LO extension belonging to the default user, within catalog CAT1, and on the user default volume.

As a precaution, a DIR should be performed on a group of files before a "wildcard" delete is requested.

2.4.6 BACKUP - Creating Working Disks

The BACKUP utility is used to create working copies of system disks, as a periodic data preservation tool for system administration, and to create or modify disks of development programs. BACKUP permits files to be transferred between source and destination disks whether the disks are hard or floppies and of the same or different capacities. Information in the descriptors of transferred files can be changed, and disk Volume Information Block (VID) data can be altered. BACKUP also permits track-to-track data transfer and will perform verification and disk data comparison and reporting in the latter mode.

Unused intra-file and inter-file space can be regained on the destination disk, and selection of files from the source disk can range from the entire disk down to a single file. The selection process can proceed automatically or interactively through dialog with the utility.

To properly utilize the variation in the two operating modes and take advantage of the many file selection provisions of BACKUP, the user should familiarize himself with the Chapter 4 description of this utility.

The minimum command line is:

```
    BACKUP
```

Execution of this line selects the two lowest numbered disk drive device names in the system tables -- usually #HD00 and #HD01 for a hard disk system, #FD00 and #FD01 for a system with an FDC and no hard disks, or #FD04 and #FD05 for a system with a UDC and no hard disks. The default operating mode is entered (file transfer, A option if hard disks or track-by-track, U option if floppy disks). In a floppy disk system, the utility prompts the user for a new volume

ID and description, after which the source data is transferred track-by-track to the destination. In a hard disk system, the user is prompted to choose transfer of all or selected files. If "all files" is chosen, transfer proceeds automatically unless a duplicate name is encountered on the destination, in which case a prompt then asks for an OK to overwrite before proceeding. If "selected files" is chosen, the user may select categories of files to be copied.

CHAPTER 3 SESSION CONTROL

CHAPTER 3

VERSAdos SESSION CONTROL

3.1 INTRODUCTION

Session control commands are commands made directly to VERSAdos, rather than being loaded as separate programs. They modify the operation of VERSAdos or of a user command, and are made available to the user when the session control task is established at session initiation (logon). The following commands are included in the universal session control category.

Command characters to the left of the line are required; those to the right are allowed, but are not required. Lowercase letters are allowed. All commands must terminate with one or more spaces or a carriage return if no arguments are required for the command. Therefore, a space or carriage return can be entered in lieu of any allowable character shown on the right-hand side of the line. Entry of any other character in that position causes session control to assume a file name is being entered. For example:

```
ASSI<space>     => session control command
ASSI.           => file name (.LO extent assumed)
ASSIS           => file name (.LO extent assumed)
ASSIG           => session control command
```

Command		Description	
OFF		Terminate session.	
LOG OF	F	Terminate session.	IDENTICAL SESSION TERMINATION
LOGOF	F	Terminate session.	
BYE		Terminate session.	
BATC	H	Submit batch job.	
CANC	EL	Cancel selected or all batch jobs.	
ELIM	INATE	Cancel all batch jobs (privileged).	
QUER	Y	Request status of batch job.	
CHAI	N	Execute chain file.	
RETR	Y	Restart execution of aborted chain file at current record.	
PROC	EED	Restart execution of aborted chain file at next record.	
OPT	ION	Set chain conditional processing option.	
R?		Display contents of chain conditional processing pseudo registers.	
END		Terminate chain processing.	
LOAD		Call the loader.	
STAR	T	Begin execution of user task.	
CONT	INUE	Restart execution of user task.	
STOP		Stop execution of user task.	
TERM	INATE	Terminate execution of user task.	
BSTO	P	Stop all tasks on Break.	
BTER	M	Terminate all tasks on Break.	
USE		Enter file descriptor defaults.	
DEF	AULTS	Display default values.	
ARG	UMENTS	Enter/display new arguments.	
NOARG	UMENTS	Clear argument list.	
DATE		Display time and date or (privileged) change date.	
TIME		Display time and date or (privileged) change time.	
^		Place the session control task in the dormant state.	

SESSION CONTROL COMMANDS

BATCH MODE PROCESSING

PASS|WORD	Specify or change user password.
SWORD|	Specify or change system security word (privileged).
SECURE|	Specify level of system security (privileged).

In the preceding commands, the character "^" corresponds to the ASCII hexadecimal code 5E, and "?" corresponds to ASCII hexadecimal code 3F. "Privileged" means that only the system administrator, user 0, may use this command.

3.2 SESSION CONTROL COMMANDS

3.2.1 Batch Mode Processing

The facility which allows a user to operate in the online or foreground mode while his job is being processed in the background mode is called batch mode processing. Although the batch job is processed concurrently with foreground, it operates under a lower priority of access to system resources. A structure comprised of several session control commands offers the means by which a user interfaces with the subsystem. This structure can be excluded from or included into the system at SYSGEN time. The commands are:

a. BATCH (submit)
b. QUERY (status)
c. CANCEL (job in queue or process)
d. ELIMINATE (all jobs in queue or process for specified user)

Argument substitution is in effect during batch processing.

Job Queueing

The batch facility allows multiple users to request execution of multiple jobs, thus placing the decision of which to do first on the operating system. A circular queue established in RAM is used to allow the dispatch of jobs in entry order.

Job queue length may be increased by a multiple of 32, chosen when the user initially generates his application operating system. As supplied, VERSAdos allows 25 batch jobs to be queued without regard to the number of online users. Because jobs are queued with reference to user number and remain queued when the user signs off, he can later initiate a new session and determine the status of and/or terminate any previously entered batch jobs through use of the QUERY and CANCEL functions of the subsystem. Where several users are sharing the same user number, the status of any job queued under that number is accessible. In addition, any queued job can be cancelled by any user with the same user number or by the system administrator (user 0).

Job Dispatching

As stated, jobs are dispatched in the order queued. The maximum number that can be processed concurrently is established by the user at system generation time (i.e., when the user initially implements his operating system) or dynamically with the DISPATCH utility.

Execution of the dispatch cycle takes place upon the queueing or termination of a batch job.

BATCH

The system administrator has the capability of putting a hold on the dispatching of any new jobs, and of releasing the hold.

Online/Batch Processing Differences

The basic differences between online and batch processing are charted as follows:

Circumstance	Online Processing Mode	Batch Processing Mode
Command input method.	Manual via terminal or command file via CHAIN command.	Command file via BATCH command.
Insufficient memory space to load program.	1. Message displayed. 2. PROMPT (=) returned. (User may issue QUERY and CANCEL commands.)	1. Message displayed. 2. A 16-second delay begun*. 3. Time-out initiates new attempt to load (256 attempts).

*As supplied. Period may be changed by user at system generation time.

3.2.1.1 Batch Command Syntax (BATCH) BATC

```
BATC[H] <cdf>,[<ldf>][ <arg 1>,...<arg n>]
```

where:

cdf	Is the descriptor of a command (chain) file (minimum of file name field required; if an extension is not specified, .CF will be supplied). A device is not allowed. <cdf> is assigned as public read only.
ldf	Is a file descriptor or device name. The file name specified need not exist but will be allocated and assigned as <ldf>.SA. <ldf> is assigned as write. The system printer (#PR), #NULL (no message printed), or #CNxx (where xx is another terminal) may be specified. If the <ldf> field is omitted, #PR is assumed.
arg 1,...arg n	Is the argument list for substitution in the chain file. May be preceded by one or more spaces.

BATCH Command Example

```
=BATCH DEMO.CF,#PR
 ssss: QUEUED (or BATCH Q FULL)
=
```

where ssss represents the session number.

QUERY

CANCEL

3.2.1.2 Batch Job Status (QUERY). After being input to the system through use of the BATCH command, a job can exist in only one of three states: waiting for execution (queued); executing (running); or completed (done), which includes terminated, aborted, or cancelled. The status of any job can be determined through use of the QUERY command -- in particular, the status of the last job submitted, a specific job (by session number), or all jobs submitted by one user (by number of the current session user). The system administrator, user number = 0, can request the status of the total batch queue. The status of a job is retained until the circular queue is filled and it, as the oldest status, is overwritten when an additional job is submitted. Following are the query command forms.

QUERY Command Forms

QUER[Y]	Obtains the status of the last job submitted. If no batch jobs have been submitted during the current session, the command defaults to all batch jobs for the user number of the current session.
QUER[Y] <ssss>	Obtains the status of the job with session number <ssss>.
QUER[Y] ALL	Obtains the status of all jobs submitted by the user number of the current session. If the user number of the current session is 0, the status of all jobs is returned.

3.2.1.3 Discontinuation of Batch Job (CANCEL). After submission to the system, a job can be cancelled (unless already completed) through use of the CANCEL command. Therefore, it is the usual practice to obtain the status of the job or jobs by using the QUERY command prior to the termination of a session. In particular, the user can cancel the last job submitted during the current session, a specific job (by session number), or all jobs submitted by one user (by current session user number). The system administrator, user number = 0, can cancel all batch jobs; therefore, he should cancel with care. The status of the cancelled job is returned.

CANCEL Command Forms

CANC[EL]	Cancels the job last submitted.
CANC[EL] <ssss>	Cancels the job with session number <ssss>.
CANC[EL] ALL	Cancels all jobs submitted by the user number of the current session. If the user number of the current session is 0, all jobs are cancelled.

ELIMINATE

3.2.1.4 <u>Cancel Batch Jobs (ELIMINATE).</u> The ELIMINATE command is used by the system administrator (logon user 0 only) to cancel all the jobs of a particular user for the current session.

<u>ELIMINATE Command Form</u>

ELIM[INATE] <user no.>	Cancels all jobs belonging to user number specified.

3.2.1.5 <u>Messages - Batch Mode Commands.</u> Use of any of the four batch mode commands BATCH, QUERY, CANCEL, or ELIMINATE can result in display of one of the following messages:

```
    ssss:  QUEUED
    ssss:  RUNNING
    ssss:  DONE STATUS=$xxxx:<message>
```

In these displays, ssss is the session number and xxxx is a description of the status. For example, $xxxx:<message> might be:

EXAMPLES:

```
$C001:  NORMAL TERMINATION FROM RUNNING
$C006:  CANCELLED WHILE RUNNING
$A006:  CANCELLED WAITING IN QUEUE
$C0C6:  CANCELLED DUE TO BREAK
$C010:  ABORTED DUE TO BUS ERROR
$C011:  TERMINATED DUE TO ERROR (ADDRESS)
```

<u>Example - Typical Session with Both Online and Batch Job Processing</u>

```
(ALL CAPS) (key illuminated)
(BREAK)    (session requested)
VERSAdos VERSION: rr.vv x/xx/xx
ENTER USER NO. =DEMO:1
x:xx:xx x/xx/xx START SESSION 0002 USER=1
=BATC GCD,#PR                (GCD is the name of a chain file.  Default
                              extension .CF is assumed. Error messages will be
                              printed.)
 0003: QUEUED
=BATC QUEENS,#NULL           (#NULL causes no messages to be printed.)
 0004: QUEUED
=E OLDFILE,NEWFILE;S         (Online operations may be performed as desired
                              while waiting for execution of batch jobs)
```

DISPATCH

Example - Typical Session with Both Online and Batch Job Processing (cont'd)

```
=QUER 0003
 0003: DONE STATUS $C001: NORMAL TERMINATION FROM RUNNING
                                      (Code $C001 indicates a normal termination
                                       following successful completion of the job.)

=QUER 0004
 0004: QUEUED
=CANC 0004
 0004: DONE STATUS $A006: CANCELLED WAITING IN QUEUE
=                                     (Code $A006 indicates that the job was cancelled
                                       before execution.)
```

NOTE

In the QUERY command, leading zeros do not have to be entered.

DISPATCH

3.2.1.6 Hold or Release New Batch Jobs (DISPATCH). DISPATCH is not a session control command, but a utility program. It is included in this section due to its related function. The DISPATCH utility allows the system administrator (only) to change dynamically the number of batch jobs that are able to execute. With this command, a hold can be put on the dispatching of any new jobs, and it can then be used to resume dispatching. (Jobs already running are not affected by this command.)

DISPATCH Command Form

DISPATCH [<number of jobs>]

where <number of jobs> is the number of jobs concurrently executing/waiting (maximum = 8, a greater than practical limit). Specifying a 0 as <number of jobs> prevents any jobs in the queue from executing until restarted with another DISPATCH command. Specifying DISPATCH without the argument <number of jobs> sets its value to the <number of jobs> established when the system was SYSGENed.

CHAIN MODE PROCESSING (CHAIN)

3.2.2 Chain Mode Processing (CHAIN)

Chain mode processing allows predefined procedures to be automatically executed. A procedure consists of any sequence of VERSAdos command lines and subcommands or task entries as required by the command and which are stored in a file. Such a procedure or chain file is normally given the file extension .CF. The CHAIN mode commands can be excluded from or included into the system at SYSGEN time.

When a chain file is invoked, the connection between logical unit 5 and the terminal is closed. The chain file is then assigned to logical unit 5. Subsequent commands are read from logical unit 5 (chain file). For a brief description of logical unit usage, see the description of the ASSIGN session control command in this manual.

Leading spaces are permitted and are ignored for command decode purposes. This allows indenting to indicate levels of nested conditional commands. Only those commands preceded by an "=" as the first character following the permissible leading spaces are treated as commands. Those without an "=" are assumed to be subcommands following a command, and are ignored.

Argument (parameter) substitution is in effect during chain file processing. A new argument list may be entered at the time the chain processing is initiated, or an existing argument list will be used. Refer to paragraph 3.2.3 for further discussion of argument substitution.

Three pseudo registers are provided to test and manipulate data during chain file processing, thus providing conditional processing capability. Each of the pseudo registers is a 16-bit data item in the data segment of the session control task. These pseudo registers are updated or changed only during chain file processing. Normally, all three are cleared to zero when chain file processing is initiated; optionally, however, they may be left with the previous contents.

The RA, or abort code pseudo register, receives the lower half of the A0 register when a user task terminates or aborts. When a user task terminates normally, the A0 register should be zero; hence, the RA pseudo register will normally be zero if the user task terminates. When a user task aborts, the abort code is in the lower half of the A0 register and will be transferred to the RA pseudo register. The RA register may be used as a general purpose register for data manipulation between user task invocations.

The RD, or diagnostic pseudo register, is used to retain status of the progress of the chain file processing. When a user task terminates or aborts, the terminate or abort TRAP #1 directive number is in the lower half of the D0 register. The upper half of the D0 register is used to convey diagnostic information which is used in chain file processing to determine whether to proceed. The upper half of the D0 register is inclusively OR'ed with the RD pseudo register. If the RD bit 15 is set, a flag is set to cause aborting the chain file processing. Current implementation of the Pascal system (compiler, assembler, linkage editor) puts zero (no errors), $C000+no. of errors, or $1000+no. of warnings in the upper half of D0 on termination or abort; hence, errors will cause an abort of chain file processing. When a user task is loaded and started during chain file processing, bits 15 and 0-11 of the RD register are reset to zero; thus, bits 14-12 reflect the retained severity of faults during the chain file processing.

The RX pseudo register is a general purpose register completely at the user's discretion. "FOR" loops normally use the RX pseudo register for maintaining the loop count but, optionally, an internal counter may be used instead of the RX pseudo register. A powerful use of the RX pseudo register is to use the indirect form of argument addressing, \(RX), especially in "FOR" loops.

Several of the OPTION utility letters (paragraph 3.2.11) are used in chain file processing, as follows:

A	Abort chain file processing if a user task terminates or aborts with the lower half of A0 <> 0.
L	List chain file commands which are executed (not conditionally skipped).
M	List chain file commands conditionally skipped (with a "\" substituted for the "=").
N	No echo of chain file commands (overrides L & M).
O	Override errors; i.e., continue chain file processing under the following error conditions:

Override?	Error condition
yes	"ER: xxxx" error messages
yes	User task terminates/aborts with D0 bit 31 set (RD bit 15)
yes	OPT A and lower A0 <> 0
no	=/ABT command
no	Chain file return nesting errors
no	Argument substitution table overflow
no	Command syntax errors

When chain file processing is interrupted due to one of the above errors, the chain file remains assigned to logical unit 5. Interactive dialog with the terminal is continued through logical unit 0 (privileged connection for session management). The CHAIN ABORTED notice is output, as well as the contents of the pseudo registers. Interactive dialog prompt in the chain aborted mode is "(CHAIN)=". If the cause of the error can be remedied, chain file processing may be resumed with a PROCEED or RETRY command. Other possible actions include:

a. [@]END or CLOSE 5	(Terminal input to terminate chain mode.)
b. [@]CHAIN xxx or [@]xxx.CF	(Start new or same processing.)
c. <session control commands>	
d. <utility commands>	(Except an edit to the chain file, because of access permission restrictions.)

NOTE

RETRY and PROCEED are not applicable if chain processing was aborted as the result of opening the door of the diskette drive containing the chain file. In this case, the error message "$100000EB FROM USR ** DEVICE STATUS CHANGED" appears.

A chain file may be terminated in the following ways:

a. =[@]END statement in the chain file.
b. [@]END or CLOS 5 terminal input to a "(CHAIN)=" prompt.
c. An "end of file" return when attempting to read a command.
d. A new chain file is invoked with "CHAIN xxx" or "xxx.CF".

3.2.2.1 Invoking Chain Mode Processing.

CHAIN Command Syntax

```
    [@]CHAI[N] <cdf>  [<arg1>[,...,<argn>]]

    or

    [@]<cdf>.CF  [<arg1>[,...,<argn>]]
```

where:

cdf	Is the file descriptor of the chain file. If the extension .CF is explicitly included, the command mnemonic (CHAIN or CHAI) is superfluous and may be omitted.
[@]	Inhibits the clearing of the three pseudo registers RA, RD, and RX to zero before initiating the chain file processing. Default is to zero the pseudo registers before initiating chain file processing.

<u>NOTES</u>:

a. An implied ARGUMENTS command may be included on the command line. One or more spaces must delimit the file descriptor and the argument list.

b. When the chain file specified cannot be assigned, an END command is automatically executed and the chain mode is terminated.

<u>CHAIN Command Example</u> (without argument substitution)

```
=CHAI DEMOCHN.CF
 =ASM DRIVER,,#PR
 =LINK DRIVER,MATHEX,#PR;IXHML=ARITH,A
 LISTM              (Subcommand)
 END                (Subcommand)
 =END
=
```

In this example, when execution of the command line is initiated, the system gets the ASCII record file DEMOCHN from the default volume. Commands in DEMOCHN are then sequentially executed. A command line in the file preceded by an equal sign (=) requires that a task be initiated to execute that line. Sub-command lines are executed by the task set up by the previous command.

CHAIN Command Example (with argument substitution)

```
=CHAI ARGDEMO.CF  GCD,#PR,#NULL,IXMH
=LIST ARGDEMO.CF
PAGE 1    LISTING OF ARGDEMO.CF

=LIST ARGDEMO.CF
=ARG
=PASCAL \1,,\2
=PASCAL2 \1,,\3
=LINK \1,,\2;\4
=END
END OF FILE

=ARG
\1:GCD
\2:#PR
\3:#NULL
\4:IXMH
=PASCAL GCD,,#PR
 Motorola Pascal Compiler Phase 1 Version  x.xx
 Copyrighted 1982 by Motorola, Inc.
 Source Lines  Intermediate Code   Errors     Warnings
     145             828              0           0

=PASCAL2 GCD,,#NULL
 M68000 Pascal Compiler Phase 2 Version x.xx
 Copyrighted 1982 by Motorola, Inc.
 Source Lines  Intermediate Code   Bytes Generated
     145             828                2026

 No Errors detected.
=LINK GCD,,#PR;IXMH
M68000 Linkage Editor Version x.xx
Copyrighted 1982 by Motorola, Inc.

=END
END CHAIN
=
```

3.2.2.2 Unconditional Chain Commands.

CHAIN Mode Command (END)

The END command is used to terminate chain mode processing. The END command may be included in the chain file, preceded by an equal sign, or entered from the terminal while the system is in the CHAIN ABORTED state.

```
    [@]END(CR)    (Terminal entry)

    =[@]END       (Within chain file)
```

The @ option causes the closing of all intermediate LUN's used for subroutine chain file calls, and for FOR loops. It may be used to abort chain file processing during execution of a nested chain file or inside a FOR loop.

NOTE

> If processing is aborted inside a FOR loop or a chain file subroutine call, the "(CHAIN ABORTED)=" prompt is issued. If END is input, only one level of nesting is terminated and several END inputs may be required to receive the END CHAIN message. If @END is input, all levels of nesting are terminated, and the END CHAIN message is output.

Display Pseudo Registers

The pseudo registers may be displayed with this command (and they are automatically displayed if chain file processing is aborted). The response to this command is output to LU=6.

```
    =/R?
```

Set Pseudo Register to a Specified Value

```
    =/RA|RD|RX = <value>
```

where <value> is an integer which may be expressed in 16 bits; is assumed to be a decimal string; and may be a hex string if preceded by "$".

EXAMPLES:

```
=/RA = 0        (Clear abort code register)
=/RX = \0       (Set RX equal to the current number of arguments)
=/RD = $00FF    (Set to a hex value)
```

Set/Reset Specified Bit in a Pseudo Register

```
    =/RA|RD|RX (<bit number>) = 1|0
```

where <bit number> must be enclosed in parentheses.

EXAMPLES:

```
=/RX(0) = 1              (Set bit 0 to a 1)
=/RX (\2) = 0            (Argument positional notation)
=/RX (/BITNO=) = 1       (Argument tag notation)
```

Operation on Pseudo Registers with Literal Values

```
    =/RA|RD|RX = RA|RD|RX <operator> <value>
```

where:

value	Is an integer which may be expressed in 16 bits; is assumed to be a decimal string; and may be a hex string if preceded by $.
operator	May be one of the following: ! = inclusive OR & = logical AND + = arithmetic add - = arithmetic subtract

EXAMPLES:

```
=/RX = RX & $FF00        (Mask low order byte)
=/RA = RX & $FF          (Transfer low order byte RX to RA)
=/RX = RX + 1            (Increment RX by 1)
=/RX = RX ! /BITMASK=    (Inclusive OR tag notation argument)
```

Chain File Comment

This command allows the user to document the execution of a chain file. The use of comments shows both the progress and the direction of conditional statements executed.

```
    =/*[<comment>]
```

EXAMPLES:

```
a.  =/*  EXECUTING FOR \0 ARGUMENTS
    =/*  ARGUMENT 1 = \1
    =/*  ASSEMBLING THE MODULE \(RX)

b.  =/IF RX(14) = 0
    =/*  NO ERRORS ASSEMBLING \(RX) MODULE
    =/ENDIF
    =/IF RX(12) = 1
    =/*  WARNINGS ASSEMBLING \(RX) MODULE
    =/ENDIF
```

Chain File Subroutine Call

```
    =/@ <filename>
```

The effect of this command is to treat a chain file as a subroutine; i.e., the named <filename> (.CF extension assumed) is executed and then control is returned to the original chain file at the next command. Nesting of chain file subroutines is subject to the restrictions of FOR loops.

The subroutine chain file is not initialized upon entry; i.e., the pseudo registers are not cleared. The present state of the conditional IF level is preserved. Upon return to the calling chain file, the IF level must be the same as when the call was made.

EXAMPLE:

```
=/@ ASMUTIL         (A chain file to assemble a program)
=/@ USMUTLNK        (Followed by a chain file to link edit the result)
=END
```

Note that if arguments are passed in a chain file subroutine, they override those currently in effect. Therefore, existing arguments should be reentered with those of the subroutine.

Chain File Pause Command

```
    =/& [<comment>]
```

The effect of this command is to provide a pause until input is received from the terminal. The command line is printed, an = prompt is printed at the left margin, and input from the keyboard is required to proceed.

NOTES:

a. In batch mode, the command line is printed as a comment, but input is not solicited.

b. A non-interactive command may be input if preceded by another = prompt. Interactive commands may produce unpredictable results since LUN=5 is assigned to the chain file. A command not preceded by an = prompt is ignored.

c. The chain file processing will proceed on a RETURN.

d. The pause for input is subject to input time-out values as determined by SYSGEN parameters (refer to the System Generation Facility User's Guide).

Abort Chain File Processing (ABT)

The Abort command (ABT) interrupts chain file processing to allow the user to enter commands from the terminal or terminate a batch session. The CHAIN ABORTED message, the pseudo registers, and (CHAIN) prompt are output. The OPT O does not override this command.

```
    =/ABT [<comment>]          (To continue chain file execution, the subcommand
                                PROC must then be entered.)
```

EXAMPLE:

```
=/ABT  ENTER CORRECT ARGUMENTS AND "PROC"
```

NOTE

RETRY is not applicable following this command.

3.2.2.3 FOR Loop Processing.

Command Syntax

```
=[@]/FOR a,b
       .
       .              (User tasks, utilities, other chain file commands, etc.)
       .
=/ENDFOR
```

The effect of this pair of commands is to process the commands between the /FOR and /ENDFOR beginning with the value "a". If the @ option is not used, the "a" value is put into the RX register. The @ option causes the current value to be maintained internally; thus, the RX register is available for other user use. If a<=b, the processing is initiated; otherwise, a "skip" is initiated (next command executed is that following ENDFOR). When the /ENDFOR is reached, the "a" value is compared to the "b" value. If a>=b, the processing continues following the /ENDFOR. Otherwise, "a" is incremented by 1 and the loop is executed again.

Nesting of FOR loops and chain subroutines are permitted to the extent that LUN's >=7 are SYSGENed into the operating system and, space permitting, in the conditional return stack. (The return stack and the argument substitution pointer table have a floating boundary; the number of levels of nesting possible decreases inversely proportional to the number of arguments). The following table shows possible combinations achievable:

MAXLU	No. Arguments	Nesting Returns	
-	79 (all null)	0	
8	67	2	(Standard VERSAdos as delivered)
10	55	4	
19	-	13	

If an ARGUMENTS command is processed when the nesting level does not permit the number of arguments to be entered, a "No. ARGUMENTS" error is output and the number of arguments allowable are entered. If the current number of arguments does not leave room for nesting, an attempt to nest will result in the error message RETURN STACK SPACE. If an insufficient number of logical units is SYSGENed, an attempt to nest another level results in the error message NESTING LEVEL. All of these errors cause the chain processing to abort in the fashion equivalent to a /ABT (which can not be overridden by option O).

EXAMPLE:

```
a.  =/FOR 1,\0            (For all arguments)
       .
       .
       .
    =/ENDFOR

b.  =/FOR 1,4             (Loop executed 4 times)
       .
       .
       .
    =/ENDFOR
```

3.2.2.4 <u>Conditional Chain File Commands</u>. The conditional chain file commands provide the capability to test pseudo registers, arguments, and data. Various portions of the chain file may be executed or skipped, based on the results of the test. The basic structure consists of:

```
    =/IF <condition>
       .
       .                 (Statements executed if condition is met or true)
       .
    =/ENDIF
```

or the alternative form:

```
    =/IF <condition>
       .
       .                 (Statements executed if condition met)
       .
    =/ELSE
       .
       .                 (Statements executed if condition not met)
       .
    =/ENDIF
```

Nesting of up to seven levels of IF's is permitted. Note that failure to provide an =/ENDIF for each level of =/IF will provide erroneous results. For debugging of complex chain files, it is suggested that OPT LM be set in order to verify the path taken.

EXAMPLE: (3 levels of IF's)

```
    =/IF RA = 0
         =/IF RD = 0
              =/IF RX = 0
                   =/* ALL PSEUDO REGISTERS = 0
              =/ELSE
                   =/* RA & RD = 0, RX <> 0
              =/ENDIF
         =/ELSE
              =/IF RX = 0
                   =/* RA & RX = 0, RD <> 0
              =/ELSE
                   =/* RA = 0, RD & RX <> 0
              =/ENDIF
         =/ENDIF
    =/ELSE
         =/IF RD = 0
              =/IF RX = 0
                   =/* RD & RX = 0, RA <> 0
              =/ELSE
                   =/* RA & RX <> 0, RD = 0
              =ENDIF
         =/ELSE
              =/IF RX = 0
                   =/* RA & RD <> 0, RX = 0
              =/ELSE
                   =/* ALL PSEUDO REGISTERS <> 0
              =/ENDIF
         =/ENDIF
    =/ENDIF
```

Conditional Testing/Execution with Literal Values

```
    =/IF RA|RD|RX <|<=|>|>=|=|<> <value>
```

EXAMPLE:

```
    =/IF RA <> 0              (RA not equal to zero)
         .
         .                    (Statements performed if condition met/true)
         .
    =/ENDIF                   (End of condition)
```

Alternative form:

```
    =/ RA <> 0
         .
         .                    (Statements executed if RA not zero)
         .
    =/ELSE
         .
         .                    (Statements performed if RA zero)
         .
    =/ENDIF
```

Conditional Testing/Execution Based on Pseudo Register Bits

```
    =/IF  RX (15) = 1         (Bit 15 of RX set)
           .
           .                  (Statements executed if condition met)
           .
    =/ENDIF

         or

    =/IF  RX (12) = 0         (Bit 12 of RX reset)
           .
           .                  (Statements executed if bit 12 reset)
           .
    =/ELSE
           .
           .                  (Statements executed if bit 12 set)
           .
    =/ENDIF                   (End of condition)
```

Conditional Testing/Execution Based on Expression Consisting of Logical Operators or Substitution Arguments

Conditional execution may be based on the combination of existing or null arguments using the logical operators ! (inclusive OR) and & (logical AND).

```
    =/IFC|IFS <expression>        /IFC = clear if argument is undefined (null),
                                  /IFS = set if argument is defined)
```

EXAMPLES:

```
    =/IFS  \1!\2                  (If either argument 1 or 2 is not null)
    =/IFC  \1&\3                  (If both arguments 1 and 3 are null)
    =/IFC  \INPUT= ! \OUTPUT=     (If either tag notation arguments null)
    =/IFS  \1!\2&\3               (3 required not null and either 1 or 2 must be
                                   present)
```

Execution if a Specified Argument Matches a Literal String

```
    =/IFEQ "<literal ASCII string>"<argument>
```

EXAMPLES:

```
    =/IFEQ "#PR"\3                (Argument position notation)
    =/IFEQ "LIST.LS"\LISTING=     (Argument tag notation)
    =/IFEQ "04"\0                 (Two digit number of arguments)
    =/IFEQ "XXXX"\(RX)            (Argument indirectly addressed)
```

NOTES:
a. An exact match of the string between the double quotes and the argument specified is required.

b. No punctuation between the second quote and the argument is allowed.

c. Double quotes within the literal string is not allowed.

NOTE

"Tag notation arguments" and the use of \(RX) are described in paragraph 3.2.3.

3.2.2.5 Aborted Chain File Restart Commands.

Chain Mode Command (PROCEED)

The PROCEED command allows restart of the aborted execution of a chain file beginning with the command following the one being executed at the time the CHAIN ABORTED message was sent. The values of the pseudo registers are not changed by the PROCEED command.

PROCEED Command syntax:

```
    PROC[EED]
```

Chain Mode Command (RETRY)

The RETRY command allows restart of the aborted execution of a chain file at the beginning of the command which was being executed when the CHAIN ABORTED message was generated.

RETRY Command syntax:

```
    RETR[Y]
```

3.2.2.6 Composite Chain File Examples.

Chain File Example 1:

The following chain file will assemble and link a list of files whose file names are arguments 1-n. A composite listing of the assemblies is generated, as well as a separate composite listing of the link maps. When all of the assemblies and link edits are complete, the composite listings are output to the link printer and all listing files are deleted. The use of subroutine chain file calls is demonstrated. The ASMTSK.CF performs all of the assemblies before returning, and then LNKTSK.CF performs all of the link edits before returning.

```
=/* PERFORM ALL OF THE ASSEMBLIES      (Message)
=/@ ASMTSK                             (Chain file subroutine call)
=/* PERFORM ALL OF THE LINK EDITS      (Message)
=/@ LNKTSK                             (Chain file subroutine call)
=/IFS \1                               (Conditional test for argument)
 =/* CREATE COMPOSITE LISTINGS OF ASSEMBLIES AND LINK EDITS    (Message)
 =/FOR 1,\0                    (FOR loop processing, to the number of arguments)
 =COPY \(RX).LS,ASMLIST.LS;A           (Append argument file to ASMLIST.LS)
 =COPY \(RX).LL,LINKMAP.LL;A           (Append argument file to LINKMAP.LL)
 =/ENDFOR                              (End of FOR loop processing)
 =/* OUTPUT COMPOSITE LISTINGS TO LINE PRINTER    (Message)
 =COPY ASMLIST.LL,#PR                  (Copy file to printer)
 =COPY LINKMAP.LL,#PR                  (Copy file to printer)
 =/* DELETE ALL OF THE LISTING FILES   (Message)
 =DEL *.LS;Y                           (Delete all files with extension .LS)
 =DEL *.LL;Y                           (Delete all files with extension .LL)
=/ENDIF                                (End of conditional test)
=/* TOTAL END OF CHAINFILE             (Message)
=@END                                  (End, regardless of nesting depth)
```

ASMTSK.CF contains the following:

```
=/* ASSEMBLES THE FILE NAMES IN THE ARGUMENT SUBSTITUTION TABLE   (Message)
=/* CAN BE EXECUTED DIRECTLY, OR AS A CHAINFILE SUBROUTINE        (Message)
=/FOR 1,\0                              (FOR loop processing all arguments)
=ASM \(RX),\(RX),\(RX);RZ=80            (Assemble file)
=\ENDFOR                                (End FOR loop processing)
=END                                    (End chain file subroutine)
```

LNKTSK.CF contains the following:

```
=/* LINKS THE FILE NAMES IN THE ARGUMENT SUBSTITUTION TABLE   (Message)
=/* CAN BE EXECUTED DIRECTLY, OR AS A CHAINFILE SUBROUTINE    (Message)
=/FOR 1,\0                              (FOR loop processing all arguments)
=LINK \(RX),\(RX),\(RX);HIMSXL=0..UTILIB.RO   (Link files)
=/ENDFOR                                (End for loop processing)
=END                                    (End chain file subroutine)
```

Chain File Example 2:

The following example illustrates parameter checking, conditional processing, the PAUSE command, and FOR loops. Arguments 1-3 are default values (volume, user, and catalog), and arguments 4-n are file names to be transferred. Argument 1 is the source, while 2 and 3 are destinations. Note that if arguments 2 and 3 are different user numbers, the chain file can only be executed by user = 0. Also note that if both arguments 2 and 3 are null, no files are transferred. First a check is performed for the presence of an argument 4 and, if none, a pause is executed to allow the user to enter from the keyboard. Although an = prompt is displayed, the user must enter an =ARG followed by the arguments. Failure to precede the input with an = prompt will cause the input to be ignored; however, the chain file processing will proceed. Failure to input the arguments will cause the termination of the processing. Note that the nested IF's require the two ENDIF's. In the FOR loop, processing begins with the fourth argument (not necessarily the first). Note that the conditional processing is properly terminated inside the FOR loop, as required. Note that the first @END totally terminates the processing, although it is in a nested IF. The second END does not require the @, although it is permissible.

```
=/IFC \4                                 (Test for argument 4)
 =/* ARG 1 = FROM <vol>:<user>..         (Message)
 =/* ARG 2 = TO   <vol>:<user>.. FOR .SA & .RO    (Message)
 =/* ARG 3 = TO   <vol>:<user>.. FOR .RO ONLY     (Message)
 =/* ARG 4 = FILENAMES TO BE TRANSFERRED          (Message)
 =/& ENTER "=ARG <arg1>,<arg2>,<arg3>,<arg4>...." AND RETURN (Pause and message)
 =/IFC \4                                (Test for argument 4)
  =/* YOUR ARGUMENTS WERE NOT ENTERED BECAUSE YOU DID NOT USE "=" FIRST
                                                (Message if argument not entered)
=@END                                    (End all levels of nesting)
 =/ENDIF                                 (End second level of test for argument 4)
=/ENDIF                                  (End first level of test for argument 4)
=/FOR 4,\0                               (FOR loop processing)
 =/IFS \2                                (Test for argument 2)
  =COPY \1\(RX).SA,\2\(RX).SA;Y          (Perform copy of .SA file)
  =COPY \1\(RX).RO,\2\(RX).RO;Y          (Perform copy of .RO file)
 =/ENDIF                                 (End test for argument 2 processing)
 =/IFS \3                                (Test for argument 3)
  =COPY \1\(RX).RO,\3\(RX).RO;Y          (Perform copy of .RO file only)
 =/ENDIF                                 (End IF argument 3 processing)
=/ENDFOR                                 (End FOR loop processing)
=END                                     (End chain file)
```

ARGUMENTS

3.2.3 Arguments Session Control Command (ARGUMENTS)

The ARGUMENTS command offers argument (parameter) substitution capability. Arguments to be substituted are specified on an ARGUMENTS command line, and their entry sequence directly determines the position of each in the current argument list. An argument's ordinal position in the current list (first, second, ... nth) is used as a concise means of referencing the argument to obtain an abbreviated notation. The notation permits concatenation of simple arguments into compound arguments so that general procedures or chain files can be implemented easily. Argument substitution is in effect for all modes of command processing: online, chain, and batch. The current argument list remains in effect until one of the following events occurs:

a. A NOARG (clear arguments) is executed.
b. A new argument list is entered.
c. The session ends.

Common space is shared between the argument substitution pointer table and chain file nesting (subroutine calls and FOR loops). A floating boundary exists between the pointer "heap" and the nesting "stack". An appropriate error message is displayed if an attempt is made to exceed the boundary. Thus, if a system is generated with a large number of logical units per task, and nesting is in effect, the number of arguments may be limited.

Argument substitution must comply with the following rules:

Maximum length of argument list = 156 characters

Argument 00 is the current number of arguments (number of the last argument)

Argument numbering = 01-79 (single digits permitted when no ambiguity exists)

Argument delimiter = "," (spaces are data, not delimiters).

String delimiter = < > (thus, <,> is taken to be a data comma rather than a delimiter).

Reference to arguments \<argument number>:

\00	is replaced with a 2-character string representing the number of arguments (or the number of the last argument).
\12	refers to argument 12 (not \1 concatenated with a 2).
\1\2	refers to a compound argument comprised of argument 2 concatenated to argument 1.
\(RX)	is replaced with the argument whose number is in the RX pseudo register; i.e., indirectly addressed.

ARGUMENTS Command Syntax

```
ARG[UMENTS]                      (Display argument list, regardless of OPT N)
ARG[UMENTS] <arg1>,...<argn>     (Enter new list and display unless OPT N is
                                  set)
```

where <argn> is an expression and can be a tag with an argument in the following syntax:

```
\<tag>=<string>
```

where:

tag	May be a string of any length beginning with an alphabetic character. The string may contain any character except an equal sign (=) or a comma (,) unless the tag is enclosed in angle brackets (< and >). In addition, since these string delimiting symbols (< and >) are stripped, double entry is required for their use within a string.
string	Is the argument to be substituted.

A tag argument can be referenced by:

a. \<argument number> - i.e., first argument (\1), second argument (\2), etc. in the current argument list.

or

b. \<tag>=

EXAMPLES:

```
=ARG GCD,,#PR,#NULL,IXMH,,<BATCH GCD,#PR>     (Argument list defined)
\1:GCD
\2:#PR
\3:#NULL                    (Message resulting from execution of
\4:IXMH                      above ARG command line)
\5:       (Null entry)
\6:BATCH GCD,#PR
=ARG <PASCAL \1,,\2>,PASCAL \1<,,>\2
\1:PASCAL GCD,,#PR          (Substitution made from previous list
\2:PASCAL GCD,,#PR           to create current list)
```

NOTE

Equivalent forms of <BATCH GCD,#PR> are:

```
BATCH <GCD,#PR>
BATCH GCD<,>#PR
```

```
=ARG A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P   (Argument list defined)
\1:A
\2:B
\3:C
\4:D
\5:E
\6:F
\7:G
\8:H
\9:I
\10:J
\11:K
\12:L
\13:M
\14:N
\15:O
\16:P
=\03\08\01\09\14 \04\05\13\15     (Concatenate parameter 3 before 8 before 1 ...
                                     before 15)
CHAIN DEMO                        (Result of last line)

=ARG ABC,DEF,GHI                  (Argument list defined)
\1:ABC
\2:DEF
\3:GHI
=ARG \1,PQR,\3                    (To change one or more arguments specified
\1:ABC                             in the previous ARG list, the entire list
\2:PQR                             must be reexpressed in the desired form)
\3:GHI
=ARG  CHAIN ,DEMO:,1,.,DEMOCHN,CF
\1:CHAIN
\2:DEMO:
\3:1
\4:.
\5:DEMOCHN.CF
\6:CF
=\1\2\3\4\4\5\4\6
CHAIN DEMO:1..DEMOCHN.CF          (Chain processing initiated)
END CHAIN
=ARG  1<<2,3>>2,<1,2,3>
\1:1<2,3>2
\2:1,2,3
=ARG 1<<2>
 WARNING: NO. "<" NOT EQUAL TO NO. ">"
\1:1<2>
```

\0 contains the number of arguments in the list. Rx contains, by default, the argument being processed in a FOR loop construct.

Example of the Use of \0 and \(RX)

A chain file consisting of the following statements:

```
    =/FOR    1,\0              (For all of the arguments consisting of file names)
     =ASM \(RX),\(RX),\(RX);RZ=80      (ASM file)
     =LINK \(RX),\(RX),\(RX);HIMX      (LINK file)
     =SPOOL P \(RX).LS,#PR             (Print .LS file through spooler to #PR)
     =SPOOL P \(RX).LL,#PR             (Print .LL file through spooler to #PR)
    =/ENDFOR                           (End FOR loop processing)
    =END
```

Example of the Use of Tag Notation

Existing arguments are displayed:

```
=ARG FILE=XYZ,DRIVE=#FD03,#PR
\1:FILE=XYZ
\2:DRIVE=#FD03
\3:#PR
```

```
Now, if the tag \FILE= is specified, XYZ is substituted.
     if         \1     is specified, FILE=XYZ is substituted.
```

Thus, following the above, the argument sequence can be changed:

```
=ARG \3,\2,\1

\1:#PR
\2:DRIVE=#FD03
\3:FILE=XYZ
```

and the tag argument can be changed:

```
=ARG \1,\2,FILE=ABC
\1:#PR
\2:DRIVE=#FD03
\3:FILE=ABC
```

permitting the following:

```
=PASCAL \FILE=,\FILE=,\FILE=
 PASCAL ABC,ABC,ABC
```

NOARGU-
MENT

BSTOP

3.2.4 No Arguments Session Control Command (NOARGUMENTS)

The No Arguments command clears the argument substitution list.

<u>No Arguments Command Syntax</u>

```
    NOARG[UMENTS]
```

<u>EXAMPLE</u>:

```
=ARG                    (Display argument list; arguments previously set)
\1:PAR1
\2:PAR2
=NOARG                  (Clear argument list)
=ARG                    (Display of cleared argument list requested)
=                       (List contains no argument)
```

3.2.5 Break Stop Session Control Command (BSTOP)

The Break Stop command is used to specify the option to issue a "STOP" to task(s) when break notification is received by the session control task. (Default is to terminate all tasks on break.) This command is identical to OPT H.

<u>NOTES</u>

a. Break notification is sent if there is no I/O in progress with the terminal. If the I/O request in progress specifies break notification, the requestor receives the notification (rather than the session control task).

b. The session control task requests break notification during I/O requests. If a prompt has been issued and input is not complete, a break will terminate the input with break error indication. If any user tasks in the same session are executing or are executable, they are neither terminated nor stopped as the result of terminating the input with the BREAK key.

c. This works in conjunction with OPT D. If -D is in effect (default), user tasks in the current session are terminated (-H) or stopped (H). If D is in effect, only the default user task (i.e., the last task started) is terminated (-H) or stopped (H).

<u>Break Stop Command Syntax</u>

```
    BSTO[P]
```

BTER

CONTINUE

DATE

3.2.6 Break Terminate Session Control Command (BTER) BTER

The Break Terminate command restores the default action to "TERM" tasks when break notification is received by the session control task. (See Break Stop "notes" in paragraph 3.2.5.) This command is identical to OPT -H.

<u>Break Terminate Command Syntax</u>

```
    BTER[M]
```

3.2.7 Continue Session Control Command (CONTINUE) CONT

The Continue command allows a task to resume execution following a STOP (from either an operator command or from a Break Stop action). The executive directive, START, is issued to task(s) specified (without registers supplied option).

<u>Continue Command Syntax</u>

```
    CONT[INUE]                    (Default user task assumed)
    CONT[INUE] <tttt> <ssss>      (Continue specified task)
    CONT[INUE] ALL <ssss>         (Continue all tasks in specified session)
```

where default <ssss> is the current session. If <ssss> is specified, it must be the current session unless the user number at logon = 0.

3.2.8 Date Session Control Command (DATE) DATE

The DATE command may be used to provide a time and date logging function, or to set the current date if the user number at logon time = 0 and if the system is not operating in the chain or batch mode. The current date may not be set in the chain or batch mode. (If user number is not 0, DATE displays the current date and time.) If user number = 0, a prompt is issued to "ENTER DATE (MM/DD/YY)=", where year must be 80 or greater. Each of the entries is validated and an error results in another prompt. To exit without changing the date, depress the BREAK key.

<u>DATE Command Syntax</u>

```
    DATE
```

<u>DATE Command Example (User 0)</u>

```
=DATE
 08:17:52 6/2/80
ENTER DATE (MM/DD/YR)=13/26/80          (Illegal MM)
ENTER DATE (MM/DD/YR)=1/32/80           (Illegal DD)
ENTER DATE (MM/DD/YR)=1.1.80            (Illegal separator)
ENTER DATE (MM/DD/YR)=1/1/79            (Illegal YR; year tested to ensure 80-99)
ENTER DATE (MM/DD/YR)=<BREAK>           (Exit, no change)
=DATE
 08:21:39 6/2/80
ENTER DATE (MM/DD/YR)=<BREAK>           (Exit, no change)
```

DEFAULTS

3.2.9 Defaults Session Control Command (DEFAULTS)

The Defaults command displays the current default values assumed by the session control task, and also displays set options. The USE command terminates with an implied DEF unless OPT N; i.e., the result of the USE command is displayed as if a DEF were entered unless option N is set.

DEFAULTS Command Syntax

```
    DEF[AULTS]
```

DEFAULTS Command Example

after initial logon, SESSION 0001 -

```
=DEF
 SYSTEM VOLUME = SYS0:
 USE DEFAULT VOLUME = SYS0:0..     (null catalog)
 USER NUMBER = 0                   (logon user number)
 USER TASK =                       (no task name; i.e., no tasks loaded)
 SESSION = 0001
 TERMINAL = CN00
 OPTION(S) SET =                   (no options set)
=
```

after logon as "DEMO:1" -

```
=DEF
 SYSTEM VOLUME = SYS0:
 USE DEFAULT VOLUME = DEMO:1..     (null catalog)
 USER NUMBER = 1                   (logon user number)
 USER TASK =                       (no default user task)
 SESSION = 0002
 TERMINAL = CN00
 OPTION(S) SET =
=LOAD DIR
 DIR 0002: LOADED
=DEF
 SYSTEM VOLUME = SYS0:
 USE DEFAULT VOLUME = DEMO:1..     (null catalog)
 USER NUMBER = 1
 USER TASK = DIR
 SESSION = 0002
 TERMINAL = CN00
 OPTION(S) SET =
=
```

NOTE

User number entered with USE DEFAULT should be the same as the logon user number, unless logon user number = 0.

LOAD

3.2.10 Load Session Control Command (LOAD)

The LOAD command creates a task and performs the load function for a program, but does not initiate execution of the program. For example, the user can load a program from a diskette and then change the diskette before initiating the program. Multiple loads are allowed, with the last one becoming the assumed default user task.

LOAD Command Syntax

```
LOAD <task descriptor> [<task arguments>]
```

where:

task descriptor	Is the file name field of the file descriptor of a load module -- i.e., a file of .LO extension.
task arguments	Is comprised of those arguments required by the task to be loaded.

LOAD Command Example

```
=LOAD BACKUP #FD00,#FD01
 BACK xxxx: LOADED  (response indicating normal completion; xxxx is session no.)
=
```

In this example, the program in file name "BACKUP" is loaded. The response message indicates the task name is "BACK" (which is used in subsequent system control commands to refer to this task.)

NOTES

The argument "#FD00,#FD01" is passed to the task "BACK" during the loading process.

BACK becomes the default user task which may be assumed by subsequent system control commands until: (a) another user task is loaded, or (b) BACK is started and either terminates or aborts.

OPTION

3.2.11 Option Session Control Command (OPTION)

The OPTION command allows any of fifteen alphabetic letter options, A through O, to be set (or cleared) for use, either singly or in the desired combinations. The resulting combination of options set is displayed unless OPT N is set.

OPTION Command Syntax

```
    OPT[ION][;<options>]
```

where <options> may be one or more of the following:

A - Aborts chain file processing when a task terminates/aborts with A0 $\neq$ 0.

B - Ring terminal bell when command prompt is issued.

C - Reserved.

D - User default task only is terminated or stopped on break (default is all user tasks).

E - Reserved.

F - Reserved.

G - Reserved.

H - Stop task(s) on break. (Also set by BSTOP command.) (Default is terminate; also reset by BTERM command.)

I - Perform task dump if task aborts.

J - Inhibit echo in SBARG subroutine (argument substitution subroutine for processing inputs to user tasks).

K - Inhibit translation of lowercase alphabetic letters to uppercase (also used by SBARG).

L - List =/xxx commands executed.

M - List =/xxx commands skipped as a result of a false condition.

N - No echo. Chain file commands are not displayed. This overrides options L and M. Also inhibits displaying the results of ARGUMENTS, OPTION, and USE commands.

O - Override error or abort and continue processing.

The default values for all options are the reset condition; i.e., -A through -O.

OPTION Command Examples

=OPT ;L	List =/XXX commands.
=OPT ;-L	(default) NO list.
	NOTE: =/ and =/ABT are printed if not in skip mode (skip mode is the action taken when IF test conditions are <u>not</u> met).
=OPT ;M	List =/XXX commands which are not executed during skip mode -- i.e., the condition was not met. The "=" is altered to a "/" and the resulting //XXX is printed.
	NOTE: The "ENDIF" is printed //ENDIF.
=OPT ;O	Override error or abort - continue processing.
	NOTES: a. Does not override =/ABT. b. Does not override syntax errors in =/XXX commands.
=OPT ;-O	(default) Abort chain processing on error or command abort.

Examples of Chain Processing Options L, M, N

```
LISTING OF INPUT CHAIN FILE
PAGE 1              LISTING OF  ELSE.CF

=OPT
=ARG A
=/IFS \1
 =/*TEST 1 SET
=/ELSE
 =/*TEST 1 CLEAR
=/ENDIF
=/IFC \1
 =*TEST 2 CLEAR
=/ELSE
 =/*TEST 2 SET
=/ENDIF
=NOARG
=/IFS \1
 =/*TEST 3 SET
=/ELSE
 =/*TEST 3 CLEAR
=/ENDIF
=/IFC \1
 =/*TEST 4 CLEAR
=/ELSE
 =/*TEST 4 SET
=/ENDIF
=END
```

PRINTOUT DURING EXECUTION

DEFAULT/NORMAL (NO OPTIONS SET)

```
=OPT
 OPTION(S) SET =
=ARG A
\1:A
 =/*TEST 1 SET
 =/*TEST 2 SET
=NOARG
 =/*TEST 3 CLEAR
 =/*TEST 4 CLEAR
=END
END CHAIN
```

VARIATIONS OF PRINTOUT DURING EXECUTION

```
=OPT
 OPTION(S) SET = L
=ARG A
\1:A
=/IFS A
 =/*TEST 1 SET
=/ELSE
=/IFC A
 =/*TEST 2 SET
=/ENDIF
=NOARG
=/IFS ~
 =/*TEST 3 CLEAR
=/ENDIF
=/IFC ~
 =/*TEST 4 CLEAR
=/ELSE
=END
END CHAIN
```

```
=OPT
 OPTION(S) SET = M
=ARG A
\1:A
 =/*TEST 1 SET
 //* TEST 1 CLEAR
//ENDIF
//ELSE
 =/*TEST 2 SET
=NOARG
 //*TEST 3 SET
//ELSE
 =/*TEST 3 CLEAR
 =/*TEST 4 CLEAR
 //*TEST 4 SET
//ENDIF
=END
END CHAIN
```

```
=OPT
 OPTION(S) SET = LM
=ARG A
\1:A
=/IFS A
 =/*TEST 1 SET
=/ELSE
 //* TEST 1 CLEAR
//ENDIF
=/IFC A
//ELSE
 =/*TEST 2 SET
=/ENDIF
=NOARG
=/IFS ~
 //*TEST 3 SET
//ELSE
 =/*TEST 3 CLEAR
=/ENDIF
=/IFC ~
 =/*TEST 4 CLEAR
=/ELSE
 //*TEST 4 SET
//ENDIF
=END
END CHAIN


 OPTION(S) SET = LMN  (N overrides LM)
END CHAIN



 OPTION(S) SET = N
END CHAIN
```

Tilde (~) represents a null argument.

START

3.2.12 Start Session Control Command (START)

The START command allows a previously loaded task to be initiated. If the task was not initiated when loaded, the START command passes register values to the task being started. In the event the task was initiated and stopped, it is treated as if a CONTINUE command were issued (the registers are not altered and the task resumes execution). The task started becomes the default task.

When a task is initiated (either as a result of STAR or load and go), the registers of the user task are initialized by VERSAdos as follows:

D0	Task name that loaded/started user task (monitor task).
D1	Session number that loaded/started user task (monitor task).
D2	User default volume name.
D3	Default user number.
D4 D5	User default catalog name.
D6	Length of command line passed (not including file name/command and terminating spaces).
D7	Bit mask of logical units assigned to user task where bit number = logical unit number.
A0	Task name of the user task (as read from loader information block).
A1	Terminal identification at logon time.
A2	Actual user number at logon time.
A3	Actual session number of user task.
A4-A7	Reserved for future use.

<u>D7 NOTES</u>

a. LUN=0 is not available to user tasks. Therefore, if set to 1, bit 0 indicates LUN 5 is the terminal device; if set to 0, bit 0 indicates LUN 5 is a file (Chain or Batch mode).

b. The command device or file (LUN=5) is assigned as public read and write when bit 0 is set to 1 (interactive mode) and public read only (chain or batch mode) when bit 0 is set to 0.

c. The log/list device or file (LUN=6) is assigned as public write only.

d. LUN's 5 and 6 are passed with keep option; all others are passed with/without keep, depending whether @ASSIGN'ed/ASSIGN'ed from the terminal (paragraph 3.3.1).

e. In the interactive mode, including the "chain aborted" mode, LUN 0 is transferred to the user task LUN 5. Thus, utility tasks using interactive mode may be executed during chain abort. During this time, LUN 5 remains assigned to the chain file; hence, the chain file can not be edited in the "chain abort" mode, but other files may be edited (unless assigned to a stopped user task). During chain file processing or batch mode, LUN 5 of the session control task is transferred to LUN 5 of the user task.

START Command Syntax

```
[@]STAR[T]  (start default user task)

                       or

[@]STAR[T] [<task name> [<task session number>]] (start specified task)
```

where:

task name	Is the 4-character name of the last task loaded.
task session number	Is the number of the session in which the task was created.

NOTES

a. If the task name and session number are not known, they may be found by means of the DEF command.

b. The optional "@" causes a prompt for another command following initiation of the task. Normally, a "wait for event" executive directive is executed following initiation. When the user task terminates or aborts, the session control task is awakened and a prompt for a new command is issued.

c. If a task name is not specified, the last task loaded (but not terminated or aborted) is assumed. If session number is specified, it must match the current session number, or the actual user number at logon must be 0. If a session number is specified, a task name must also be specified before the session number.

STOP

3.2.13 Stop Session Control Command (STOP)

The STOP command causes a task to discontinue execution. The executive directive, STOP, is issued to the task(s) specified. Task(s) which have been stopped may be continued or terminated by subsequent commands. (All user tasks in the same session which were stopped are terminated automatically when the session is terminated by LOGOFF or BYE commands.)

STOP Command Syntax

```
    STOP                          (Default user task assumed)
    STOP <tttt> [<ssss>]          (Stop specified task)
```

where:

tttt	Is a 4-character task name (valid for the specified session).
ssss	Default <ssss> is current session and, if <ssss> is specified, it must be the current session unless the user number at logon = 0.

See notes for TERMINATE command.

TERMINATE

The TERMINATE command causes the specified task(s) to be discontinued and removed from the system. All I/O is halted, all files and devices are closed and released, and all allocated memory is deallocated. By default, a TERM ALL command is issued when the BREAK key is pressed. This command issues the executive directive, TERMT, to the specified tasks.

TERMINATE Command Syntax

```
TERM[INATE]                    (Default user task assumed)
TERM[INATE] <tttt> <ssss>      (Terminate specified task)
TERM[INATE] ALL <ssss>         (Terminate all tasks in specified session)
```

where:

tttt	Is a 4-character task name (valid for the specified session).
ssss	Default <ssss> is current session and, if <ssss> is specified, it must be the current session unless the user number at logon = 0.

NOTES

a. The first character of a task name must begin with either A-Z or ".". Session control tasks begin with "&", and they may not be directly terminated. An entire session may be terminated by user 0 with "TERM ALL xxxx", where xxxx = session number. A test is performed to determine whether or not terminating a task will cause the operating system to crash. Such tasks may not be terminated.

b. User session tasks have session numbers consisting of ASCII characters. Server system tasks (beginning with ".") have binary session numbers; e.g., the spooling task. Binary session numbers must be preceded by "&". Examples:

```
TERM  .SPL  &1        (spooling task)
TERM  PR    &1        (printer task)
```

TIME

3.2.15 Time Session Control Command (TIME)

The TIME command may be used to provide a time and date logging function, or to set the current time if the user number at logon time = 0 and if the system is not operating in the chain or batch mode. The current time may not be set in the chain or batch mode. If user number = 0, a prompt is issued to "ENTER TIME (HR:MIN)=". Each of the entries is validated and an error results in another prompt. To exit without changing the time, depress the BREAK key. If the TIME command is entered by a user other than 0, only the current time and date are displayed.

TIME Command Syntax

```
    TIME
```

TIME Command Example (User 0)

```
=TIME
 09:22:37 4/2/82
ENTER TIME (HR:MIN)=13:9          (Time changed)
=TIME
 13:9:12 4/2/82
ENTER TIME (HR:MIN)=<BREAK>       (Exit, no change)
```

TIME Command Example (Other Users)

```
=TIME
 09:10:42 4/6/82
=
```

USE

3.2.16 Use Session Control Command (USE)

The USE command establishes default values to be used when abbreviated file names are specified. Default values for volume name, user number, and catalog may be specified for use until a subsequent USE command is issued or the user logs off. Only the fields entered are changed; the fields not supplied remain unchanged -- e.g., if only the user number is entered, the previous defaults for volume and catalog remain in effect. If a volume is specified, the volume is established as a user default volume for the duration of the session (or until a new default volume is specified).

When a user or system utility task is started, the most current user default volume is passed to the user task in register D2. The default user number is passed in register D3 (the actual user number at logon time is passed in A2). Default catalog is passed in D4 and D5. (See description of START, paragraph 3.2.12.)

USE Command Syntax

```
    USE [<volume name>:][<user number>][.<catalog>]
```

where:

volume name	Is the optional 1- to 4-alphanumeric character volume ID field of a file descriptor.
user number	Is the optional 1- to 4-decimal digit user number field of a file descriptor.
catalog	Is the optional 1- to 8-alphanumeric character catalog field of a file descriptor.

NOTE

The changed result is displayed as if a DEF command had been entered. The resulting display may be inhibited by setting OPT N.

Supplying Use Values at Logon Time

At logon time, the initial entry is processed as if a USE command were specified. Prior to processing the initial logon entry, the assumed default values are:

Volume	= Previously specified system volume
User	= (Must be entered at logon time)
Catalog	= All spaces (null or none)

To log on, the user must supply a user number, even if 0; volume and catalog may or may not be supplied. (Session 0001 requires specification of the system default volume, which also becomes the user default volume until changed by a subsequent USE command or the user logs off.)

NOTE: Changing the default catalog requires user number and/or volume -- e.g.,

```
<user number>.<catalog>
<volume name>:<user number>.<catalog>
:.<catalog> (default volume and user number)
```

To reset a default catalog to all spaces from any non-space value requires a special entry (because omitting a catalog preserves the previous entry).

```
<catalog>=&     resets default catalog to all spaces
```

Examples:

```
<user number>.&
<volume name>:<user number>.&
:.&
```

Session 0001 Logon

Session 0001 may be initiated by any user numbers. Session 0001 requires entry of the system default volume, even if only confirming the assumed default by entry of ":" followed by the user number. (Default system volume is specified at system generation. As supplied, the volume is SYS: for hard-disk-based systems, SYS0: for floppy-disk-based systems.) Time and date must also be entered, which is normally reserved for user 0, but is permissible for any user during this session. Upon completion of logon, a chain file, <default system volume>:0.PRIV.UPSYSTEM.CF is executed on behalf of user 0. The contents of this chain file can be changed by the system administrator by using the CRT editor. Suggested commands include start of spooling, SWORD, SECURE, and output of a message indicating the system has been started.

^ SESSION CONTROL COMMAND

HARD DISK SESSION CONTROL

^

3.2.17 ^ Session Control Command

The ^ session control command is used to cause the session control task to go into the dormant state. This command removes the conflict which arises when the session control task is prompting for input at the time a user task has outstanding I/O to the terminal. Then, at completion of user task I/O, the session control task is automatically reactivated. Should the ^ command be entered when no conflict exists, the user task may be restarted by depressing the BREAK key.

^ Command Syntax

```
     ^
```

^ Command Examples

=@DIR =^	The @ option entered ahead of DIR causes an immediate prompt for another command while the DIR.LO file is loaded and started, but prevented from using the terminal. This prompt inhibits output to the terminal from the DIR command. (Session control gives priority of terminal access to any task in the foreground over a task running in batch.) Entry of the ^ command puts session control into the dormant state, allowing DIR (the batch task) to output the default volume directory entries. The session control task is restarted and a new prompt issued when DIR activities are completed.
=@STAR (or @CONT) ^	Any task started or continued can have conflicting I/O to the terminal and additional tasks may need to be started or continued in a particular sequence. Thus, the ^ command may be used to cause the session control task to go dormant at the appropriate time.

3.3 VERSAdos HARD DISK SYSTEM SESSION CONTROL

In addition to the universal session control commands described in paragraph 3.2, versions of VERSAdos supplied with the multiuser hard-disk-based systems provide the following capabilities. These commands can be added to or excluded from the single-user version when performing a system generation. These commands are also executed as part of the session control task established at session initialization. In the command descriptions below, characters within square brackets are optional.

COMMAND	DESCRIPTION
ASSI[GN]	Assign device/file to logical unit.
CLOS[E]	Dissolve logical unit number assigned to device/file.
HELP	Display available session control commands.
NEWS	Display contents of 0.PRIV.NEWS.NW

ASSIGN

3.3.1 Assign Session Control Command (ASSIGN)

The ASSIGN command allows a user to assign a logical unit number (LUN) to a device or file. The assignment is made for the session control task with the access permission specified, and is passed to user task(s) that are subsequently started by the session control task. When the command is preceded by an "@", the session control task keeps the assignment and passes the assignment to user tasks with the FHS KEEP option. When not preceded by an @, the assignment is passed only to the next task started without the FHS KEEP option.

ASSIGN Command Syntax

```
[@]ASSI[GN] <lun>,<file/device><space><access permission>

     or

[@]ASSI[GN] <lun>,<file/device>;O=[$]nnnn
```

where:

@	Causes the control task to "keep" the assignment and pass to all subsequent tasks (refer to design specification regarding FHS function of change lu assignment). NOTE: May not be used with an exclusive access permission.
lun	Is a logical unit number. As supplied, VERSAdos supports nine LUN's and normally uses the following convention:

0 - reserved for session management
1 - input function
2 - output function
3 - listing function (when distinguished from system output)
4 - load or second input (as in a merge)
5 - system input from a command device or file
6 - system output to a log/list device or file

7 } used for chain file subroutine returns and FOR
8 } processing

NOTE

To ascertain conditions under which the preceding conventions may not be followed, refer to the description of the ASSIGN command in the VERSAdos Data Management Services and Program Loader User's Manual, RMS68KIO.

file/device	Is a device name or the file name field of a file descriptor.

access permission Is one of the following valid access permission codes:

EREW	Exclusive read and exclusive write
ERPW	Exclusive read and public write
PREW	Public read and exclusive write
PRPW	Public read and public write
EW	Exclusive write
PW	Public write
ER	Exclusive read
PR	Public read

In the alternate form of the ASSIGN command syntax shown above, the letter O is the only option allowed, and nnnn is the value used in the options field of the trap #3 parameter block:

```
nnnn=    0 PR
         2 PW
     $   A PW with overwrite existing file
     $  40 PR, position record pointer at end of file
```

ASSIGN Command Example

ASSI 3,#PR PW	Assign LUN=3 to #PR (line printer) with public write access.
or	
ASSI 3,#PR;O=2	

NOTE

If the access permission includes write capability, option bit 6 is set to position the current record pointer to the end of file (previously, it was by default always set to the beginning of the file, which had the effect of only allowing sequential files to be assigned for write purposes).

CLOSE

HELP

3.3.2 Close Session Control Command (CLOSE)

The CLOSE command is used to dissolve a logical unit assignment. Since terminal assignments made without the KEEP option are dissolved by the start of the first using task, the main use of the CLOSE command is dissolution of assignments which specified the KEEP option -- @. The CLOSE command can be used to release an assignment to LUN 5 or LUN 6; however, a user task <u>will not be started</u> unless both LUN's 5 and 6 are assigned (on-line dialog is through LUN 0, which cannot be closed). In the chain mode (on line), closing LUN 5 has the effect of both aborting the chain mode processing and issuing an END command.

In the batch mode, closing LUN 5 or LUN 6 will abort the batch job.

<u>CLOSE Command Syntax</u>

```
    CLOS[E] <lun>
```

where:

lun	Is a logical unit number.

<u>CLOSE Command Example</u>

CLOS 3	Dissolves assignment to LUN 3.

3.3.3 Help Session Control Command (HELP)

The HELP command causes output of a display of the session control tasks. Required characters and allowable characters for each command are shown -- the latter in lowercase.

<u>HELP Command Syntax</u>

```
    HELP
```

<u>HELP Command Example</u>

```
=HELP
SESSION CONTROL COMMANDS:
LOG OFf       LOGOFf        OFF           BYE
LOAD          STARt         STOP          CONTinue
TERMinate     ^             DATE          TIME
USE           DEFaults      ARGuments     NOARGuments
BSTOp         BTERm         OPTions       NEWS
HELP          ASSIgn        CLOSe         CHAIn
END           RETRy         PROCeed       R?
BATCh         CANCel        QUERy         ELIMinate
PASSword      SWORD         SECURE
(ENTRY OF LOWER CASE LETTERS OPTIONAL)
=
```

NEWS

3.3.4 News Session Control Command (NEWS)

Execution of the NEWS command causes an attempt to assign the file PRIV.NEWS.NW on the system default volume, user 0, to the LUN 1 communication channel. If successful, the file contents are directed to LUN 6, the system console. Thus, the file can be used to provide information to users.

The file belongs to the system administrator (user = 0); therefore, its contents can only be altered by user 0 through use of the CRT editor. Any user, however, can invoke the NEWS command.

Information of a general nature is contained in the file, as supplied.

NEWS Command Syntax

```
NEWS
```

3.3.5 Dissemination of News and System Information

Provision (in addition to that afforded by the NEWS session control command) has been made for the system administrator (user = 0) to disseminate information to users. Immediately following the logon message and prior to display of the first prompt, an attempt is made to assign the file 0.PRIV.BULLETIN.NW on the system default volume, user 0. If successful, the contents of the file are sent to LUN 6 (the logon terminal).

Should an attempt to log on be rejected, the file 0.PRIV.REJECT.NW is similarly output.

These files belong to the system administrator. Therefore, only user 0 can alter their contents.

Data in these files is changed, as desired, through use of the CRT editor. Some information of a general nature is contained in the files, as supplied.

CHAPTER 4
UTILITIES

COMMON
UTILITIES

CHAPTER 4

VERSAdos UTILITIES

4.1 INTRODUCTION

This chapter describes utilitarian functions of VERSAdos. Paragraph 4.2 describes those utilities which are supplied for use with both floppy disk and hard disk-based systems. Utilities for hard disk systems only are described in paragraph 4.3.

VERSAdos utilities are invoked by simply entering the name of the load image file (LO) which performs the function. The session control task calls the loader to create a task and load the file, and then starts the function. Since user-created files of type LO can also be loaded and started by simply specifying their name, any which perform a useful function may also be considered utility commands. These are created through use of the LINK command (linkage editor).

It is sometimes useful to load a utility but, rather than have execution begin automatically, start execution manually at the desired time. For example, this two-step operation will allow a diskette to be changed between loading and execution, which can be helpful in a two-drive system. The session control command LOAD, used in conjunction with the session control command START, allows this to be done. If the utility to be loaded requires arguments, these must be specified on the LOAD command line.

Error messages which may occur during execution of a utility program are described in the VERSAdos Messages Reference Manual, M68KVMSG.

4.2 COMMON UTILITIES

The utilities described in the following paragraphs may be used by all systems, regardless of disk storage type.

BACKUP

The BACKUP utility transfers data from one disk to another. Two major operating modes are provided by BACKUP: "track-to-track data transfer" and "file transfer". Before using BACKUP, new disks must have been formatted for VERSAdos, using the INIT utility.

Track-to-track mode is faster, but both the source and the destination disk must be of the same kind (hard or diskette) and capacity (except in single-to-double sided diskette backup). Information transferred in this mode cannot be selected or modified. Variations of the mode allow unverified transfer of data, transfer with verification, and single byte-to-byte comparison of two disks. Reporting of source destination disk differences is provided.

The file transfer mode provides more latitude. Source and destination disks can be of different kind and size. Individual files or families of files can be selected for transfer. File descriptor fields information can be specified on the destination disk. Indexed sequential files can be packed to reclaim internal file space. Files can be packed together to reclaim disk space. A starting point at which file transfer should begin can be specified on the source disk. Files can be selected by date range and/or file/family, or can be selected one at a time. When source data exceeds capacity of the destination disk, the file transfer mode permits insertion of additional destination disk(s). The additional disk(s) must have been initialized previously with the INIT utility.

The description and ownership of the destination disk may be specified in either mode.

If, during the backup procedure, the user tries to change the status of the disk being backed up from 'not write protected' to 'write protected', BACKUP will abort with a Device Status Change Error.

The desired BACKUP mode is selected by specifying one or more options in the command line options field.

BACKUP Command Syntax

```
BACKUP [<input field>,<output field>][,<list field>][;<options>]
  or
BACKUP [,,<list field>][;<options>]
```

where:

input field	Source; may be any <volume>:[<user>][.<catalog>] or device name (no file name or extension allowed). If not specified, defaults to lowest numbered hard disk device (if present; usually #HD00) or the lowest numbered floppy disk device. Refer also to Tables 4-1 through 4-3.
output field	Destination; may be any <volume>:[<user>][.<catalog>] or device name (no file name or extension allowed). If not specified, defaults to second lowest hard disk device (if present; usually #HD01) or second lowest floppy device. Refer also to Tables 4-1 through 4-3.

list field — Is a file name or device name. If option A or S is specified, the names of those files created on backup volumes are put in the list file. If option V or B is specified, any discrepancies found in the verify process are sent to the list file. The list file is not used for option U. If input and output fields are not specified, list field must be preceded by <u>two</u> commas. The default value for this field is the logon terminal.

options — (File transfer mode): Entering one of the main options A or R selects this mode. A variation is obtained by entering one or more of the sub-options -- Y, N, S, or P. Refer also to Table 4-3.

A - Append option. Files are transferred from the source disk to the destination disk. This is the default option when the backup involves a hard disk, or when Y, N, S, or P is entered without entering A or R. Source and destination disks may be of different types.

Y - When a file to be backed up already exists on the destination disk, the Y modifier causes the file to be overwritten.

N - When a file to be backed up already exists on the destination disk, the N modifier prevents the file from being overwritten (the new version is <u>not</u> copied).

S - The S modifier allows the user to enter a starting point in the input volume directory for the backup. This is especially useful if a backup is aborted, since it prevents one from having to start over from the beginning.

P - The P modifier specifies that all unused space within an indexed sequential file is to be reclaimed, thereby generating tightly packed files on the output disk and more free space, although the backup will proceed more slowly.

The A option also allows selection of files by date range and/or by file/family (when "S" for "SELECT FILES" is entered when prompted); if neither date range nor file/family is requested, files may be selected one at a time.

R - Reorganize option (destination disk must be a floppy). The destination floppy is first initialized, followed by an option "A" copy. The effect of this is to consolidate available disk space which exists between files into one large block at the end of the disk. The initialization that takes place does not do any formatting or validation, so the floppy must have been initialized previously using the INIT utility.

P - The P modifier specifies that all unused space within an indexed sequential file is to be reclaimed, thereby generating tightly packed files on the output disk and more free space, although the backup will proceed more slowly.

options

(Track-to-track mode): Entering one of the main options U, V, or B selects this mode. A variation is obtained by entering one or more of the sub-options C or T. Source disk and destination disk must be of equal capacity (except in single-to-double sided diskette backup). Refer also to Table 4-3.

U - Track-by-track option. The input disk will be copied track-by-track to the output disk. This is the default option when both disks are floppies or if the C sub-option is specified without the V or B option. Destination disk must have 0 bad sectors, as ascertained during formatting with INIT.

C - The C modifier specifies that if read or write errors occur during the transfer, BACKUP should continue from the point beyond the error (rather than terminating).

V - Verify option. The input and output disks will be compared, byte for byte, and any discrepancies will be listed to the list field.

T - The T modifier truncates the verify listing so that if more than one discrepancy is found in a given sector, only the first discrepancy will be printed. May not be specified without V or B.

B - Backup option. An option U backup is performed, followed by an option V backup to verify the data. Recommended when destination disk is suspected of having bad sectors. If bad sectors are found, lock them out using the INIT utility and run the backup again with a file-by-file option.

C - The C modifier specifies that if read or write errors occur during the transfer, BACKUP should continue from the point beyond the error (rather than terminating).

T - The T modifier truncates the verify listing so that if more than one discrepancy is found in a given sector, only the first discrepancy will be printed. May not be specified without V or B.

<u>NOTE</u>

The track-to-track options (U, V, B) should be used only when the source and destination disks are of identical media formats (i.e., media configuration attributes and parameters in the configuration areas on both disks are identical). The BACKUP utility will not produce an error if disks of different formats are used, but the output disk will be unusable.

TABLE 4-1. Destination User Number Field Values

	OUTPUT FIELD		
INPUT FIELD	NULL	SPECIFIED VALUE	*
NULL	Current default	Output value	Current default
SPECIFIED VALUE	Current default	Output value	Input value
*	Source values	Invalid	Source values

NOTE: This table depicts the user number that will be assigned to files created on the destination volume, depending on how the user number was specified in the input and output fields of the command line. Null obtains current user number default value. * represents all (or any) valid values for that field ("wildcard"). Multiple user numbers cannot be backed up to a single user number in one operation.

TABLE 4-2. Destination Catalog Field Values

	OUTPUT FIELD			
INPUT FIELD	NULL	SPECIFIED VALUE	* (FIELD)	* (CHARACTER)
NULL	Current default	Output value	Current default	Invalid
SPECIFIED VALUE	Current default	Output value	Input value	Invalid
* (FIELD)	Source values	Invalid	Source values	Invalid
* (CHARACTER)	Source values	Invalid	Source values	Invalid

NOTE: This table depicts the catalog name that will be assigned to files created on the destination volume, depending on how the catalog name was specified in the input and output fields of the command line. Null obtains current catalog field default value. * represents all (or any) valid values for that field or character position in the field ("wildcard"). Multiple catalogs cannot be backed up to a single catalog in one operation.

NOTE

See also the "Use of the * Character" paragraph in the COPY utility description, paragraph 4.2.3.

TABLE 4-3. Destination Disk Volume ID Field Values

		DESTINATION VID FIELD		
LOGON USER NUMBER	BACKUP OPTION	VOLUME ID	USER NUMBER	DESCRIPTION
= 0	R	Note (a)	Note (b)	Note (e)
≠ 0	R	Note (a)	Current default	Note (c)
Any	U or B	Note (d)	Source value	Note (d)

NOTES:

(a) User is prompted for input. Input is required. Default not allowed.

(b) User is prompted. No input obtains current default user number.

(c) User is prompted. No input obtains source description (if logon user number is zero or the same as the source user number) or a blank field (if logon user number is non-zero and not the same as the source user number).

(d) If source disk was loaded <u>before</u> destination disk, user is prompted; no input obtains corresponding source field value. If source disk was loaded <u>after</u> destination disk or is a non-VERSAdos disk, no prompt is issued. Source sector 0 is transferred as is.

(e) No input obtains same description as source disk.

<u>Examples</u>

In the following examples, operator inputs are shown underscored for clarity and are followed by a carriage return. The underscore is not typed. Note that in some examples, more operator inputs will be requested if the user is user 0.

<u>Example 1</u> - Disk backup, minimum command line, on a system with no hard disks

```
=BACKUP
 COPY FROM DEVICE FDxx TO DEVICE FDxx (Y/N) ? Y
 STARTING TRACK-BY-TRACK BACKUP PROCESS
 ENTER NEW VOLUME NAME
 ENTER DESCRIPTION (MAX 20 CHARACTERS)
```

The system selects the first two drive numbers (normally zero and one) for xx. Using the default option U, all files on the source are copied (track by track) onto the destination disk. If specified, a new volume ID and user description are created on the destination disk.

Example 2 - Disk backup with verify

```
=BACKUP #FD01,#FD02;B
 STARTING TRACK-BY-TRACK BACKUP PROCESS
 ENTER NEW VOLUME NAME
 ENTER DESCRIPTION (MAX 20 CHARACTERS)
 THE OUTPUT IS ...
 STARTING VERIFY PROCESS
```

The disk in #FD01 is copied to the disk in #FD02, and then they are compared. If specified, a new volume ID and user description are created on the destination disk.

Example 3 - Volume backup, floppy disks

```
=BACKUP VOL1:,VOL2:
 STARTING TRACK-BY-TRACK BACKUP PROCESS
 ENTER NEW VOLUME NAME
 ENTER DESCRIPTION (MAX 20 CHARACTERS)
```

All files on VOL1 are copied onto VOL2. If specified, a new volume ID and description are created on VOL2; otherwise, the volume ID on the destination is changed to VOL1 and the description is also changed. If using hard disks rather than floppies, option A will go into effect.

Example 4 - Append files

```
=BACKUP VOL2:*.*,VOL3:;A
 STARTING FILE-BY-FILE BACKUP PROCESS
 COPY ALL FILES, SELECT FILES, OR QUIT (A/S/Q)? A
```

Files on VOL2 are copied onto the remaining space on the drive 1 disk (or VOL3) until an identical file descriptor is encountered on the destination volume. The following message is then displayed:

```
DUPLICATE FILE - OK TO COPY (Y/N/Q)?     (full file descriptor of output file
                                          displayed here)
```

A Y response causes the destination file to be overwritten with the source file. An N response or just a carriage return causes the source file to be bypassed and execution continued until another identical file descriptor is found or all files are copied. The volume ID field in the file descriptor of each transferred file is changed to VOL3. A Q response causes BACKUP to terminate at that point.

Example 5 - Append files automatically

```
=BACKUP VOL1:,VOL2:;AY
 STARTING FILE-BY-FILE BACKUP PROCESS
 COPY ALL FILES, SELECT FILES, OR QUIT (A/S/Q)? A
```

The unprotected files in the default user number and catalog on VOL1 are added to VOL2. Any file on VOL2 having the same user number, catalog name, file name, and extension as the file to be copied from VOL1 is replaced by the file from VOL1.

NOTE

When the Y option is used, any destination volume file which has file name and extension fields identical to the file name and extension fields of a file on the source volume, is deleted and replaced by the source volume file, which could have different content. If a choice of deleting or not is desired, the BACKUP command using the A option alone should be used.

Example 6 - Track-by-track backup with continue

```
=BACKUP #FD01,#FD02;UC
 STARTING TRACK-BY-TRACK BACKUP PROCESS
      .
      .
      .
```

The backup will go to completion even though bad sectors exist on one of the disks. This option is useful when dealing with disks that have known bad sectors which are not being used or the user is attempting to salvage as much data as possible. Anytime a bad sector is encountered, the data for that sector on the receiving disk must be considered useless.

Each time a bad sector is encountered, the error message handler will report the error. The record number field represents the sector that caused the problem. If the command field says READ, the bad sector is on the input disk; a command field of WRITE indicates the error was on the output disk.

Example 7 - Append files starting with a specific file

```
=BACKUP SYS:99,VOL1:;AS
 STARTING FILE-BY-FILE BACKUP PROCESS
 COPY ALL FILES,SELECT FILES, OR QUIT (A/S/Q)? A
 ENTER RESTART FILENAME (INCLUDING USER NUMBER)
 99..TEST.SA
```

A file-by-file backup of files under user number 99 with current default catalog name will take place, starting with file SYS:99..TEST.SA and continuing to the end of the directory. This works with select files as well.

Example 8 - Append files (disk capacity exceeded)

```
=BACKUP VOL1:*.*,VOL2:*.*;A
 STARTING FILE-BY-FILE BACKUP PROCESS
 COPY ALL FILES, SELECT FILES, OR QUIT (A/S/Q)? A
 **OUTPUT DISK FULL** CONTINUE (Y/N)? Y
 REPLACE OUTPUT DISK WITH NEW INITIALIZED DISK
 ENTER THE NEW VOLUME NAME WHEN READY TO PROCEED VOL3
```

The file-by-file backup filled the first disk, and the user is given the chance to mount another volume and enter its existing volume name. The new disk must also have been initialized with INIT. (NOTE: The volume name cannot be changed at this point, even if option R was originally selected. Also, the disk need not be in the same drive). If the hard disk drive needs to be spun down to swap cartridges, this must be done after answering Y to ...CONTINUE (Y/N)? and prior to entering the new volume name.

Example 9 - Selective backup by criterion

```
=BACKUP VOL1:*.*,VOL2:;A
 STARTING FILE-BY-FILE BACKUP PROCESS
 COPY ALL FILES,SELECT FILES,OR QUIT (A/S/Q)? S
 DO YOU WANT A DATE RANGE (Y/N)? Y
 ENTER DATE RANGE FROM-TO (MM/DD/YY-MM/DD/YY) 5/20/80-2/15/81
 or, if only one date is desired, enter 6/22/80
```

Files with a date before May 20, 1980 or after February 15, 1981 are not considered for selection if other selection criteria exist, or are not copied if date range is the only selection criterion. Only files on the source disk are used for selection.

```
DO YOU WANT FILE/FAMILY SELECTION (Y/N)? Y
ENTER FILE/FAMILY SELECTIONS ONE AT A TIME (UP TO 10).
HIT RETURN ON EMPTY LINE TO TERMINATE SELECTIONS.
ENTER NAME  *.SA
ENTER NAME  PAY.*
ENTER NAME  22..SYSTEM.SY
ENTER NAME  CATB.*.*
ENTER NAME(CR)
```

NOTE

> If volume, user number, or catalog is omitted from a descriptor, the usual default is not used. Instead, the corresponding entry from the input field of the command line is used (or *, for user number and catalog, if a device was specified).

The following files are selected: All files with the extension SA; all files with the file name PAY; all files with a user number of 22, the file name of SYSTEM, and an extension of SY; and all files with a catalog name CATB.

A maximum of ten selections by family category can be specified. A file will be accepted if it passes any of the ten selection-by-category tests. If the Date Selection feature is active, the file must pass the date criterion before being considered for selection according to the family categories. Note that criteria expressed on the command line must also be satisfied. For example, had user #15 been specified on the command line, the entry CATB.*.* would not have obtained the descriptor VOL1:*.CATB.*.* as it does in the example as given, but the descriptor VOL1:0015.CATB.*.*.

Example 10 - Selective backup by file descriptor

```
=BACKUP VOL1:*.CAT1,VOL2:.CAT2;A
 STARTING FILE-BY-FILE BACKUP PROCESS
 COPY ALL FILES, SELECT FILES, OR QUIT (A/S/Q)? S
 DO YOU WANT A DATE RANGE (Y/N)? N
 DO YOU WANT FILE/FAMILY SELECTION (Y/N)? N
 OK TO COPY (Y/N/Q)? VOL2:0915.CAT2.FILE1.SA Y
 OK TO COPY (Y/N/Q)?...
```

All unprotected files of any user on VOL1 having a catalog name of CAT1 are made available for copying. Each file descriptor is displayed as modified for the destination by the entries in the output field.

Example 11 - Reorganized backup, hard disk to diskette

```
=BACKUP SYS:915,SAVE:;R
 WARNING--INPUT DISK IS LARGER THAN OUTPUT DISK
 STARTING FILE-BY-FILE BACKUP PROCESS
 COPY ALL FILES, SELECT FILES, OR QUIT (A/S/Q)? A
 ENTER NEW VOLUME NAME SAV2
 ENTER DESCRIPTION (MAX 20 CHARACTERS) PASCAL STUFF
 THE OUTPUT IS VOL = SAV2   USER# = 0317   DESC = PASCAL STUFF
```

Volume SAVE is initialized (all files on it are lost). Its volume ID becomes SAV2, its user number becomes the user's current default (317), and its description becomes PASCAL STUFF. All user number 915 files on SYS: under the user's default catalog are then copied to SAV2 under the user's default user no. (317) and catalog.

BUILDS

4.2.2 Build S-Record Utility (BUILDS)

The BUILDS utility transforms a binary load module into a file of ASCII-encoded information which may then be transported to another system for further use. The format of the records in the file is Motorola S-record, so called because each record begins with a byte containing the code for an ASCII "S" -- for start of record.

The ASCII (American Standard Code for Information Interchange) code is commonly used to represent information being transported between systems or contained in the various storage media. The alphabetical character "H", for example, is represented by the hexadecimal number 48 in ASCII. Any means of representing "48" which suits the particular nature of a transmission or storage medium can now be used, and the information inherent in the original "H" will be preserved.

The types of S-records generated by BUILDS are:

S0	Starting record. Contains task name.
S1	Data record for block starting at address that fits in 16 bits.
S2	Data record for block starting at address that requires 24 bits.
S8	Ending record if program start address requires 24 bits.
S9	Ending record if program start address fits in 16 bits.

Refer to Appendix A for additional information on the makeup and function of S-records.

BUILDS Command Syntax

```
BUILDS <input field>[,<output field>]
```

where:

input field	Is the descriptor of the load module file. A minimum of the file name field is required. .LO is the default extension. Beginning with version 4.0 of VERSAdos, BUILDS accepts .SY files. .LO was the only allowable extension for previous versions.
output field	Is the descriptor of a file in which the generated S-records are to be stored. If not specified, the extension will be .MX. If an output field is not specified, the generated S-record file is stored on the default volume under the input field file name with extension of .MX.

Build S-Record Utility Examples

```
=BUILDS PAYIN.LO,PAYREC.XY
```

VERSAdos searches the default volume, user number, and catalog for PAYIN.LO. A file of S-records is then created from the load module code. The file is stored on the default volume under the name PAYREC and is given the specified extension .XY.

```
=BUILDS PAYIN,PAYREC
```

VERSAdos searches the default volume, user number, and catalog for PAYIN, assuming an extension of .LO. BUILDS then creates a file of S-records and stores it on the default volume under the specified name PAYSREC, supplying a default extension of .MX.

```
=BUILDS PAYIN
```

VERSAdos searches the default volume, user number, and catalog for PAYIN, assuming an extension of .LO. BUILDS then creates a file of S-records and stores it on the default volume under the default name PAYIN, supplying a default extension of .MX.

If the input file is empty, the message

```
    LOADMODULE IS EMPTY
```

is displayed. The resulting .MX file will have nothing in it.

If a file by the same name as the output file name already exists, the question

```
    FILE EXISTS - OK TO OVERWRITE (Y/N)?
```

is displayed. Entering Y will cause BUILDS to replace what is currently in the file with the usual S-record output. Entering N will cause BUILDS to stop without changing the existing file.

CONNECT

4.2.3 Connect Utility (CONNECT)

The CONNECT utility allows the user at a terminal on a VERSAdos system to communicate with a second computer which is connected to a second port. It produces the same effect as physically disconnecting the terminal from the VERSAdos system and connecting it to the second computer, without having to touch any cables.

<u>CONNECT Command Syntax</u>

```
    CONNECT <device name>[,<exit character>][;L[=<line #>]]
```

where:

device name	specifies the device name of a second terminal port.
exit character	specifies a character which, when entered, allows the user to discontinue the CONNECT mode. The default exit character is CTRL-V. To select a different exit character, enter any alphabetic character (A to Z) in this field. The exit character then becomes the corresponding control character (CTRL-A to CTRL-Z).
line #	specifies the line on which the CONNECT reminder message should be displayed. This option is functional only when CONNECT is invoked from an EXORterm 155 or VME/10. The default is line 25. A different line may be specified by entering the appropriate number, from 10 to 25, in this field.

EXORterm 155 and VME/10 users may utilize the L option to achieve a friendlier interface. When L=n is specified, CONNECT performs the following functions on the terminal from which it was invoked before connecting the terminal to the other port.

a. Reset of the display screen.

b. Set up of the virtual screen (i.e., the area which scrolls while in CONNECT mode) as lines 1 through 1-n.

c. Display of the following message on line n:

```
        CONNECTed to #CNxx -- type CTRL-x to exit.
```

The CONNECT utility logically short circuits two ports -- the port to which the user's terminal is attached and the port specified in the first argument on the command line. All characters and break signals entering the first port are sent to the second port instead of following normal VERSAdos channels. Conversely, characters and break signals entering the second port are sent directly to the user's terminal port.

To discontinue the CONNECT mode of operation, enter the exit character from the terminal on which CONNECT was invoked.

Only the system administrator (user 0) can invoke the CONNECT utility, because it uses privileged operating system calls. Furthermore, CONNECT may be run only if both ports used fall into the following categories:

VME/10 keyboard and display screen
VM02 local ports
MVME400 ports

CONNECT is not supported for other serial ports.

CONNECT may be invoked from a chain file. Because the computer expects to receive input from a terminal whenever CONNECT mode is invoked, it continues to do so even when invoked from a chain file. Therefore, input is taken from the terminal from which the chain file was invoked, rather than from the chain file. CONNECT will not work when executed in batch mode.

CONNECT Utility Examples

Example 1

=CONNECT #CN02	The user's terminal is connected to the device attached to port #CN02. CTRL-V will exit from CONNECT.

Example 2

=CONNECT #CN01,X	The user's terminal is connected to the device attached to port #CN01. CTRL-X will exit CONNECT.

Example 3

=CONNECT #CN01;L=24	The user's terminal is connected to the device attached to port #CN01. CTRL-V will exit CONNECT. The reminder message is displayed on line 24 as follows:

```
CONNECTed to #CN01 -- type CTRL-V to exit.
```

4.2.4 Copy Utility (COPY)

The COPY utility copies a file onto the same volume under a new file name or onto another volume under the same or a new name. No data conversion is performed. Options to the basic command allow a file to be appended to the end of an existing file, packing of data in an indexed sequential file, character-by-character comparison of existing files with display of byte differences within records, and character-by-character comparison of a copied file and the original with display of byte differences within records. COPY output can be directed to the system printer.

COPY Command Syntax

```
    COPY <input field>,<output field>[,<list field>][;<options>]
```

where:

input field	May be any file name or device name.
output field	May be any file name or device name.
list field	May be a file name or device name.
options	May be A, B, N, P, Y, V, or V and T, as follows:

A - Append option. Causes the file specified in the input field to be appended to the end of the file specified in the output field.

B - Copy Verify option. Causes the output field file to be created from the input field file, and then compared against the input field file.

N - This option will automatically bypass copying an input field file to an output field file if the output field file already exists.

P - Pack option. Specifies that all unused space within an indexed sequential file is to be reclaimed, thereby generating tightly packed output files requiring less disk space.

Y - This option will automatically overwrite output field file names that currently exist.

V - Verify option. Verify-only function which causes the output field file to be compared against the input field file. Differences, when found, will be displayed.

T - This option, which can only be used with the V option, will output only the first difference encountered for a given record.

Any copy request that contains an invalid option will result in display of the appropriate status message, task termination, and display of the HELP panel.

Copy Utility Examples - Single Volume Operations

Example 1 - On the same volume, duplicate a file and give it a new name.

```
=COPY VOL2:..INDX.CF,NINDX
```

When executing this command line, the operating system assumes default values for user number and catalog name and uses the default volume for the output field volume ID. If the default user owns INDX.CF (or is system administrator), the file is rewritten on the disk called VOL2 under the new name NINDX. The extension of the input field file, .CF, is supplied. The input field file itself is unchanged.

A comparison of the copied file with the original can be obtained by specifying option B. The command line would then be:

```
=COPY VOL2:..INDX.CF,NINDX;B
```

If the output field file (the file must exist) and input field file require comparison, but no copy function is desired, option V is used. A valid comparison is indicated by the return of the system prompt (=). Otherwise, differences between the two files are displayed as described below.

A listing of differences can be obtained only by using the V or B option, with the output directed to the printer as follows:

```
=COPY VOL2:..INDX.CF,NINDX,#PR;<option>
```

Figure 4-1 shows a typical example.

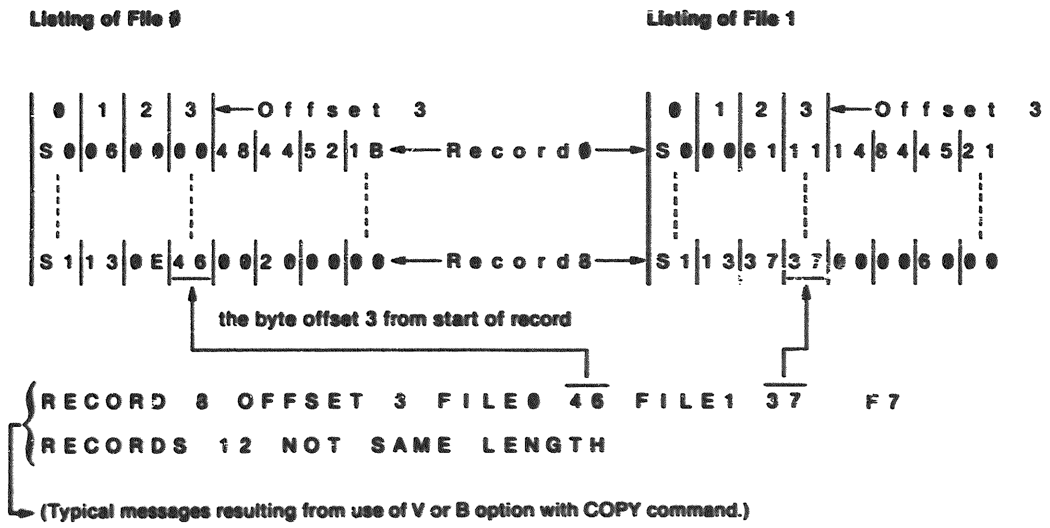

Listing of File 0
Listing of File 1
0 1 2 3 ←Offset 3
S 0 0 6 0 0 0 0 4 8 4 4 5 2 1 B ←Record 0→ S 0 0 0 6 1 1 1 1 4 8 4 4 5 2 1
S 1 1 3 0 E 4 6 0 0 2 0 0 0 0 0 ←Record 8→ S 1 1 3 3 7 3 7 0 0 0 0 6 0 0 0
the byte offset 3 from start of record
RECORD 8 OFFSET 3 FILE0 46 FILE1 37 F7
RECORDS 12 NOT SAME LENGTH
(Typical messages resulting from use of V or B option with COPY command.)

FIGURE 4-1. COPY Command Example

Example 2 - On the default volume, append one file to a second under the name of the second file.

```
=COPY INDEX.CF,XREF.CF;A
```

Default values for volume name, user number, and catalog name are assumed. If the proper file ownership exists, INDEX.CF is affixed to the end of file XREF.CF, as requested by the specified A option. INDEX.CF is unchanged.

Note that the append option requires that all files appended be of the same type and attributes.

Example 3 - On the same volume, copy an existing indexed sequential file to a new file name, but pack the file.

```
=COPY ERRORMSG.SA,SAVE.ERRORMSG.SA;P
```

Default values for volume, catalog, and user number are assumed. The system will search the default volume for the input field file and copy it to the output field file, packing it so that minimum disk space will be used for the output field file.

Copy Utility Examples - Two Volume Operations

Example 4 - Duplicate a file under the same file name on another volume.

```
=COPY INDEX.SA,VOL2:
```

Default volume ID, user number, and catalog name are assumed. The system searches the default volume for the specified file. If proper ownership exists, an identical file is created on VOL2 under the input field file name, INDEX.SA, which is itself unchanged.

Example 5 - Verify two files on different volumes, creating a file of differences to list and save.

```
=COPY XREF1.LO,VOL1:..XREF2.LO,POUT.SA;V
```

Default values are assumed for volume ID, user number, and catalog name. The system searches the default volume for the source file. VOL1 is searched for the destination file. Character-by-character comparison requested by the V option is performed and differences placed in a new file named POUT.SA on the default volume, user number, and catalog.

A listing of this file of differences can now be obtained. Although the COPY command could be used, a better method is to use the LIST command because it provides formatting and pagination, and skips perforated folds. The command line would be:

```
=LIST POUT.SA,#PR
```

Copy Utility Examples - Single Volume Operations

Example 1 - On the same volume, duplicate a file and give it a new name.

```
=COPY VOL2:..INDX.CF,NINDX
```

When executing this command line, the operating system assumes default values for user number and catalog name and uses the default volume for the output field volume ID. If the default user owns INDX.CF (or is system administrator), the file is rewritten on the disk called VOL2 under the new name NINDX. The extension of the input field file, .CF, is supplied. The input field file itself is unchanged.

A comparison of the copied file with the original can be obtained by specifying option B. The command line would then be:

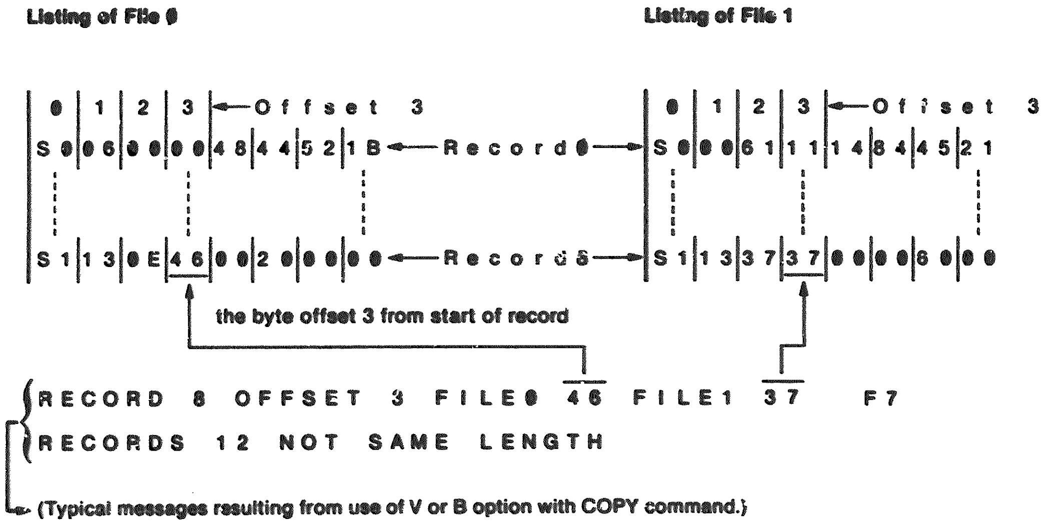
```
=COPY VOL2:..INDX.CF,NINDX;B
```

If the output field file (the file must exist) and input field file require comparison, but no copy function is desired, option V is used. A valid comparison is indicated by the return of the system prompt (=). Otherwise, differences between the two files are displayed as described below.

A listing of differences can be obtained only by using the V or B option, with the output directed to the printer as follows:

```
=COPY VOL2:..INDX.CF,NINDX,#PR;<option>
```

Figure 4-1 shows a typical example.



Listing of File 0
Listing of File 1
0 1 2 3 ← Offset 3
S 0|0 6|0 0|0 0|4 8|4 4|5 2|1 B ← Record 0 →
S 0|0 0|6 1|1 1|1 4|8 4|4 5|2 1
S 1|1 3|0 E|4 6|0 0|2 0|0 0|0 0 ← Record 8 →
S 1|1 3|3 7|3 7|0 0|0 0|8 0|0 0
the byte offset 3 from start of record

```
RECORD 8 OFFSET 3 FILE0 46 FILE1 37   F7
RECORDS 12 NOT SAME LENGTH
```

(Typical messages resulting from use of V or B option with COPY command.)

FIGURE 4-1. COPY Command Example

Example 2 - On the default volume, append one file to a second under the name of the second file.

```
=COPY INDEX.CF,XREF.CF;A
```

Default values for volume name, user number, and catalog name are assumed. If the proper file ownership exists, INDEX.CF is affixed to the end of file XREF.CF, as requested by the specified A option. INDEX.CF is unchanged.

Note that the append option requires that all files appended be of the same type and attributes.

Example 3 - On the same volume, copy an existing indexed sequential file to a new file name, but pack the file.

```
=COPY ERRORMSG.SA,SAVE.ERRORMSG.SA;P
```

Default values for volume, catalog, and user number are assumed. The system will search the default volume for the input field file and copy it to the output field file, packing it so that minimum disk space will be used for the output field file.

Copy Utility Examples - Two Volume Operations

Example 4 - Duplicate a file under the same file name on another volume.

```
=COPY INDEX.SA,VOL2:
```

Default volume ID, user number, and catalog name are assumed. The system searches the default volume for the specified file. If proper ownership exists, an identical file is created on VOL2 under the input field file name, INDEX.SA, which is itself unchanged.

Example 5 - Verify two files on different volumes, creating a file of differences to list and save.

```
=COPY XREF1.LO,VOL1:..XREF2.LO,POUT.SA;V
```

Default values are assumed for volume ID, user number, and catalog name. The system searches the default volume for the source file. VOL1 is searched for the destination file. Character-by-character comparison requested by the V option is performed and differences placed in a new file named POUT.SA on the default volume, user number, and catalog.

A listing of this file of differences can now be obtained. Although the COPY command could be used, a better method is to use the LIST command because it provides formatting and pagination, and skips perforated folds. The command line would be:

```
=LIST POUT.SA,#PR
```

Example 6 - Copy an existing file to the specified output file, but change the protect codes on the output file.

```
=COPY TEST.SA,TEMP.SA(AABB)
```

Default values for volume, user number, and catalog are assumed. The file TEMP.SA will be identical to TEST.SA but its protection code will have been changed.

COPY Utility Examples - Use of the * Character

The * (sometimes referred to as the "wildcard") can be used to represent one or more characters within a file descriptor field or to represent one or more entire fields. Its general meaning is "any" or "all" of the element it represents. As an entire field or fields, the asterisk can be used in the source or destination of both file descriptors. There must be a one-to-one source destination correspondence of asterisk-represented fields. When the asterisk is used to represent one or more characters within a field or fields, however, it must appear in corresponding character positions in like fields in the other file descriptor. If the asterisk appears at the end of a field, it can represent zero or more contiguous characters. If it does not appear at the end of the field, it may represent only one character. It is illegal to copy with family mode specified in the input field and a device in the output field. Only family type copy and/or verify request will output a status message as to the number of files copied and/or verified unless an error occurs.

Example 7 - Asterisk used for catalog, file name, and extension fields.

```
=COPY FIX:22.*.*.*,FIX:24.*.*.*;B
```

All files on volume FIX: belonging to user 22 are copied to user 24 and then verified. Note that when copying between user numbers, the default user number (supplied at logon or with the USE command (paragraph 2.3.1)) must be the destination user number. In other words, in the above example, user 24 may copy user 22's files (unless protected); but user 24 may not copy his files to user 22. However, user 0 may copy files between any two user numbers.

Example 8 - Asterisk used for extension fields.

```
=COPY WORK.*,PAY.*
```

Default values are assumed for volume ID, user number, and catalog name. All files named WORK with any extension are copied under the name PAY and given the corresponding extension.

Example 9 - Asterisk used for non-specified following characters.

```
=COPY ST*.*,GO*.*
```

Default values are assumed for volume ID, user number, and catalog name. All files of any extension whose names start with ST (regardless of following characters) are copied under corresponding new names starting with GO and having corresponding extensions.

Example 10 - Asterisk used for beginning characters.

```
=COPY **CSBUG.*,**XBUG.*
```

Default values are assumed for volume ID, user number, and catalog name. All source volume files of any extension with names beginning with any two characters and ending with CSBUG are copied to the destination file under names corresponding in the first two characters but ending in XBUG. Destination extensions also correspond to source extensions.

Example 11 - Use of asterisk to change catalog name.

```
=COPY CATA1.*.*,CATA2.*.*
```

All source files under the catalog name CATA1 are copied on the destination volume under the catalog name CATA2.

The following dialog typifies use of the COPY command with the verify option:

```
=COPY C*.*,F*.*;B
 COPY ALL OR SELECT FILES (A/S) ? A
 SYS:24..FOBOL.LO
 START VERIFY
   FILE 1 SYS:24..COBOL.LO
   FILE 2 SYS:24..FOBOL.LO
 SYS:24..FVT.RO
 START VERIFY
   FILE 1 SYS:24..CVT.RO
   FILE 2 SYS:24.. FVT.RO

 FILES COPIED   =   2
 FILES VERIFIED =   2
```

DEL

4.2.5 Delete Utility (DEL)

This utility removes a file name from a disk directory and frees all space allocated to that file. A single file, a list of files, or a family of files with the same file descriptor parameters can be deleted by executing a single command. For later reference, a listing file containing displayed messages as DEL executes can be created.

DEL Command Syntax

```
    DEL <input field>[,<output field>][;<option>]
```

where:

input field	May be any file name(s) or device name(s), such as NAME-1[/NAME-2/NAME-N].
output field	May be any file name or device name. The default value is the session terminal. If a file is specified without extension, the default is .LS.
option	Y, which automatically deletes all files in a family without prompting the user about each file.

Delete Utility Examples

Example 1 - Delete specified files.

```
         DEL TEST1.SA/TEST2.SA
```

Default volume ID, user number, and catalog name values are used to find the files specified in the input field. A message is displayed indicating the deletion of the specified file when deletion is selected:

```
         DELETED   VOL:21..TEST1.SA
         DELETED   VOL:21..TEST2.SA
```

Deletion involving the family notation allows the user to delete all files that satisfy the user family request. The user can request automatic deletion of all files or prompting of each individual file satisfying the family request by respectively selecting or not selecting the Y option. If the user requests prompting of each individual file, he will be given the opportunity to delete the existing file (Y), to bypass deleting the existing file (N), or to terminate the deletion process (Q).

Unless otherwise specified, the default volume, user number, and catalog will be used in the deletion process.

Example 2 - Delete all *.SA files, requesting a prompt for each file.

```
DEL *.SA,#PR
```

In this example, the user will be prompted for each file satisfying the family request. The confirmation message for each file selected for deletion will be directed to the printer.

```
OK TO DELETE (Y/N/Q)     VOL:21..TEST3.SA     Y
OK TO DELETE (Y/N/Q)     VOL:21..TEST4.SA     N
```

The confirmation message "DELETED VOL:21..TEST3.SA" -- where "VOL:21.." is the default volume, user number, and catalog -- is then output to the printer.

Example 3 - Automatically delete all *.LO files.

```
DEL *.LO;Y
```

In this example, automatic deletion of all *.LO files belonging to default user number 14 on the default volume SYS, under default catalog "blank", will occur, with the confirmation message being shown on the requesting device. For example:

```
DELETED  SYS:14..PATCH.LO
DELETED  SYS:14..TEST.LO
DELETED  SYS:14..LIB.LO
DELETED  SYS:14..DIRECT.LO
```

DIR

4.2.6 Directory Utility (DIR)

Information describing the disk space allocation, location, and other attributes of each file in a volume are stored in the volume directory. Part or all information entered for a file can be obtained through use of the Directory utility. The basic command obtains complete file names under the default volume, user number, and catalog established at logon or with USE (paragraph 2.3.1), and four options (used alone or in combination) provide greater file and volume detail.

DIR Command Syntax

```
DIR [<input field>][,<output field>][;<options>]
```

where:

input field	May be any file name or device name.
output field	May be any file name or device name.
options	Z - The Z option outputs the names of all files controlled by the default user number and user number 0.
	E - For any file name specified in the input field, the E option outputs all directory information (file type, data block and FAB sizes, update and access dates, and file's starting physical sector number).
	A - For any file name specified in the input field, the A option outputs all directory information, plus the physical sector numbers of all FAB's and data blocks of the file. Valid for volume owner or system administrator.

NOTE

Only the system administrator can use the DIR utility with the A option on a system volume. Others will get a PROTECT CODE ERROR and the E option will be used.

S - The S option is provided as an aid to the user working with a large number of files on a volume. It arranges the DIR command output according to user number, catalog name, file name, and extension, and may be used alone or in combination with the A, Z, or E option.

Sufficient memory is allocated to sort 200 (the default value) files. Should a number greater than this be encountered, the message FILE COUNT EXCEEDED - RETRY WITH LARGER 'S' OPTION is displayed. The user may respond by invoking the command as before, but with a specified number of files in the options field -- for example, ;S=300 -- in which case up to 300 files could be sorted. A number close to the actual number should be used since a larger risk of exceeding available memory exists with larger numbers.

The message NOT ENOUGH MEMORY FOR THIS 'S' OPTION might be encountered when using the S option. In this case, the user may reduce the number specified in the S option, or try again when memory is freed on completion of others tasks.

Directory Utility Examples

Example 1 - Display default catalog files alphabetically.

=DIR ;S   Null input and output fields.

```
DIR VERSION 111781 3   6/25/82 09:50:37

FIX:0021..EXAMP2.RO
FIX:0021..EXAMP2.SA
FIX:0021..EXORSTAT.SA
FIX:0021..FORCE.RO
FIX:0021..PASCALIB.RO
FIX:0021..PASPROG.LO
FIX:0021..SORT.RO
FIX:0021..SORT.SA

NUMBER FILES RETRIEVED =      8
```

Default user volume ID, user number, and catalog name values are used for the corresponding null fields within the command line input field. Output defaults to the system terminal, which displays the complete descriptor of each file belonging to the default user, with the default catalog name, on the default volume. File names are arranged in alphanumeric order.

Example 2 - Use of "wildcard" to obtain all catalogs.

=DIR VOL2:.*,#PR1;S

```
DIR VERSION 111781 3   6/25/82 11:59:38

VOL2:0021..AFILE.SA
VOL2:0021..MNEMONS.TX
VOL2:0021..TEST2.LO
VOL2:0021.PUBS.MNEMONS.SA
VOL2:0021.ZZ.SAMPLE.SA
VOL2:0021.ZZ.TEST.LO

NUMBER FILES RETRIEVED =      6
```

In this example, the "wildcard" asterisk (*) is used in the catalog field to get a directory of all files belonging to the default user on all catalogs on volume VOL2. The S option causes alphabetic listing, and output is directed to printer #PR1.

Example 3 - Minimum command, Z option (obtains user 0 and logon user file names from default volume).

=DIR ;Z   Null input and output fields, Z option.

```
DIR VERSION 111781 3   6/28/82 09:51:50

FIX:0000..IPL.SY
FIX:0000..DUMP.SY
FIX:0000..IOE.SA
FIX:0000..VERSADOS.SY
FIX:0000..TASKDUMP.SY
FIX:0000..ACCT.LO
FIX:0000..ASM.LO
       :
FIX:0000..SPLQUEUE.SQ
FIX:0021..PASCALIB.RO
FIX:0021..EXORSTAT.SA
FIX:0021..EXAMP2.SA
FIX:0021..SORT.SA
FIX:0021..SORT.RO
FIX:0021..FORCE.RO
FIX:0021..PASPROG.LO
FIX:0021..EXAMP2.RO

NUMBER FILES RETRIEVED =    189
```

Default volume ID, user number, and catalog name values established at session initiation or with USE are supplied by VERSAdos for the corresponding null fields within the command input field. Output defaults to the system terminal, which displays the names of all files on the default volume controlled by the logon user and user number 0.

Example 4 - Directory information for a specified file.

=DIR SORT.SA,#PR;E   File name input field, E option (contiguous file), device name output field.

```
DIR VERSION 111781 3   6/28/82 11:43:57

FIX:0021..PASPROG.LO
                  LOG    # OF                      REC   KEY FAB DB    DATE        DATE
  START    END    EOF   RECORDS   WC   RC   FT     LEN   LEN LEN LEN  CHANGED    ACCESSED
 $1A482 $1A52C     -       -      PP   PP    C      -     -   -   -   6/26/82    6/28/82


NUMBER FILES RETRIEVED =      1
```

=DIR PASPROG.LO;E   File name input field, E option (sequential file)

```
DIR VERSION 1117e1 3   6/28/82 09:53:24

FIX:0021..SORT.SA
                      LOG    # OF                           REC   KEY FAB DB    DATE       DATE
  START    END        EOF  RECORDS   WC   RC   FT           LEN   LEN LEN LEN CHANGED   ACCESSED
 $1A498 $1A498        $A     101     PP   PP    S             0     0   1   4 6/26/82    6/26/82


NUMBER FILES RETRIEVED =       1
```

In both examples, default values are used for volume, user number, and catalog. In the first example, output is directed to a printer. The E option results in the display or printout of the following information:

START END	Starting and ending sectors of the specified file(s). Useful only for contiguous files, as sequential and indexed sequential are shown with the same sector number for start and end.
LOG EOF	Logical sector number in which the file ends (noncontiguous files).
# OF RECORDS	Number of records in the file (noncontiguous files).
WC	Write protect code.
RC	Read protect code.
FT	File type: ID=indexed sequential, C=contiguous, S=sequential
REC LEN	Record length (noncontiguous files); if 0, then variable length records.
KEY LEN	Length of keys (noncontiguous files).
FAB LEN	Number of sectors in File Access Block (noncontiguous files).
DB LEN	Data Block size in sectors (noncontiguous files).
DATE CHANGED	
DATE ACCESSED	

NOTE: Refer to the VERSAdos Data Management Services and Programs Loader User's Manual, RMS68KIO, for further description of key length, FAB's, and DB's.

Example 5 - Directory and disk space allocation information for a specified file.

```
=DIR VOL2:21.PUBS.MNEMONS.SA;A        A option (noncontiguous file).
```

```
DIR VERSION 111781 3   6/28/82 10:03:42

VOL2:0021.PUBS.MNEMONS.SA
                    LOG    # OF                        REC  KEY FAB DB   DATE        DATE
  START    END      EOF  RECORDS  WC  RC  FT           LEN  LEN LEN LEN CHANGED   ACCESSED
   $34     $34      $12    164    PP  PP  ID            0    0   1   8  6/28/82    6/28/82

  FAB PSN          DATA START    # REC  # SECT      DATA START     # REC   # SECT
   $34                  $2C         9       1            $35          68        8
                        $30        15       2            $45          72        8
  SIZE         33/$21

TOTAL SIZE           33/$21

NUMBER FILES RETRIEVED =      1
```

For the specified noncontiguous file, option A provides the starting sector of each data block, the number of records held by each data block, and the number of sectors used by each data block, in addition to the information provided by the E option. Output defaults to the system terminal.

Example 6 - Volume ID display.

```
=DIR #FD10            Device number input field, null output and option fields.
```

```
DEVICE  FD10  IS VOLUME VOL2
USER NUM= 0021    DESC= EXCRMACS
```

VERSAdos determines the volume ID, user number, and description of the diskette in the specified drive, and outputs this information to the default system terminal.

DISMOUNT

4.2.7 Disk Dismounting Utility (DISMOUNT)

This utility performs the complementary function to the MOUNT utility. It forces the operating system to release control of a mounted VERSAdos disk and sets the driver to reject all I/O requests for the device until a new MOUNT command is issued. Dismounting of a foreign disk also sets the driver to reject all I/O requests for the device. DISMOUNT is not functional on EXORmacs Development Systems.

<u>NOTE</u>

The DISMOUNT utility is neither necessary nor allowed for disks connected to the IPC controllers -- i.e., VERSAmodule Floppy Disk Controller (FDC), M68KVM20, and VERSAmodule Universal Disk Controller (UDC), M68KVM21.

DISMOUNT Command Syntax

```
    DISMOUNT <input field>
```

where:

```
    input field    1) Is a disk device mnemonic of the form #HDcd (hard disk)
                                                                or
                                                               #FDcd (floppy disk)

                      where c specifies a SYSGEN-specified controller number
                            d specifies a SYSGEN-specified drive number
                                                  or
                   2) Is a VERSAdos volume name consisting of one to four
                      characters of the form xxxx:

                      where the first character is alphabetic and the remaining
                      characters are alphanumeric.
```

For VERSAdos disks, the volume name and device mnemonic are interchangeable on the command line. For foreign disks, the device mnemonic must be specified.

Dismounting Disks

Follow the steps below to dismount a disk.

a. Take the drive offline. For floppies, this is done by opening the drive door and removing the diskette. For hard disks, the procedure is drive-dependent. Consult the manufacturer's instructions.

b. Invoke the DISMOUNT utility as described earlier.

In normal operation, the only output that should be generated is the utility sign-on message.

```
=DISMOUNT #FD02
DISMOUNT Version xxxxxx x

=DISMOUNT TEST:
DISMOUNT Version xxxxxx x
```

DUMP

4.2.8 Dump Utility (DUMP)

During program development or problem investigation, it is often useful to examine the contents of portions of a file or volume. The DUMP utility provides the user with this facility.

Memory contents are "dumped" in compact hexadecimal form to the terminal or, optionally, are directed to the printer or to another file. Displayable ASCII characters are shown separately.

The basic command provides display of the contents of all the logical sectors comprising a specified file. An option to the basic command allows the starting and ending logical sector numbers of a portion of a file to be specified. Instead of a file name, a physical device name or volume name can be specified in the command input field in order to dump from disk to console or, optionally, from disk to file or from disk to printer.

Also provided is an interactive level under which the DUMP utility maintains control after execution of the initial command line. This level allows file portions to be dumped or changed one by one without entering the complete command line for each portion.

DUMP Command Syntax

```
    DUMP <input field>[,<output field>][;<option>]
```

where:

input field	May be (1) any file descriptor (minimum of file name <u>and</u> extension field required), (2) any device name, or (3) any volume name.
output field	May be any device or file name.
option	May be one of the following:

S=nn,T=mm   or   I

where: nn represents the logical sector number of the starting sector. If S=nn is not specified, nn defaults to zero.

mm represents the logical sector number of the ending sector. If T=mm is not specified, mm defaults to the last sector number of the file.

Each logical sector number (nn or mm) is an up to 8-digit number expressed in decimal or, if preceded by a dollar sign, hexadecimal representation.

NOTE

Sector numbers are physical when only device or volume name is given as <input field> on the command line.

I causes entry into the DUMP utility interactive level. When execution of the DUMP command line is initiated and this level is entered, a greater-than symbol (>) prompt is displayed. Following the prompt, interactive level subcommands may be entered, as described in the following paragraphs. If the I (interactive) option is selected, all other normally valid options which were specified on the command line are invalid. If other valid options were specified, the following message is displayed:

```
ONLY 'I' OPTION PROCESSED
```

Quit Dump Subcommand Syntax (QUIT)

```
>Q[UIT]
```

where:

Q or QUIT	Is the subcommand for quitting the DUMP utility and returning control to VERSAdos.

Dump Specified Sectors Subcommand Syntax (D)

```
>D <starting sector>,[<ending sector>]
```

where:

D	Is the subcommand mnemonic for dumping a specified portion of a file.
starting sector	Is a sector number within the file/device specified when the interactive level was entered. The number may be expressed in decimal or, if preceded by a dollar sign ($), in hexadecimal representation.
ending sector	Is a sector number within the file/device specified when the interactive level was entered. The number may be expressed in decimal or, if preceded by a dollar sign ($), in hexadecimal representation.

Dump Succeeding or Preceding Sector Subcommand Syntax (^)

```
>[^](CR)
```

where:

(CR)	Carriage return only dumps the next succeeding sector.
^(CR)	Circumflex (hex 5E), followed by a carriage return, dumps the previous sector.

Provision has been made within the DUMP utility for changing a file. The contents of a specified sector may be read into a sector change buffer, modified, and written back to disk. Modification is performed in a manner similar to the technique used in the PATCH utility except that DUMP does not recognize assembler mnemonics. The general procedure is:

a. The I option is specified on the DUMP command line to enter the interactive level.

b. The contents of a sector are read into the sector change buffer, using the R subcommand.

c. The M subcommand is used to enter the change mode and display a particular starting byte. On the same M subcommand line, the desired change is effected, using one of the PATCH utility change modes.

d. When the buffer contents are correct, typing a period (.) returns to the DUMP prompt (>), and then the W subcommand is used to write out the sector to disk, thus making the actual change to that sector in the file.

Two additional subcommands, S and F nn, are provided in the interactive level. S allows the sector change buffer contents to be displayed. F nn allows the buffer to be filled with any one byte value of the user's choice.

Subcommand Syntax (R)

```
>R nn
```

where:

R	Is the Read subcommand mnemonic.
nn	Are one or more digits (decimal assumed unless preceded by $).

Execution of this command causes the logical sector nn of the file, device, or volume specified on the DUMP command line to be read into the sector change buffer.

Fill Subcomand Syntax (F)

```
>F nn
```

where:

F	Is the Fill subcommand mnemonic.
nn	Are one or two digits (hexadecimal assumed).

Execution of this command fills each byte in the sector change buffer with the hex digits nn.

Show Subcommand Syntax (S)

```
>S
```

where:

S	Is the Show subcommand mnemonic.

Execution of this command displays the contents of the sector change buffer.

Enter Change Mode Subcommand Syntax (M)

```
>M nn
```

where:

M	Is the Enter Change Mode subcommand mnemonic.
nn	Are two digits (hexadecimal assumed).

Execution of this command causes entry into the change mode at the byte offset nn from the start of the sector change buffer. The specified byte is displayed and, on the same command line, changes to the buffer may be effected (starting at the byte displayed), exactly as described for the PATCH utility. Byte offsets are analagous to addresses as described in the PATCH utility. Changes can be made only within the boundaries of the current sector change buffer. The change is applied to disk only after execution of the W subcommand.

Write Subcommand Syntax (W)

```
>W [nn]
```

where:

W	Is the Write subcommand mnemonic.
nn	Is a one-or-more digit sector number (decimal assumed unless preceded by $).

Execution of the W command without a sector number may be performed only after a sector change buffer has been read. It will write the current contents of the sector change buffer back out to the sector from which it was originally read.

Execution of the W nn command writes the current contents of the sector change buffer to sector nn. Unlike the W subcommand given without a sector number, W nn may be used even if the sector change buffer has not yet been read in. The initial content of the sector change buffer is all zeros. This may be modified by reading a sector into the change buffer, using the M nn subcommand to set specific bytes in the change buffers, or using the F nn subcommand to fill the change buffer with constant byte values. In every case, the W nn command may be used. This form of the W command may not be used on non-contiguous files.

Out Subcommand Syntax (OUT)

```
>OUT <name>
```

where:

OUT	Is the Out subcommand mnemonic.
name	Is a file or device name.

Execution of this subcommand directs the listing to the name specified. <name> may be a file or device. The listing will continue to be directed to <name> until another Out subcommand is executed.

HELP Subcommand Syntax

```
>HELP
```

Execution of this subcommand causes the following display:

```
Legal options in interactive mode are:

'D S,T'      To dump from sector S to sector T; a leading '$' implies hex.

<CR>         To dump the next sector in succession.

^<CR>        To dump the previous sector.

'R S'        To read sector S into change buffer.

'S'          To display the sector change buffer.

'F S'        To fill the sector change buffer with S; S is assumed to be 1-byte
             hex.

'M S'        To enter change mode at offset S; S in the range from 0 to FF
             (always hex).

 'W'         To write the previously read sector change buffer.

'W N'        To write out to sector N a change buffer that has not previously
             been read in.  Cannot be used with non-contiguous files.

'OUT NAME'   To redirect listing to file or device name.

'QUIT'       To quit.
```

DUMP Utility Examples

The listing below resulted from the following procedure.

=DUMP PATCH.LO,#	Executing this line produces a scrolled display which may at any time be stopped by keying in CTRL-W. It may be restarted by pressing any other character.
=DUMP PATCH.LO,#PR;I	The interactive mode is called and the printer is attached.
>D $C,$D (other file portions may be dumped similarly)	Executing this line lists the specified sectors and returns the interactive mode prompt.
>QUIT =	DUMP is exited and control returned to VERSAdos.

Had the sectors of interest been known initially, the listing shown in Figure 4-2 could also have been produced without entering the interactive mode. The command line would be:

```
=DUMP PATCH.LO,#PR;S=$C,T=$D
```

```
DUMP VERSION  121181 3   2/24/82 18:24:42  DUMP OF FIX:0..PATCH.LO
     SN=$C              12
00     0C 10 00 30 6D 1A 0C 1C  00 39 63 0C 0C 10 00 41    ...0m....9c....A
10     6D 0E 0C 10 00 46 6E 08  52 88 B1 C9 6F E2 42 80    m....Fn.R...o.B.
20     20 5F 4A 8C 4E 75 4A AE  01 9A 66 00 FD E4 2D 7C     _J.NuJ...f...-|
30     00 00 00 01 01 9E 60 00  FF 32 4A AE 01 9A 66 C0    ......`..2J...f.
40     FD D0 2D 7C FF FF FF FF  01 9E 60 00 FF 1E 4A AE    ..-|......`...J.
50     01 9A 66 00 01 5A 2D 7C  0C 00 00 01 01 9E 42 AE    ..f..Z-|......B.
60     01 C2 48 EE 03 00 01 C6  2D 7A F6 28 01 96 52 88    ..H.....-z.(..R.
70     B1 C9 6E 2E 0C 10 C0 27  66 24 52 88 B1 C9 6E 22    ..n....'f$R...n"
80     0C 10 00 27 66 02 60 16  0C 10 00 5E 66 44 2D 7C    ...'f.`....^fD-|
90     FF FF FF FF 01 9E 52 88  B1 C9 6C 06 6C 34 61 42    ......R...l.`4aB
A0     60 CC 53 AE 01 92 4A AE  01 C2 66 14 1C 7C FF FF    `.S...J...f..|..
B0     01 C2 4C FA 03 00 F6 10  2D 7A F5 DC 01 92 60 AE    ..L.....-z....`.
C0     4A AE 01 9E 6A 00 FE A4  2D 7A F5 CC 01 92 60 0C    J...j...-z....`.
D0     FE 9A 43 FA F9 E6 41 FA  F9 F5 61 00 05 00 60 0C    ..C...A...a...`.
E0     FD 30 48 E7 00 C0 20 3A  F5 AA 61 00 03 7C 60 04    .0H... :..a..|`.
F0     2C 0F 60 36 66 34 4A AE  01 C2 66 06 4C DF 03 00    ,.`6f4J...f.L...


     SN=$D              13
00     60 22 24 48 4C DF 03 00  42 81 12 32 B8 00 B2 10    `"$HL...B..2....
10     67 12 D3 6E 01 BE 12 1C  D3 6E 01 C0 15 81 58 00    g..n....n.......
20     42 AE 01 BA 52 AE 01 92  4E 75 4C DF 03 00 4A 9F    B...R...NuL...J.
30     4A 86 66 0C FE 48 60 00  FC D8 B3 C8 67 18 52 88    J.f..H`.....g.R.
40     61 00 03 FA 61 00 FE AE  66 64 61 00 04 00 2D 41    a...a...fda...-A
50     01 D0 60 00 FC EC 20 3A  F5 78 43 FA F9 7C 61 00    ..`... :.xC..|a.
60     04 44 43 FA F9 6A 41 FA  F9 79 61 00 04 48 60 00    .DC..jA..ya..H`.
70     FC A0 52 88 0C 10 00 4D  66 02 52 88 61 00 03 BE    ..R....Mf.R.a...
80     2D 4F 01 9A 60 00 FD 04  4A AE 01 BA 66 04 61 00    -O..`...J...f.a.
90     03 96 20 3A F5 2A 43 FA  F9 74 61 00 04 08 43 FA    .. :.*C..ta...C.
A0     F9 60 41 FA F9 71 61 00  04 0C 60 00 F8 D2 61 00    .`A..qa...`...a.
B0     04 30 60 00 FC 5C 20 3A  F4 DA 00 BA F5 14 08 00    .0`..\ :........
C0     00 00 67 10 43 FA F8 7C  41 FA F8 98 61 00 C4 0E    ..g.C..|A...a...
D0     60 00 FC 3E 28 40 2F 00  43 FA F8 34 20 3A F4 B4    `..>(@/.C..4 :..
E0     7E 01 60 12 61 00 02 82  60 02 60 12 66 00 FC 22    ~.`.a...`.`.f.."
F0     12 C1 52 80 52 87 0C 87  00 00 00 0A 6F E6 0C 87    ..R.R.......o...
```

FIGURE 4-2. Typical DUMP

DUMPANAL

4.2.9 Dump Analysis Utility (DUMPANAL)

The DUMPANAL utility analyzes the contents of a system crash dump. It will list various system tables and any memory locations as they appear in the dump file.

A system crash dump is saved on disk as a result of the firmware-resident debug monitor BD command, or by using the DUMP button on the front panel of an EXORmacs.

DUMPANAL is an interactive operating system debugging tool. Output can be directed to the user's terminal or can be redirected to a printer.

DUMPANAL Command Syntax

```
    DUMPANAL [<input field>][,<output field>]
```

where:

input field	Is the file descriptor of the dump file to be analyzed. If no file descriptor is supplied, then the user's default volume and user ID are assumed with a filename DUMP.SY.
output field	Is the name of the device where output will be listed. If no output field is given, the default is the user's interactive terminal.

DUMPANAL responds by displaying a prompt for entry of subcommands:

```
    DMPA:>
```

Quit Subcommand Syntax (QUIT)

```
    DMPA:>Q[UIT]
```

where:

Q or QUIT	Is the subcommand for terminating the DUMPANAL utility and returning control to VERSAdos.

Address Dump Subcommand Syntax (AD)

```
    DMPA:>AD <address1>[,<offset>] <address2>
```

where:

AD	Is the subcommand to dump a block of memory in hex format.
address1	Is the starting address of the block of memory to be dumped.
offset	Is a hex value that will be added to <address1> to determine the starting address of the block of memory dumped.
address2	Is the ending address of the block of memory dumped.

Channel Control Block Subcommand Syntax (CCBS)

```
DMPA:>C[CBS]
```

where:

CCBS or C	Is the subcommand to dump the contents of all Channel Control Blocks known to the system.

Free Memory Subcommand Syntax (FREE)

```
DMPA:>F[REE] <partition#>
```

where:

FREE or F	Is the subcommand used to list the free memory list for any memory partition.
partition #	Is the memory partition number.

Global Segment Table Subcommand Syntax (GST)

```
DMPA:>G[ST]
```

where:

GST or G	Is the subcommand to list all entries in the Global Segment Table. Each entry describes the size and attributes of a globally or locally shareable segment. See Appendix E, paragraph E.5, of the Real-Time Multitasking Software User's Manual, M68KRMS68K, for a description of the Global Segment Table.

Map Subcommand Syntax (MAP)

```
DMPA:>M[AP]
```

where:

MAP or M	Is the subcommand for listing the limits and size of each memory partition and the free memory list for partition 0. See Appendix E, paragraph E.9, of the Real-Time Multitasking Software User's Manual for a description of the Memory Map and Free Memory List.

Memory Dump Subcommand Syntax (MD)

```
DMPA:>MD <address>[,<offset>] <# bytes>
```

where:

MD	Is the subcommand used to dump a block of memory in hex format.
address	Is the starting address of the block of memory to be dumped.
offset	Is a hex value that will be added to <address> to determine the starting address of the block of memory dumped.
# bytes	Is the number (expressed in hex) of bytes to be dumped.

Offset Subcommand Syntax (OFF)

```
DMPA:>OFF <address>
```

where:

OFF	Is the subcommand used to set a default memory address offset for use by the MD and AD subcommands.
address	Is a hex value to be used as a memory address offset.

Output Subcommand Syntax (OUT)

```
DMPA:>OUT <device>
```

where:

OUT	Is the subcommand used to select an output device.
device	Is the output device. # selects the user's terminal; #PRn selects a printer.

Periodic Activation Table Subcommand Syntax (PAT)

```
DMPA:>P[AT]
```

where:

PAT or P	Is the subcommand used to list all of the tasks with entries in the Periodic Activation Table as a result of having delayed or requested periodic activation.

See Appendix E, paragraph E.10, of the Real-Time Multitasking Software User's Manual for a description of the information found in the Periodic Activation table.

READY Subcommand Syntax (READY)

```
DMPA:>R[EADY]
```

where:

READY or R	Is the subcommand used to list selected information about each task currently on the 'Ready' list (these tasks are waiting to run). The information is taken from the Task Control Block associated with each task. See Appendix E, paragraph E.2, of the Real-Time Multitasking Software User's Manual for a description of a Task Control Block.

Register Subcommand Syntax (REGS)

```
DMPA:>REGS
```

where:

REGS	Is the subcommand used to list the contents of the program counter, status register, and all address and data registers at the time of the system crash.

Stack Subcommand Syntax (STACK)

```
DMPA:>STACK
```

where:

STACK	Is the subcommand used to dump 200 (hex) bytes of the system stack area.

System TCB Subcommand Syntax (STCB)

```
DMPA:>S[TCB]
```

where:

STCB or S	Is the subcommand for listing selected information about each system task currently known to the system. The information listed is taken from the Task Control Block associated with each task. See Appendix E, paragraph E.2, of the Real-Time Multitasking Software User's Manual for a description of a Task Control Block.

System Parameters Subcommand Syntax (SYSP)

```
    DMPA:>SY[SP]
```

where:

SYSP or SY	Is the subcommand for listing the system parameters. See Appendix E, paragraph E.1, of the Real-Time Multitasking Software User's Manual for a description of system parameters.

Tables Subcommand Syntax (TABL)

```
    DMPA:>T[ABL]
```

where:

TABL or T	Is the subcommand for listing all of the tables recognized by DUMPANAL. The TABL subcommand is the equivalent of entering the following subcommands: SYSP REGS STACK TRAP MAP TCB READY PAT UST GST CCBS TRACE

<u>Task Subcommand Syntax (TASK)</u>

```
DMPA:>TASK <task name>[ <session>]
```

where:

TASK	Is the subcommand used to request a detailed list of information about one task. The information listed is found in the Task Control Block and Task Segment Table for the given task. See Appendix E, paragraphs E.2 and E.3, of the Real-Time Multitasking Software User's Manual for descriptions of the Task Control Block and Task Segment tables.
task name	Task name can be entered as a 4-character ASCII value or as an 8-character hex value preceded by $.
session	Session can be entered as a 4-character ASCII value or as an 8-character hex value preceded by $. If no session is entered, the first task found with the given task name will be listed.

<u>Task Control Block Subcommand Syntax (TCB)</u>

```
DMPA:>TCB
```

where:

TCB	Is the subcommand for listing selected information about each task currently known to the system. The information listed is taken from the Task Control Block associated with each task. See Appendix E, paragraph E.2, of the Real-Time Multitasking Software User's Manual for a description of a Task Control Block.

<u>Trace Subcommand Syntax (TRAC)</u>

```
DMPA:>TR[AC]
```

where:

TRAC or TR	Is the subcommand used to list the System Trace Table entries. The entries are listed in chronological order from the oldest to the most recent. The time that each entry was built and the difference in time between entries is also listed. See Appendix E, paragraph E.7, of the Real-Time Multitasking Software User's Manual for a description of Trace entries.

<u>Trap Subcommand Syntax (TRAP)</u>

```
DMPA:>TRAP
```

where:

TRAP	Is the subcommand for listing the Trap Instruction Assignment Table and the Trap Instruction Owner table. These tables indicate which Trap instructions are currently owned by server tasks. See Appendix E, paragraph E.1, of the Real-Time Multitasking Software User's Manual for a description of the TIAT and TIOT tables.

<u>User Semaphore Table Subcommand Syntax (UST)</u>

```
DMPA:>US[T]
```

where:

UST or US	Is the subcommand to list all entries in the User Semaphore Table. See Appendix E, paragraph E.6, of the Real-Time Multitasking Software User's Manual for a description of the User Semaphore Table.

<u>User TCB Subcommand Syntax (UTCB)</u>

```
DMPA:>U[TCB]
```

where:

UTCB or U	Is the subcommand for listing selected information about each user task currently known to the system. The information listed is taken from the Task Control Block associated with each task. See Appendix E, paragraph E.2, of the Real-Time Multitasking Software User's Manual for a description of a Task Control Block.

DUMPANAL - Summary of Subcommands

Program Control Commands

OUT <device>	Send output to <device>.
QUIT	Terminate program.

Memory Dump Commands

OFF <address>	Set default memory address offset to <address>.
MD <address1>[,<offset>]	<# bytes> Dump <# bytes> starting at <address1> + <offset>.
AD <address1>[,<offset>]	<address2> Dump memory <address1> to <address2>.
REGS	List registers at time of crash.
STACK	Dump $200 bytes of system stack area.

List System Table Commands

CCBS	Channel Control Blocks.
FREE n	Free Memory List for memory partition n.
GST	Global Segment Table.
MAP	Memory Map.
PAT	Periodic Activation Table.
READY	Task Ready List.
STCB	System Tasks.
SYSP	System Parameters.
TABL	All system tables.
TASK <name>[<session>]	Detailed information about one task.
TCB	All tasks.
TRAC	System Trace Table.
TRAP	Trap Instruction Tables.
UST	User Semaphore Table.
UTCB	All user tasks.

FREE

4.2.10 Free Utility (FREE)

Knowledge of unallocated space on a disk is often needed for file creation or editing, or before copying a file. The FREE utility determines and displays in decimal and hexadecimal representation the total number of available sectors and the size of the largest available block of contiguous sectors in decimal and hexadecimal representation for a specified volume.

FREE Command Syntax

```
FREE [<volume ID>][,<list field>]
```

where:

volume ID	is the ID of the volume from which allocation information is required. Device names are not allowed.
list field	is the name of any file or physical device. If it is a file, the default extension is .LS.

NOTE

Only the system administrator can use the FREE utility on a system volume. Others will get a PROTECT CODE ERROR.

FREE Utility Example

Example of minimum command line - null input and output fields.

```
=FREE
```

VERSAdos supplies the default volume ID established at session initiation (the null user number and catalog name fields are ignored) and directs the following message to the default system output device:

```
VOLUME AAAA
        DDDDDDDD/HHHHHHHH  TOTAL SECTORS AVAILABLE     (decimal/hex numbers with
         DDDDDDDD/HHHHHHH  LARGEST CONTIGUOUS SECTORS   leading zeros blanked)
                     DDD%  OF SECTORS ARE AVAILABLE
```

INIT

4.2.11 Initialize Utility (INIT)

The Initialize utility, INIT, must be used on all new disks. Additionally, new hard disks and double-sided diskettes must be formatted -- i.e., certain sector addressing information must be written on them. (New single-sided diskettes must be initialized, but ordinarily do not require formatting.)

INIT performs both of these tasks. Its interactive dialog permits the user to input information necessary for the Volume Identification Block (VID), including a volume ID of four alphanumeric characters (first character must be alphabetic) and a 20-character description (any ASCII characters). It will check user-specified configuration data against the actual disk configuration. If the disk is already formatted accordingly, it need not be reformatted. If an error occurs during this check, the first line of the configuration change request is output again.

The INIT utility will write configuration information onto the disk. It will also generate a bootable disk from .SY files that require an absolute load address or that are position-independent, having a load address specified by the user at INIT time. During validation, the physical addresses of bad hard disk sectors are flagged in the Sector Allocation Table (SAT) and written in the Sector Lockout Table (SLT) by the INIT command. Detailed descriptions of logical disk and file structure, including the SAT and the SLT, are provided in the REPAIR utility discussion.

CAUTION

INIT MAY BE USED TO REINITIALIZE AND/OR REFORMAT A PREVIOUSLY USED DISK TO ALLOW ITS RE-USE. HOWEVER, IT IS RECOMMENDED THAT THE DIR UTILITY BE USED FIRST TO IDENTIFY THE CONTENTS OF THE DISK AND MAKE SURE NO WANTED FILES ARE PRESENT. FORMATTING WITH INIT DESTROYS ALL DATA ON THE DISK. INIT ALWAYS INITIALIZES THE DIRECTORY INFORMATION.

NOTE

Operation of the INIT utility differs slightly between IPC controllers (i.e., VERSAmodule Floppy Disk Controller (FDC), M68KVM20, and VERSAmodule Universal Disk Controller (UDC), M68KVM21) and non-IPC disk controllers.

INIT Command Syntax

```
INIT <device name>|<volume name>[;<option>]
```

where:

device name	Is the drive in which the disk to be initialized is mounted; i.e., #FD00-#FD03 or #HD00-#HD03.
volume name	Is the name assigned to the volume; e.g., VOL1. NOTE: Volume names must not be the same as device names; e.g., FD00-FD03.
option	V, which allows the user to define load address of bootstrap. This option is ignored if 0 was not entered as user number.

Initialize Utility Example

```
=INIT WIN6:
OK TO INITIALIZE #FD02  (Y/N)?    VOLUME WIN6  Y       (The volume ID, if one
                                                        exists, of the diskette
                                                        in drive FD02 is dis-
                                                        played after the ? as
                                                        a reminder.)

Data Density of media (S-single,D-double) D >(CR)      (Initializations done on
Track Density of media (S-single,D-double)D >(CR)       non-IPC controllers allow
S-Single or D-Double sided diskette       D >(CR)       the user to change the
Media format (M-Motorola,I-IBM standard)  I >(CR)       current configuration be-
Physical sectors/track on media(8 bits)    16 >(CR)     fore the disk is initial-
Number of cylinders on media(16 bits)      80 >(CR)     ized. See discussion be-
Interleave factor on media(8 bits)         1 >(CR)      low for responses. These
Bytes/physical sector on media(16 bits)    256 >C       lines will not print out
                                                        for initializations done
                                                        on IPC controllers.)

DO YOU WANT TO FORMAT DISK (Y/N) ?  Y                  (If a new disk.)
START FORMAT
ENTER NEW VOLUME NAME WIN6
ENTER USER NUMBER 0                                    (Displayed for logon user
                                                        0 only; user 0 will own
                                                        volume.)

ENTER DESCRIPTION (MAX 20 CHARACTERS)  FLOYD'S WORK DISK
DO YOU WANT THE BOOT STRAP (Y/N) ? N                   (Displayed only if user 0
DO YOU WANT A DUMP AREA (Y/N) ? N                       was entered above.)
DO YOU WANT TO VALIDATE SECTORS (Y/N)? N
=
```

The following responses to the configuration information are recognized.

(CR) - Retain current value and issue prompt for the next item.

Q - Terminate. (If the initializer is terminated with the Q command, the disk is unusable because its VID has been cleared.)

C - Configure device with current configuration information. This response must be entered before the configuration process can begin.

CAUTION

ON DISK CONTROLLER BOARDS THAT REQUIRE MOUNT, IF A FLOPPY DISK IS NOT MOUNTED WITH THE MOUNT UTILITY, IT IS CONSIDERED A FOREIGN DISK AND CAN BE REINITIALIZED WITHOUT USER NUMBER PROTECTION.

Initialize Utility Example - Hard Disk System

```
=INIT #HD10;V
OK TO INITIALIZE #HD10  (Y/N) ? Y                         (The volume ID of the disk
                                                           in drive HD10, if one
                                                           exists, is displayed
                                                           after ? as a reminder.)

Physical sectors/track on media(8 bits)      32 >(CR)   (Initializations done on
Number of cylinders on media(16 bits)        256 >(CR)   non-IPC controllers allow
Interleave factor on media(8 bits)           8 >(CR)     the user to change the
Bytes/physical sector on media(16 bits)      256 > C     current configuration be-
                                                         fore the disk is initial-
                                                         ized. See discussion be-
                                                         low for responses. These
                                                         lines will not print out
                                                         for initializations done
                                                         on IPC controllers.)

DO YOU WANT TO FORMAT DISK (Y/N)  ?  Y                  (If a new disk.)
START FORMAT
TRACK BY TRACK FORMAT  (Y/N) ? N                        (See discussion below.)
ENTER NEW VOLUME NAME SASI
ENTER USER NUMBER 0                                     (Displayed for logon user
                                                         0 only; user 0 will own
                                                         volume.)

ENTER DESCRIPTION (MAX 20 CHARACTERS)  TEST DISK
DO YOU HAVE LOCKOUT TABLE ENTRIES (Y/N) ? N
   VALIDATING SECTORS....
   0 BAD SECTORS ENCOUNTERED
DO YOU WANT A DIAGNOSTIC TEST AREA (Y/N) ? N            (See discussion below.)
DO YOU WANT THE BOOT STRAP (Y/N) ? Y
FILE NAME IS: WIN:0000..IPL.SY
ENTER NEW NAME IF NEEDED  WIN:0..VERSADOS.SY            (V option)
THE CURRENT LOAD ADDRESS IS  $00040000
ENTER NEW LOAD ADDRESS $     (CR)
DO YOU WANT A DUMP AREA (Y/N) ? N
=
```

If a non-IPC hard disk is being formatted, the user must choose either the format track or the format disk option. (If a floppy is being formatted, a format track is always used.) Format track allows the user to read or write other devices on the same controller board while the hard disk is being formatted. If a format disk is requested, no other devices can be accessed until formatting is completed on the hard disk.

Hard disks have an SLT for reserving sectors; floppies do not. The access to the SLT provided by the INIT command allows a user to reserve a number of sectors for special use. Reserved sectors are flagged (the corresponding bits are set) in the SAT, and the appropriate information is entered in the SLT. Bad sectors discovered during validation are also flagged in the SAT and entered in the SLT. Reference the REPAIR command for a description of SLT.

Another structure allowed on hard disk is the diagnostic test area. When a diagnostic test area is requested, two complete tracks are reserved on each recording surface -- one on the inner and one on the outer portion of each recording surface. The reserved tracks are used to store diagnostic programs and test patterns used for determining disk system viability during periodic maintenance. Sectors in the diagnostic area are flagged in the SAT as unavailable.

If the bootstrap is opted for, the specified file will be copied to the initialized disk. It will be set up so that the firmware-resident debug monitor BO command, if told to use this disk, will load the file and execute it. For EXORmacs, VMC 68/2, and VME/10 systems, this file is a bootstrap program which loads the actual operating system from another file. The defaults for volume, user number, and catalog are the system volume, 0, and null, respectively.

If option V was specified on the command line, and the bootstrap is to be copied to the intialized disk, then the user can specify the load address. For example, if the user wants the bootstrap to be loaded at address $1000 instead of $0000, then $1000 should be entered when asked for. For this option to be operable, the user must have entered 0 for user number at logon or with the USE command.

The option of specifying a dump area allows an area of the desired size (of diskette capacity, for example) to be allocated under the file name DUMP.SY. Following a system crash, the resident debug monitor can then access the dump file via the pointer in the disk volume information directory installed by the INIT command. Note that sectors are always validated for hard disk initialization -- a choice is not offered.

NOTES

The file DUMP.SY is write protected. This is done so that the file cannot be deleted or overwritten during a BACKUP or COPY. If DUMP.SY is deleted or moved to another area on the disk, the dump area pointers in the VID (reference the REPAIR utility for the format of VID) must also be changed because this is the information that the firmware-resident debug monitor uses when a memory dump is requested. If DUMP.SY does not reside where the VID pointers indicate, a memory dump may cause valid data (either files or system tables) to be destroyed.

For VMC 68/2 and VME/10 systems, the V option must be used, and the load address is $E00.

Incorporating Manufacturer's Bad Sector Data into the SLT

New hard disks are delivered with a list describing cylinder, head, and sector number for each sector known to be bad. These locations should be entered into the SLT to ensure that data will never be stored in them.

The following steps can be used to find the Physical Sector Numbers (PSN's) of these bad sectors for entry into the SLT using INIT:

a. Find the sectors per track (SPT), spiral offset (SPO), and number of heads (HDS) for your drive type. Table 4-4 provides this information for initializations done on IPC controllers. For initializations done on non-IPC controllers, consult manufacturer's documentation.

b. For each sector in the manufacturer's bad sector list, do the following:

   - Locate the cylinder number (C), head number (H), and sector number (S).

   - Compute the PSN, using the following formula:

```
OFFSET = (S-SPO*(H-1)) MOD SPT
IF OFFSET <0 THEN OFFSET = OFFSET + SPT
PSN = (C*HDS+H-1) * SPT + OFFSET
```

TABLE 4-4. IPC Controller Disk Drive Parameters

DRIVE CODE	MANUFACTURER	MODEL	TYPE	MEDIA	SECTORS/ TRACK (SPT)	DATA HEADS (HDS)	CYLINDERS	CAPACITY FORMATTED (MByte)	CAPACITY UNFORMATTED (MByte)	SPIRAL OFFSET (SPO)
0	Control Data Corp. (CDC)	9448-32	CMD	Removable Fixed	64 64	1 1	823 823	13.5 13.5	16 16	4
1	Control Data Corp. (CDC)	9448-64	CMD	Removable Fixed	64 64	1 3	823 823	13.5 40.5	16 48	4
2	Control Data Corp. (CDC)	9448-96	CMD	Removable Fixed	64 64	1 5	823 823	13.5 67.4	16 80	4
3	Control Data Corp. (CDC)	9455	LMD	Removable Fixed	64 64	2 2	206 206	6.8 6.8	8.5 8.5	0
4	Control Data Corp. (CDC)	9457	LMD	Removable Fixed	64 64	2 2	624 624	20.4 20.4	25 25	0
5	Control Data Corp. (CDC)	9762	SMD	Removable	64	5	823	67.4	80	4
6	Control Data Corp. (CDC)	9764	SMD	Removable	64	19	411	128	150	4
7	Control Data Corp. (CDC)	9766	SMD	Removable	64	19	823	256	300	4
8	Control Data Corp. (CDC)	9730-80	MMD	Winchester	64	5	823	67.4	80	4
9	Control Data Corp. (CDC)	9730-160	MMD	Winchester	64	10	823	135	160	4
10	PRIAM	3350	--	Winchester	64	3	561	27.6	33	4
11	PRIAM	6650	--	Winchester	64	3	1024	50.3	66	4

LIB

4.2.12 Library Utility (LIB)

The purpose of the Library utility is to complete the process of making functionally related groups of proven routines, subprograms, and programs available for relocation and linking into executable load modules by the linkage editor. Since the Library command operates on files of relocatable object type produced by assembly or compilation of source ASCII (.SA extension) files, the creation of a library begins early in the software development process. Program parts that the user knows will be useful should be pulled together into .SA files through use of the CRT editor, and then assembled or compiled into .RO files. These can then be combined into a library file or files, using the library utility. At linkage time, both the user-created library files and those supplied with the system high-level languages can then be made available for the resolution of external references, relocation, and linking.

The capability to manipulate individual modules within a library file is provided by the command set of the Library utility.

Each copy of a file module added to a library file is given a name taken from the first record of the copied module. The user should ensure that each module name within a library file is unique, for two reasons:

a. The linkage editor, in its first pass in which it searches for externally defined symbols, examines the modules in a library file for a symbol. Upon finding a symbol, it thereafter associates that symbol with the name of the module in which the symbol was found. On the relocation and linking pass, the editor tries to access the module by name. An error will result if another module of the same name resides within the same library file.

b. As the number of modules increases, a descriptive name can be a valuable memory aid.

Relocatable object module names come into existence through use of an assembler, compiler, or linkage editor. Use of the assembler directive, IDNT, allows a name to be given to the resulting .RO module. If the directive is not used, however, the default name ASM is assigned and will become the name of the library module made from that .RO module. Therefore, consistent use of the IDNT directive is recommended to eliminate this source of duplicate library module names.

The Library utility is a language comprised of seven commands: ADD, REPLACE, DELETE, COPY, LIST, QUIT, HELP, and CHANGE. When the utility is brought into memory, it also builds an internal directory of names of all modules in the .RO file cited on the command line. This directory is updated thereafter as the utility is used.

LIB Command Syntax

```
LIB <libname>[,<output field>]
```

where:

LIB	Is the command mnemonic.
libname	Is a standard VERSAdos file descriptor. The descriptor can be fully specified or, as a minimum, the file name field specified with established default values assumed. The default value for the file name extension field is .RO.
output field	May be any file name or device name. The default value is the session terminal. If a file is specified without extension, the default is .LS. Only the LIST command will cause output to be directed to the specified output field.

The Library utility responds to the command entry by displaying a greater-than symbol (>), indicating that the subcommand set of the utility is now available for use. Following the > prompt, the user may quit the utility or construct a library through use of the following seven subcommands. If a file called <libname> exists, it is made available for modification. Otherwise, a file of that name is created.

Quit Subcommand Syntax (QUIT)

```
>QUIT
```

where:

QUIT	Is the subcommand mnemonic.

Add Subcommand Syntax (ADD)

```
>A[DD] <filename>[<option>]
```

where:

ADD or A	Is the subcommand mnemonic.
filename	Is the file from which a copy of the first module therein is added to the library file in accordance with the option specified. The default file extension is .RO.
option	Will determine where in the library the new module is to be added. It may be one of the two options which follow. If no option is selected, the module will be added at the end of the library.

/	This option will cause the module to be added at the beginning of the library.
/<modname>	This option will cause the module to be added after the file <modname> which is already in the library.

Multiple modules may be added by specifying each module, with its option (if specified), separated from the next module by a comma. An attempt to ADD a module from a file containing multiple modules results in a display of the following message:

```
WARNING: SOURCE CONTAINS MORE THAN ONE MODULE
```

Example:

```
>ADD     FILE3,FILE2/,FILE1/FILE2
```

The end result of this is that:

FILE2	will be at the beginning of the library.
FILE1	will follow FILE2 in the library.
FILE3	will be at the end of the library.

Delete Subcommand Syntax (DELETE)

```
>DEL[ETE] <modname>[,<modname2>...,<modnameN>]
```

where:

DELETE or DEL	Is the subcommand mnemonic.
modname	Causes the module called modname to be deleted from the library file.
,modname2,modnameN	Is an optional field in which the names of other modules to be deleted from the library file may be entered.

Replace Subcommand Syntax (REPLACE)

```
>R[EPLACE] <filename>/<modname>[,<filename2>/<modname2>]...
```

where:

REPLACE or R	Is the subcommand mnemonic.
filename	Is the file from which the first module will be taken to replace the module specified in the following field. The default extension is .RO.
modname	Is the module in the library which will be replaced by the first module from the file specified in the preceding field.
,filename2/modname2	Signifies that multiple module replacements can be specified on a single line, if desired. (NOTE: modules may be named to capacity of command line.)

The module name in the replacement file can be identical to the name of the library module being replaced. If the module names are different, the name of the libarary module is replaced with the name of the replacement module. Identical module names are not allowed in a library file.

Copy Subcommand Syntax (COPY)

```
    >CO[PY] <filename>/<modname>[,<filename2>/<modname>,<filename3>/
                     <modname>]...
```

where:

COPY or CO	Is the subcommand mnemonic.
filename	Is the name of a VERSAdos file which will be created to contain the module specified in the following field. The default extension is .RO.
modname	Is the name of a module in the library file, a copy of which will be made and placed in the created file called filename.RO.
filename2/modname,filename3/modname	Signifies that copying of multiple modules can be specified on a single line, if desired.

This command allows copies to be made of selected modules within a library file. Modules within a library file are unchanged.

List Subcommand Syntax (LIST)

```
    >LIST [,<modname>]...
```

where:

LIST	Is the subcommand mnemonic. If no module names are specified, library directory entries for all modules in the library are listed.
modname	Causes the directory entry for a specified module in the library to be listed. More than one module within a library can be specified.

The LIST subcommand supplies the following directory information for each specified module:

Module name	Description
Version number	Name of source file from which .RO module
Revision number	was created
Language processor type	Date and time the .RO file was created

The above information is taken from the identification record of the .RO file.

Change Subcommand Syntax (CHANGE)

```
    >CH[ANGE] <modname>
```

where:

CHANGE or CH	Is the subcommand mnemonic.
modname	Is the name of a module in the library file. One or more items in the identification section of the first record in the module can be changed in response to displayed messages as follows:

DISPLAY		USER RESPONSE
MODNAME	ENTER NEW MODULE NAME	(CR) or <modname1>(CR)
OLD-nnn	ENTER NEW VERSION #	(CR) or nnn(CR)
OLD-nnn	ENTER NEW REVISION #	(CR) or nnn(CR)
OLD DESCRIPTION		
ENTER DESCRIPTION UP TO nn CHARACTERS		(Note that nn is the length of the old description. The new description cannot exceed this length. If shorter, unused spaces are filled with blanks.)

Help Subcommand Syntax (HELP)

```
    >HELP
```

where HELP is the subcommand mnemonic.

This subcommand will give the user a list of the subcommands and their corresponding command line syntax.

LIST

4.2.13 List Utility (LIST)

All or part of a file can be printed or displayed through use of the LIST utility. The command allows row and column page format, a heading, and line numbers to be specified. Portions of a file can be obtained by citing beginning and ending line numbers. The file should be sequential or indexed sequential and contain displayable ASCII characters.

LIST Command Syntax

```
    LIST <input field>[,<output field>][;<options>]
```

where:

input field	May be any file descriptor (minimum of file name field required; if not specified, default extension of .SA assumed). Multiple file names separated by "/" may be specified. File should be sequential or indexed sequential, containing only displayable ASCII characters.
output field	May be any physical device or file name. If output field is a file name with no extension specified, the extension will default to .LS.
options	I - Go into interactive mode. If this option is specified, no other options specified on the command line will be processed. Interactive mode entry is indicated by display of a greater-than symbol (>).
	H - Prompt user for a heading before performing list. If the H option is not selected, the heading defaults to the input field exactly as it is keyed in.
	F - Prompt user for nonstandard page format. With this option, the user can select a nonstandard count for the number of rows and columns to appear on a single page. If this option is not selected, the default row count is 60 and the default column count is 80 for non-printer devices. If the output device is a printer, row and column count will default to the values defined for that device at system generation time.
	L=nn - Selecting the L= option will result in line numbers being placed on the output listing. nn is the first (base) line number assigned to the first record listed. Line numbers require eight characters at the beginning of the output record. nn can be up to six decimal digits. The minimum value for nn is 1.
	S=nn - Defines the beginning line to list. nn is hexadecimal if preceded by $; otherwise, decimal is assumed. nn can be up to five decimal digits.

T=nn - Defines the ending line to list. nn is hexadecimal if preceded by '$'; otherwise, decimal is assumed. nn can be up to five decimal digits.

NOTE: If neither the S= nor T= option is selected, the default value (entire file) is assumed. If S= alone is specified, the T= default (end of file) is assumed. If T= alone is specified, the S= default (beginning of file) is assumed.

LIST Utility Interactive Mode Subcommands

With the exception of QUIT, all interactive subcommands will prompt the user for further input if he simply keys in the subcommand letter followed by a carriage return. However, entering the subcommand parameters on the same line as the subcommand letter eliminates further prompting for that subcommand. The folowing subcommands are provided:

>HELP

Causes the following list of available subcommands to be displayed:

```
Legal options in interactive mode are:
    'H'    - Define heading
    'F'    - Define nonstandard row and column counts
    'L'    - Put line numbers on listing output
    'D'    - Display (list) a range of logical records
    'QUIT' - To quit
```

>H <string of up to 60 characters>

The heading subcommand is used to specify a new heading. If only H is entered, the user is prompted with the following line:

```
Enter heading - 60 characters maximum:
```

>F nn,mm

The format subcommand sets the row count (nn) and column count (mm) for the listing page. nn and mm can be up to six decimal digits. If only F is entered, the user is prompted with the following line:

```
Enter row and column counts separated by comma:
```

```
>L nn
```

The Line subcommand specifies line numbers for the listing. The beginning line number is nn, which must be greater than 0 and may be up to six decimal digits long. If only L is entered, the user is prompted with the following line:

```
Enter base line number:
```

```
>D nn,mm
```

The Display subcommand causes a range of lines from line number nn through line number mm to be listed. If only nn is specified, only line nn is displayed. If nn is greater than mm or greater than the last line in the file, then no lines are displayed. The smallest allowable value for nn is 1, which corresponds to the first record (line) in the file. If only D is entered, the user is prompted with the following line:

```
Enter starting,ending line numbers:
```

```
>QUIT
```

This subcommand causes the list utility to terminate.

List Utility Messages

The VERSAdos Messages Reference Manual contains general error messages that may be output by the LIST utility. In addition to these messages, several special-purpose messages may be output by the LIST utility.

If the output field of the command line is a file that already exists, the following message is displayed:

```
Output file exists - OK to overwrite (Y/N) ?
```

The user can enter Y(CR) to overwrite the existing file. Any other response will terminate LIST.

If the I (interactive) option is selected, all other normally valid options which were specified on the command line are ignored. Any option which could have been specified on the command line can be invoked in interactive mode using subcommands. If any valid options are specified along with the I option on the command line, the following message is displayed:

```
Only 'I' option processed
```

Processing will then continue as if only the I option had been selected.

If a syntax error is detected while processing the D (Display) subcommand line, the following message is displayed:

```
Enter range of lines as follows:
'D S,T' to list from line S to line T
A leading '$' implies hex input
S and T must be larger than 0
```

If a syntax error is detected while processing the F (Format) subcommand line, the following message is displayed:

```
Enter row and column counts as follows:
'R,C' where R is the row count per page
and C is the column count.  R must be
greater than 9.  C must be greater than 0.
```

The user must then correctly enter the row and column counts before proceeding with any other interactive commands.

MBLM

4.2.14 Build Load Module Utility (MBLM)

The Build Load Module utility can be used to create a load module from object code produced by the M68000 Family Cross Macro Assembler (refer to manual M68KXASM). For transporting ease, the cross assembler output is in Motorola S-record form, which cannot be transformed directly by the linkage editor into a load module. The module produced by executing MBLM is a loadable object file (of .LO extension), which can be called for loading and execution as any VERSAdos utility is invoked, by simply naming the file. MBLM is functional only with VERSAdos for EXORmacs and VME/10 systems.

When a VERSAdos utility is invoked, a user task is created. The information needed for the loader information block of the .LO file is obtained through an interactive dialog with the MBLM utility.

As a module is built, a pass is executed for each memory segment to be built. The first pass searches for S-records which lie in the bounds of the first segment. The second pass searches for S-records which lie in the bounds of the second segment, and so on. Consequently, S-records in the input files need not be in a particular order.

Memory space required is equivalent to 5K (the size of MBLM), plus space for the largest specified segment. Since MBLM dynamically allocates work space for a buffer work space segment, the required memory space need not be contiguous.

MBLM Command Syntax

```
MBLM <input field>[,<output field>]
```

where:

input field	May be from one to four file descriptors (minimum of file name field required -- see paragraph 1.4) separated by slash (/) characters as filename1/filename2, etc.
output field	May be any file descriptor. If no output field is specified, the command defaults to the last specified input field file descriptor, and supplies an .LO extension. If an existing file descriptor is specified in the output field, the user is given the choice of re-using the descriptor or terminating.

Build Load Module Utility Example

In the following example, values are shown for the task control parameter block (and loader information block) and memory segment parameters, which can be entered by the user as a general means of creating a load module. In addition, brief explanations of the various parameters are given.

<u>Task Name</u> - A 4-character alphanumeric string of the user's choice, beginning with an alphabetic. Serves as identification for inter-task communication.

Task Session - A 4-digit hexadecimal number that serves as an extension to the task name to uniquely identify a task and facilitate protection of inter-task directives issued by the executive.

Task Priority - A 2-digit hexadecimal number which enables a user to assign relative task importance. The task priority becomes a factor in the allocation of system resources.

Task Attributes - A 4-digit hexadecimal number which identifies the requested task as being a system task, a memory resident task, a combination of both, or neither.

8000 = system
4000 = memory resident
0000 = neither

Task Options - A 4-digit hexadecimal number.

8000 = monitor specified
4000 = monitor propagated

Task Entry Point - The address in hexadecimal of the task entry point.

Command Line Address - An address in hexadecimal of a read/write buffer into which the loader will move the command line characters.

Command Line Length - A hexadecimal value may be entered here if some length less than the VERSAdos 160-character maximum is desired.

Segment Name - A 4-character alphanumeric string. After parameters for the required number of memory segments have been entered, entering a carriage return alone or a slash and a carriage return instead of a segment name will begin the building of the load module. A maximum of four segments is allowed.

Segment Attributes - A 4-digit hexadecimal number specifying the use of the corresponding segment:

4000 = read only
2000 = locally shareable
1000 = globally shareable

Segment Logical Address - A 4-digit hexadecimal address at which the segment will be located in the target task segment table. The address where search for S-records begins.

Segment Length - A 4-digit hexadecimal number giving the length of a segment in multiples of 256 bytes.

```
=MBLM BLMNEW.SA/PATCH.SA,XXXX.LO
** MACS BUILD LOAD MODULE - VER x.xx**
ENTER TASK NAME (TTTT)> XXXX
ENTER TASK SESSION>0                             Task Control Block Parameters
ENTER TASK PRIORITY>$00
ENTER TASK ATTRIBUTES>$0
ENTER TASK OPTIONS>$4000
ENTER TASK ENTRY POINT>$0
ENTER COMMAND LINE ADDRESS>$0000                 Loader Information Block Data
ENTER COMMAND LINE LENGTH>$00
ENTER SEGMENT NAME>SEG1
ENTER SEGMENT ATTRIBUTES>$0000
ENTER SEGMENT LOGICAL ADDRESS>$0                 Segment 1 Parameters
ENTER SEGMENT LENGTH>$1000 (or as required)
ENTER SEGMENT NAME>SEG2
ENTER SEGMENT ATTRIBUTES>$0                      Segment 2 Parameters
ENTER SEGMENT LOGICAL ADDRESS>$2000
ENTER SEGMENT LENGTH>$400 (or as required)
ENTER SEGMENT NAME>(CR)                 (Termination of command input and resumption
                                         of execution.   Could also be /(CR)).
** ERROR ** DUPLICATE FILE NAME **      (The file XXXX.LO specified  in  the  output
REUSE FILE (Y/N)? Y                      field on the command line already exists).
** THATS ALL FOLKS **
=
```

MIGR

The Migrate utility allows any ASCII file on an MDOS diskette to be copied to a VERSAdos file. No data conversion is performed during the copy. Binary files cannot be copied using this command.

MIGR Command Syntax

```
    MIGR <#FDnn>/<filename1>[,<filename2>]
```

where:

#FDnn	Is the floppy disk drive unit on which the MDOS diskette is mounted.
filename1	Is the input file name and is the name of the MDOS ASCII file to be copied. This name must be expressed in MDOS format which consists of the following fields: FILENAME = One to eight alphanumeric characters (first character must be alphabetic). Alphabetic characters are uppercase in all MDOS file name fields. SUFFIX (optional) = One or two alphanumeric characters (first character must be alphabetic). If a suffix is prescribed, it must be preceded by the period (.) field delimiter. If a suffix is not prescribed, the default suffix, source ASCII (.SA), is used. LOGICAL UNIT NUMBER = (not required).
filename2	Is the output field and consists of the name VERSAdos will assign to the copied MDOS ASCII file. This name must comply with the rules for VERSAdos file names described in paragraph 1.5. Except for the file name and extension fields, default values for permitted null fields in the output file name will be those established at session initiation. Defaults for the output file name and extension fields are the MDOS input file name and suffix, respectively.

MIGR Command Execution

Except in the case where an output file exists under the name specified on the command line, copying of the MDOS ASCII record proceeds automatically after the migrate command is properly entered. Where the output file name specified already exists, overwrite of the file may be requested by entering a "Y" in response to the following display:

```
    FILE EXISTS - OK TO OVERWRITE (Y/N)?
```

Completed execution of the command is indicated by the following message:

```
    MIGRATION VERSION '<version nbr>' COMPLETED
    =
```

Migrate Utility Examples

```
    =MIGR #FD01/TESTFILE
```

The MDOS file TESTFILE.SA on device #FD01 will be copied to the VERSAdos default output volume and named TESTFILE.SA.

```
    =MIGR #FD01/NAMEFILE,TESTFILE
```

The MDOS file NAMEFILE.SA on device #FD01 will be copied to the VERSAdos default output volume and named TESTFILE.SA.

```
    =MIGR #FD01/NAMEFILE,VOL0:1.CTLG0001.TESTFILE.AL
```

The MDOS file NAMEFILE.SA on device #FD01 will be copied to the specified VERSAdos output volume and catalog. The full file name will be VOL0:01.CTLG0001.TESTFILE.AL.

MOUNT

Recent advances in disk storage technology have produced hard and floppy disk drives in a wide variety of physical configurations with very high storage capacities. Many of these drives -- floppies in particular -- support many different media formats. The MOUNT utility, in conjunction with the INIT and DISMOUNT utilities, enables the user to access disks of different formats without a re-SYSGEN of VERSAdos and, in most cases, without knowledge of the media format of a particular disk or of the characteristics of the drive and controller that are being used.

The MOUNT utility has several modes of operation, briefly described as follows:

- Disks containing VERSAdos file directories are mounted with the invoking command line being the only required user input. All such disks are referred to as VERSAdos disks.

- Disks containing VERSAdos directories that were initialized with older versions of the INIT utility can be mounted with a minimum of user interaction when first mounted and may be automatically mounted thereafter.

- Disks not containing a VERSAdos file directory (foreign) that are used for user application-dependent data storage may be mounted with user-specified media format information. This information is solicited interactively with user-oriented input and output.

- When executed from user 0 with the MOUNT Drive (D) option, the user may change the necessary configuration data to replace a drive with a different drive type than was specified in the VERSAdos SYSGEN.

The MOUNT utility is not functional for EXORmacs.

NOTE: The MOUNT utility is neither necessary nor allowed for disks connected to the IPC controllers -- i.e., VERSAmodule Floppy Disk Controller (FDC), M68KVM20, and VERSAmodule Universal Disk Controller (UDC), M68KVM21.

MOUNT Command Syntax

```
    MOUNT <input field>[;<option>]
```

where:

input field	Is a disk device mnemonic of the form #HDcd (Hard Disk) or #FDcd (Floppy Disk) where c specifies a SYSGEN-specified controller number d specifies a SYSGEN-specified drive number
option	D specifies that MOUNT Drive mode is to be entered. This option may be specified only when the utility is executed by logon user 0.

Mounting VERSAdos Disks - Automatic Mode

The following steps should be used to mount a disk initialized with a version of the INIT utility dated after January 16, 1983. (For the revision date, refer to the message displayed when INIT is invoked.)

- Set the drive to online status. For floppies, this is done by inserting the diskette and closing the drive door. For hard disks, the procedure is drive-dependent. Consult the manufacturer's instructions.

- Invoke the MOUNT utility with no options, as described earlier.

The output of the utility is shown in the following examples. The volume name of the disk is displayed with the MOUNT confirmation message. The total number of VERSAdos sectors indicates the maximum capacity of the disk.

```
=MOUNT #FD02
MOUNT Version xxxxxx x
TEST has been mounted
Total Vdos sectors                        2552

=MOUNT #FD22
MOUNT Version xxxxxx x
SYS2 has been mounted
Total Vdos sectors                        1001
```

NOTES:

- If the logon user number is not user 0 and does not match the user number of the disk being mounted, the following message will be displayed:

  ```
  Only the owner of the disk or user zero may mount
  ```

- If the configuration area of the disk becomes damaged, one of the following error messages may occur:

  ```
  Configuration Error Code $xx

  The disk cannot be accessed properly with the given configuration
  ```

  If this condition occurs, the disk may be accessed by mounting a good disk having the same configuration and replacing it with the damaged disk without executing DISMOUNT in between. The user may then attempt to restore the damaged area using the DUMP and/or REPAIR utility.

Mounting VERSAdos Disks - User-Assisted Mode

Disks initialized with earlier versions of the INIT utility do not contain media format information. All such disks consist of hard disk fixed media, hard disk removable media, and single- or double-sided floppy media initialized on an IPC controller. At present, only the floppy disk media is portable to a non-IPC controller.

The following steps should be used to mount a disk initialized with a version of the INIT utility dated prior to November 12, 1982:

- Put the drive online. For floppies, this is done by inserting the diskette and closing the drive door. For hard disks, the procedure is drive-dependent. Consult the manufacturer's instructions.
- Invoke the MOUNT utility with no options, as described. The user will then be prompted with the following message:

  ```
  Is this an IPC format disk (CR or N-No,S-Single sided,D-Double sided)?
  ```

  If the user responds with a (CR) or N, the utility will enter interactive mode as described for mounting foreign disks. Otherwise, the user identifies the disk as single- or double-sided. The volume name and total number of VERSAdos sectors are then displayed as in automatic mode. An invalid response to the above prompt will cause the same prompt to be reissued.

Prior to terminating, the MOUNT utility attempts to write to sector 0 of the disk a code that will allow automatic mode to be used for all subsequent mounts of that disk. If the disk is write protected, the following message will be displayed:

```
The VID cannot be modified for automatic mount
```

MOUNT Utility Examples - User-Assisted Mode

Example 1

```
=MOUNT #FD22
MOUNT Version xxxxxx x
Is this an IPC format disk (CR or N-No,S-Single sided,D-Double sided)?X (Invalid
                                                                        response)
Is this an IPC format disk (CR or N-No,S-Single sided,D-Double sided)(CR)
Data Density of media (S-single,D-double) D
S-Single or D-Double sided diskette       D
F-Floppy diskette or H-Hard disk          F
Vdos sector size                              256
Total Vdos sectors                            3991
Physical sectors/track on media(8 bits)       26
Number of heads on drive(8 bits)              2
Number of cylinders on media(16 bits)         77
Interleave factor on media(8 bits)            13
Bytes/physical sector on media(16 bits)       256
No configuration data is available for this disk

Data Density of media (S-single,D-double) D > (CR)
S-Single or D-Double sided diskette       D > (CR)
Physical sectors/track on media (8 bits)    26 > (CR)
Number of cylinders on media(16 bits)       77 > Q
```

Example 2

```
=MOUNT #FD22
MOUNT Version xxxxxx x
Is this an IPC format disk (CR or N-No,S-Single sided,D-Double sided)? S
XYZ  has been mounted
Total Vdos sectors                   1001
=DISMOUNT #FD22
DISMOUNT Version xxxxxx x
=MOUNT #FD22
MOUNT Version xxxxxx x
XYZ  has been mounted
Total Vdos sectors                   1001
```

Example 3

```
=MOUNT #FD22
MOUNT Version xxxxxx x
Is this an IPC format disk (CR or N-No,S-Single sided,D-Double sided)? S
XYZ  has been mounted
Total Vdos sectors                   1001
MOUNT $0003 $100000E4 FROM IOS ** WRITE PROTECTED DEVICE
 CMD=WRITE OPT=$6000 LU=1 DEVICE=XYZ
The VID cannot be modified for automatic mount
```

Mounting Foreign Disks

Follow the steps below to mount a foreign disk.

- Set the drive to online status. For floppies, this is done by inserting the diskette and closing the drive door. For hard disks, the procedure is drive-dependent. Consult the manufacturer's instructions.

- Invoke the MOUNT utility with no options, as described. The utility then issues the following prompt:

```
No configuration data is available for this disk
```

- The utility enters interactive mode to allow changes to the media-related configuration attributes and parameters. The operation of interactive mode is described later.

- Input a C to exit interactive mode. A configure request is then issued to the driver with the user-specified data. If an error is returned from the configure request, the utility displays the following:

```
Configuration Error Code $xx
```

  where xx is a device-dependent error code describing the first error detected in the configuration data. Refer to the VERSAdos Data Management Services and Program Loader User's Manual, RMS68KIO, for descriptions of the various Configuration Error Codes.

- After a successful configuration has been performed, the new configuration is validated by attempting to read track 0-1 boundary sectors and the last sector of the disk. If either attempt returns an I/O error, the error is displayed followed by the message:

  The disk cannot be accessed properly with the given configuration

  Interactive mode is then reentered to allow changes to be made to the configuration data.

MOUNT Utility Example - Foreign Disk

```
=MOUNT #FD02
MOUNT Version xxxxxx x
Data Density of media (S-single,D-double) S
Track Density of media (S-single,D-double)S
S-Single or D-Double sided diskette       S
Media format (M-Motorola,I-IBM standard)  I
F-Floppy diskette or H-Hard disk          F
Vdos Sector size                                256
Total Vdos sectors                              640
Physical sectors/track on media(8 bits)         16
Number of heads on drive(8 bits)                2
Number of cylinders on media(16 bits)           80
Interleave factor on media(8 bits)              1
Bytes/physical sector on media(16 bits)         128
Max. number of cylinders on drive(16 bits)      80
Precomp. cylinder number on drive(16 bits)      40

No configuration data is available for this disk

S-Single or D-Double sided diskette       S >(CR)
Media format (M-Motorola,I-IBM standard)  I >(CR)
Physical sectors/track on media(8 bits)         16 >(CR)
Number of cylinders on media(16 bits)           80 >(CR)
Interleave factor on media(8 bits)              1 >(CR)
Bytes/physical sector on media (16 bits)        128 > 0   (Invalid value)
Data Density of media (S-single,D-double) S > C
Configuration Error Code $0044
Data Density of media (S-single,D-double) S >(CR)
Track Density of media (S-single,D-double)S >(CR)
S-Single or D-Double sided diskette       S >(CR)
Media format (M-Motorola,I-IBM standard)  I >(CR)
Physical sectors/track on media (8 bits)       16 >(CR)
Number of cylinders on media(16 bits)          80 > 100   (Too many cylinders)
Interleave factor on media(8 bits)             1 >(CR)
Bytes/physical sector on media(16 bits)        0 > 128
Data Density of media (S-single,D-double) S > C
MOUNT $0003 $100000E7 FROM IOS ** INVALID SECTOR ADDRESS
 CMD= READ OPT=$6000 LU=1 DEVICE=FD02

The disk cannot be accessed properly with the given configuration
```

```
Total Vdos sectors                          800
Data Density of media (S-single,D-double) S > D
Track Density of media (S-single,D-double)S > D
S-Single or D-Double sided diskette       S > D
Media format (M-Motorola,I-IBM standard)  I >(CR)
Physical sectors/track on media(8 bits)     16 >(CR)
Number of cylinders on media(16 bits)       100 > 80
Interleave factor on media(8 bits)          1 >(CR)
Bytes/physical sector on media(16 bits)     128 > 256
Data Density of media (S-single,D-double) D > C
FD02 has been mounted                                  (Successful mount)
Total Vdos sectors                          2552
```

MOUNT Drive Mode (D Option)

Use of the MOUNT Drive mode is restricted to logon user 0. If the D option is specified from a non-zero user number, the following message will be displayed:

```
Only user zero can use D (mount drive) option
```

The MOUNT Drive mode allows permanent changes to be made to all configuration attributes that are supported by the driver for the device. Changing the drive-related information allows a different type of drive to be used without requiring a new SYSGEN. The default values for the media attributes and parameters may also be changed without running a new SYSGEN.

The D option causes interactive mode to be entered so the user can view and change any of the configuration attributes and parameters. When the user exits interactive mode by entering C, the utility issues a configure command to the driver to set the current configuration as specified, followed by a configure default command to set the default configuration to match. If either configuration attempt results in an error, the following message is displayed:

```
Configuration Error Code $xx
```

where xx is a device-dependent error code describing the first error detected in the configuration data. Refer to the VERSAdos Data Management Services and Program Loader User's Manual, RMS68KIO, for descriptions of the various Configuration Error Codes. After the error is displayed, interactive mode is reentered.

CAUTION

DUE TO IMPLEMENTATION RESTRICTIONS IN VERSAdos 4.2, THE DEFAULT VALUES FOR MEDIA-RELATED ATTRIBUTES AND PARAMETERS MUST BE SET TO ACCOMMODATE THE LARGEST CAPACITY DISK THAT WILL BE USED. IF A DISK IS USED THAT HAS A TOTAL NUMBER OF VERSAdos SECTORS LARGER THAN THAT OF THE DEFAULT DISK TYPE, FILES MAY BE READ AND WRITTEN NORMALLY, BUT DELETING FILES COULD RESULT IN A DAMAGED DIRECTORY.

MOUNT Utility Examples - MOUNT Drive Mode

Example 1

```
=MOUNT #FD03;D
MOUNT Version xxxxxx x
Data Density of media (S-single,D-double) D
Track Density of media (S-single,D-double)D
S-Single or D-Double sided diskette       D
Media format (M-Motorola,I-IBM standard)  I
F-Floppy diskette or H-Hard disk          F
Vdos sector size                             256
Total Vdos size                              2552
Physical sectors/track on media(8 bits)      16
Number of heads on drive(8 bits)             2
Number of cylinders on media(16 bits)        80
Interleave factor on media(8 bits)           1
Bytes/physical sector on media(16 bits)      256
Max.  number of cylinders on drive(16 bits)   80
Precomp.  cylinder number on drive(16 bits)   40
Data Density of media (S-single,D-double) D >(CR)
Track Density of media (S-single,D-double)D > S
S-Single or D-Double sided diskette       D > S
Media format (M-Motorola,I-IBM standard)  I >(CR)
F-Floppy diskette or H-Hard disk          F >(CR)
Vdos sector size                             256 >(CR)
Physical sectors/track on media(8 bits)      16 >(CR)
Number of heads on drive(8 bits)             2 > 1
Number of cylinders on media(16 bits)        80 >(CR)
Interleave factor on media(8 bits)           1 >(CR)
Bytes/physical sector on media(16 bits)      256 >(CR)
Max.  number of cylinders on drive(16 bits)   80 >(CR)
Precomp.  cylinder number on drive(16 bits)   40 >(CR)
Data Density of media (S-single,D-double) D > C           (Valid configuration)
```

Example 2

```
=MOUNT #FD03;D
MOUNT Version xxxxxx x
Data Density of media (S-single,D-double) S
Track Density of media (S-single,D-double)S
S-Single or D-Double sided diskette       S
Media format (M-Motorola,I-IBM standard)  I
F-Floppy diskette or H-Hard disk          F
Vdos sector size                             256
Total Vdos sectors                           1280
Physical sectors/track on media(8 bits)      16
Number of heads on drive(8 bits)             1
Number of cylinders on media(16 bits)        80
Interleave factor on media(8 bits)           1
Bytes/physical sector on media(16 bits)      256
```

```
Max. number of cylinders on drive(16 bits)    80
Precomp. cylinder number on drive(16 bits)    40
Data Density of media (S-single,D-double) S >(CR)
Track Density of media (S-single,D-double)S > D
S-Single or D-Double sided diskette       S > D
Media format (M-Motorola,I-IBM standard)  I >(CR)
F-Floppy diskette or H-Hard disk          F > H  (Double-sided hard invalid)
Vdos sector size                            256 > C
Configuration Error Code $0042
Data Density of media (S-single,D-double) S > Q
```

Interactive Mode

Upon entry to interactive mode, the current values of all configuration attributes and parameters for the specified drive are displayed. For a detailed description of disk configuration attributes and parameters, consult the VERSAdos Data Management and Program Loader User's Manual, RMS68KIO.

The user is prompted for changes to the current values of all configuration attributes. The prompt for each attribute contains a description of the attribute and its current value in mnemonic form. The user may change the attribute by entering one of the mnemonics given in the prompt, or only a (CR) may be entered. If only a (CR) is entered, the current value of the attribute is retained and the prompt for the next attribute is issued. When neither a (CR) nor a valid mnemonic is entered, the input is rejected and the same prompt is reissued to obtain new data.

Next, the user is prompted for changes to all configuration parameters in similar fashion. The prompt for each parameter contains a description of the parameter and its current value in decimal. The user may enter a new value for the parameter, or only a (CR). If a new value is entered, it is assumed to be decimal unless preceded by a '$' for hexadecimal. If an invalid digit for the specified base is entered or the value contains more significant bits than the maximum number specified in the prompt, the same prompt is reissued to obtain new data. If only a (CR) is input, the current value for the parameter is retained and the prompt for the next parameter is issued.

After the user has been prompted for all parameters, the user is prompted again for changes to the configuration attributes.

Interactive mode may be exited only by entering one of the following in response to any of the prompts:

'Q' or 'q' - Exit interactive mode and terminate the utility without mounting the disk or changing the current or default configurations.

'C' or 'c' - Exit interactive mode and attempt to configure the drive with the user-specified configuration data.

MOUNT Utility Example - Interactive Mode

```
Data Density of media (S-single,D-double) D > X    (Invalid attribute value)
Data Density of media (S-single,D-double) D >(CR)
Track Density of media (S-single,D-double)D > S    (Valid attribute value)
S-Single or D-Double sided diskette       D >(CR)
Media format (M-Motorola,I-IBM standard)  I >(CR)
Physical sectors/track on media(8 bits)     16 > 257    (Number too large)
Physical sectors/track on media(8 bits)     16 > 2A0    (Invalid decimal digit)
Physical sectors/track on media(8 bits)     16 > $8Z    (Invalid hex digit)
Physical sectors/track on media(8 bits)     16 > $F
Number of cylinders on media(16 bits)       80 > 25
Interleave factor on media(8 bits)          1 >(CR)
Bytes/physical sector on media(16 bits)     256 > 35768
Data Density of media (S-single,D-double) D >(CR)
Track Density of media (S-single,D-double)S >(CR)
S-Single or D-Double sided diskette       D >(CR)
Media format (M-Motorola,I-IBM standard)  I >(CR)
Physical sectors/track on media(8 bits)     15 > 16
Number of cylinders on media(16 bits)       25 >(CR)
Interleave factor on media(8 bits)          1 >(CR)
Bytes/physical sector on media(16 bits)     35768 > 65539  (Number too large)
Bytes/physical sector on media(16 bits)     35768 > 16
Data Density of media (S-single,D-double) D > Q
```

PATCH

The PATCH utility allows disk-resident files in executable form (load module files) to be changed without need for editing and reassembling the corresponding source files. Changes are accomplished by entering hexadecimal or ASCII literal information, or assembly language statements, into the appropriate memory locations. The versatility of the utility is enhanced by a command structure that offers several means by which memory location contents are displayed and changed. Checksums are also calculated over corresponding affected and unaffected memory ranges to aid verification of change accuracy. The utility may be used to modify any contiguous load module file. An example of PATCH use is shown in Figure 4-3.

If the input to the PATCH utility is a chain file, the user may comment the chain file for documentation purposes by putting an asterisk (*) in the first column of any documentation input.

PATCH Command Syntax

```
    PATCH <filename>
```

where:

filename	Is the name of the file to be changed. Extension .LO is default.

The PATCH utility responds to the command entry by displaying a greater than (>) symbol, indicating that the user may either exit the utility or enter the display/change mode. Following the prompt, any of the subcommands described below may be entered.

Quit Subcommand Syntax (QUIT)

```
    >Q[UIT]
```

where:

Q or QUIT	Is the subcommand for quitting the PATCH utility and returning control to VERSAdos.

Before returning to VERSAdos, PATCH displays the following message:

```
    CHECKSUM = xxxxyyyy
```

where xxxx and yyyy are each the sum of the values in all modified bytes in the patched load module; xxxx is the sum of the values as they were before modification, and yyyy is the sum of the values after modification. If the originator of a patch provides this checksum along with the patch, anyone who repeats the patch may verify that he has performed it correctly by checking that his checksum value matches the original.

Set Address Offset Subcommand Syntax (O)

```
>O <offset>
```

where:

O	Is the Set Address Offset subcommand mnemonic.
offset	Is up to eight digits (hexadecimal assumed; leading $ is optional).

The Set Address Offset subcommand is used to set or change an address offset or to display the current offset value. An offset supplied by this subcommand is automatically added to the address specified on the display/change mode entry command line (described below) before execution of any subsequent display/change mode command. For change or debugging ease, displayed addresses will include both the user requested address and the offset address.

Use of an offset is convenient when debugging a routine whose listing addresses do not match the logical machine addresses. An example would be a subroutine assembled with a listing address of zero and a logical starting address (found from the link map) of $546. Debugging this routine would be facilitated by setting an offset address of $546, using the command "O 546". Now, for example, keying in the Patch subcommand "M 100" opens the byte at location $100 plus $546, or $646. The displayed address will include the listing address of $100 and the actual address of $646. The dollar sign ($) is optional when entering the offset value, since hexadecimal is automatically assumed.

Once set, an offset remains effective until changed by execution of another Set Address Offset command. To obtain display of the current value, simply key in the command "O". The offset value is not changed.

Display/Change Mode Entry Subcommand Syntax (M)

```
>M <hhhh>[;DI]
```

where:

M	Is the Display/Change Mode Entry subcommand mnemonic.
hhhh	Are hexadecimal digits representing the address of a memory location whose contents are to be displayed and/or changed. The dollar sign ($) preceding hexadecimal data is optional.
DI	Is required when invoking the assembler and disassembler option.

Display/Change Mode without Assembly/Disassembly

Following entry of the M subcommand, the specified location address is displayed on the next screen line followed, after a space, by the location contents in hexadecimal and, again after a space, by the ASCII character representing the contents, if displayable. The latter display is provided as an aid in visualization when text such as a message is being modified or added to a program. Undisplayable codes are represented by a single quotation mark, a period, and another single quotation mark: '.'. The user should note that the cursor remains on the same line a space apart from the preceding character. Any of the display/change modes may now be used, as follows:

Mode 1 - Sequential Display/Change. Hexadecimal Entry with Continue.

    <h...h>

where: h...h is a string of up to 50 hexadecimal digits to be entered into sequentially higher memory locations starting at the address displayed at left. Execution is initiated automatically by entering the fiftieth digit or, at any point in the string, by a (CR). The address one higher than that of the last changed location is displayed on the next line of the screen. Further use of the display/change mode can now be made.

Mode 2 - Sequential Display/Change. Hexadecimal Entry with Return.

    <h...h>^

where: h...h is a string of up to 50 hexadecimal digits, and ^ is a circumflex (hex 5E). Action is similar to that of Mode 1 except that following execution, the address one lower than that of the first changed location is displayed on the next line. Further use of the display/change mode can now be made.

Mode 3 - Sequential Display/Change. ASCII Entry with Continue.

    '<c...c>'

where: 'c...c' is a string of up to 50 ASCII characters directly preceded by a single quotation mark (hex 27) and directly followed by a single quotation mark (hex 27). Action is similar to that of Mode 1 except that codes for the specified ASCII characters are entered into sequentially higher memory locations starting at the displayed address at left on the command line. In order to embed an apostrophe within a string, two consecutive apostrophes must be entered.

Mode 4 - Sequential Display. ASCII Entry with Return.

    '<c...c>'^

where: 'c...c' is a string of up to 50 ASCII characters directly preceded by a single quotation mark (hex 27) and directly followed by a single quotation mark (hex 27), and ^ is a circumflex (hex 5E). Action is similar to that of Mode 3 except that following execution, the address one lower than that of the first changed location is displayed on the next line. Further use of the display/change mode can now be made.

Mode 5 - Sequential Display. Single Step or Continuous Scroll.

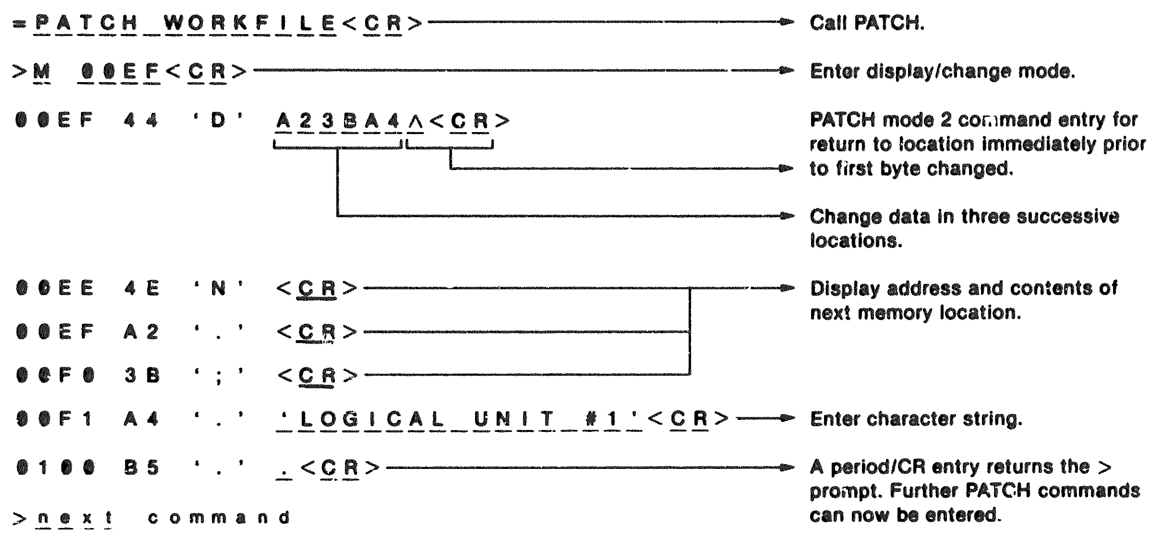
    (CR)

A carriage return keystroke can be used alone to display the address and contents of the next higher location in hexadecimal and ASCII representation on the next screen line. For quick review of the contents of a memory range, the return key can be held down to cause a scroll upward display of addresses and contents.

Mode 6 - PATCH Utility Prompt Recapture.

.(CR)

Entering a period followed by a carriage return at any time while in the display/change mode causes the PATCH prompt to be displayed on the next screen line. The display/change mode can now be reentered or the QUIT command used to exit the utility.



```
=PATCH WORKFILE<CR>
>M 00EF<CR>
00EF  44  'D'  A23BA4^<CR>
00EE  4E  'N'  <CR>
00EF  A2  '.'  <CR>
00F0  3B  ';'  <CR>
00F1  A4  '.'  'LOGICAL UNIT #1'<CR>
0100  B5  '.'  .<CR>
>next  command
```

Call PATCH.
Enter display/change mode.
PATCH mode 2 command entry for return to location immediately prior to first byte changed.
Change data in three successive locations.
Display address and contents of next memory location.
Enter character string.
A period/CR entry returns the > prompt. Further PATCH commands can now be entered.

FIGURE 4-3. PATCH Utility Example

Display/Change Mode with Assembly/Disassembly

When the optional ;DI argument is included on the M subcommand line, assembly language can be used in a patch. The code at the specified address is disassembled and displayed, and a prompt appears to allow entry of replacement code.

An offset may be utilized with this instruction as follows:

a. It is automatically added to the address in the subcommand M <hhhh>;DI.

b. It may be added to any address within an instruction by including a '+O' in the instruction. For example, to patch the following code in at location $300:

```
        LEA     TABLE,A0
BLIP    ADD.L   #7,A0
        CMP.B   #$0D,(A0)
        BNE     BLIP
```

Assuming TABLE is at address $50, the following code could be entered to PATCH:

```
O 300                      (Set the offset to $300.)
0;DI                       (Start modifying at the offset.)
LEA       $50(PC),A0       (Note (PC) specified explicitly.)
ADDQ.L    #7,A0            (Note quick form (Q) specified explicitly.)
CMP.B     #$0D,(A0)
BNE       0+4              (Branch to the value of offset +4.)
```

Because the offset was used in the BNE instruction, the code may be entered anywhere in memory just by using a different offset.

The following points should be considered when using the assembly/disassembly option:

Disassembler:

a. If what lies at a particular location in memory is a valid form of some instruction, the disassembler will return that instruction. This may happen even if the location contains 'data'. If it is not a valid instruction, the disassembler returns a DC.W $xxxx (always hex), regardless of whether the data was defined in bytes, words, or long words.

b. The disassembler makes the decision whether to represent a numeric value within a disassembled instruction in decimal or hex based upon whether or not it thinks the value represents an address. Would-be addresses are displayed in hex; everything else is in decimal. For example,

```
MOVE.L #$1234,$5678
```

disassembles to:

```
21FC000012345678          MOVE.L #4660,$5678
```

Note that the hexadecimal form of decimal 4660 can still be readily found by looking at the hex for the instruction.

c. The disassembler returns BT for BRA (BRA and BT are both valid mnemonics for the same opcode). Similarly, DBF is returned for DBRA.

d. For some instructions, there is more than one assembly language equivalent. The disassembler may choose a form different from the one which generated the code originally. For example,

```
    MOVE.L    #10,A1
and MOVEI.L   #10,A1
```

generate the same code. In this case, the former would be chosen as the disassembled form.

Assembler:

a. The one-line assembler is not symbolic, and does not support macros or structured constructs. Instructions such as BTSTW and BCLRW, which existed in those MC68000 processors with R9M and prior mask sets but do not exist in current mask sets, are not supported. Users not dealing with software written for the experimental versions of the MC68000 need not be concerned with this restriction.

b. If the "quick" form of an instruction is wanted, it must be specified. For example,

```
MOVE.L    #3,D0          is a 6-byte instruction, whereas
MOVEQ.L   #3,D0          is a 2-byte instruction.
```

c. If the PC-relative addressing mode is wanted, it must be specified. For example,

```
LEA       $3F0,A0        encodes $3F0 as an absolute address, whereas
LEA       $3F0(PC),A0    encodes $3F0 as a PC-relative value.
```

d. Operand expressions may include references to the current location (character "*") and the current PATCH offset (letter "O"). Numbers may be expressed in hex (precede with a "$" character) or decimal, which is the default in all cases. ASCII strings are also supported. The only valid operators in an expression are + and -. The following are examples of valid statements for the one-line assembler in PATCH:

```
BRA.S     *+$24
MOVE.L    #255-32,$1F0+O
CMP.W     #'SA',24(A0)
```

e. The offset is NEVER applied automatically to an address in an instruction. If you wish it applied, you must specify "+O".

f. The BHS and BLO instructions are not supported. Use BCC for BHS, and BCS for BLO.

g. Comments may be put at the end of a line, or on a line that begins with an asterisk.

h. The only pseudo-op accepted is DC.W, and the operand must fit into 16 bits.

PRTDUMP

The Print Dump utility is an adjunct to a system function commonly referred to as "post mortem dump", but more accurately termed "task dump". Task dump is called by the system when a task aborts and either the I option is set for the session (OPT I -- refer to Chapter 3) or the task dump attribute is set in the task's loader information block. (The task dump attribute is incorporated in a load module by linking with the linker's D attribute option set.) Task dump provides the capability which gives the user the choice of dumping (or not) his memory space to a file on disk for examination. All or portions of the dump file can then be displayed or printed by using the PRTDUMP utility.

When an abort occurs, a display of the contents of the processor registers will appear, followed by an interactive dialog:

a. TDMP: DO YOU WANT A DUMP? (Y/N)>

   Keying in an N returns control to VERSAdos; a Y causes a second and third display:

b. TDMP: DEFAULT DUMP FILE IS (VOLUME ID):(user no.).TASKDUMP.DUMPmmss.DU

   where:

volume ID	Is the user's default volume ID.
user no.	Is the user's logon number.
TASKDUMP	Is the default catalog name.
DUMPmmss	Is the default file name made up of DUMP plus the minutes and seconds of the system clock at time of abort.
.DU	Is the default file name extension.

c. TDMP: ENTER NEW FILENAME OR HIT CR >

   Use of a file descriptor for the dump file comprised of all default values of b. above is obtained by keying in a (CR), or the user may specify the desired values in any fields in the file descriptor.

d. TDMP: DUMP FILE IS (vol. ID):(user no.).(catalog).(filename).(extension)

   This is the descriptor under which the dump is created, and should be noted for later access.

<u>Using the Print Dump Utility</u>

Once the display in d. above is reached and the dump file descriptor noted, the utility can be invoked to print all or portions of the dump file.

PRTDUMP Command Syntax

```
PRTDUMP <filename>[,<device>][;<options>]
```

where:

filename — Is the dump file descriptor established in c. above. Note that except for the file name field (which is required) in the file descriptor, Print Dump will use the following default values for unspecified fields:

volume ID	= user's default volume ID
user number	= user's default (not logon) number
catalog	= TASKDUMP
extension	= .DU

All file descriptor fields, whether specified or obtained from default values, must match those of the dump file descriptor in order for the Print Dump utility to access the dump file.

device — Is #PR for printed output or # for output displayed on the user's logon terminal. If the device field is not specified, # is used.

options — May be one of the following:

A - Outputs all of the dump task's memory space plus the Task Control Block information (i.e., all the information retained by the dump).

R - Outputs all of the dump task's read/write memory space plus the Task Control Block information. Does not output memory segments with the read-only attribute.

If the A or R option is specified, the requirements of the command line are provided, and control is returned to VERSAdos.

No option - Specifying no options results in the Print Dump utility entering the interactive mode.

The Print Dump Interactive Mode

Display of the prompt PRTD:> indicates that the Print Dump interactive mode has been entered. Any of several interactive mode commands described below can now be used.

PRTD:>ALL	Provides the same output as the A option.
PRTD:>RAM	Provides the same output as the R option.
PRTD:>TCB	Outputs Task Control Block information.
PRTD:>SEGM <segname>	Outputs the contents of the specified segment.

where:

segname	Is written as one to four characters of which the first character must be alphabetic and the remaining characters alphanumeric. Alphabetic includes the characters "A" through "Z", "." (period), "_" (underscore), and "&" (ampersand). Alphanumeric includes all of those, plus "0" through "9" and "$" (dollar sign). Refer to the M68000 Family Linkage Editor User's Manual, M68KLINK, for more segment name information.

PRTD:>REGS	Outputs the contents of the dumped task's registers.
PRTD:>MD <address>[,<offset>][<sp><length>]	Outputs the contents of a block of memory.

where:

address	Is the point at which output is begun.
offset	Is a hexadecimal value that is added to <address>.
sp	Is a space and is required if a length is specified.
length	Is a hexadecimal value defining the size of the memory block to be output. If not specified, the default value 16 is used.

PRTD:>AD <addr1>[,<offset>][<sp><addr2>]	Outputs the contents of a block of memory.

where:

addr1	Is the address at which output is begun (subject to modification by any offset specified).
offset	Is a hexadecimal value that is added to <addr1> to determine the starting address.
sp	Is a space and is required if <addr2> is specified.
addr2	Is the address at which output is stopped. If not specified, 16 bytes are output.

PRTD:>OFFS <offset>	Specifies a hexadecimal value that is added to <address> or <addr1>, respectively, for all subsequent MD or AD command lines. This value is ignored if an offset is specified on an MD or AD command line.
PRTD:>OUTP <device>	Changes a previously specified output device to the device specified in the device field.
PRTD:>HELP	Prints a summary of command line format and interactive mode commands for PRTDUMP.
PRTD:>QUIT	Terminates execution of any PRTDUMP command and returns control to VERSAdos.

RENAME

4.2.19 Rename Utility (RENAME)

The RENAME utility is used to give a file a new name, a new catalog name, or both and, if desired, assign a new protect key. The user number may also be changed if RENAME is executed while logged on as user 0.

<u>RENAME Command Syntax</u>

```
    RENAME <input field>,<output field>[,<list device>]
```

where:

input field	May be any file name.
output field	May be any file name.
list device	May be any file name or device name. The default value is the session terminal. If a file is specified without extension, the default is .LS.

The "wildcard" asterisk (*) may be used as described under the COPY utility.

<u>RENAME Utility Examples</u>

<u>Example 1</u> -

```
    =RENAME  TEST.SA,PAYIN.SA(AABB)
```

The file name of the input field file and its file protection code (the public read code PPPP need not be specified) are changed to PAYIN.SA and AABB, respectively. The defaults are used for volume ID, user number, and catalog.

<u>Example 2</u> -

```
    =RENAME  WORK.SA(BBCC),USEIT.SA(PPPP)
```

The file name of the input field file and its file protection code are changed to USEIT.SA and PPPP (owner only can write; any user can read), respectively. The defaults are used for volume ID, user number, and catalog.

<u>Example 3</u> -

```
    =RENAME  *.SA,*.FT
```

Default values are used for the volume ID, user number, and catalog fields. Any file having the extension .SA is given the extension .FT.

<u>Example 4</u> -

```
    =RENAME  CAT1.*.*,PAY.*.*
```

The default value for the volume ID and user number fields are used. The catalog field for all files of catalog name CAT1 is changed to PAY.

Example 5 -

```
    =RENAME  *.*(AABB),(CCDD)
```

Default values are used for volume ID, user number, and catalog name. The protection field of those files belonging to the default user and having the value AABB is changed to the value CCDD. The output file name cannot be specified if only the protection codes are being changed.

Example 6 -

```
    =RENAME  TEST.SA(AAAA),(PPPP)
```

Default values are used for volume ID, user number, and catalog fields. The protection field of the file is changed to public write/read. The output file name cannot be specified if only the protection codes are being changed.

Example 7 -

```
    =RENAME  T*.*,A*.*
```

Default values are used for volume ID, user number, and catalog fields. All files belonging to the default user which have a file name beginning with T have that character changed to A.

Example 8 -

```
    =RENAME  TEST.SA,PAYIN.SA,LIST
```

The file name of the input field file is changed to PAYIN.SA. All displayed messages are directed to the file LIST.LS. Default values are used for volume ID, user number, and catalog fields.

REPAIR

4.2.20 Repair Utility (REPAIR)

The REPAIR utility detects errors in a VERSAdos disk or diskette which has bad disk control table information. Errors are found and identified and an attempt is made to correct errors with minimal loss of data. REPAIR may also be run to change disk contents or to dump sectors or disk structures. REPAIR can be run to process the entire volume or only a particular file on the volume. REPAIR may also be executed to recover deleted files, unless the files' Data Blocks (DB's) and File Allocation Blocks (FAB's) have been reallocated to other uses after the files were deleted.

REPAIR Command Syntax

```
REPAIR <device>[,#PRn][;<option>]
```

or

```
REPAIR <vol>:[<user no>][.<cat.filename.ext>][,#PRn][;<option>]
```

where:

device	Is a device mnemonic representing the device on which the disk resides. Volume repair is performed -- i.e., all the structures on the disk are checked and processed by REPAIR.
vol:user no	Is a volume name followed by a colon and optionally by a user number. Volume repair is done only if a <cat.filename.ext> is _not_ specified. If <vol>: only is specified (no <user no> or filename), the default <user no> is that assigned to the disk when initialized.
cat.filename.ext	Is an optional field consisting of a catalog name, a file name, and an extension, each preceded by a period. If specified, file repair is done. Only the SAT, primary block entry, and the file's DB's and FAB's, if any, are checked and processed. Other structures on the disk are assumed to be correct.
#PRn	Is the listing device; if omitted, default is # (display on CRT screen).
options	May be omitted for default processing in the interactive mode, or may be one of the following characters:

N - Non-Interactive Mode (NIM). In this mode, no disk structures are displayed, and the user cannot make changes unless an error is found since execution proceeds automatically. If an error is found, one of three actions is taken:

a. For a critical error, execution terminates. The user should then enter the interactive mode (default; no options) to make the appropriate correction.

b. The messages "ER: <problem>" and "R: <corrective action>" are displayed, followed by the request "REPAIR (Y/N)?".

  . The user may enter Y to obtain the indicated change.

  . The user may enter N to terminate or obtain display of an alternative course of action, according to the nature of the error.

c. The message "R: <corrective action>" is displayed, with the utility automatically making the indicated change.

C - Check only mode (not valid with R option). In this mode, the utility proceeds as in the non-interactive mode, displaying a message and terminating when an error is encountered. When both the C and N options are specified, the C option takes precedence. Any I/O error causes termination. Check only mode should be tried first if disk errors are suspected.

R - File recovery mode (not valid with the C option). In the file recovery mode, when only a volume name is specified on the REPAIR command line, the PDE of each deleted file is shown and the user is given the choice of asking the utility to attempt recovery or of skipping to the next file. When the optional field <user no> <.cat.file name.ext> is fully specified on the REPAIR command line, attempted recovery of the specified file is automatic. In such cases where deleted files of the same name exist (possible because the first character of the name of a file is zeroed on deletion), the utility offers the choices of skipping or of asking for a recovery attempt. If REPAIR with the R option is invoked for a file which is found (i.e., which has not been deleted), no action is taken.

(No option) - Default Processing - When no option is specified, the utility operates in the Interactive Mode (IM) under user control. All deleted files are skipped. Dumps of sectors or structures may be made in this mode. Considerable knowledge of VERSAdos disk and file structure is required to use this mode effectively; therefore, its use is not recommended for first-time users. If #PR is specified in the command line, it is ignored in this mode.

After the command line has been correctly entered, REPAIR identifies itself with the message:

```
VERSAdos REPAIR UTILITY - VERSION x
```

Suggestions for Using the REPAIR Utility

Four diagrams showing disk and file structure (Figures 4-4 through 4-7) and a list of field names are given in this section for the user's reference. It is recommended that the N option be used in the beginning until the disk and file structures are learned.

If allocation conflicts are detected on the disk and it is not clear to which other disk structure(s) the sectors have also been allocated, execution should be terminated and a note made about the Physical Sector Number (PSN) of sectors with allocation conflict. The utility should be run again in interactive mode, and the given sectors marked allocated (A $<psn>,n) in the Sector Allocation Table (SAT) during the UPDATE SAT loop, described later in this section. Then an error message will be given at the first reference to the same sector(s). It is then up to the user to determine to which structures the sector(s) actually belong. Note that REPAIR assumes that the first association is correct -- see following paragraph.

The utility will not recognize a space allocation conflict until a second structure tries to claim the same space. The assumption is made that the first association is correct and there is no allocation error because no information exists for structures not yet processed. If multiple allocation errors are suspected, the structures should be examined as they are displayed to discover such errors. The utility will normally give error messages about content errors when multiply allocated sectors do not belong to the disk structure at hand, but it is possible for the wrong sectors to pass the tests for the structure being examined.

Any severe disk problems should be analyzed by an adequate dump of the problem area. Since the main purpose of the utility is to detect errors, the repair process is improved if the user can supply corrections (often the utility deletes or truncates some structure to clear an error unless directed to do otherwise).

A particular REPAIR utility version should correspond to the version of VERSAdos in use, since the utility assumes certain reserved fields in the disk structures. If a reserved field error is discovered, the user is given the choice of inhibiting (or not) the utility from zeroing out the field (then terminating or skipping the file if doing file check). The version incompatibility must then be resolved before the disk can be processed.

If file structures are deleted during file repair, some of the sectors may not be deallocated in the disk SAT because the utility did not process them. Therefore, it is recommended that a disk check be run to check the SAT after file repair involving extensive truncation or deletion of data.

Disk Structures

Listed below are mnemonics representing various disk structures.

VID	Volume identification block, always sector 0, length of one sector.
SAT	Sector allocation table, or map, variable length.
CFGA	Configuration area (media format information), variable length.
SDB	Secondary directory block (list of catalogs), length of one sector.
SDE	Secondary directory block entry (catalog entry).
PDB	Primary directory block (list of file names), length of four sectors.
PDE	Primary directory block entry (file entry).
FAB	File allocation block (list of data blocks), variable length.
DB	Data block (block of sequential records), variable length.
HDR	Header (first few bytes of each SDB, PDB, or FAB); contains linkage information to next structure of same type. FAB's have forward and backward links; other structures have only forward links.
SLT	Sector lockout table.
DTA	Diagnostic test areas.

Several separate test routines are performed during REPAIR processing. They are:

VID CHECK	Performed unless file repair only is selected. During this test, the volume identification block sector is checked. All remaining structures are reachable only if the VID is intact and readable. A segment is allocated to accommodate two SAT buffers for the disk and allocation SAT's.
SAT CHECK	Performed unless file repair only, in NIM (N option), is selected. The SAT on disk is referred to as "disk SAT" -- it contains the sector allocation map for the disk. Under normal circumstances, the disk SAT is checked and updated as necessary and at the end of REPAIR, it is written back to disk if changed. If the SAT is in poor shape, the user has the option to have REPAIR re-create it; then the disk SAT is ignored and the allocation SAT is written to disk at end of REPAIR. (The SAT cannot be re-created if only file repair is done in NIM because then only the file's sectors are checked and processed for allocation -- the resulting allocation SAT is not complete).

Another SAT-type map is built by REPAIR to keep track of sectors that have been processed and have been allocated so far. This SAT is checked for multiple allocation errors. It is referred to as "allocation SAT". Note that in NIM if file repair only is done, then REPAIR will not be able to detect multiple allocation errors except among the sectors associated with the file.

CFGA CHECK

SDB CHECK

PDB CHECK

FILE CHECK

Update Loop

In the interactive mode, an "update loop" is entered for file structures VID, SAT, SDB, PDB, PDE, DB, and FAB as they are checked during the applicable sections of the REPAIR process. The update loop is usually preceded by a dump of the structure, an error message, and/or a REPAIR action message ("R:..."). The query UPDATE xxx? is displayed, where xxx is the appropriate disk structure mnemonics.

The user must give one of the following responses to each UPDATE xxx? request (note that for each particular structure, there is some variation in allowed response):

Q - Always terminates REPAIR with the message REPAIR TERMINATED. Any sectors updated in memory are written to disk. If I/O error, then error message is given ("IO ERROR, DO = x FROM WRITE xxx, PSN = x", where xxx is a disk structure mnemonic); otherwise, the error is ignored.

R - If given as response to a proposed action for error ("R: ... "), then the indicated repair is enabled and the update loop is terminated; otherwise, WHAT? is sent.

D - Deletes the current structure and terminates the loop. If the structure contains a link to the next structure of the same type, the link is used to continue processing (the user may have changed this field from its original value -- the sectors of the current structure are not written to disk no matter how much they were changed.)

A - Allocate - used only in final SAT CHECK when SAT is not to be recreated by REPAIR.

D $<psn>[,n] -

Delete or deallocate sectors from SAT, where <psn> is a physical sector number (in hexadecimal) within the SAT, and where n (if given) is the decimal number of consecutive sectors affected. n defaults to 1.

A $<psn>[,n] -

Allocate sectors in SAT, where <psn>[,n] are as above.

$[,n] - Dump the structure or, if n (decimal) is given, dump nth sector of the structure (if n is greater than the length of the structure, then n = 1 is used; in PDE update loop, n is ignored). Display to listing default (LDF), usually console screen.

$<psn>[,n] [#PR] -

Dump any sector(s) to the listing device, where n is the decimal number of consecutive sectors to be dumped (default is 1).

$<psn> <offset> xx[,xx]... -

Update a sector of the structure, where <psn> is as above, <offset> is a hexadecimal value (0-$FF), and xx is either a hexadecimal value (0-$FF) or a period (.) followed by an ASCII character. The new values specified (xx[,xx]...) replace consecutive bytes starting with the byte at the given <offset> in the specified sector.

<offset> xx[,xx] -

Same as immediately above, but changes are always made to the first sector of the structure. Here <offset> must start with a numeric character (its value must still be 0-FF). Not allowed in VID check.

N - Terminates loop and turns off interactive mode until a later structure is processed or until an error is discovered.

C - Terminates loop and continues as if in the check only mode until an error is encountered or until execution is completed.

(CR) (Carriage return only.) Terminates loop.

NOTES

This version of the REPAIR utility dumps sectors directly from disk; only the current structure is dumped as it appears in memory. Therefore, other structures having changes made by the utility that are also memory resident may not be displayed.

If REPAIR is used to update the structure of the disk, REPAIR should be performed on the entire volume with the C option. This will ensure the integrity of the new disk structure.

Error messages to invalid UPDATE xxx? responses are:

WHAT?	-	If syntax error, or unallowed response.
PSN ERROR	-	If PSN is out of range, only the current structure can be updated.
VALUE ERROR	-	If byte value for sector change is invalid.
OFFSET ERROR	-	If offset is past sector boundary or too many new byte values are given -- updates are allowed only within the specified sector.

The BREAK key will abort REPAIR with "BREAK! - REPAIR ABORTED". Sectors changed in memory are not written to disk.

In IM during final SAT CHECK, all unused sectors marked allocated are listed. If some structure had to be deleted or truncated because of bad links or other problems, then by dumping the unused allocated sectors the user may recognize sectors that belong to the deleted or truncated structure. The utility can then be reexecuted to reconstruct the structure.

During REPAIR processing, if the volume name in the VID or a file name or an extension in the PDE is in error, the utility will request entry of a correction. The erroneous name or extension will be displayed with periods (.) replacing any non-alphanumeric characters. When making the new entry, only the exact number of characters input will be replaced in the erroneous name or extension, with characters beyond those input remaining unchanged. For example, if a file name should be PROGRAM and it appears as PR....M, at least the characters PROGRA must be entered. If only the missing characters OGRA are entered, the name would become OGRA..M and would still be in error.

Disk and File Structure Tables

The tables below contain the name (symbol), offset in bytes, length in bytes, and description of each field within the VERSAdos disk and file structures.

VOLUME ID BLOCK (VID) - always Sector 0

Symbol	Offset	Length	Field contains
VIDVOL	0	4	Volume ASCII identifier
VIDUSN	4	2	User number
VIDSAT	6	4	Start of SAT
VIDSAL	10 ($A)	2	Length of SAT
VIDSDS	12 ($C)	4	Secondary directory start
VIDPDL*	16 ($10)	4	Primary directory PSN list start
VIDOSS	20 ($14)	4	Start of Initial Program Load (IPL) file
VIDOSL	24 ($18)	2	IPL length
VIDOSE	26 ($1A)	4	IPL execution address
VIDOSA	30 ($1E)	4	IPL load address
VIDDTE	34 ($22)	4	Generation date
VIDVD	38 ($26)	20 ($14)	Volume descriptor
VIDVNO	58 ($3A)	4	Initial version/revision
VIDCHK	62 ($3E)	2	VID checksum
VIDDTP	64 ($40)	64 ($40)	Diagnostic test pattern
VIDDTA	128 ($80)	4	Diagnostic test area directory
VIDDAS	132 ($84)	4	Start of dump area
VIDDAL	136 ($88)	2	Length of dump area
VIDSLT	138 ($8A)	4	Start of SLT
VIDSLL	142 ($8E)	2	Length of SLT
VIDCAS	144 ($90)	4	Configuration area starting sector
VIDCAL	148 ($94)	1	Configuration area length
VIDIPC	149 ($95)	1	IPC format disk type code
VIDRS1	150 ($96)	98 ($62)	Reserved
VIDMAC	248 ($F8)	8	VERSAdos disk (ASCII string "EXORMACS")

*Currently not implemented; =0

Configuration Area (CFGA)

The configuration area contains a disk configuration parameter block identical to the disk configuration IOCB used for Configure, Configure Default, and Configuration/Status requests (see VERSAdos Data Management Services and Program Loader User's Manual). This block contains the configuration in effect when the disk was initialized. This information is used to set up the current configuration when the disk is mounted to allow data access.

DISK CONFIGURATION BLOCK

SYMBOL	OFFSET	LENGTH	CONTENTS
IOSDST			DEVICE STATUS
OR			OR
IOSCEC	0	1	CONFIGURATION ERROR CODE
IOSCTP	1	1	CHANNEL TYPE
IOSDTP	2	1	DEVICE TYPE
IOSDRC	3	1	DRIVER CODE
IOSATM	4	2	ATTRIBUTES MASK
IOSPRM	6	2	PARAMETERS MASK
IOSATW	8	2	ATTRIBUTES WORD
IOSREC	10($A)	2	VERSAdos SECTOR SIZE
IOSRSZ	12($C)	4	TOTAL VERSAdos SECTORS
IOSWTO	16($10)	4	WRITE TIME-OUT (UNUSED)
IOSRTO	20($14)	4	READ TIME-OUT (UNUSED)
IOSSPT	24($18)	1	PHYSICAL SECTORS PER TRACK ON MEDIA
IOSHDS	25($19)	1	NO. OF HEADS ON DRIVE
IOSTRK	26($1A)	2	NO. OF CYLINDERS ON MEDIA
IOSILV	28($1C)	1	INTERLEAVE FACTOR ON MEDIA
IOSSOF	29($1D)	1	SPIRAL OFFSET ON MEDIA
IOSPSM	30($1E)	2	PHYSICAL SECTOR SIZE OF MEDIA
IOSPSD	32($20)	2	PHYSICAL SECTOR SIZE OF DRIVE
IOSTRKD	34($22)	2	NUMBER OF CYLINDERS ON DRIVE
IOSPCOM	36($24)	2	PRECOMPENSATION CYLINDER # (usually .5 total cyl)
IOSSPTD	38($26)	1	PHYSICAL SECTORS PER TRACK ON DRIVE
IOSDRSV	39($27)	7	RESERVED
	40($28)	60($D8)	UNUSED

Sector Lockout Table (SLT)

The SLT is a contiguous segment of disk that describes the sectors on the disk which have been locked out. The number of sectors in the SLT is maintained in the VID and is determined by the number of sectors on the disk. Each entry (six bytes) in the SLT consists of two fields. The first field is four bytes and contains the physical sector number (PSN) of the start of a lockout range. The second field is two bytes and contains the number of contiguous sectors to be locked out. The first zero entry terminates the list. If no SLT exists, the PSN of the SLT in the VID is set to zero.

Diagnostic Test Areas (DTA)

The DTA is one sector that describes the areas on the disk available to diagnostic programs as read/write test areas. Each entry (six bytes) in the DTA consists of two fields. The first field is four bytes and contains the physical sector number (PSN) of the start of a test area. The second field is two bytes and contains the length of the test area. The first zero entry terminates the list. If no DTA exists, the PSN of the DTA in the VID is set to zero.

SECONDARY DIRECTORY BLOCK (SDB) HEADER

Symbol	Offset	Length	Field contains
SDBFPT	0	4	PSN of next SDB (zero if none)
	4	12 ($C)	Reserved (0)
SDBSTR	16 ($10)		Start of first secondary directory entry

SECONDARY DIRECTORY ENTRY (SDE)

Symbol	Offset	Length	Field contains
SDBUSN	0	2	User number
SDBCLG	2	8	Catalog name
SDVPDP	10 ($A)	4	PSN of first PDB for catalog (zero = empty entry)
SDBACI	14 ($E)	1	Reserved
SDBRS1	15 ($F)	1	Reserved

PRIMARY DIRECTORY BLOCK (PDB) HEADER

Symbol	Offset	Length	Field contains
DIRFPT	0	4	PSN of next PDB (zero if none)
DIRUSN	4	2	User number
DIRCLG	6	8	Catalog name
	14 ($E)	2	Reserved
DIRSTR	16 ($10)		Start of first primary directory entry

PRIMARY DIRECTORY ENTRY (PDE)

Symbol	Offset	Length	Field contains
DIRFIL	0	8	File name
DIREXT	8	2	Extension
DIRRSI	10 ($A)	2	Reserved
DIRFS	12 ($C)	4	File starting PSN (contiguous) or first FAB pointer (noncontiguous)
DIRFE	16 ($10)	4	Physical EOF Logical Sector Number (LSN) (contiguous) or last FAB pointer (noncontiguous)
DIREOF	20 ($14)	4	End of file LSN (zero if contiguous)
DIREOR	24 ($18)	4	End of file Logical Record Number (LRN) (zero if contiguous)
DIRWCD	28 ($1C)	1	Write access code
DIRRCD	29 ($1D)	1	Read access code
DIRATT	30 ($1E)	1	File attributes
DIRLBZ*	31 ($1F)	1	Last data block size (zero if contiguous)
DIRLRL	32 ($20)	2	Record size (zero if variable record length or contiguous)
DIRRSZ	34 ($22)	1	Reserved
DIRKEY	35 ($23)	1	Key size (zero if null keys or non-Indexed Sequential Access Mode (non-ISAM))
DIRFAB	36 ($24)	1	FAB size (zero if contiguous)
DIRDAT	37 ($25)	1	Data block size (zero if contiguous)
DIRDTEC	38 ($26)	2	Date file created or updated
DIRDTEA	40 ($28)	2	Last date file assigned
DIRRS3	42 ($2A)	8	Reserved

- File attributes (DIRATT) (bits 4-7 user defined)

Symbol	Value	Meaning
DATCON	0	Contiguous
DATSEQ	1	Sequential (variable or fixed record length)
DATISK	2	Keyed ISAM - No duplicate keys
DATISD	3	Keyed ISAM - Duplicate keys allowed; null keys allowed

* In current implementation, DIRLBZ=DIRDAT

FILE ACCESS BLOCK (FAB) HEADER

Symbol	Offset	Length	Field contains
FABFLK	0	4	Pointer to next FAB (zero if none)
FABBLK	4	4	Pointer to previous FAB (zero if none)
FABUSE*	8	1	Fraction of FAB in use
FABPKY**	9	1	Length of previous FAB's last key
	10 ($A)	FABPKY	Last key in previous FAB (zero if this is first FAB)
FABSEG	10+FABPKY		Start of first segment descriptor

* Divided into 16 parts. 0 ---> empty FAB - 16 ---> full FAB

** Current implementation always has FABPKY=DIRKEY. Later implementations may use partial key.

FAB SEGMENT DESCRIPTOR

Symbol	Offset	Length	Field contains
FABPSN	0	4	PSN of data block start (zero if rest of FAB empty)
FABREC	4	2	Number of records in data block
FABSGS	6	1	Number of sectors in use in data block
FABKEY*	7	1	Key of last record in data block **

* Current implementation always has FABKEY=DIRKEY

** The key may not be equal to the last key in the data block, but it is always less than the first key of next data block unless duplicate keys are allowed. If in first FAB, FABPKY for first FAB has byte for byte correspondence with length of keys established for file and values of bytes are zeros.

Data Blocks (DB)

Data blocks can contain three different record types, none of which allows a record to be split between data blocks:

FIXED LENGTH
VARIABLE LENGTH
CONTIGUOUS

Fixed Length Records: The data content of the fixed length record is specified by the user. There are no embedded control characters generated by File Management Service (FMS). Any unused locations in the data block will be zeroed out.

Variable Length Records: Variable length records have a 2-byte field, preceding each record, which contains the number of data bytes, followed by the data. The data portion of the record can be binary or ASCII. With ASCII specification and a formatted write request, FMS will compress spaces. A space compression character is indicated by a data byte having the sign bit (bit 7) set, while the remaining bits (6-0) contain a binary number representing the number of spaces to be inserted in place of the compressed character. FMS will automatically expand these compressed characters into spaces when such files are read, using formatted ASCII I/O. A zero filler byte is stored at the end of the data portion of a record if the record length is odd. Any unused locations in the data block will be zeroed out.

Contiguous Records: Records for contiguous files (load modules) are 256 bytes in length. The data content, completely specified by the user, contains no embedded control characters generated by FMS.

Example 1 - Recovering a Deleted File

A file named TEST2.LO was inadvertently deleted from volume VOL2. Assuming no files have been allocated in its space, the file is recovered as follows:

```
=REPAIR VOL2:21..TEST2.LO
  VERSADOS REPAIR UTILITY -VERSION 032282 3
  *** FILE CHECK ***
  00 45 53 54 32 20 20 20  4C 4F 00 00 00 00 00 D0   .EST2   LO......
  10 00 00 7A 00 00 00 00  00 00 00 00 00 00 00 00   ...z............
  20 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
  30 00 00 00 00 00 03 8E  03 8E 00 00 00 00 00 00   ..
UPDATEPDE?
  00 54
UPDATEPDE?
  Q
REPAIR TERMINATED
=
```

Since "deleting a file" destroys only the first character of the file name, entering offset 00 and the ASCll character 54 restores the letter "T" to the file name.

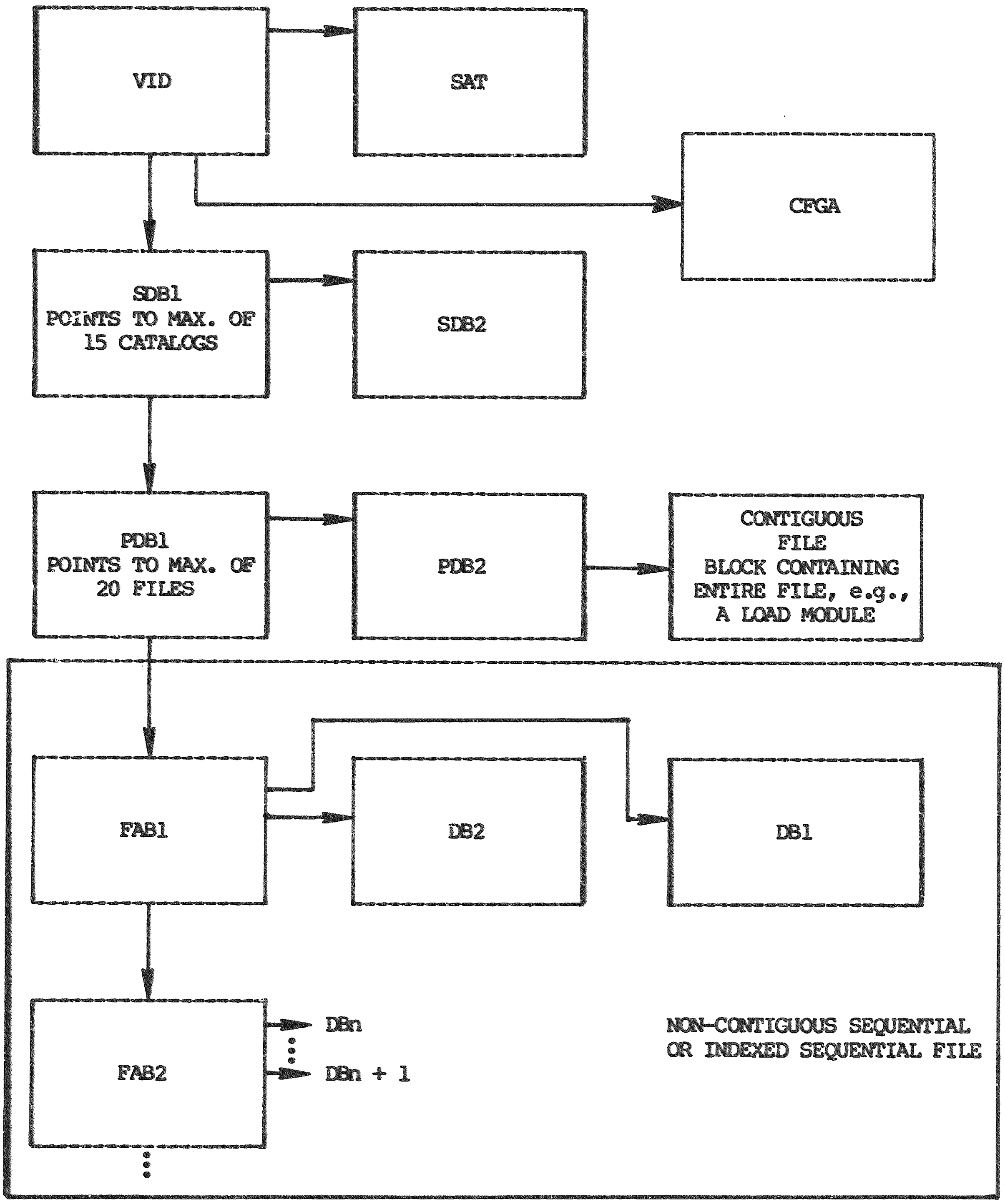


FIGURE 4-4. VERSAdos Disk Structure

(Contains Header And Entries For Up To 20 Files)

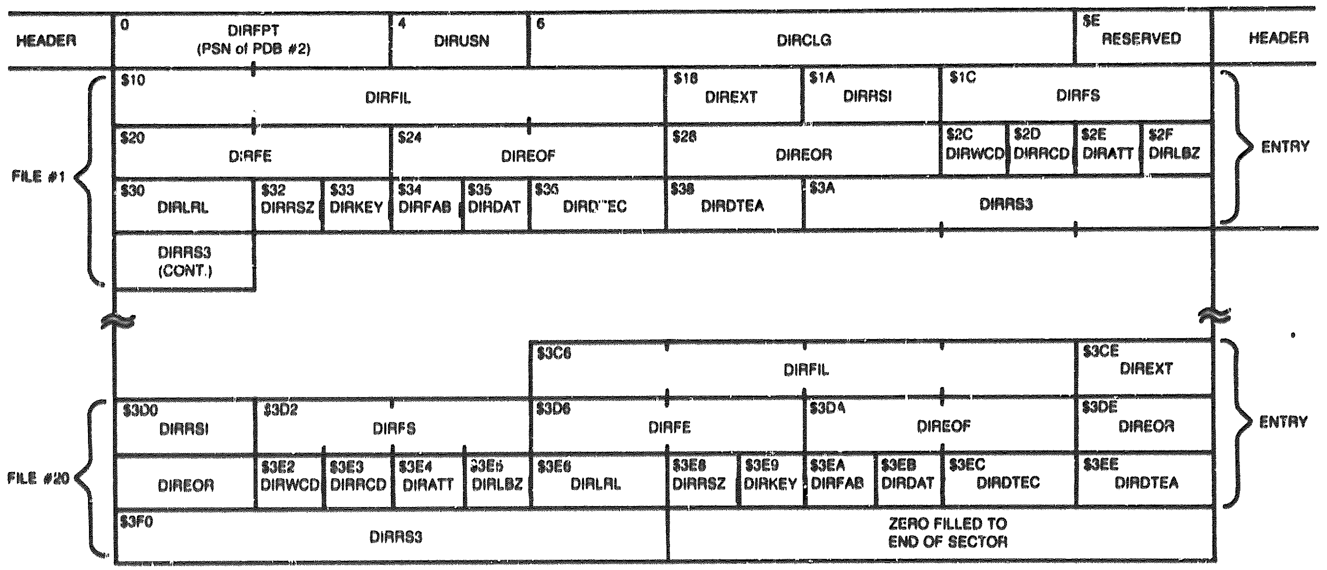


FIGURE 4-5. Primary Directory Block - Four Sectors

```
+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|0                              |4              |6                              |
|            VIDVOL             |    VIDUSN     |            VIDSAT             |
+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|A              |C         VIDSDS               |10                             |
|    VIDSAL     |      (PSN OF SDB #1)          |            VIDPDL             |
+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|14                             |18             |1A                             |
|            VIDOSS             |    VIDOSL     |            VIDOSE             |
+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|1E                             |22                             |26             |
|            VIDOSA             |            VIDDTE             |               |
+-------+-------+-------+-------+-------+-------+-------+-------+               |
/                                                                               /
/                                   VIDVD                                       /
|                                                                               |
+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|3A                             |3E             |40                             |
|            VIDVNO             |    VIDCHK     |                               |
+-------+-------+-------+-------+-------+-------+                               |
/                                                                               /
/                     VIDDTP DIAGNOSTIC TEST PATTERN                            /
|                                                                               |
+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|80                             |84                             |88             |
|            VIDDTA             |            VIDDAS             |    VIDDAL     |
+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|8A                             |8E             |90                             |
|            VIDSLT             | VIDSLL        |            VIDCAS             |
+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|94     |95     |96                                                             |
|VIDCAL |VIDIPC |                                                               |
+-------+-------+                                                               |
/                             VIDRS1 (RESERVED)                                 /
/                                                                               /
|                                                                               |
+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|F8                                                             |
|                   ASCII STRING "EXORMACS"                     |
+-------+-------+-------+-------+-------+-------+-------+-------+
```

FIGURE 4-6. Volume Identification Directory - Sector 0

```
                                +-------+-------+-------+-------+-------+-------+
                                |0        SDBFPT                |4              |
                      HEADER--> |     (PSN OF SDB #1)           |               |
+-------+-------+-------+-------+-------+-------+-------+-------+               |
|                                                                               |
|                                RESERVED (ZERO)                                |
+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
    |10             |12                                                             |
CAT |    SDBUSN     |                            SDBCLG                             |
#1  +-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
    |1A                             |1E     |1F     |20                             |
    |            SDBFDP             |SDBACI |SDBRS1 |                               |
    +-------+-------+-------+-------+-------+-------+                               |
    |                                                                               |
    /                                       .                                       /
    /                                       .                                       /
    |                                       .                                       |
    |                                                                               |
    +-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
    |F0             |F2                                                             |
CAT |    SDBUSN     |                            SDBCLG                             |
#15 +-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
    |FA                             |FE     |FF     |
    |            SDBFDP             |SDBACI |SDBRS1 |
    +-------+-------+-------+-------+-------+-------+
```

FIGURE 4-7. Secondary Directory Block - One Sector

SCRATCH

4.2.21 Scratch Utility (SCRATCH)

The SCRATCH utility allows the owner of a disk or the system administrator (user 0) to erase the volume identification directory (sector 0) of a disk so that the disk may be reused. Optionally, a disk may also be formatted.

SCRATCH Command Syntax

```
SCRATCH <input field>
```

where:

input field	Is the device name of the drive in which the disk to be scratched is mounted (e.g., #FD00, #FD01, or #HD01), or the volume name of the disk to be scratched.

Scratch Utility Examples

```
=SCRATCH #FD01
 OK TO SCRATCH (Y/N) FD01? Y                    (Displayed if volume ID does not
DO YOU WANT TO FORMAT DISKETTE (Y/N)? Y          exist.)

             or

 OK TO SCRATCH (Y/N) FD01 VOLUME VOLN? Y        (Displayed if volume ID exists.)
 DO YOU WANT TO FORMAT DISKETTE (Y/N)? Y
=
                                                (The INIT command may now be used to
                                                 set up the VID of the diskette, as
                                                 desired.)
```

SNAPSHOT

4.2.22 Snapshot Utility (SNAPSHOT)

The SNAPSHOT utility will copy the contents of an EXORterm 155 screen to a file or device. It is useful in saving transient information displayed on the screen which cannot be conveniently directed to a file or device by a more direct method.

SNAPSHOT Command Syntax

```
    SNAPSHOT [<terminal to copy>][,<file/device for output>]
```

where:

terminal to copy	Is the optional device name of the terminal on which the information to be copied is displayed. If this field is not specified, the default value will be the terminal from which SNAPSHOT was invoked.
file/device for output	Is the optional name of the device or file to which the screen contents should be copied. The default value for this field is #PR, the common device name for the first system printer. If a file name is specified, an extension of .SA and the current default volume, user number, and catalog are applied where the file name is not fully qualified. The file thus created is an indexed sequential file of the type used by the editor, simplifying subsequent modification.

When SNAPSHOT is invoked on a terminal, several thousand characters are sent from the terminal to the computer. This causes a delay of several seconds, even over a 9600-baud port. When transmission speeds of less than 9600 baud are used, more time is required to copy screen contents.

It is important to note that the SNAPSHOT utility will work only with EXORterm 155 terminals. Attempts to use it with other terminal types will cause the receiving terminal to lock up and display garbage until its RETURN key is pressed.

SNAPSHOT Utility Examples

Example 1

=SNAPSHOT	The contents of the screen on the terminal from which SNAPSHOT was invoked are copied to device #PR.

Example 2

=SNAPSHOT #CN12,SHOWSBUG	The contents of the screen of terminal #CN12 are copied to file SHOWSBUG.SA under the user's default volume, user number, and catalog.

SYSANAL

4.2.23 System Analysis Utility (SYSANAL)

The SYSANAL utility provides a means of looking at RMS68K system tables and at any part of memory while VERSAdos is running. SYSANAL is an interactive operating system debugging tool. Output can be directed to the user's terminal or can be redirected to a printer.

SYSANAL Command Syntax

```
    SYSANAL [<output field>]
```

where:

output field	Is the name of the device where output will be listed. If no output field is given, the default is # (user's interactive terminal).

SYSANAL responds by displaying a prompt for entry of subcommands:

```
    SYSA:>
```

Quit Subcommand Syntax (QUIT)

```
    SYSA:>Q[UIT]
```

where:

QUIT or Q	Is the subcommand for terminating the SYSANAL utility and returning control to VERSAdos.

Address Dump Subcommand Syntax (AD)

```
    SYSA:>AD <address1>[,<offset>] <address2>
```

where:

AD	Is the subcommand to dump a block of memory in hex format.
address1	Is the starting address of the block of memory to be dumped.
offset	Is a hex value that will be added to <address1> to determine the starting address of the block of memory dumped.
address2	Is the ending address of the block of memory dumped.

Channel Control Block Subcommand Syntax (CCBS)

```
SYSA:>C[CBS]
```

where:

CCBS or C	Is the subcommand to dump the contents of all Channel Control Blocks known to the system.

Free Memory Subcommand Syntax (FREE)

```
SYSA:>F[REE] <partition #>
```

where:

FREE or F	Is the subcommand used to list the free memory list for any memory partition.
partition #	Is the memory partition number.

Global Segment Table Subcommand Syntax (GST)

```
SYSA:>G[ST]
```

where:

GST or G	Is the subcommand to list all entries in the Global Segment Table. Each entry describes the size and attributes of a globally or locally shareable segment. See Appendix E, paragraph E.5, of the Real-Time Multitasking Software User's Manual, M68KRMS68K, for a description of the Global Segment Table.

Map Subcommand Syntax (MAP)

```
SYSA:>M[AP]
```

where:

MAP or M	Is the subcommand for listing the limits and size of each memory partition and the free memory list for partition 0. See Appendix E, paragraph E.9, of the Real-Time Multitasking Software User's Manual for a description of the Memory Map and Free Memory List.

Memory Dump Subcommand Syntax (MD)

```
SYSA:>MD <address>[,<offset>] <# bytes>
```

where:

MD	Is the subcommand used to dump a block of memory in hex format.
address	Is the starting address of the block of memory to be dumped.
offset	Is a hex value that will be added to <address> to determine the starting address of the block of memory dumped.
# bytes	Is the number (expressed in hex) of bytes to be dumped.

Offset Subcommand Syntax (OFF)

```
SYSA:>OFF <address>
```

where:

OFF	Is the subcommand used to set a default memory address offset for use by the MD and AD subcommands.
address	Is a hex value to be used as a memory address offset.

Output Subcommand Syntax (OUT)

```
SYSA:>OUT <device>
```

where:

OUT	Is the subcommand used to select an output device.
device	Is the output device. # selects the user's terminal; #PRn selects a printer.

Periodic Activation Table Subcommand Syntax (PAT)

```
SYSA:>P[AT]
```

where:

PAT or P	Is the subcommand used to list all of the tasks with entries in the Periodic Activation Table as a result of having delayed or requested periodic activation.

See Appendix E, paragraph E.10, of the Real-Time Multitasking Software User's Manual for a description of the information found in the Periodic Activation table.

Ready Subcommand Syntax (READY)

```
SYSA:>R[EADY]
```

where:

READY or R	Is the subcommand used to list selected information about each task currently on the 'Ready' list (these tasks are waiting to run). The information is taken from the Task Control Block associated with each task. See Appendix E, paragraph E.2, of the Real-Time Multitasking Software User's Manual for a description of a Task Control Block.

System TCB Subcommand Syntax (STCB)

```
SYSA:>S[TCB]
```

where:

STCB or S	Is the subcommand for listing selected information about each system task currently known to the system. The information listed is taken from the Task Control Block associated with each task. See Appendix E, paragraph E.2, of the Real-Time Multitasking Software User's Manual for a description of a Task Control Block.

System Parameters Subcommand (SYSP)

```
SYSA:>SY[SP]
```

where:

SYSP or SY	Is the subcommand for listing the system parameters. See Appendix E, paragraph E.1, of the Real-Time Multitasking Software User's Manual for a description of system parameters.

Tables Subcommand Syntax (TABL)

```
SYSA:>T[ABL]
```

where:

TABL or T	Is the subcommand for listing all of the tables recognized by SYSANAL. The TABL subcommand is the equivalent of entering the following subcommands:

SYSP
TRAP
MAP
TCB
READY
PAT
UST
GST
CCBS
TRACE

Task Subcommand Syntax (TASK)

```
SYSA:>TASK <task name>[ <session>]
```

where:

TASK	Is the subcommand used to request a detailed list of information about one task. The information listed is found in the Task Control Block and Task Segment Table for the given task. See Appendix E, paragraphs E.2 and E.3, of the Real-Time Multitasking Software User's Manual for descriptions of the Task Control Block and Task Segment tables.
task name	Task name can be entered as a 4-character ASCII value or as an 8-character hex value preceded by $.
session	Session can be entered as a 4-character ASCII value or as an 8-character hex value preceded by $. If no session is entered, the first task found with the given task name will be listed.

Task Control Block Subcommand Syntax (TCB)

```
SYSA:>TCB
```

where:

TCB	Is the subcommand for listing selected information about each task currently known to the system. The information listed is taken from the Task Control Block associated with each task. See Appendix E, paragraph E.2, of the Real-Time Multitasking Software User's Manual for a description of a Task Control Block.

Trace Subcommand Syntax (TRAC)

```
SYSA:>TR[AC]
```

where:

TRAC or TR	Is the subcommand used to list the System Trace Table entries. The entries are listed in chronological order from the oldest to the most recent. The time that each entry was built and the difference in time between entries is also listed. See Appendix E, paragraph E.7, of the Real-Time Multitasking Software User's Manual for a description of Trace entries.

Trap Subcommand Syntax (TRAP)

```
SYSA:>TRAP
```

where:

TRAP	Is the subcommand for listing the Trap Instruction Assignment Table and the Trap Instruction Owner table. These tables indicate which Trap instructions are currently owned by server tasks. See Appendix E, paragraph E.1, of the Real-Time Multitasking Software User's Manual for a description of the TIAT and TIOT tables.

User Semaphore Table Subcommand Syntax (UST)

```
    SYSA:>US[T]
```

where:

UST or US — Is the subcommand to list all entries in the User Semaphore Table.

See Appendix E, paragraph E.6, of the Real-Time Multitasking Software User's Manual for a description of the User Semaphore Table.

User TCP Subcommand Syntax (UTCB)

```
    SYSA:>U[TCB]
```

where:

UTCB or U — Is the subcommand for listing selected information about each user task currently known to the system. The information listed is taken from the Task Control Block associated with each task.

See Appendix E, paragraph E.2, of the Real-Time Multitasking Software User's Manual for a description of a Task Control Block.

SYSANAL - Summary of Subcommands

Program Control Commands

OUT <device>	Send output to <device>.
QUIT	Terminate program.

Memory Dump Commands

OFF <address>	Set default memory address offset to <address>.
MD <address1>[,<offset>]	<# bytes> Dump <# bytes> starting at <address1> + <offset>.
AD <address1>[,<offset>]	<address2> Dump memory <address1> to <address2>.

List System Table Commands

CCBS	Channel Control Blocks.
FREE n	Free Memory List for memory partition n.
GST	Global Segment Table.
MAP	Memory Map.
PAT	Periodic Activation Table.
READY	Task Ready List.
STCB	System Tasks.
SYSP	System Parameters.
TABL	All system tables.
TASK <name>[<session>]	Detailed information about one task.
TCB	All tasks.
TRAC	System Trace Table.
TRAP	Trap Instruction Tables.
UST	User Semaphore Table.
UTCB	All user tasks.

TRANSFER

4.2.24 Transfer ASCII File Utility (TRANSFER)

The ASCII file transfer utility is provided so that files comprised of printable ASCII characters in the range $20-$7E, such as source programs or S-record files (Appendix A), can be transported between a host computer system and one of the VERSAdos systems (download) or in the other direction (upload). The communication link can be a direct connection between serial ports (RS-232) on the two systems, or voice grade phone lines using a type 103 modem. Data can be transferred at 300 baud via the modem/phone line link, or at rates up to 9600 baud using a direct connection.

When transferring a file via modem, connection is made through a port on a Multi-Channel Communication Module (MCCM). For direct connection file transfer, an MCCM port or a port on the DEbug Module or the second port on the VM02 can be used.

The ASCII file transfer utility differs somewhat from the usual VERSAdos utility in that three separate programs are involved. Two of these are Pascal programs installed on the remote host system, and the third is an assembly language program installed on the VERSAdos system. The two host programs, DLOAD and ULOAD, run on the host in coordination with the TRANSFER program running on the VERSAdos system. TRANSFER uses the standard VERSAdos file handling and input/output services.

Source code for the three programs is supplied so that modifications can be made to accommodate any differences in Pascal implementation. Most of the modifications will be required in the DLOAD and ULOAD programs. As a familiarization aid, two example sets of these Pascal programs are also supplied. One will run on the VERSAdos system. The other has been run on an IBM 370 after compilation by the Pascal/VS program.

On VERSAdos versions prior to 2.0, a patch must be made to the TRANSFER.LO module for ULOAD to work properly. The patch and associated change to the source file is presented below:

TRANSFER.LO			TRANSFER.SA		
Address,	Old value,	New value	Line number,	Old contents,	New contents
$1ABE	$53DA	$5388	1295	SUB.L #1,A2	SUB.L #1,A0

The utility operates in three modes:

a. The download mode, during which the file is actually transferred from the host system to the VERSAdos system.

b. The upload mode, during which the file is actually transferred from the VERSAdos system to the host system.

c. The transparent mode, in which the EXORterm or other terminal functions as a simple terminal and appears to be communicating directly with the host system.

The general operating sequence is to enter the transparent mode; initiate upload or download, following completion of which the transparent mode is automatically reentered; then initiate another file transfer or terminate.

Use of the ASCII file transfer utility to communicate data between two computer systems -- either by direct connection between serial ports or via phone lines using modems -- places certain constraints on the process which must be accommodated. Namely, both systems must be configured for the same baud rate, type of modem, and character makeup.

VERSAdos, as shipped, contains default values which set up the serial ports on the DEbug and MCCM Modules as follows: 9600 baud, one start bit, eight data bits, no parity bit, and one stop bit. If the host computer supports this protocol, file transfer is possible by direct connection through port 2 on the DEbug Module or an MCCM port. However, the user should note that the bit time is about one millisecond; therefore, the processor should be dedicated solely to the data transfer else overrun is likely. Successful data transfer at this rate in a multiuser environment is possible through an MCCM port, but may not succeed when port 2 on the DEbug Module is used.

To use the ASCII file transfer utility for data transfer via modem using an MCCM port, for example, the following considerations should be observed prior to invoking the utility.

First, the VERSAdos system hardware must be configured.

a. Select an MCCM port for the file transfer.

b. On the terminal/modem configuration header corresponding to the selected port, ensure that the configuring jumpers are installed in the B-C position (MOD). This causes the VERSAdos system to appear like a terminal to the modem. Refer to the following chart for correspondences between MCCM header, I/O panel port, and VERSAdos nomenclature. Note that the I/O port designators differ from the VERSAdos designators. In the System Generation Facility User's Manual, M68KSYSGEN, these ports are called terminals.

MCCM	PORT DESIGNATOR		VERSAdos
HEADER	I/O PANEL	VERSAdos	MNEMONIC
K8	1	0	CN10
K7	2	1	CN11
K10	3	2	CN12
K9	4	3	CN13

The mnemonics (device names) given above are for the first (or only) MCCM.

Next, a new operating system must be generated which reflects the new hardware configuration and supports a baud rate of 300 and a type 103 modem. The configuration can be changed by modifying parameters in the SYSGEN command file. Refer to the System Generation Facility User's Manual.

A special step must be taken when using ASCII file transfer between two VMC systems with the connection made between local ports. When operating in this mode, upload will not function properly unless the type-ahead feature on the local (home) port is deconfigured. To deconfigure the type-ahead feature, it is necessary to set bit 7 of the attributes word in the associated device control block (DCB) to 1. The standard value for the attributes word is $0100. This must be changed to $0180 to deconfigure type-ahead. The device control block is contained in module IOC.SA which is assembled at SYSGEN time..

After the user has configured the VERSAdos system hardware and operating system, he should ensure that the host system supports the just-established data communications protocol. As mentioned, because the Pascal modules are implementation-dependent, it is highly likely that some slight adjustment for the user's environment will be necessary. The best approach is to acquire an understanding of each program so that the part that needs to be changed can be isolated and the precise change determined. To aid you in this endeavor, source listings are extensively annotated.

The following example typifies use of the utility to download an ASCII file via modem from a host IBM 370 system running under TSO to a VERSAdos system. The source file is HOST.DATA; the destination file name is HOME.SA. Note that syntax appropriate to the system executing the command must be observed when a command is entered. For clarity, user input is underlined, while utility displays are not. The dialog begins after the user has initiated a session on the VERSAdos system.

```
Step 1.  =TRANSFER #CN20;HD
          ASCII FILE TRANSFER - VERSION 1.00
          **TRANSPARENT MODE - TYPE CTRL-A TO TERMINATE TRANSFER PROGRAM
```

Step 2. The user now establishes a phone line link with the host and initiates a session on that system.

```
Step 3.  EX DLOAD 'HOST.DATA'  (sent from VERSAdos system to host and executed
                                there)
         ENTER RECEIVING FILE NAME OR 'Q' TO RESUME TRANSPARENT MODE
```

```
Step 4.  HOME.SA
         DOWNLOAD INITIATED
         KEY IN 'T' TO TERMINATE TRANSFER AND RESUME TRANSPARENT MODE
                'D' TO TOGGLE RECORD DISPLAY
         (records are displayed as they are downloaded)
         DOWNLOAD COMPLETED 000312 RECORDS TRANSFERRED
         **TRANSPARENT MODE - TYPE CTRL-A TO TERMINATE TRANSFER PROGRAM
```

Step 5. <u>(CTRL-A)</u>

The command line in Step 1 invokes the assembly language program on the VERSAdos system and specifies the port through which data is to be sent. Options H and D select half duplex communication and request display of records as they are downloaded, respectively.

Step 3 shows a command line sent to TSO, specifying a command on the TSO CLIST that will invoke the Pascal download program. The name of the source file is also specified. TSO syntax is used on this command line.

Entering the name of the receiving file in Step 4 initiates the download. The user is given the options of viewing the file records as they are received and of stopping the transfer. (At present, the process cannot be restarted at the stopping point but must begin again from Step 1.)

An upload file transfer is quite similar to the above download example except that upload messages are provided and the command line sent to TSO (Step 3) would be:

```
EX ULOAD 'HOST.DATA'
ENTER SENDING FILE NAME OR 'Q' TO RESUME TRANSPARENT MODE
```

This command specifies the command on the TSO CLIST that will invoke the upload program.

The capability of transferring ASCII files in either direction between a VERSAdos system and a remote system was developed using a data communications link between an EXORmacs and a system comprised of an IBM 3033 (or the AMDAHL equivalent) running under the MVS operating system equipped with the TSO option.

4.2.24.1 ASCII File Transfer Command Format. Commands specified to the file transfer utility are interpreted by the VERSAdos operating system and, if the host is a not an applicable VERSAdos system, by the operating system of the host. Those commands specified to VERSAdos must be expressed in the syntax required for the command, file descriptor, device name, and option fields within the standard command line. Commands specified to a non-VERSAdos system are expressed in the format of that system.

TRANSFER Command Syntax

```
    TRANSFER <device name>[;<options>]
```

where:

device name — Is the VERSAdos mnemonic for the port through which file transfer is to be made.

NOTE

The as-shipped default value for the device name field is CN20. The user can install his own default by altering the value of the DEFPORT equate in the source file TRANSFER.SA, then re-assembling and relinking the file.

options — may be one or more of the following options:

B - Remove trailing blanks from record prior to transmission. The default is B; -B may be specified when trailing blanks are desired.

C - Transmit checksums with records. On checksum verification failure, make three additional transmission attempts. The default value (3) can be changed using the form C=n in place of the character C alone. This form must be followed by a comma if an additional option is specified. The default is C; -C may be used when checksums are not to be transmitted with records.

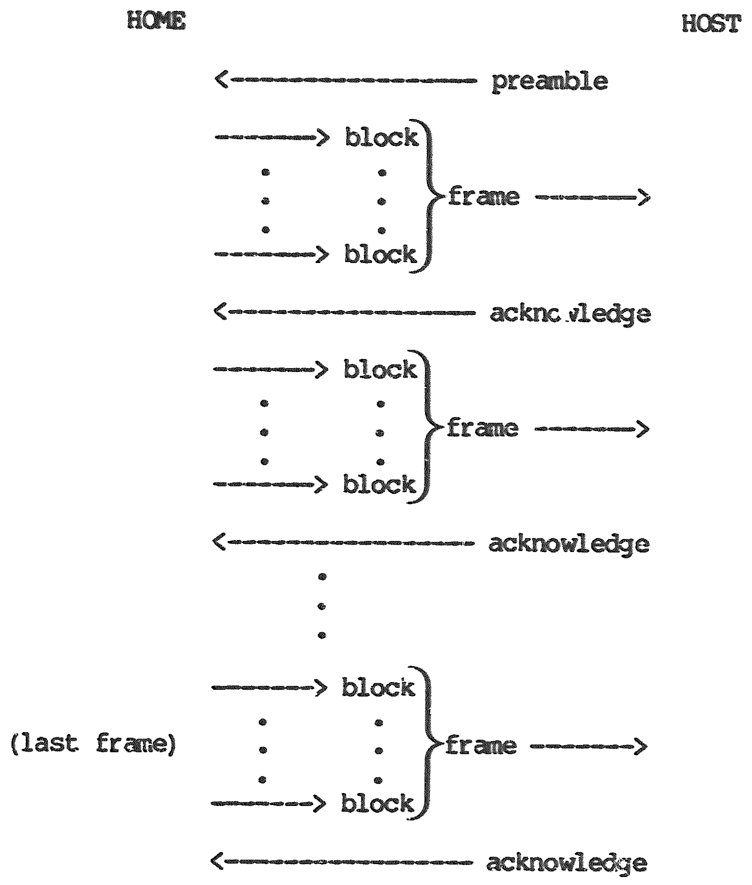
D - Display records on home terminal when transmitted. This option must be specified when it is desired. The default is -D, which causes transmitted records not to be displayed.

H - Set half duplex communication for transparent mode (assumes no characters are echoed from host). H causes data to be transferred a line at a time, with a carriage return signifying the end of each line. -H causes data to be transmitted a character at a time. The default is H.

4.2.24.2 <u>Transfer Protocol</u>. The following diagram depicts the data transfer protocol used during the download process.

```
         HOME                                   HOST

                  preamble <----------

               acknowledge ---------->

                         ┌ block <-------
                         │   .        .
         <------ frame  <    .        .
                         │   .        .
                         └ block <-------

               acknowledge ---------->

                         ┌ block <-------
                         │   .        .
         <------ frame  <    .        .
                         │   .        .
                         └ block <-------

               acknowledge ---------->
                                .
                                .
                                .
                         ┌ block <-------
                         │   .        .
         <------ frame  <    .        .   (last frame)
                         │   .        .
                         └ block <-------

               acknowledge ---------->
```

The following diagram depicts the data transfer protocol used during the upload process.



Following are definitions of the terms used in the data transfer protocol diagrams:

Preamble - The preamble is a string of characters which identify parameters of the ensuing process. Preamble contents can be modified by altering constants in the Pascal source host programs and recompiling. Parameters identified in the preamble include:

- upload or download
- block size expressed as number of characters
- number of blocks per frame
- communication option
- character conversion option

<u>Block</u> - A block is a single burst of transmitted characters. It consists of packed, compressed records from the source file plus other control information.

<u>Frame</u> - A frame is a transmitted sequence of from one to nine blocks containing a checksum calculated over the group. An acknowledgement is transmitted back to the sender after each frame is received.

Preamble Format

An actual file transfer is initiated by the invoked Pascal program sending a preamble record that alerts the TRANSFER program running on the VERSAdos system so that special requirements of the host/home communication link can be accommodated. This first record or preamble is comprised of the following fields:

CHARACTER POSITION	FIELD	DESCRIPTION
1-8	Preamble	The character string "PREAMBLE" occupies this field.
9	Process Type	The character "U" or "D" identifies the upload or download process, respectively.
10	Communication Option	Identifies the nature of the ensuing communications. As released, the options characters are "A" (host is VERSAdos system), and "B" (IBM 370 is host).
11	Character Conversion Option	The character "A" (VERSAdos system is host) or "B" (IBM is host) avoids or selects a special conversion, respectively.
12-14	Block Length	Three ASCII decimal digits (blanks represent non-significant preceding digits) identify the number of characters per block.
15	Number of Blocks	One ASCII decimal digit identifies the number of blocks per frame.

All characters in the preamble are encoded as ASCII characters. Fields 10 and 11 require additional discussion.

The communication option, if A, informs the TRANSFER program that the host is a VERSAdos system. If option B is sent, TRANSFER recognizes that the host is an IBM 370 system. The protocol differs between the two. For example, if the host is an IBM, an acknowledgement is not sent immediately after receipt of data as it is if the host is a VERSAdos system, but only after a $11 (DC1) control character is received indicating that the host port is ready to receive.

The character conversion option, if an A, informs TRANSFER that, since the host is a VERSAdos system, no conversion is needed. A 'B' character, on the other hand, causes TRANSFER to accommodate the different I/O required by the IBM host.

In particular, lowercase characters are automatically converted to uppercase before calculation of the checksum for a frame prior to the data being uploaded. Conversion is not required for downloading, but two IBM 370 idiosyncrasies are accommodated. The first results from the fact that the IBM Pascal runtime package automatically forces an end of line when it recognizes the $5E (^) code. This restriction is accommodated by having the host download program edit the source file data and change all $5E's to $7E's (~) before sending the record. The second arises from the fact that the IBM Pascal runtime package interprets the '~' character (ASCII code $7E) differently if it is passed as a character constant than if it is read from a file. This restriction is accommodated in the lookup tables in both the upload and download Pascal programs on the host by using the $7E code in both the $7E and $5F positions.

It can be seen from the above that a certain customizing of the Pascal upload and download -- and perhaps the TRANSFER -- programs may be necessary before successful ASCII file transfer is achieved on a particular home/host system configuration.

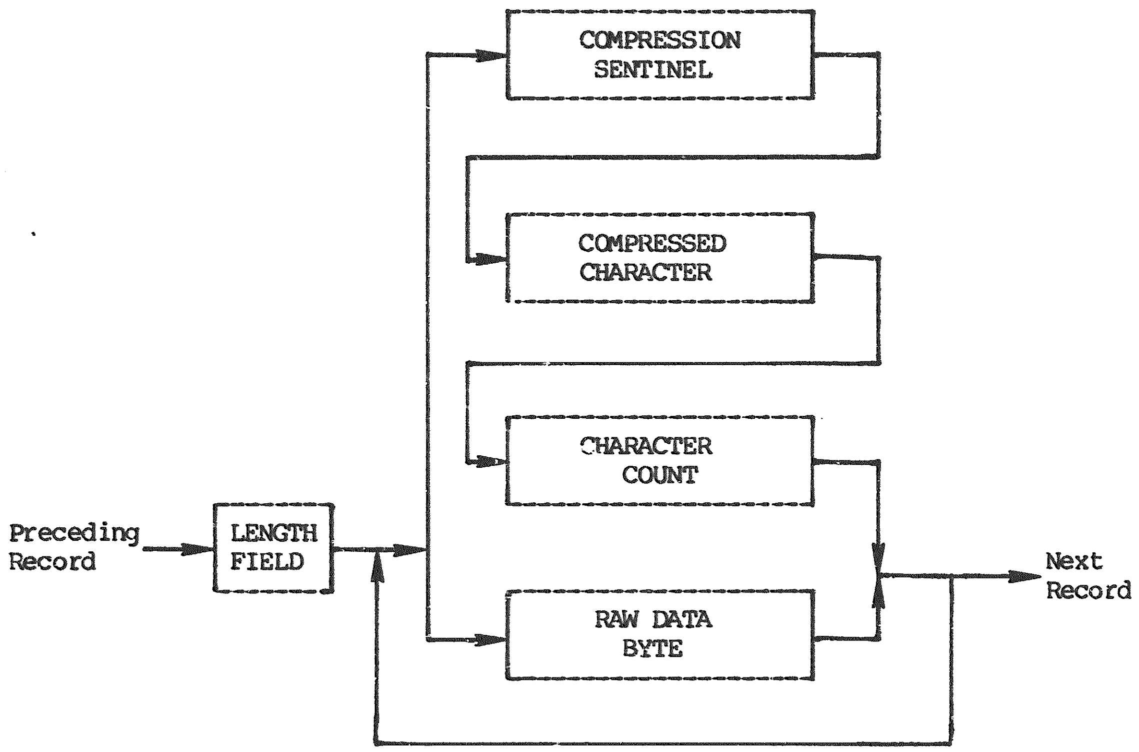
Frame Format

A frame is comprised of one to nine blocks and can be considered to be a single character string formed of packed, compressed ASCII records. Although a frame is of a fixed length, records are variable length and may cross frame boundaries, thus conserving space. Due to encoding restrictions, an ASCII record is limited to a maximum of 255 bytes.

A frame is comprised of two control fields and a data stream, as follows:

CHARACTER POSITION	FIELD	DESCRIPTION
1-2	Checksum	The checksum, in ASCII hex, calculated over the data field only. The wildcard value 00 ($3030) is always accepted. Checksum value is (sum mod 255) + 1, where sum is the sum of the ASCII codes from byte position 3 to the end of the frame.
3	Last Frame	The character "A" indicates not last frame. Character "Z" identifies the last frame in an upload or download transmission.
4-N	Data	Data encoded as raw data and compression packets (explained on following page) and contained in records of variable length, where N is the number of characters in a frame as computed from the preamble.

The data stream is a contiguous succession of records, each of a length defined by a preceding length field and containing raw data and/or compression packets. Records are encoded using the following format:



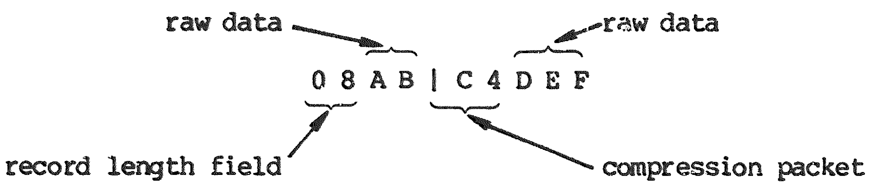
Record Encoding Diagram

The length field is encoded from a 2-byte value, thereby limiting the maximum record length to FF or 255 characters, not including the record length field.

A record body is made up from raw data and compression packets in any order required. It is unlikely that a record would comprise a single compression packet or raw data only, but this is permitted subject to a maximum length of 255 characters.

The compression packet is a device to increase transmission efficiency in the presence of data containing characters that are repeated four or more times in succession. Packets are encoded using the leading sentinel $7C (the symbol "|"), followed by the character whose repetitions are being compressed, followed by the compression factor or the number of times that character is to be repeated in the destination file. The compression factor value is conveyed by a single character from the alphanumeric sequence 4,5,...9,A,B,...Z, in which A has the value 10, B has the value 11, etc., through Z which has the value 35.

For example, the string QQQQQQQ would be encoded |Q7 ($7C5137 in ASCII hex). A further example of the use of a compression packet would be the encoding of the short record: ABCCCCDEF. This would be represented as:



From the above, it can be concluded that any data not encoded in a compression packet is represented as raw data - i.e., as one ASCII character per byte of source file data.

Acknowledge Format

In the download mode, the home system sends a two-character acknowledgement to the host system following successful reception of the preamble and each frame. If the B option was specified (trailing blanks suppression) when TRANSFER was invoked, NB is sent. If the B option was not selected, NØ is sent.

Similarly, RB or RØ sent on detection of a frame checksum mismatch; TB or TØ sent to terminate the download program following the last frame in a transmission.

In the upload mode, the host system sends a one-character acknowledgement. Following successful reception of the preamble and each frame, an N is sent. On detection of a checksum mismatch, an R (for retransmit) is sent and a T for acknowledge and terminate after the last frame.

In the upload mode only, detection of a T in the first character position causes unconditional termination of the process.

4.2.24.3 File Transfer Reliability. Transfer of ASCII files is affected by some factors over which the utility has no control. Because the interval required to perform I/O differs on a VERSAdos system versus that required by an IBM system, and because, in a multiuser environment, processor reliability is not precisely predictable, a format of one block per frame is recommended. This allows the interleaved acknowledgements to synchronize block transmission, thus assuring reliable data transfer.

In applications where experimentation for the purpose of maximizing the rate of data transfer is desired, the number of blocks per frame can be adjusted. This is done by modifying the constant section of the download Pascal program running on the host and recompiling the program. Since baud rate and concurrent user activity on the VERSAdos system can also affect I/O synchronization, the user may wish to experiment with these factors also.

As mentioned earlier, data can be transferred at higher rates by direct connection. In one instance on a single-user system, a rate of 9600 baud was achieved using direct connection through port 2 on an EXORmacs debug module.

When a new VERSAdos operating system is generated for data transfer by direct connection, standard RS-232 protocol is observed at the serial port. However, when VERSAdos is generated for data transfer via modem, the clear-to-send line is not asserted by the new code because this could cause loss of carrier.

4.2.24.4 <u>Transfer Programs and Files</u>. The following programs and files are supplied with the current release of the ASCII File Transfer utility:

(1)	TRANSFER.SA	-	source for VERSAdos system program
(2)	ULOAD.SA	-	source for VERSAdos system host upload program
(3)	DLOAD.SA	-	source for VERSAdos system host download program
(4)	TRANSFER.LO	-	load module from TRANSFER.SA
(5)	TRANSFER.LF	-	chain file to link TRANSFER
(6)	TSO.ULOAD.SA	-	source for IBM host upload program
(7)	TSO.DLOAD.SA	-	source for IBM host download program
(8)	TSO.DLOAD.CF	-	example CLIST for IBM (host) download
(9)	TSO.ULOAD.CF	-	example CLIST for IBM (host) upload

<u>Note on Communication between Two EXORmacs Systems</u>

When establishing communication between two VERSAdos systems, a problem arises when trying to log onto the host system in transparent mode. Entering a 'break' from the home system will simply terminate the transfer program. The break is not sent to the host system to initiate a logon sequence. At the present time, there is no mechanism to send a break out from either a DEbug or MCCM serial port.

To overcome this problem, a utility called BREAK can be used. To use the BREAK utility, perform the following steps:

a. Bring the TRANSFER utility up on the home system but do not attempt to initiate a logon sequence from the home system.

b. On the host system, log on as user 0 and enter the following command:

```
=BREAK #CNxx
```

Where #CNxx is the device name for the port on the host system which is connected to the home system. The effect of this utility is to queue an attention event to $EET indicating that a break has occurred on the indicated port. The entry exit task will then initiate a logon sequence to that port. The logon message should appear at the home terminal where the transfer utility has been invoked. The normal logon dialog can then be followed.

The MCCM firmware and the local terminal driver will always wait for a carriage return to terminate an input request from the TRANSFER utility on the home system. For this reason, only messages that are sent from the host system that are terminated with a carriage return will appear on the home terminal. The user at the home terminal will thus see the first line of the logon message but the line "ENTER USER NO. =" will not appear since it is not terminated with a carriage return. The user number should be entered even though the second line of the logon message did not appear. The same can be said for the security word and password if they are active. Note also the system prompt (=) will never appear at the home terminal since it too is not terminated by a carriage return.

Because all incoming characters are ignored between VERSAdos I/O requests, characters at the beginning of a line may be dropped. For example, if only a carriage return/line feed is sent, both may be missed and TRANSFER may appear to hang.

TRANSFER does not support a "true" transparent mode of operation. The transparent mode of the TRANSFER utility simply provides a mechanism for entering the upload or download mode. It is not a general-purpose mode and does not allow other types of dialog between home and host system.

UPLOADS

4.2.25 Upload S-Records Utility (UPLOADS)

UPLOADS is a VERSAdos utility which is used to migrate S-records (see Appendix A) from some external source to a VERSAdos system. The S-records must be received through a port of a Multi-Channel Communications Module (MCCM) which is connected to the source system via a direct RS-232 hardware configuration.

NOTE

Modems cannot be used because direct connection of the CTS line is required to provide the source system with the information needed to prevent buffer overrun.

The VERSAdos file which will contain the received S-records is specified as an operand when the UPLOADS command is entered. UPLOADS either "allocates" this output file if it is new, or "positions" the file following the first existing S9 record, for appending, if the file already exists.

NOTE

There cannot be embedded S9 trailer records within an S-record file. UPLOADS prevents this by using a positioning technique and changing any previously existing S8/S9 records to nulls. Also, if several S-record files are sent to UPLOADS, any S8 or S9 records received will be ignored unless at the end of the file (the processing of the QUIT directive will transfer a pending S8 or S9 record to the file).

UPLOADS Command Syntax

```
UPLOADS <output field>
```

where <output field> may be any file name (default extension is .MX).

UPLOADS Subcommands

STATUS	This subcommand allows the remote user to see if there were any errors while transmitting the S-records. Only the last error will be displayed.
QUIT	The QUIT subcommand is entered to terminate the UPLOADS program and cause the newly created S-record file to be saved.

UPLOADS Command Example

Using an EXORmacs Development System, allocate a new file and receive S-records from a Remote Hardware Development Station (RHDS).

NOTE

> In the following example, information that was entered by the user is underlined. All other information can be assumed to be responses to user commands (in this case, MACSbug or VERSAdos).

After connecting port 2 of the RHDS to the MCCM of the EXORmacs, apply power and RESET to start the following interactive sequence:

TERMINAL DISPLAY	BRIEF EXPLANATION

`PC=000000 SR=2700=.S7..... US=B238B238 SS=0000055C`
`D0=FFFFFFFF D1=FFFFFFFF D2=FFFFFFFF D3=FFFFFFFF`
`D4=FFFFFFFF D5=FFFFFFFF D6=FFFFFFFF D7=FFFFFFFF`
`A0=FFFFFFFF A1=FFFFFFFF A2=FFFFFFFF A3=FFFFFFFF`
`A4=FFFFFFFF A5=FFFFFFFF A6=FFFFFFFF A7=0000055C`
`-----------------------------------------------`	

MACSbug 3.x*	"RHDS" MACSbug prompt.
MACSbug 3.x* TM *TRANSPARENT* EXIT=$01= CTRL A	Enter transparent mode. (Hold CTRL key down and press A.) Transparent mode allows the terminal (connected to port 1 of the RHDS) to be virtually connected to whatever is connected to port 2 of the RHDS (which is connected via a direct line to an MCCM on an EXORmacs system operating under VERSAdos). The next step is to initiate a user session under VERSAdos to enable the EXORmacs system to receive the S-records and have them saved on disk.
<BREAK key>	Begin VERSAdos logon procedure.
VERSAdos VERSION: mm.nn, mm/dd/yy ENTER USER NO.=FIX:<user number>	Enter default drive and user number.
=UPLOADS FILE01	When the prompt (=) appears, the UPLOADS program can be invoked, and the name of the file it will use to store any S-records that are received from this point on can be entered.

UPLOADS will respond with:

```
                    U P L O A D   "S"   RECORDS
                           Version x.y
                 Copyrighted 1981 by MOTOROLA, INC.

volume=FIX
 catlg=
  file=FILE01
   ext=MX

UPLOADS Allocating new file

Ready for "S" records,...
```

By now, the UPLOADS program has verified the name of the output file which was entered, found that it did not exist previously, and allocated such a file. Also, as the last line indicates, the UPLOADS program is ready to receive S-records (via the MCCM communication link).

The next step is to "DISCONNECT" the "virtual" terminal connection with VERSAdos, and communicate with MACSbug in order to construct and transmit the S-records. This is done by exiting the transparent mode.

```
(CTRL A)
```

(Hold CTRL key down and press A.) The link through port 2 to EXORmacs (via the MCCM) has now become one-way. Data can be transmitted out of port 2, but VERSAdos cannot send messages to port 2. (This is why any error status message must be saved and sent only when the link can receive again.)

```
MACSbug 3.n*DU2 700 7FC
```

MACSbug has been instructed to create S-records from memory, starting at location $700, and transmit them through port 2 until the ending address of $7FC has been reached.

```
MACSbug 3.n*TM
```

Having transmitted the memory images desired, the UPLOADS session can be terminated within VERSAdos and it can be verified that there were no errors.

It is now necessary to reconnect the terminal (on port 1) with the MCCM to EXORmacs (on port 2).

QUIT

The QUIT directive is entered to check for any errors, save the new file, and terminate the UPLOADS program. The program should respond with the following:

```
volume=FIX
 catlg=
  file=FILE01
   ext=MX

*STATUS*   No error since start of program

Upload of S-Records complete.
```

=OFF

The program is now complete but the VERSAdos session is still active. Typing the OFF command frees the port on EXORmacs for another user. (BYE is also acceptable.)

The file FIX:<user number>..FILE01.MX has been released and is now ready for any future requirements.

ERROR MESSAGE GENERATOR (EMFGEN)

4.2.26 Error Message File Generator Program (EMFGEN)

The Error Message File Generator program (EMFGEN) reads sequential records from ERRORMSG.SA (the input file) and generates indexed sequential records in ERRORMSG.SY (the output file). Upon execution, the current ERRORMSG.SY file is deleted and a new one created. This program enables the user to add his own error messages and/or reformat existing messages. New messages should be added at the end of the ERRORMSG.SA file, rather than at the beginning. A description of the ERRORMSG.SA file can be found in the VERSAdos Messages Reference Manual, Appendix E.

This program uses the error message handler program for any errors it might encounter, and the error message handler program requires the ERRORMSG.SY file to execute properly. Since the error message file generator program is bulding a new ERRORMSG.SY file, the error message handler program is not able to access it. Consequently, it is recommended that the ERRORMSG.SA file be copied to another user number to generate the new ERRORMSG.SY file. When this function has been successfully completed, the new ERRORMSG.SY file can then be copied back to user 0.

When EMFGEN is initiated, the input file ERRORMSG.SA must be available under default user volume and user number.

4.2.26.1 EMFGEN Command Syntax.

EMFGEN is called from VERSAdos as follows:

```
EMFGEN
```

4.2.26.2 Error Message File Format for ERRORMSG.SA.

The ERRORMSG.SA format can have two kinds of records: comment and data input. Comment records give the user the flexibility of documenting pertinent information, while data input records are used as input to create keyed records for the output file, ERRORMSG.SY.

General Record Formats

a. Comment record

COLUMN	REMARKS
01	Contains the identifier (*) which will denote this as a comment record.
02-80	Contains user comments.

b. Data input record

COLUMN	REMARKS
01-08	The hex notation of these eight bytes will be converted to four binary bytes and used as the key value of the error message in the output file.
09	Contains a delimiter which must be a space character.
10-80	Contains the error message text and substitution sentinels.

Data Input Record Key Value Format

The basic format of the key value field is the same as the format of register D0 following a trap call. The subfield definition is as follows:

BITS	CONTENT
00-15	Error status code
16-26	Directive number
27-30	Trap number
31	Reserved for internal error classification

a. Error status code - This is a unique hex value identifying the type of error.

b. Directive number - This is a unique number within a trap call that identifies for the trap logic the appropriate path of execution to be followed.

c. Trap number - This number identifies the type of trap to execute. A trap number of zero (0) is used for non-trap-related messages to be displayed by the error message handler program.

Substitution Sentinel Format

The basic substitution sentinel can have one of the three following formats:

a. \Soo
b. \Sooxx
c. \Syyyy

where:

\	Is a flag identifying this as a substitution sentinel.
S	Is a substitution sentinel code.
oo	Is a decimal offset to be added to a base register address supplied to the error message handler program. The information at the offset will be processed according to the substitution code and inserted into the error message.
xx	Is a length value identifying the number of characters to move. This is used only with substitution sentinel code C.
yyyy	Is a key value that is required only for substitution sentinel code K.

Except for sentinel I and K, the decimal offset (oo) plus the number of bytes involved processing the sentinel cannot exceed 48.

Substitution Sentinel Codes

The following substitution sentinel codes and their interpretations are used to construct error messages that are displayed by the error message handler program.

a. Substitution sentinel codes \Boo, \Woo, \Loo

For each of these codes, the offset is added to the base address supplied to the error message handler program and respectively one, two, or four bytes of data specified by the resultant address are converted to decimal ASCII.

b. Substitution sentinel code \Xoo, \Yoo, \Zoo

For each of these codes, the offset is added to the base address supplied to the error message handler program and respectively one, two, or four bytes of data specified by the resultant address are converted to hex ASCII.

c. Substitution sentinel code \Cooxx

The offset is added to the base address supplied to the error message handler program, and xx bytes of data from this address are transferred to the error message.

d. Substitution sentinel code \Kyyyy

The record containing the hex key value yyyy will be retrieved from the ERRORMSG.SY file. This part of the error message will be preceded by a carriage return, a line feed, and have all embedded sentinels expanded before continuing the interpretation of the message. For example, if the key value yyyy contained the message "CMD=\T00 OPT=\T02 LU=\B05", then the embedded sentinels would be interpreted and this part of the message would be preceded by a carriage return and line feed prior to being inserted into the error message being built. Nested K sentinels are invalid.

e. Substitution sentinel code \Too

This sentinel requires that a code be translated into its ASCII prose equivalent. To accomplish this, a dynamic key value will be built by the error message handler program, the corresponding record will be read from ERRORMSG.SY, and the ASCII prose equivalent will be inserted into the error message being built. The format for the dynamic key value is "ttoocccc".

where:	tt	Is the original trap number plus the bit setting required to indicate a trap four (bit number 29).
	oo	Is the original offset in this substitution sentinel.
	cccc	This value is determined by using the contents of the address determined by adding the offset to the base address supplied to the error message handler program.

f. Substitution sentinel \Aoo

Part of the required information that is passed to the error message handler program is register A0. If the user wishes to have the address specified by this register converted into hex ASCII and displayed in the error message being built, then this sentinel should be used. The offset used with this sentinel must be zero (00).

g. Substitution sentinal \Ioo

The user may pass to the error message handler program the address of a message plug pool which contains multiple strings of text separated by the hex delimiter "FF". This sentinel will result in the deblocked text being inserted into the basic error message, thus providing the caller with a way to include variable message content from a source external to the parameter blocked that is passed to the error message handler program. Plugs within the plug pool are numbered beginning with one, so "\I02" will access the text following the first hex delimiter "FF". The plug pool always starts with a text string, never with the delimiter "FF".

h. Substitution sentinel \Doo

The standard error message displayed by the error message handler will include the task name, session number, binary key value requested, caller, and a message delimiter. If the user wishes to eliminate this part of the error message, he should use this sentinel. It is recommended that this sentinel be the last one in the error message. The offset used with this sentinel must be zero (00).

Key Values Used with Substitution Sentinel Code \Kyyyy

The following hex key values are examples of sentinel codes that can be used:

KEY VALUE	MESSAGE
00000000	\I01
0000000C	CMD=\T00 OPT=\T02 LU=\B05
0000000D	CMD=\T00 OPT=\T02 LU=\B05 DEVICE=\C0604
0000000E	CMD=\T00 OPT=\T02 LU=\B05 PSN=\Z08 DEVICE=\C0604
0000000F	FILE=\C0604 :\W10 .\C1208 .\C2008 .\C2802
00000010	CMD=\T00 OPT=\T02 LU=\B05 PSN=\Z36

Error Message Key Values

The following will define the key values currently associated with a given error message set. Non-trap-related error message key values from hex 00008000 through 0000FFFF are reserved for customer use.

UTILITIES	00000000 - 000002FF
SESSION MANAGEMENT	00000300 - 000004FF
SYMBUG	00000500 - 000006FF
PASCAL RUN TIME	00001000 - 00002FFF

Record Examples

The following will illustrate some of the types of error message records that could exist.

```
*
*
*         Any record with an asterisk (*) in column one is a comment record.
*
*
0000010C  Invalid entry
08010006  Duplicate segment name (GTSEG)
10000084  Invalid data buffer \K000C BUFF=\Z12
18000017  Nonexistent file name \K000C \K000F
*
*
*          MESSAGE KEY VALUE      TYPE ERROR MESSAGE

              00000100            Non-trap-related
              08010006            Trap 1 related
              10000084            Trap 2 related (IOS)
              18000017            Trap 3 related (FHS)
```

4.2.26.3 EMFGEN Output. Upon completion, the program outputs the new ERRORMSG.SY file under default user volume and user number, along with the following message:

```
'XXXXXX X  NNNN RECORDS NOW IN ERRORMSG.SY FILE'
```

where: XXXXXX X Represents the program revision code.

NNNN Represents the number of records in the file.

All errors encountered by this program will be processed via the error message handler program. The potential errors that can occur are as follows:

a. An assign error via the file handling system on ERRORMSG.SA
b. An assign error via the file handling system on ERRORMSG.SY
c. A read error via the I/O system on ERRORMSG.SA
d. A write error via the I/O system on ERRORMSG.SY
e. Key value XXXXXXXX will not be added to file ** input error
f. Key value XXXXXXXX error message added but truncated to 84 bytes

4.2.27 Error Message Handler

The error message handler program is a system server task that will provide standardized error message displays in response to exception conditions, thus relieving the user task from the responsibility of maintaining his own error message list.

The program will retrieve the key value of the message to be displayed from the ERRORMSG.SY file, which must reside on the system volume, expanding any embedded sentinels, and then present the error message to the user-specified output device. The expansion of sentinels C, I, and T will replace nondisplayable characters, except null, with a period (.).

4.2.27.1 Program Requirements. The following must be true at program input:

a. The error message file ERRORMSG.SY on system volume, user number 0.

b. Register A0 equals the address of the error message parameter block.

c. Register D0 equals the directive 2, to initiate the error message handler program.

Program output consists of the display of an error message.

4.2.27.2 Error Message Parameter Block Format. The error message parameter block is the mechanism by which the user and the error message handler program communicate. Its format is as follows:

FIELD BYTE SIZE	FIELD DESCRIPTION
Four	Contains the binary key value of the error message to be displayed.
Four	Address of the parameter block for trap-related calls.
One	Available for future use.
One	Logical unit number, assigned to the user, of the device or file to which the output error message is transmitted.
Two	Available for future use.
Four	Contains the start address of the user message plug pool if one exists.
Four	Contains the end address of the user message plug pool if one exists.

Binary Key Value

In the case of trap-related calls, the contents of register D0, as it is returned to the user task, must be stored in this field since it will contain the binary key value of the error message to be displayed. For non-trap-related error messages, this field needs to be set to the appropriate binary key value for the desired error message.

Address of Parameter Block for Trap-Related Calls

If the error message to be displayed is the result of a trap call, this field contains the address of the trap parameter block associated with the call. Pertinent information from this parameter block can then be extracted and used to build the resultant error message. If the error message is non-trap-related, this field should be set to zero.

Error Message Plug Pool

The error message plug pool contains multiple strings of text separated by the hex delimiter 'FF'. The user may specify not only which plug text he would like inserted into the error message, but also where he would want it inserted. This flexibility allows the user to include variable message content and can be accomplished via the substitution sentinel code \Ioo. The maximum plug pool length for a given plug pool is 96 bytes.

4.2.27.3 Program Initiation. The error message handler program is initiated via a trap 4, directive 2 system service request.

Example for a trap-related call

```
.
.
.
LEA      IOSBLK,A0           A0=address of IOS parameter block
TRAP     #2                  Initiate the IOS trap call
BEQ      CONTINUE            Jump if the trap was successful
MOVEM.L  D0/A0,EMHPBLK       Initialize the error message handler parameter
                             block with the binary key value and the address of
                             the IOS parameter block
LEA      EMHPBLK,A0          A0=address of error message handler parameter
                             block
MOVE.L   #2,D0               D0=directive 2
TRAP     #4                  Initiate call to error message handler program
.
.
.
```

Example of a non-trap-related call

```
   .
   .
   .
MOVE.L   #EMHBKV,EMHPBLK     Initialize the error message handler parameter
                             block with the binary key value
LEA      EMHPBLK,A0          A0=address of error message handler parameter block
MOVE.L   #2,D0               D0=directive 2
TRAP     #4                  Initiate call to error message handler program
   .
   .
   .
```

4.2.27.4 Error Message Format. All error messages displayed from the error message handler program will display a standard error message format which will consist of the following, unless requested otherwise:

Task name
Session number
The binary key value requested
The type of call -- user, IOS, FHS, service request, loader
Message delimiter "**"
User message

4.2.27.5 Errors. Errors internal to the error message handler program will have the same basic format as all other error messages. The user message part of the total error message will be standard in all cases with "EMH ERROR #n" as the message, where n represents the internal error number. The message "EMH ERROR #n" is followed by a hex value which is the contents of D0. This value normally represents the error code that the message handler encountered when trying to output the requestor's original message.

The value of n and its interpretation are as follows:

ERROR NUMBER	MEANING
0	Available for future use.
1	Error encountered changing the user's output logical unit number.
2	Error encountered trying to assign ERRORMSG.SY. This will occur if an exclusive read or write assignment exists on the system volume.
3	Error encountered trying to read the user's binary key value.
4	Error encountered trying to read the binary key value of sentinel code \K.
5	Available for future use.

6	Error encountered attempting to write to the user's output device.
7	Error due to nested K sentinels.
8	Error message being built expanded such that the source creating the message was destroyed.
9	Error encountered receiving logical unit number from user.
10	Error encountered sending logical unit number back to user.
11	Error encountered trying to move the caller's A0 parameter block or the caller's plug pool.
12	An error has occurred because the plug pool size specified by the caller exceeds the maximum allowed size of 96. The D0 value displayed after the error message handler number will contain the plug pool size requested.
13	An error has occurred because upon examination of the plug pool, an insufficient number of plug pool delimiters was discovered. The D0 value displayed after the error message handler number will contain the number of delimiters originally requested.

If a user is executing multiple tasks from a given terminal, it is potentially possible that an error message will not be displayed. This will occur when one task has the output device in a busy state, requesting input, and another task would like to output an error message to that device.

HARD DISK SYSTEM UTILITIES

SPL

4.3 HARD DISK SYSTEM UTILITIES

Beginning with revision 3.0, VERSAdos supports several utilities which are supplied for use with hard-disk based systems only. These are the Dispatch utility (DISPATCH), the Account utility (ACCT), the Validate Password utility (VALID), the Invalidate Password utility (NOVALID), the SESSIONS utility, and the Spooling utilities (SPL and SPOOL). DISPATCH is used in batch mode and is described in Chapter 3. ACCT, VALID, and NOVALID are part of the security subsession and are described in Chapter 5. Detailed explanations of SPL, SPOOL, and SESSIONS are provided in the following paragraphs.

4.3.1 Spooler Task Utility (SPL)

The purpose of a spooling capability is to increase the overall throughput of a given system by increasing the amount of time the processor and terminals are available during I/O with the slower peripherals, usually the printer(s). The general approach requires that input be read from and output be written to auxiliary storage concurrently with job execution. The form of the input/output should be suitable for later processing or output operations.

VERSAdos provides a spooling function by means of a system task which is installed by the system administrator (user 0) when the function is desired. To provide control, this spooler task -- named SPL -- utilizes a spooler queue file of the name and extension SPLQUEUE.SQ. SPLQUEUE.SQ contains a description of the name, size, location, and queue position of each spooler file. The latter is a file created by the spooler task on a volume designated the "spooler" volume, whose function is to contain all spooler files.

Once SPL is resident, interface with any peripherals designated spooling devices is under control of the task. Interface with the spooler queue file is made via the utility SPOOL, which permits the user to view the queue file and effect certain changes in his queue file status and requirements as described below. The SPOOL utility also allows the user to request output of non-spooler files on designated spooling devices. (Non-spooler files are those for which output, at creation, was not directed to a spooling device and for which, consequently, the spooling task did not create a copy on the spooler volume and a corresponding entry in the spooler queue file.) The spooling function can be removed from the system by the administrator, when required.

The operating system VERSAdos has been tailored so that all printers are designated as Centronics-compatible spooling devices. To remove the spooling device status from a printer, or to add a printer and designate it as a spooling device, the user's operating system file -- VERSADOS.SY -- must be regenerated using the SYSGEN process. This permits the appropriate descriptions to be installed in the Device Control Blocks. Refer to the section describing SYSGEN for details.

Installing the Spooling Utility

The task SPL must be installed by the system administrator (user 0) at the time the system is brought up. At that time, the administrator designates the volume to hold the spooler queue file and all spooler files subsequently created by the task.

SPL Command Syntax

```
    SPL <spooler volume name>
```

where:

spooler volume name	Is the volume to use for spooler files.

Examples:

=SPL	SYS0	Designate volume SYS0 as the spooling volume.
=SPL	SYS0:	Designate volume SYS0 as the spooling volume.

If the spooling loader is called again after spooling is active, the message "ER. DUPLICATE TASK" will appear.

Once installed, the spooling function remains in force until discontinued by the system administrator or the system is rebooted. To remove the function in mid-session, the session control command TERM is used in conjunction with the spooling task name and a special session number as follows:

```
    TERM .SPL &1
```

If the disk device containing the spooler queue file is to be taken offline, SPL must be terminated and restarted as follows:

ACTION	EFFECT
TERM .SPL &1	Terminate spooler task utility to close spooler queue file.
Take device offline.	Remove current spooling device.
Put new device online and ready device.	Install new spooling device.
SPL <spooler volume name>	Install spooler utility.

SPOOL

In general, user interface with the spooling function merely requires requesting that the output of some process be made to a designated spooling device with the spooling task handling the user's requirements from that point on. To accommodate the normal exigencies of computer use, a utility named SPOOL is provided which offers a repertoire of nine commands that allow limited manipulation of the spooling function with respect to the user's own jobs. Provided capabilities include starting, stopping, continuing, cancelling, adding and viewing jobs, changing the forms ID of a spooler device, and changing the number of desired copies. Details of the syntax and use of these commands are provided below.

Spooling Utility Commands

The following table lists the SPOOL utility commands and gives a brief description of the general functions they provide. Full descriptions are provided in the following pages.

COMMAND	ACTION
CANCEL	Cancel a spooler job
CONTINUE	Continue a spooler job
COPIES	Change number of copies of a job output
FORMS	Change forms ID for a spooler device
HELP	List the commands and command syntax
PRINT	Add a non-spooler file to the spooler queue
QUEUE	View the spooler queue file
QUIT	Terminate SPOOL (interactive mode only)
START	Output a spooler file now

General Command Notation

Any command may be entered by using the first four characters of the command. Also, PRINT may be entered as "P", FORMS as "F", START as "S", and QUEUE as "Q". The delimiter preceding <argument> may be either a blank space or a comma. The interactive mode prompt is ">". In interactive mode, the user is prompted for a new command until "QUIT" is entered or SPOOL terminates because of an error. Both upper- and lowercase letters are valid.

Specifying SPOOL and one of the utility's nine commands and its arguments, if any, on the command line causes execution of the command and return to VERSAdos control. Specifying SPOOL alone causes entry into the utility's interactive mode, which is indicated by display of a greater than symbol (>) at screen left. Any SPOOL utility command can then be used.

SPOOL Command Syntax

```
    =SPOOL [<command>[,<argument>]]

                    or

    =SPOOL [<command>[<space><argument>]]
```

where:

command	Is any SPOOL utility command.
space or ,	Is a required delimiter if an argument follows.
argument	Is a value on which the specified command operates.

HELP

QUEUE

4.3.2.1 Help Command Syntax (HELP). The HELP command is used to view the available commands and their syntax:

```
    HELP
```

HELP Command Example

```
=SPOOL
>HELP     (this outputs the list of commands)
>         (enter any command here)
```

4.3.2.2 Queue Command Syntax (QUEUE). The QUEUE command is used to display spooler queue file information.

```
    Q[UEU[E]] [<argument>]
```

where:

Q, QUEU, or QUEUE	Is the command.
argument	May be one of the following: . user number . "S" for session . device name . "F" or "FORM"

Information in the queue file for each job includes:

a. A spooler device designation such as PR1 that shows to which device the job file will be sent for output.

b. A 4-character forms ID - either the default ID "STND" if the job file is a spooler file or one of the user's choice (via the PRINT command) if a non-spooler file.

c. A job ID consisting of the user's user number plus four digits added by the spooling task to provide unique identification for each user job.

d. The volume ID, catalog name, file name and extension of the file to be output by that job.

e. The name of the utility which generated the job file.

f. The current status of that job.

Any user may invoke the QUEUE command at any time. If a spooler device has never had an assignment made to it, the device name will not be in the form-device listing.

The following table lists the various forms the QUEUE command argument may take and the corresponding function provided.

SPECIFY	ACTION
NO ARGUMENT	List all spooler queue file entries.
USER NUMBER	List all entries for that user.
S	List all entries for the current session.
DEVICE NAME	List all entries for the specified device.
FORM	List the spooler device names, the current forms ID setting and current status.

QUEUE Command Examples

NON-INTERACTIVE MODE	INTERACTIVE MODE	ACTION
	=SPOOL	
=SPOOL Q	>QUEUE	List the entire queue.
=SPOOL QUEUE 313	>QUEUE 313	List all entries for user #313.
=SPOOL QUEUE S	>QUEUE S	List all entries for the current session.
=SPOOL QUEUE #PR	>Q #PR	List all entries for specified device.
=SPOOL QUEUE F	>Q FORM	List all spooler device names and their current forms ID and current status.
	>QUIT (or any other command)	

Example of listing the entire queue

```
DEV   FORM   JOBID     VOL    CATALOG    FILENAME      TASK   COPY  STATUS

PR    STND   3130100   SPOL              @1234567.89   RASM      1  WAIT-FORMS

PR1   ABCD   3130413   VOL3   TESTFILE   TEST3.SA      SPOL      2  OUTPUT NOW

PR2   STND    440003   SPOL              @ABCDEFG.34   LINK      1  WAIT-CONT
```

The following table explains the messages displayed in the Status column.

STATUS	MEANING
READY	Ready to output.
ACTIVE	File not closed, being built.
OUTPUT NOW	File being output to a spooler device.
WAIT-CONT	File cannot be output until a CONTINUE is issued. Forms ID for the device matches the entry forms ID.
WAIT-FORMS	File cannot be output until the forms ID is changed to match the entry forms ID and a CONTINUE is issued.
IO ERR DO=zz	File was being output but encountered an I/O error. The file cannot be output until the reason for the error has been corrected and a CONTINUE is issued.
CANCEL	The file is currently being output and a CANCEL was issued for it. The file will be canceled as soon as the next file record is output.

The following type of display results when a list of device names and their forms ID's are requested.

```
DEVICE   FORMS ID    STATUS  (OK = OK TO OUTPUT; WAIT = WAITING FOR CONTINUE)

PR       1234        OK
PR1      ABCD        OK
PR2      STND        WAIT
```

CANC

4.3.2.3 <u>Cancel Command Syntax (CANC)</u>. The CANCEL command is used to cancel spooler output and delete the spooler file.

```
    CANC[EL] [<argument>]
```

where:

CANC or CANCEL	Is the command.
argument	Is an optional value which may be a job ID or a file name. A non-spooler file name may be specified if it exists in the spooler queue file. Non-spooler file name default values for volume, catalog, and user number are the standard default values.

The CANCEL command may be specified at any time. Both the spooler file and the corresponding queue file entry are deleted. Only the queue file entry is deleted for a non-spooler file. User 0 may cancel any job; otherwise, only the file owner may cancel a job. User 0 must specify either a file name or a job ID. Failure to do so will result in an error message. Present versions of VERSAdos do not allow use of the CANCEL subcommand to cancel all spooler jobs. Future versions will allow user 0 to either cancel all spooler jobs or output a message.

The following table lists the CANCEL command argument forms and the general functions provided.

SPECIFY	ACTION
NO ARGUMENT	Cancel all jobs for the logged on user number
FILENAME	Cancel the specified file
JOBID	Cancel the specified spooler job

<u>CANCEL Command Examples</u>

NON-INTERACTIVE MODE	INTERACTIVE MODE	ACTION
	=SPOOL	
=SPOOL CANCEL	>CANCEL	
OK TO CANCEL ALL YOUR FILES (Y/N)?	OK TO CANCEL ALL YOUR FILES (Y/N)?	
		Cancel all user jobs if a Y response is given.
=SPOOL CANCEL 3130049	>CANCEL 3130049	Cancel this job.
=SPOOL CANCEL @ABCDEFG.34	>CANCEL @ABCDEFG.34	Cancel this spooler file.
=SPOOL CANC VOL3:313..OUTFILE.SA	>CANC OUTFILE.SA	Cancel this non-spooler file.
	>QUIT (or any other command)	

CONT

When no argument is specified, the spooler prompts with the message:

```
    OK TO CANCEL ALL YOUR FILES (Y/N)?
```

and waits for entry of Y (yes) or N (no). If Y is entered, all files for the logged-on user number are removed from the queue; if N, the spool prompt (>) is returned to.

4.3.2.4 Continue Command Syntax (CONT). The CONTINUE command is used to restart spooler output to a spooler device.

```
    CONT[INUE] <device name>
```

where:

CONT or CONTINUE	Is the command.
device name	Is the name of a spooler device.

CONTINUE is used to start output to the device after the forms ID associated with the device is changed, or to restart output after an I/O error is encountered on the specified device. CONTINUE is ignored if the device is busy. Any user may issue the CONTINUE command.

CONTINUE Command Examples

NON-INTERACTIVE MODE	INTERACTIVE MODE	ACTION
	=SPOOL	
=SPOOL CONTINUE #PR	>CONT #PR	Start output on this device.
=SPOOL CONT #PR1	>CONT #PR1	Start output on this device.
	>QUIT (or any other command)	

FORM

4.3.2.5 Forms Command Syntax (FORM). The FORMS command is used to stop spooler output when the current job is completed, and to change forms ID for the specified spooler device.

```
    F[ORM[S]] <device name>,<forms ID>
```

where:

FORMS, FORM, or F	Is the command.
device name	Is the name of a spooler device.
forms ID	Is a name formed of four alphanumeric characters.

Execution of the FORMS command stops output to the specified device (the current job is first completed) and marks that device with the specified forms ID. Then the spooling task will send only jobs with the matching forms ID to the device for output. The spooling utility command CONTINUE must be used to start output to a device following change of that device's forms ID. The forms ID is similarly changed back to the standard default value by specifying 'STND' in the forms ID field of the Forms command. Any user can issue the Forms command.

FORMS Command Examples

NON-INTERACTIVE MODE	INTERACTIVE MODE	ACTION
	=SPOOL	
=SPOOL FORMS #PR,ABCD	>FORMS #PR,ABCD	Change the forms ID to "ABCD" for this device; suspend output until a CONTinue command is issued.
=SPOOL CONT #PR	>CONT #PR	Output spooler files to this device which have a forms ID of "ABCD".
=SPOOL F #PR,STND	>F #PR,STND	Change forms ID to "STND" (standard) and suspend output until a CONTinue command is issued.
=SPOOL CONT #PR	>CONT #PR	Output standard forms ID spooler files.
	>QUIT (or any other command)	

PRIN

4.3.2.6 Print Command Syntax (PRIN). The PRINT command is used to add a non-spooler file name to the spooler queue file.

```
P[RIN[T]] <argument>
```

where:

PRINT, PRIN, or P	Is the command.
argument	Is specified as follows:

<non-spooler filename>,<device name>[,<forms ID>]
[,<# of copies>]

where:

non-spooler filename	Is the descriptor of a VERSAdos file. A minimum of the file name and extension fields is required. Standard default values are used for unspecified fields in the descriptor.
device name	Is the name of a spooler device.
forms ID	Is a name formed of four alpha-numeric characters.
# of copies	Is a decimal number from 1 through 255.

The default values for the optional "forms ID" and "# of copies" fields are "STND" and 1, respectively. A specified non-spooler file must exist at the time the PRINT command is issued -- i.e., the file cannot be in the process of being created through use of the CRT editor or by assembly, etc. Execution of the PRINT command immediately creates an entry in the spooler queue file for the specified file. On completion of output, the queue entry (only) is deleted.

PRINT Command Examples

NON-INTERACTIVE MODE	INTERACTIVE MODE	ACTION
	=SPOOL	
=SPOOL PRINT TEST.SA,#PR	>PRINT TEST.SA,#PR	Add this non-spooler filename to the spooler queue file. Device name=PR, forms ID=STND, # of copies=1
=SPOOL PRINT TEST1.SA, #PR1,ABCD,4	>PRINT TEST1.SA, #PR1,ABCD,4	Add this non-spooler filename to the spooler queue file. Device name=PR1, forms ID=ABCD, # of copies=4
=SPOOL P SYS:313..TEST2.LS, #PR,,2	>P SYS:313..TEST2.LS, #PR,,2	Add this non-spooler filename to the spooler queue file. # of copies=2
	>QUIT (or any other command)	

STAR

4.3.2.7 Start Command Syntax (STAR). The START command starts immediate output of a job file, regardless of existing spooler task priority.

```
    S[TAR[T]] <argument>
```

where:

START, STAR, or S	Is the command.
argument	Is specified in either of two forms: <job ID>[,<# of copies>] or <spooler filename>[,<# of copies>]

where:

job ID	Is the 5- to 8-digit number in the queue file for that job.
# of copies	Is a decimal number from 1 through 255.
spooler filename	Is the name of a spooler file.

START Command Examples

NON-INTERACTIVE MODE	INTERACTIVE MODE	ACTION
	=SPOOL	
=SPOOL START 3130149,4	>START 3130149,4	Output four copies of this job next.
=SPOOL START 3130035	>START 3130035	Output this job next using the number of copies already in the Queue entry.
=SPOOL START @1234567.89	>START @1234567.89	Output this spooler file next using the number of copies already in the Queue entry.
	>QUIT (or any other command)	

COPI

SPOOLING UTILITY ERROR MESSAGES

4.3.2.8 Copies Command Syntax (COPI). The COPIES command syntax is used to change the desired number of copies.

```
    C[OPI[ES]] <argument>,<# of copies>
```

where:

COPIES, COPI, or C	Is the command.
argument	Is the 5- to 8-digit job ID existing in the queue file for that job, or the name of a spooler file (or non-spooler file if an entry for the file exists in the spooler queue file).
# of copies	Is a decimal number from 1 through 255.

The COPIES command may be used at any time. User 0 can specify any file name. Other users can specify only their own files. A minimum of the file name and extension fields is required when specifying a non-spooler file. Standard default values are used for other fields in the descriptor, if not specified.

COPIES Command Examples

NON-INTERACTIVE MODE	INTERACTIVE MODE	ACTION
	=SPOOL	
=SPOOL COPIES 3130010,4	>COPIES 3130049,4	Output four copies of this job.
=SPOOL COPIES TEST.SA,1	>COPIES TEST.SA,1	Output one copy of this file name.
	>QUIT (or any other command)	

4.3.2.9 Spooling Utility Error Messages

Syntax errors found before any command is executed result in a list of the commands and their basic syntax, followed by the prompt (>) for correct input. For any other errors, refer to the VERSAdos Messages Reference Manual.

SESSIONS

4.3.3 Sessions Utility (SESSIONS)

The SESSIONS utility is used to determine the current online sessions and the batch jobs in queue for execution. Information is displayed by device number (terminal) and sessions number for online sessions and by user number and session number for batch jobs.

SESSIONS Command Syntax

```
    SESSIONS
```

SESSIONS Utility Example

```
=SESSIONS
 ONLINE SESSIONS (TERMINAL/SESSION):
 CN12   0043
 CN10   0068
 CN11   006B
 CN21   0040
 CN00   0067
 CN13   006D
 BATCH JOBS     (USER/SESSION):
  212   007C
=
```

CHAPTER 5 SECURITY ADMINIS- TRATION

CHAPTER 5

SECURITY ADMINISTRATION

5.1 INTRODUCTION

This chapter provides information on the system security capability, as well as procedures for modifying the security facilities to suit the user's purposes. The security package can be excluded from or included in the system at SYSGEN time.

5.2 VERSAdos SECURITY ADMINISTRATION

Four levels of system and user security are provided for multiuser hard-disk M68000 family VERSAdos systems. (Security can be established for single-user systems by modifying the SYSGEN command file.) VERSAdos security is attained using a security level flag, a system security word residing in the system memory space, and a user password file on disk. Interfacing with these are three system control commands which allow the administrator and users to establish and observe the desired system and personal security. Two system utilities provide further interface between administrator and password file to permit control of system access by user number. The system security level is established --usually at the time the system is initially brought up -- by the administrator (user 0). Personal security is established by the users for themselves.

The four security levels are discriminated according to the state of eight bits representing a combination of user 0 or not and session 1 or not established in the security level flag partly by the administrator and partly by the boot load file. According to the level, a user will be granted access to the system at logon only if he matches the system security word, his password, both, or if no match is required. The levels are:

Level 0 - No security. Any user number is given logon access in all sessions.

Level 1 - An administrator-specified system security word exists. To log on, each user must match the system security word on request.

Level 2 - A password file exists which contains a list of user numbers validated by the administrator and a user-specified password (or not) corresponding to each valid user number. If a password exists for a valid user, a match is requested at logon. If no password exists, no match is needed and, therefore, is not requested.

Level 3 - Both a system security word and a password file exist. The logon requirements of both level 1 and level 2 must be met by a user.

The diagram in Figure 5-1 shows how the security level flag is set to reflect the four levels of security.

```
<-------- USER = 0 ------->|<------- USER ≠ 0 -------->
---------------------------------------------------------
| PW  |  SW  |  PW  |  SW  |  PW  |  SW  |  PW  |  SW  |
---------------------------------------------------------
  SESSION 1  |     ≠ 1     |  SESSION 1  |     ≠ 1
```

LEVEL	PW/SW PAIRS		REQUIRED MATCH
0	0	0	None
1	0	1	System word
2	1	0	Password
3	1	1	Both

FIGURE 5-1. VERSAdos Security Level Flag

5.2.1 The Initial Use of VERSAdos Security

The bootload file, VERSADOS.SY, as supplied, writes all zeros into the system security level flag when the system is booted. Therefore, level 0 security exists before session 1 is initiated. For the system to come up in another level requires that the bootload file be changed using the PATCH utility, since no provision is made in the security program for such a change. (Paragraph 5.3 provides instructions for making this change.)

When the administrator initially establishes level 1 security through use of the SECURE command, he must supply the word that all users will subsequently be required to match for access to the system. This system security word, or S-word, is established in system memory by means of the SWORD command, and remains in force until the security level is changed to 0 or 2 or the S-word is changed through execution of another SWORD command.

When the administrator initially establishes level 2 security, he must execute a VALID command (paragraph 5.2.5) to create a password file on the default system volume which must then and subsequently be available to the security program. The VALID command uses the list of user numbers supplied by the administrator to build an indexed sequential file in which each user number is the key to a corresponding password record. The password record created for each user number specified on a VALID command line is set to zero (nulled) so that (1) no password match is required for a user's initial access to the system, and (2) the administrator can remove the match password requirement for a user who has forgotten his password.

Subsequent to the administrator's validation of a user's number, the user may, if he chooses, install his personal password in the password file through use of the Password (PASS) command (paragraph 5.2.4). A match will then be requested when that user number is supplied at the initiation of subsequent sessions until the password is set to zero or security level 0 or 1 is placed in force.

The initial establishment of level 3 security by the administrator requires that both a system security word (S-word) is supplied and a password file is created, as described above.

SECURE

SWORD

5.2.2 Secure System Control Command (SECURE)

The SECURE command is for use by the system administrator (user 0) to specify a desired level of system security for subsequent sessions. An appropriate level for the administrator himself and an appropriate level for all other users are communicated to the SECURE command. Upon execution, the state of the system security flag is changed to correspond to the prescribed conditions. This correspondence is shown in the security level flag diagram shown earlier. Note that the SECURE command can only affect the flag bits for sessions other than session 1. The state of the other bits is set during bootload, as previously mentioned. This means that only the running system is altered by the SECURE command. The bootload file is not affected.

<u>SECURE Command Syntax</u>

```
    SECURE
```

The following dialog is initiated with the SECURE command:

```
    LEVELS ARE 0/NONE, 1/SWORD, 2/PWORD, 3/BOTH
    USER = 0:     (The administrator specifies a security level for himself)
    USER NOT 0:   (The administrator specifies a security level for all other
                   users)
   =
```

The specified security levels are echoed to the terminal.

If BREAK is entered, the existing level of security is displayed and remains unchanged. Likewise, if a non-digit or (CR) without a digit is entered, the existing level is displayed and remains unchanged.

5.2.3 S-Word System Control Command (SWORD)

The SWORD command is for use by the system administrator (user 0) to change the system security word in a running system (the bootload file is not affected). An S-word is comprised of up to eight ASCII characters.

<u>SWORD Command Syntax</u>

```
    SWORD
```

The following dialog is initiated with the SWORD command:

```
    ENTER NEW SECURITY WORD  (The administrator specifies a security word. The
                              entry is not echoed to the terminal.)
   =
```

A response of (CR) clears the current S-word, effectively removing the match S-word requirement for all users. Thus, although the system security flag might reflect level 1 or level 3, the security program on finding a null S-word, does not request a match when a user logs on. The administrator must subsequently enter an S-word through use of the SWORD command to restore the match S-word requirement.

A response of BREAK causes the input to be ignored and the existing security word to remain unchanged.

PASS

5.2.4 Password System Control Command (PASS)

The PASS command can be used by all users to enter characters into or clear the password portion of their own password record. A password entry will only be accepted for a user number subsequent to validation of that user number by the administrator, using the VALID utility.

The PASS command is used in two circumstances: 1) a null password exists, or 2) an actual password exists. In the first case, the security program on finding a null password does not request a match and immediately allows the user to specify a password. In the second case, the user must match his password before being allowed to change or clear his password. A password is comprised of from one to 12 ASCII characters.

PASS Command Syntax

```
    PASS
```

where:

The following dialog is initiated with the PASS command:

ENTER EXISTING PASSWORD	(This message is displayed only if a non-null password exists. The user responds with his password, which is not echoed to the terminal. Failure to enter the password within the number of tries specified in SYSGEN results in being logged off.)
ENTER NEW PASSWORD	(The user enters (CR) to clear his password or specifies a password, which is not echoed to the terminal. Entry of BREAK causes the existing password to be retained.)

=

Should the user desire to eliminate the requirement of matching his password to log on, he may respond to the ENTER NEW PASSWORD message with only a carriage return, which clears his password but does not invalidate his user number. Should a user not know his password, the administrator must clear the password through use of the VALID command to remove the match password requirement for the user. If the user enters BREAK when a new password is requested, the input is ignored and the password remains unchanged.

VALID

5.2.5 Valid System Utility (VALID)

VALID is a system utility for use by the administrator to control access to the system by user number for security levels 2 and 3. Control is achieved by means of an indexed sequential file on the default volume that is comprised of password records to which user numbers are the keys. Specifying a user number to the VALID command results in the creation in the file of a record containing a null password entry for that user number. The null value can be replaced by an actual password supplied to the PASS command by the user. The security program then requires, for security levels 2 and 3, that the password be matched before allowing a session to be initiated for that user number.

Password match is not required for null passwords. A user can clear his own password record using the PASS command (but only after matching his password). The administrator can clear the password record for any user number using the VALID command.

A single user number or a range of user numbers can be specified to the VALID command. (A user number is any decimal number from 0 through 9999.) Specifying a non-valid user number creates a record containing the user number and a null password in the password file. Specifying a valid user number clears the password portion of the record for that number. Specifying a valid user number with the V option will create a record if not existing, but will not reset the password in an existing record. The V option allows the system administrator to maintain a chain file of VALID (and NOVALID) commands. Without the V option, the chain file would reset all of the existing passwords.

<u>VALID Command Syntax</u>

```
    VALID <user no.>[<separator><last user no.>][;<option>]
```

where:

user no.	Is a user number.
separator	Is a space, comma, or dash separator.
last user no.	Is the final number in a range of user numbers.
option	V, which maintains chain file of VALID and NOVALID commands (user 0 only).

<u>VALID Command Example</u> (commands entered in the sequence shown)

=VALID 11	(A record containing a null password is created in the password file for user number 11.)
=VALID 1-12	(Records containing null passwords are created in the password file for user numbers 1 through 10 and 12. User number 11 is ignored.)
=VALID 19 20	(Records containing null passwords are created in the password file for user numbers 19 and 20. A space separator may be used between user numbers in a sequence.)

NOVALID

5.2.6 Novalid System Utility (NOVALID)

NOVALID is a system utility for use by the administrator to delete records from the password file. An entire record is deleted -- both the user number key and the password. Access to the system is subsequently not granted to the user of that number when the security level requires a password match.

A single user number or a range of user numbers can be specified to the NOVALID command. Invalid numbers within a range of valid numbers are ignored. The record for user 0 cannnot be deleted through use of the NOVALID command.

NOVALID Command Syntax

```
    NOVALID <user no.>[<separator><last user no.>]
```

where:

user no.	Is a user number.
separator	Is a space or comma or dash separator.
last user no.	Is the final number in a range of user numbers.

NOVALID Command Example (commands entered in the sequence shown)

=NOVALID 256	(The record, including user number key and password, is deleted from the password file for user number 256.)
=NOVALID 250,260	(The records, including user number key and password, are deleted from the password file for user numbers 250 through 255 and 257 through 260, since user number 256 was previously deleted).

ACCT

5.2.7 Account System Utility (ACCT)

The Account utility allows the administrator to monitor individual and collective utilization of the system. The administrator can create a file of user number records in each of which the utility will automatically maintain the following data:

a. The time and date of last logoff.
b. A count of the number of sessions.
c. A running total of the hours, minutes, and seconds of system use.

A running total of the aggregate system use is also kept. To ease association of user with user number and for further classification, provision is made for user name and department number in each user number record.

The facility offered by the Account utility is derived through use of two files. One, PRIV.ACCOUNT.PW, is an indexed sequential file of password records on the default system volume. This file is established by the administrator to allow users to control access to their individual files. When the system user accounting function is desired, the administrator can open the file PRIV.ACCOUNT.OW to contain the data described above, which the system subsequently supplies. (The accounting function is available, but is not externally accessible until the .OW file is opened.)

The PRIV.ACCOUNT.OW file can be opened in either of two ways. When a password file exists, invoking the ACCT command with the ;U option creates a record for each valid user number (and deletes existing records for numbers that are no longer valid). Alternatively, ACCT executed with the ;N option set opens the file and allows a record to be created for a user number on an individual basis.

Record data is entered and updated in several ways. The ACCT ;C option permits the user name and department fields to be changed in an existing record. The ;N option allows a new record to be created for a user number. The ;U option, in addition to creating and deleting records as explained above, causes data in all fields other than user number, department, and name to be updated for existing records or initiated in records created by execution of the command. A password file must exist for the ;U option to be functional.

The record for a user number can be deleted by specifying the ;X option on the ACCT command line.

System usage information in the PRIV.ACCOUNT.OW file can be displayed by several categories. These include:

a. User number for which name, department, time of last logoff, accumulated usage time, total number of sessions, and date of last reset are displayed.

b. Department for which each user number is listed giving data as in item a., plus total department usage time.

c. All records in the .OW file.

The latter displays data as in item a. for each user number plus the total usage time accumulated by all user numbers in the file. Usage totals are calculated only at time of request. Therefore, consumed system time credited to a user number that is deleted prior to a total usage request does not contribute to the total.

Another display provided by the ACCT utility is the password corresponding to a user number. The ;P option provides this function.

If desired, updatable information in an individual user number record can be reset. The ;R option nulls data in the cumulative time, session count and, of course, data/time of reset fields. The ;Z option provides the same function for all records in the .OW file.

The table below summarizes the functions supplied by execution of the ACCT utility and its 10 options.

ACCT Command Syntax	Corresponding Options
(1) ACCT [;<option>]	A, U, Z
or	
(2) ACCT <user no.>[;<option>]	B, P, R, X
or	
(3) ACCT <user no.>.<department>.<name>[;<option>]	C, N
or	
(4) ACCT <department>[;<option>]	D

where:

user no.	is one to four decimal digits.
department	is one to eight alphanumeric characters.
name	is one to 20 alphanumeric characters.
option	is one or more of the following: A, B, C, D, N, P, R, U, X, Z.

Summary of option functions

A - Display all records; show cumulative usage.
B - Display individual record.
C - Change record field(s): user no., name, department.
D - Display all records for department; show cumulative usage.
N - Create new record from console.
P - Display user number password.
R - Reset usage in individual record.
U - Update all records from password file.
X - Expunge individual record.
Z - Reset usage in all records.

Example 1 File creation/update

```
=VALID 313
=NOVALID 5
=ACCT ;U

     or

=ACCT 5;X
=ACCT 313.DEPTX.RON WOOD;N
```

Example 2 Individual record reset

```
=ACCT 22;R
```

Example 3 Complete usage data reset

```
=ACCT ;Z
```

Example 4 Single user display

```
=ACCT 212
```

```
USER-DEPT-----OWNER NAME----------LAST LOG OFF------USAGE--------#S-RESET
0212 DOC      KEN                 00/00/00 00:00:00   00:00:00   00 801219119

                                  (NOT USED SINCE       NO          DATA RESET
DEPARTMENT    USER'S NAME            CREATED)      ACCUMULATED      12/19/80
 NAME/NO.                                              USAGE          11:19
                                                               SESSION
                                                                COUNT
```

Example 5 Display all users

```
=ACCT ;A
```

```
USER-DEPT-----OWNER NAME----------LAST LOG OFF------USAGE--------#S-RESET
0000 RD554    SYSTEM ADMINISTRATOR12/22/80 10:50:59   08:47:57   39 8012191119
0001 LANG     JOHN                12/19/80 16:09:50   00:23:23   02 8012191119


9999 OPERATION OEM RELEASE         000/00/00 00:0:00   00:00:00   00 8012191119
USAGE=    39:30:24


          HRS MIN SEC

TOTAL USAGE FOR ALL USERS
```

MODIFYING THE USER SESSION MANAGEMENT

Example 6   Display users for a specified department

```
             =ACCT SYST;D

USER-DEPT-----OWNER NAME----------LAST LOG OFF------USAGE--------#S-RESET
0004 SYST     STEVE               12/16/80 16:42:06  00:00:00   00 8012191119
0011 SYST     ARNOLD              12/19/80 11:20:54  00:00:00   01 8012191119
7616 SYST     RON                 12/22/80 07:42:11  00:00:10   01 8012191119
USAGE=    00:00:10
```

5.3 MODIFYING THE OPERATION OF USER SESSION MANAGEMENT

The value of certain parameters in VERSAdos may be altered by the user to modify the operation of User Session Management. The parameters are in the CMDLIST module of the &EET task.

The VERSADOS.SY boot file may be patched using the PATCH utility.

&EET	This block of memory holds the identification message which is printed in response to a request for service BREAK. The identification message is limited to 80 ASCII characters, and is terminated by a byte with binary zero value. A return or line feed is not required.
&EET + $51	Two bytes which hold the number of concurrent batch jobs to run. The number is initialized to 1 in the system as delivered. Note that if more than one job is allowed to execute concurrently, a job which depends on the completion of another batch job may not be queued until the first is completed.
&EET + $53	The security flag byte which sets the initial logon security level. If it is desired to require password and/or system security word protection for session 0001, this byte may be patched. Once VERSAdos is running, the security may be changed, using the SECURE command. The binary configuration of the security flag byte is illustrated in Figure 5-2 (refer also to paragraph 5.2).
&EET + $54	Eight bytes which contain the initial system word. The NOTE following Figure 5-2 also applies with respect to execution of the SWORD command after start-up.
&EET + $5C	This block holds the System Default Volume name (four ASCII bytes). VERSAdos is delivered with this initialized to "SYS ". If the volume name is less than four characters, the remainder of the field must be filled with trailing spaces. When this field contains the actual System Default Volume name, the user may enter ":<user number>" in response to "ENTER DEFAULT VOLUME:USER NUMBER".

It is not necessary for user = 0 to start SESSION 0001; however, only user = 0 can initiate the printer spooling task from the terminal. Upon completion of session 0001 logon (if the $EET+$72 bit = 1), the chain file, 0.PRIV.UPSYSTEM.NW, is initiated as if logged on under user = 0. The printer spooling task may be initiated from this chain file as well as security commands, regardless of the user starting session 0001.

&EET + $69	This byte governs the number of attempts to log on or enter system security word or password before the logon is rejected.
&EET + $6A,6B	Low limit priority, high limit priority of user/utility tasks created by the session control task in the online/interactive mode.
&EET + $6C,6D	As above for chain file mode.
&EET + $6E,6F	As above for the batch mode.

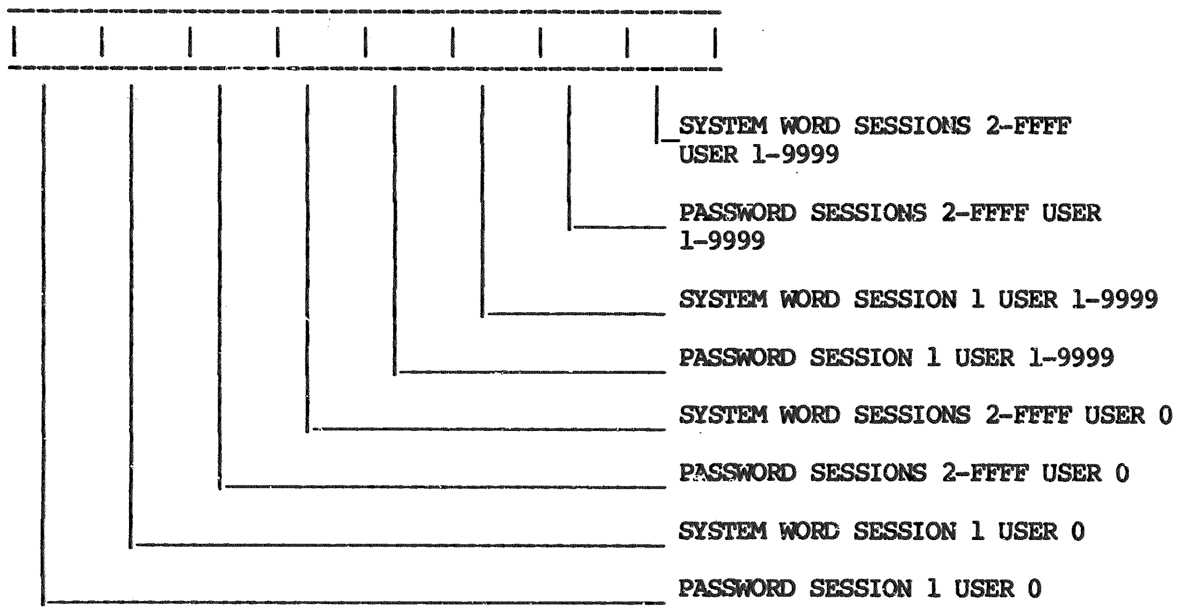

SYSTEM WORD SESSIONS 2-FFFF USER 1-9999
PASSWORD SESSIONS 2-FFFF USER 1-9999
SYSTEM WORD SESSION 1 USER 1-9999
PASSWORD SESSION 1 USER 1-9999
SYSTEM WORD SESSIONS 2-FFFF USER 0
PASSWORD SESSIONS 2-FFFF USER 0
SYSTEM WORD SESSION 1 USER 0
PASSWORD SESSION 1 USER 0

FIGURE 5-2. Security Flag Byte

<u>NOTE</u>

This byte is in read only memory, and is used only to initialize the operating security level. Display of this location during operation will not subsequently indicate the current security level after a SECURE command has been executed. Likewise, changing this location after start-up initialization will not affect the operating security level.

Each task created is given a low and high limit priority. The high limit is constrained to the high limit of the creating task. When a task is started, the high limit is used as the run priority. Each time the task is dispatched, the run priority is decremented by one. When the low limit has been reached, (compute bound task), or each time a call to a server task (trap) is acknowledged, the run priority is restored to the high limit. The current high limit for the Session Control Task is $C8 which is, therefore, the absolute maximum for a user task.

&EET + $71	This byte contains the number of times input time-out (no response to a command input prompt) may occur before the session is automatically terminated.
&EET + $72	This byte governs system options on a bit-oriented basis as follows:

BIT	FUNCTION
0	Provision for enforcing usage of batch mode for the utilities ASM, PASCAL, PASCAL2, and LINK; not yet implemented.
1	Executes the chain file PRIV.UPSYSTEM.NW, upon completion of session 0001 logon.
2-7	Unassigned.

&EET + $74 This long word is the millisecond delay to retry loading a user task in batch mode when memory is not available. A count of 256 is decremented before aborting the batch job. As supplied, the delay is 16 seconds (initial value) x 256 = 68 minutes (approximately).

CHAPTER 6 LIBRARY OF EXECUTABLE FUNCTIONS MACROS, CONSTANTS, AND DATA STRUCTURE CONFIG-URATORS

CHAPTER 6

LIBRARY OF EXECUTABLE FUNCTIONS, MACROS, CONSTANTS, AND DATA STRUCTURE CONFIGURATORS

6.1 INTRODUCTION

Relocatable object file modules and source files which offer common functions of use to the programmer exist under user 0 on the system volume (or the ASM: diskette for floppy systems). Included are executable functions, macros, system constants (equates), and structures containing data used by the executive to configure larger structures. Executable functions are contained in modules in the library file UTILIB.RO. File UTILIB.RO was assembled into section 14 for linking. The remainder are contained in source files (.SA extension).

Successful use of these functions generally requires that several processor registers be supplied with particular values, such as pointers, prior to a call. The entire machine context might also have to be saved and restored on return from a call. To aid in this endeavor, entry and exit conditions are described for each routine.

In the following discussion, the terms input field, output field, and options field refer to the corresponding fields in a command line. Character class refers to the codes (constants) discussed in paragraph 6.4. Dividing command line characters into classes aids in minimizing the code required for parsing. Unless otherwise noted, register values are given in hexadecimal.

6.2 EXECUTABLE FUNCTIONS

The .RO modules in the library file UTILIB.RO offer commonly used functions, which are listed below. A call to any of these functions can be included in a source program, provided the required XREF declaration is included and UTILIB.RO is available to the linkage editor so that its modules can be searched for the corresponding XDEF and the proper module reserved for the final linking process.

IDENTIFIER	FUNCTION
PARSE	Resolve command line into fields; check syntax.
EDITFILE	Transfer file descriptor or device name to FHS parameter block.
OPTION	Determine option field value(s).
TIMECONV	Convert internal time to ASCII representation of hours, minutes, and seconds.
ODATCONV	Convert ordinal date from binary to ASCII.

IDENTIFIER	FUNCTION
GDATCONV	Convert Gregorian date from binary to ASCII.
DATEOG	Convert ordinal date to Gregorian date.
DATEGO	Convert Gregorian date to ordinal date.
BDCONV	Convert binary value to ASCII decimal.
BHWCVT	Convert 2-byte binary value to string of ASCII hex digits.
BHLCVT	Convert 4-byte binary value to string of ASCII hex digits.
ASCVAL	Convert ASCII hex or ASCII decimal string to a binary value.
CVT2	Convert two ASCII decimal digits to a binary value.
SBARG	1) Perform argument substitution from chain file or console command input. 2) Convert lowercase characters from command line to uppercase (disabled when user session management option K is set). 3) Insert carriage return in byte appended to end of command line for this purpose by user (required for SBARG call) and echo command line to output device. The echo is disabled when user session management option U is set. 4) Overwrite with a carriage return the command line comprised of a single space character.

6.2.1 PARSE

The purpose of PARSE is the resolution of command line characters into fields and validation of field syntax. Error-free execution of this routine is a prerequisite for successful calls to the EDITFILE and OPTION functions.

PARSE Command Line Routine

	INSTRUCTION	IDENTIFIER
CALL:	BSR	PARSE

	REGISTER	CONTENTS
ENTRY CONDITIONS:	D6	Command line length (number of characters)
	A0	Command line starting address
EXIT CONDITIONS:	D0	Returned Status:

HEX VALUE	MEANING
8X XX XY XX	Call error
4X XX XY XX	No return character
2X XX XY XX	Invalid character
8X XY XX	Error in input field
4X XY XX	Null input field
2X XY XX	Invalid character in input field
8Y XX	Error in output field
4Y XX	Null output field
2Y XX	Invalid character in output field
8X	Error in option field
4X	Null option field
Y	Number of commas in output field

REGISTER	CONTENTS
D1	Current character
D2	Class code of current character
D6	Remaining character count
A0	Pointer to current character
A1	Output field starting address
A2	Options field starting address
A3	Last field starting address

		0	1	2	3	4	5	6	7
REGISTERS	D	*	*	*				*	
ALTERED	A		*	*	*	*			

6.2.2 EDITFILE

The purpose of EDITFILE is to resolve an input or output field of a command line into its respective subfields, validate subfield syntax, and install the validated field values in the corresponding areas in an FHS parameter block which the user has previously created. Notification of the effect of EDITFILE on each subfield is provided by the state of appropriate bits in the long word returned in data register zero (D0). A value of 1 in a returned D0 bit signifies that the result described for that bit has occurred.

SUBROUTINE EDITFILE - EDIT DEVICE/FILENAME

	INSTRUCTION	IDENTIFIER
CALL:	BSR	EDITFILE
	REGISTER	**CONTENTS**
ENTRY CONDITIONS:	A0	Pointer to starting address of file descriptor or device name field.
	A6	Address of FHS parameter block to receive edit output.
EXIT CONDITIONS:	D0	Returned Status:

BIT(S)	MEANING
31	Call error
30	Volume field error
29	User no. field error
28	Catalog field error
27	File name field error
26	Extension field error
25	Write protection key field error
24	Read protection key field error
23	Error occurred during pass 1 pre-edit
22	Volume field found, FHS block modified
21	User no. field found, FHS block modified
20	Catalog field found, FHS block modified
19	File name field found, FHS block modified
18	Extension field found, FHS block modified
17	Write protection key found, FHS block modified
16	Read protection key found, FHS block modified
15	Null field(s) found
14	Volume field null FHS block not modified
13	User no. field null FHS block not modified
12	Catalog field null FHS block not modified
11	File name field null FHS block not modified
10	Extension field null FHS block not modified

9 & 8	Error code
	00 Syntax error
	01 Field count error
	10 Invalid character
	11 Naming convention error
7 "*"	Found in field(s)
6 "*"	Found in volume field
5 "*"	Found in user no. field (A -2 will be inserted in the FHS parameter block user number field)
4 "*"	Found in catalog field
3 "*"	Found in file name field
2 "*"	Found in extension field
1 0 =	File name
1 =	Device name ("#" found)
0 0 =	Terminated by field terminator

<u>FIELD TERMINATORS</u>

Space
Carriage Return
Semicolon
Comma

1 =	Terminated by slash continuation character

D1	Last character processed
A0	Address of terminator character + 1

		0	1	2	3	4	5	6	7
REGISTERS	D	*	*	*	*	*			
ALTERED	A	*			*	*	*		

<u>NOTE</u>

NUL is not recognized as a valid field terminator.

6.2.3 OPTION

The purpose of OPTION is to process one option character within a command line option field. The status return indicates (1) whether the character was preceded by "-" (used to disable a previously set option), (2) whether the character was followed by an "=", (3) whether a numeric value (also returned) followed the "=" or an alphabetic character followed the "=" (file name assumed), or (4) whether a carriage return was found (end of line). This subroutine is repeatedly called until the status returned in D0 indicates a carriage return was found instead of an option.

OPTION PROCESSOR

PARSE Command Line Routine

	INSTRUCTION	IDENTIFIER
CALL:	BSR	OPTION

	REGISTER	CONTENTS
ENTRY CONDITIONS:	A2	Address of location containing option field character to be examined (the current character)
EXIT CONDITIONS:	D0	Returned Status:

BIT(S)	MEANING
7	Error
6	Carriage return found (end of option field)
5	Invalid character
4	Equal sign found D3 if numeral follows equal sign, contains binary equivalent of numeric value; otherwise, contains ASCII character value truncated to five bits A3 if alphabetical character follows equal sign, contains starting address of character
3	Current character following equal sign is alphabetical
2	Minus sign (not option indicator) found
1	Number conversion error

D1	Contains the current option character Bit 31 will be set if the disable option (¬) was found
D2	Contains the class code of the current character
D3	If bit 4 is reset, then D3 contains the binary equivalent of the character following the equal sign If bit 4 is set, then D3 contains the current option character value truncated to five bits
A2	Address of next option character to examine
A3	If bit 3 is set, then A3 contains the starting address of the non-numeric character following the equal sign If bit 4 is set, then A3 contains the starting address if value converted and stored in D3

		0	1	2	3	4	5	6	7
REGISTERS	D	*	*	*	*				
ALTERED	A			*	*	*			

6.2.4 Argument Substitution and Associated Functions - SBARG

The main benefit derived from an argument substitution capability is the elimination of the need to completely specify a command line when a new parameter is to be passed or of the need to write an entire chain file when new arguments are to be processed in the same manner. For this latter reason, argument substituion finds its greatest use in chain mode processing.

Argument substitution capability is provided by the segment .ARG and, consequently, is similar to that provided by session management under the VERSAdos operating system.

A call to the SBARG routine will provide these functions:

a. Substitution of specified arguments.

b. Conversion of lowercase to uppercase (provided option K is not set).

c. The appending of a carriage return to end of data in the data buffer (for compatibility with the Input/Output Service (IOS)).

d. Echo input from the terminal to the output device (unless option J is set).

e. Overwrite with a carriage return the command line comprised of a single space character.

The following source lines represent a generalized, typical sequence for calling SBARG. Note that the input buffer must always be one byte larger than that specified for the I/O call to accommodate the carriage return appended to end of data. Also, since SBARG attaches to the segment .ARG, the calling task cannot have more than three segments attached at the time of call. About 70 bytes of stack space are required by SBARG for the saving, use, and restoration of registers D1 through D3 and A1 through A5.

<u>Calling Sequence</u>

```
    LEA <IOSBLK>,A0               A0 = Normal I/O parameter block address
    TRAP #2                       The I/O call
    BNE <error processing>        Refer to the VERSAdos Data Management Services
    BSR SBARG                     and Program Loader User's Manual, RMS68KIO,
                                  for a description of the standard IOSBLK form.
```

<u>Entry Conditions</u> (following I/O call)

A0 = Contains address of I/O parameter block

Exit Conditions

D0 =	Zero indicates successful call (or .ARG segment non-existent)
D0 =	$00000101 - Substitution caused buffer overflow
D0 =	$00000102 - Argument not found
D0 =	$080400XX - Attach segment error
A0 =	Contains I/O parameter block address; the record length in the parameter block has been adjusted to reflect the length of the substituted argument

In the case of a non-existent .ARG segment, no argument substitution is performed, lowercase is not translated to uppercase, and D0 is set to zero.

6.2.5 TIMECONV

A call to the routine TIMECONV will result in the conversion of the binary representation of a number of milliseconds to the ASCII representation of the equivalent value expressed in hours, minutes, and seconds. The number of milliseconds passed to the function can be obtained from an arbitrary source but often would be obtained from the system clock via execution of the executive directive GTDTIM.

	INSTRUCTION	IDENTIFIER
CALL:	BSR	TIMECONV

	REGISTER	CONTENTS		
ENTRY CONDITIONS:	D0	Number of milliseconds to be converted.		
EXIT CONDITIONS:	D0	MMHH	where:	MM = minutes
	D1	--SS		HH = hours
				SS = seconds

6.2.6 Binary-to-ASCII Date Conversion - ODATCONV and GDATCONV

A call to one of these functions will provide conversion from the binary representation of the ordinal date to its ASCII equivalent, or conversion of the binary representation of the Gregorian date to its ASCII equivalent. Ordinal date is the number of days elapsed since January 1 of the current year, and can be obtained via execution of the executive directive GTDTIM. The Gregorian date is the current calendar day expressed in a binary representation as shown below.

	INSTRUCTION	IDENTIFIER
CALL:	BSR	ODATCONV
or	BSR	GDATCONV

	REGISTER	CONTENTS
ENTRY CONDITIONS:		
ODATCONV	D0	Ordinal day number (binary).
GDATCONV	D0	Gregorian date as YYYYMMDD in which YYYY is a 16-bit binary field, MM an 8-bit binary field, and DD an 8-bit binary field.
EXIT CONDITIONS:		
Both functions	D0	DDMM
	D1	YYYY

6.2.7 Binary-to-Binary Date Conversion - DATEGO and DATEOG

The DATEOG and DATEGO functions provide conversion from the binary representation of the ordinal date to the Gregorian date, or vice versa, respectively. Ordinal date is the number of days elapsed since January 1 of the current year, expressed in binary. Gregorian date is the current calendar day expressed as shown below. The ordinal date can be obtained by executing the executive directive GTDTIM. This directive is described in the M68000 Family Real-Time Multitasking Software User's Manual.

	INSTRUCTION	IDENTIFIER
CALL:	BSR	DATEGO
or	BSR	DATEOG

	REGISTER	CONTENTS
ENTRY CONDITIONS:		
DATEGO	D0	Gregorian date as YYYYMMDD in which YYYY is a 16-bit binary field, MM is an 8-bit binary field, and DD is an 8-bit binary field.
DATEOG	D0	Ordinal day number (binary).
EXIT CONDITIONS:		
DATEGO	D0	Ordinal day number (binary).
DATEOG	D1	Gregorian date (as above).

6.2.8 Binary-to-ASCII Decimal Conversion - BDCONV

This function provides conversion of binary value in the range 0-9999 to a decimal value expressed in ASCII representation. Leading zeros are suppressed.

	INSTRUCTION	IDENTIFIER
CALL:	BSR	BDCONV

	REGISTER	CONTENTS
ENTRY CONDITIONS:	D0	Binary value to converted ASCII decimal equivalent of input value.
EXIT CONDITIONS:	D0	ASCII decimal equivalent of input value.

6.2.9 Binary-to-Hexadecimal Conversion - BHWCVT and BHLCVT

These functions perform conversion of a binary value to a hexadecimal value expressed as an ASCII character string. Leading zeros are suppressed. BHWCVT, in addition to converting a binary word to four hexadecimal digits, stores the digits beginning at a specified address. BHLCVT converts a long binary word to eight hexadecimal digits but has no storage provision.

	INSTRUCTION	IDENTIFIER
CALL:	BSR	BHWCVT
or	BSR	BHLCVT

	REGISTER	CONTENTS
ENTRY CONDITIONS:		
BHWCVT	D0.W	Binary value to be converted.
	A0	Starting address to store converted value.
BHLCVT	D0.L	Binary value to be converted.
EXIT CONDITIONS:		
BHWCVT	D0	Destroyed.
	A0	Starting address of converted value plus four.
BHLCVT	D2	Four high order bytes of converted value.
	D3	Four low order bytes of converted value.

6.2.10 ASCII Decimal/ASCII Hexadecimal-to-Binary Conversion - ASCVAL

A call to function ASCVAL will convert an ASCII character string representing a decimal or hexadecimal value to the equivalent binary value. The function examines the first character to determine the conversion type wanted. If a dollar sign ($) is encountered, hexadecimal conversion is performed; otherwise, decimal conversion is performed. An empty character string field passed to this function in A0 produces a zero binary value. Conversion terminates when the first invalid character is encountered.

	INSTRUCTION	IDENTIFIER
CALL:	BSR	ASCVAL
	REGISTER	**CONTENTS**
ENTRY CONDITIONS:	A0	Starting address of ASCII character string to be converted.
EXIT CONDITIONS:	D1	The character which caused termination.
	D2	The class code of the terminating character (refer to Table 6-2).
	D3	The binary result of the conversion.
	A0	The address of the character following the terminating character.

6.2.11 ASCII Decimal-to-Binary Conversion - CVT2

Conversion of two ASCII decimal digits to the equivalent binary value is provided by the function CVT2. An empty character string field passed to this function in A0 produces a zero binary value. Conversion terminates when a non-decimal digit is encountered.

	INSTRUCTION	IDENTIFIER
CALL:	BSR	CVT2
	REGISTER	**CONTENTS**
ENTRY CONDITIONS:	A0	The address of the most significant ASCII decimal digit.
EXIT CONDITIONS:	D0	The binary result of the conversion.
	A0	The address of the terminating character.

6.3 MACROS, SYSTEM CONSTANTS, AND DATA STRUCTURE CONFIGURATORS

Several text files are supplied on the system volume which hold data of possible use to the application programmer. File contents include system constants, data structure configurators (blocks of parameters for the fields of data structures), and macros. The files are all of extension .SA and belong to user 0. In a floppy-diskette-based system, that diskette with a volume ID of DIAG contains these files.

The usefuless offered by these files is generally that of convenience. The degree of usefulness is determined to a large extent by the programmer's style -- whether he expresses offsets numerically or symbolically, whether he prefers to code an executive call or utilize a macro for that purpose, or whether he has an excellent memory or must refer often to listings and manuals until the often-used symbols and values are learned. In the latter regard, listings of these files can provide quick reference to the mnemonic symbols for offsets from the various parameter blocks.

6.3.1 Constants and Configurators for Executive Calls

The data structure configurators and the constants used to invoke the various executive directives are provided in the file SYST:0..EXE.SA (hard) or DIAG:0..EXE.SA (floppy). Full descriptors of the directives are given in the M68000 Family Real-Time Multitasking Software User's Manual, M68KRMS68K.

6.3.2 Disk Structure Configurators and Constants

The configurators for the Volume ID (VID), the Secondary Directory Block (SDB), the Primary Directory Block (PDB), and the File Access Block (FAB) disk structures, plus related constants, are provided in the file SYST:0..FME.SA (hard) or DIAG:0..FME.SA (floppy). Descriptions of these data structures are given with the REPAIR utility description in this manual.

6.3.3 Macros, Constants, and Configurators for IOS and FHS Calls

Configurators for the data structures by means of which IOS and FHS calls are executed are provided in the file SYST:0..IOE.SA (hard) or DIAG:0..IOE.SA (floppy). Also included are error message handler constants for non-trap-related messages, and five macros for constructing IOS and FHS parameter blocks and generating the code for IOS and FHS calls. Full information on the Input/Output and File Handling Services is provided in the VERSAdos I/O and File Management Services Manual, RMS68KIO.

6.3.4 Constants and Configurators for Task Control Blocks

The configurator and related constants for a task control block are provided in the file SYST:0..TCB.SA (hard) or DIAG:0..TCB.SA (floppy). The create task control block executive directive is described in the M68000 Family Real-Time Multitasking Software User's Manual.

CHARACTER CLASSIFICATION CODES

6.3.5 Executive Request Macro and Constants for the Requested Directive

A macro for setting up and calling often used Trap #1 routines and the required constants plus related error codes and messages are provided in the file SYST:0..TR1.SA (hard) or DIAG:0..TR1.SA (floppy). The requirements for each executive directive that can be called by executing a Trap #1 instruction are more fully described in the M68000 Family Real-Time Multitasking Software User's Manual.

6.3.6 Constants and Configurators for Task Segment Tables

The configurators and related constants for a task segment table are provided in the file SYST:0..TST.SA (hard) or DIAG:0..TST.SA (floppy). A description of a task segment table can be found in Appendix E of the M68000 Family Real-Time Multitasking Software User's Manual.

6.4 CHARACTER CLASSIFICATION CODES

As mentioned briefly in the introduction to this chapter, a character class code is a constant serving as a classifying device for reducing the amount of code required for determining the attributes of characters comprising a VERSAdos command line. Such attributes include, for example, the position of a character with respect to a preceding delimiter, whether it bounds or lies within a field, its validity in that field, and so on.

In the VERSAdos operating system, 53 ASCII characters are used in the various fields of a command line. Many of them are used in several fields, which complicates the task of determining their validity. To minimize this job, the characters are divided into eight natural groups. The groups are given eight different weightings, which further allows characters to be assigned a constant value determined by summing the weights of each field in which a character appears. For example, Table 6-1 shows that the character "A" appears in (1) the file descriptor group (wt.=$10), (2) the alphabetical group (wt.=$8), (3) the alphanumeric group (wt.=$4), and (4) the hexadecimal digit group (wt.=$2). Thus, "A" is assigned a constant value of $1E, as shown in the character class code chart, Table 6-2. Assigning such codes (constant values) to character groups and subgroups allows greater discrimination of a character than its expected membership in a valid group such as a command line field. The codes also allow advantage to be taken of the MC68000 microprocessor's address register indirect with index addressing mode. This, essentially, is a double indexing mode that facilitates scanning the data in a memory buffer relative to its beginning.

Note that Figure 6-1, the listing of the XDEF CLASCODE DC.B values for the code constants, maps directly into the familiar ASCII code conversion table shown in Figure 6-2. The left-most column of characters in Figure 6-2 corresponds to the first line of DC.B values. (Only the CR and LF characters from this column are used in a VERSAdos command line.) Table 6-2 demonstrates this mapping.

TABLE 6-1. Command Line Character Classes

CLASS NAME	CLASS MEMBERS	SYMBOLS	WEIGHT/ CONSTANT
Field Terminator	Carriage Return, Line Feed, Comma, Semicolon, Space	[CR] [LF] , ; [SP]	80/$80
Special Flag	Pound Sign, Dollar Sign, Equals Sign, Dash, Asterisk	# $ = — *	40/$40
Descriptor Punctuator	Colon, Period, Left Parenthesis, Right Parenthesis, Slash	: . ( ) /	20/$20
File Descriptor	Alphabeticals A-Z, Digits 0-9, Asterisk, At Sign, Ampersand	A-Z 0-9 * @ &	10/$10
Alphabetical	Alphabet Characters A through Z	A-Z	8/$8
Alphanumeric	Alphabeticals A-Z, Digits 0-9	A-Z 0-9	4/$4
Hexadecimal Digit	Alphabeticals A-F, Digits 0-9	A-F 0-9	2/$2
Decimal Digit	Digits 0-9	0-9	1/$1

NOTE 1: Value of constant for character that is a member of more than one class determined by adding weights of all classes in which character appears. Example: Asterisk constant = 40 + 10 = $50

NOTE 2: NUL and all other ASCII character codes not included in one of above classes are invalid command line codes.

TABLE 6-2. Character Class Code Chart

b4 b3 b2 b1	0 0 0		0 0 1		0 1 0		0 1 1		1 0 0		1 0 1		Bit 7 Bit 6 Bit 5
0 0 0 0	NUL	0			SP	80	0	17	@	10	P	1C	
0 0 0 1							1	17	A	1E	Q	1C	
0 0 1 0							2	17	B	1E	R	1C	
0 0 1 1					#	40	3	17	C	1E	S	1C	
0 1 0 0					$	40	4	17	D	1E	T	1C	
0 1 0 1							5	17	E	1E	U	1C	
0 1 1 0					&	10	6	17	F	1E	V	1C	
0 1 1 1							7	17	G	1C	W	1C	
1 0 0 0					(	20	8	17	H	1C	X	1C	
1 0 0 1					)	20	9	17	I	1C	Y	1C	
1 0 1 0	LF	80			*	50	;	20	J	1C	Z	1C	
1 0 1 1							;	80	K	1C			
1 1 0 0					,	80			L	1C			
1 1 0 1	CR	80			—	40	=	40	M	1C			
1 1 1 0					.	20			N	1C			
1 1 1 1					/	20			O	1C			

KEY

*	50
CHARACTER	HEX CLASS CODE

```
 SECTION 14
 XDEF CLASCODE
CLASCODE IDNT 0,0 CHARACTER CLASSIFICATION CODE TABLE
***************************************************************************
*  CHARACTER CLASSIFICATION CODE TABLE
***************************************************************************
CLASCODE EQU *
 DC.B 0,0,0,0,0,0,0,0,0,0,$80,0,0,$80,0,0
 DC.B 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 DC.B $80,0,0,$40,$40,0,$10,0,$20,$20,$50,0,$80,$40,$20,$20
 DC.B $17,$17,$17,$17,$17,$17,$17,$17.$17,$17,$20,$80,0,$40,0,0
 DC.B $10,$1E,$1E,$1E,$1E,$1E,$1E,$1C,$1C,$1C,$1C,$1C,$1C,$1C,$1C,$1C
 DC.B $1C,$1C,$1C,$1C,$1C,$1C,$1C,$1C,$1C,$1C,$1C,0,0,0,0,0
 DC.B 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 DC.B 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 END
```

FIGURE 6-1. Assignment of Character Class Code Values

BITS 4 thru 6	—	0	1	2	3	4	5	6	7
	0	NUL	DLE	SP	0	@	P		p
	1	SOH	DC1	!	1	A	Q	a	q
	2	STX	DC2	"	2	B	R	b	r
	3	ETX	DC3	#	3	C	S	c	s
	4	EOT	DC4	$	4	D	T	d	t
	5	ENQ	NAK	%	5	E	U	e	u
BITS 0 thru 3	6	ACK	SYN	&	6	F	V	f	v
	7	BEL	ETB	'	7	G	W	g	w
	8	BS	CAN	(	8	H	X	h	x
	9	HT	EM	)	9	I	Y	i	y
	A	LF	SUB	*	:	J	Z	j	z
	B	VT	ESC	+	;	K	[	k	{
	C	FF	FS	,	<	L	/	l	/
	D	CR	GS	-	=	M	]	m	}
	E	SO	RS	.	>	N	^	n	≈
	F	SI	US	/	?	O	—	o	DEL

FIGURE 6-2. ASCII Code Conversion Table

CHAPTER 7 BOOTLOAD FILE — IPL.SY

CHAPTER 7

THE BOOTLOAD FILE -- IPL.SY

7.1 RESIDENT DEBUG MONITOR BO COMMAND FUNCTION

The general function of the resident debug monitor Boot (BO) command is to access a program on disk, move it into memory space, and transfer control to that program. If the program itself can load and transfer control to a second program, the BO command can be used to bring in an operating system or to implement system tests and diagnostics. Paragraph 1.1 lists the resident debugger furnished with each VERSAdos system, and paragraph 7.4 describes the BO command execution sequence.

7.2 FUNCTIONS OF IPL.SY

In most VERSAdos environments, the file IPL.SY is the program accessed and given control by the BO command. (This does not apply to VM01-based systems. See paragraph 7.3.1 for modifications required to boot VERSAdos in VM01-based systems using the BO command.) Usually, IPL.SY loads the VERSAdos operating system. However, IPL can be used to load other programs such as test or diagnostic programs. A review of the BO command line syntax shows that the final command line field can contain a string of up to 249 ASCII characters.

IPL.SY examines the passed string looking for the syntax given below. If <filename> is not found, VERSADOS.SY is loaded.

BO command syntax:

```
    BO [<device>][,<controller][,<filename>[;<option>]]
```

where:

device controller	Are specified as described in the appropriate firmware-resident debug monitor user's manual. Defaults for both are 0.
filename	Is an ASCII character string with a leading alphabetic character. The user number, catalog, and extension of the specified file may be specified; defaults are 0, blank, and SY, respectively. The volume cannot be specified. Default <filename> is VERSADOS.SY.
option	Is one or both of the following:

L=$n	Will cause <filename> to be loaded beginning at the specified hexadecimal address (preceding $ is required).
H	Will cause control to be returned to the debug monitor after <filename> is loaded. (Default: control is given to <filename>).

NOTE

Delimiting space between options is not allowed when both options are specified.

Examples of string subfields, BO command line:

```
,TESTFIL1;HL=$7F00
,0.CAT2.TESTFIL1.SY;L=$7F00,H
```

Upon encountering TESTFIL1, IPL.SY attempts to access and load this file using the device and controller previously specified in the corresponding fields on the BO command line. (In the first example, user 0, a default catalog name of eight blanks, and a default extension of .SY are assumed.) If the attempt is successful, TESTFIL1 is loaded beginning at the specified address 7F00, and control is returned to the firmware-resident debugger in compliance with the specified H option.

In this example, since the firmware-resident debugger is given control when the load is completed and execution halts, the user can take further action by utilizing any of the various alternatives afforded him by the command set. He can, for example, GO to a point other than the start of TESTFIL1 to begin execution.

Where control is transferred on completion of the execution of TESTFIL1 would be determined by the program itself. For example, a diagnostic program might display the result of a test on the EXORmacs Control Panel Status Display and then halt, giving the user the choice of returning to the test program, or returning to the firmware debugger, or of some other option. The disk-resident diagnostics described in the EXORmacs Development System Maintenance Manual, M68KEMM, illustrate the latter alternative.

7.3 IPL.SY MODIFICATION FOR WINCHESTER/SASI DRIVERS

Each bootable target disk should be initialized, using the INIT utility and the V option. The user should always respond with a Y when asked if a bootstrap is wanted. The INIT utility will identify the file name it will use by default; however, the user will be given the opportunity to enter a new name if so desired. The user's response, as defined in the table below, should be to accept the default value by pressing carriage return or to enter the name of the file that should replace the default file. The current load address will be displayed and the user will be given an opportunity to enter a new load address. The user should respond in accordance with the load addresses defined in the following table.

TARGET SYSTEM	BOOTSTRAP	NEW NAME	LOAD ADDRESS
EXORmacs	Yes	(CR)	$0
VM01 64K version	Yes	File name	$10100
VM01 32K version	Yes	File name	$10E00
VM02	Yes	(CR)	$E00
VMC 68/2	Yes	(CR)	$E00
MVME110	Yes	(CR)	$40000
VME/10	Yes	(CR)	$E00

File name should identify the bootable operating system that was created at SYSGEN time for the specific target system (i.e., SYST:9100..VERSADOS.SY).

7.3.1 VID Modification for VM01-Based Systems

No IPL.SY file is supplied with VM01-based systems. Modifications shown below must be made to the media Volume Identification Directory so that the VERSAbug BO command can boot VERSAdos without the IPL.SY program. The PATCH utility must be used for these changes.

a. VM01 64K version

   The long word at $E00 in the VID should be changed to the STACK value used to create the EXEC process (refer to the appropriate SYSCMD.SA file).

   The long word at $E04 should be changed according to the calculations of the following formula:

   WHERLOAD + START-UP ADDRESS - STARTRMS

   The values of WHERLOAD and STARTRMS can be obtained from the appropriate SYSCMD.SA file, and the value of START-UP ADDRESS can be obtained from the SYSGEN listing, SYSGEN.LS.

b. VM01 32K version

   The long word at $10E00 of the VID should be changed to the STACK value used to create the EXEC process.

When booting an alternate, untested VERSAdos program, the location of the formerly used VERSAdos program should be noted. This information will be required if the new program fails and the old one must be rebooted for troubleshooting.

7.4 BO COMMAND EXECUTION SEQUENCE

The following sequence of events takes place when the BO command is executed:

a. Sector 0 (the volume ID) on the disk corresponding to the valid device and controller is accessed and transferred into the firmware-resident debugger scratchpad RAM.

b. Locations $F8-$FF of sector 0 are checked to ensure that they contain the string "EXORMACS". If they do not, an error occurs and execution stops. For EXORmacs systems, a "B8" error is displayed on the EXORmacs front panel.

c. The disk sectors containing the IPL.SY load module are identified by examining the data locations in sector 0, as shown:

OFFSET (bytes)	CONTENTS
$14 - $17	First sector to transfer (first sector of IPL.SY load module).
$18 - $19	Number of sectors to transfer.

   The firmware-resident debugger then reads the IPL.SY load module into memory. Control is passed to the address specified in location $4 of the load module, giving the IPL program control.

d. IPL will search the directory for the requested file. When the file is found, the program sectors are accessed and transferred into memory starting at the address specified with the L=$n option on the BO command line, if given, or the value supplied to the Loader Information Block when the program was linked, if the L=$n option was not specified.

e. If a string field was not specified, register A5 is set equal to register A6.

f. The status register is updated to reflect supervisor mode and interrupt level seven.

g. The supervisor stack pointer is loaded with address specified in location 0 of sector 1 of the load module, plus a "bias" if the L=$n option was specified.

h. The program counter is loaded with the execution start address specified in location 4 of sector 1 of the load module, plus a "bias" if the L=$n option was specified.

The program just loaded now has control of execution.

If an error occurs during any processing from step d. on, IPL will display an error status on the panel lights, if operating on an EXORmacs system.

When an error occurs on a VMC 68/2 system, press SOFTWARE ABORT and display the long word pointed to by A7. This long word contains the address of a message describing the problem.

7.5 RESETTING SECTOR 0 POINTERS TO IPL.SY LOCATION

For VERSAdos 4.2 and subsequent versions, the user may upgrade an old IPL.SY file by copying the new version onto the old disk with the INIT, COPY, or BACKUP command. Because the new file is larger than previous revisions, adjustments must be made to the pointers in sector 0 which tell the BO command where to find the IPL.SY file. The INIT utility will copy the current version over the old and adjust sector 0 so that it points to the new file location. The COPY and BACKUP utilities will successfully replace the old file with the new, but will not adjust sector 0. In such cases, the following procedure may be used to change sector 0 so that it points to the new address of the IPL.SY file.

a. Display the directory of IPL.SY to find its location on the disk:

```
=DIR 0.&.IPL.SY;A

DIR VERSION 111781 3   3/22/83 15:20:20
SYS:0000..IPL.SY
                    LOG   # OF                     REC   KEY FAB DB    DATE     DATE
 START   END        EOF  RECORDS  WC  RC  FT       LEN   LEN LEN LEN CHANGED  ACCESSED
 $42AC  $42CD        -      -     PP  PP   C        -     -   -   -  3/ 2/83   3/ 7/83
 SIZE         34/$22
TOTAL SIZE           34$22
NUMBER OF FILES RETRIEVED =   1
```

b. Locate the start field and the total number of sectors for this version of IPL.SY. In the example above, IPL.SY starts at $42AC and is $22 sectors long.

c. Do an interactive DUMP on the volume to change sector 0 of the disk. The spooler and any printer tasks may have to be terminated before the volume can be accessed.

```
   =DUMP #HD00;I
   >R 0                     (Read sector 0 into the change buffer)
   >M 16                    (Begin changing at offset $16 in the change buffer)
    00 '.'?42               (Change the Physical Sector Number to the one found in
    00 '.'?AD                step 2, plus 1, bypassing Loader Information Block)
    00 '.'?(CR)             (Skip the next byte)
    00 '.'?21               (Enter the length of IPL.SY, found in step 2, minus 1)
    00 '.'?.(CR)            (Enter a period to terminate the interaction)
   >D                       (Display the changes)
    SN=$0        0
00    53 59 53 20 00 00 00 00   00 01 00 81 00 00 00 8A    SYS ............
10    00 00 00 00 00 00 42 AD   00 21 00 00 00 00 00 00    ......B..!......
20    00 00 00 00 03 ED 20 20   20 20 20 20 20 20 20 20    ......
30    20 20 20 20 20 20 20 20   20 20 30 33 30 30 F4 F7                0300..
40    0F 1E 2D 3C 4B 5A 69 78   87 96 A5 B4 C3 D2 E1 F0    ..-<KZix........
50    0F 1E 2D 3C 4B 5A 69 78   87 96 A5 B4 C3 D1 E1 F0    ..-<KZix........
60    F1 F2 F4 F8 F9 FA FC FE   FF 7F BF DF EF 6F AF CF    .............o..
70    4F 8F 0F 07 0B 0D 0E 06   0A 0C 04 08 04 02 01 00    O...............
80    00 00 00 00 00 00 00 00   00 00 00 00 00 82 00 08    ................
90    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
A0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
B0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
C0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
D0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
E0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
F0    00 00 00 00 00 00 00 00   45 58 4F 52 4D 41 43 53    ........EXORMACS
   >W 0                     (Write sector 0 back out to the disk)
   >Q                       (Quit DUMP and try to boot up)
   =
```

d. If the starting Physical Sector Number of the IPL.SY file ($42AC in the example above) is too large to fit in a word value, then place it into a long word value starting at offset $14 in sector 0. If the length of the IPL.SY file should ever be larger than one byte, use the word which begins at offset $18. (The example above uses only part of the actual fields.)

For more information on file structure, refer to the description of the REPAIR utility.

APPENDIX A

S-RECORD OUTPUT FORMAT

APPENDIX A

S-RECORD OUTPUT FORMAT

The S-record format for output modules was devised for the purpose of encoding programs or data files in a printable format for transportation between computer systems. The transportation process can thus be visually monitored and the S-records can be more easily edited.

S-RECORD CONTENT

When viewed by the user, S-records are essentially character strings made of several fields which identify the record type, record length, memory address, code/data, and checksum. Each byte of binary data is encoded as a 2-character hexadecimal number: the first character representing the high-order four bits, and the second the low-order four bits of the byte.

The five fields which comprise an S-record are shown below:

```
+--------+---------------+-------------+-----------------------+-----------+
|  type  | record length |   address   |       code/data       |  checksum |
+--------+---------------+-------------+-----------------------+-----------+
```

where the fields are composed as follows:

FIELD	PRINTABLE CHARACTERS	CONTENTS
type	2	S-record type -- S0, S1, etc.
record length	2	The count of the character pairs in the record, excluding the type and record length.
address	4, 6, or 8	The 2-, 3-, or 4-byte address at which the data field is to be loaded into memory.
code/data	0-2n	From 0 to n bytes of executable code, memory-loadable data, or descriptive information. For compatibility with teletypewriters, some programs may limit the number of bytes to as few as 28 (56 printable characters in the S-record).
checksum	2	The least significant byte of the one's complement of the sum of the values represented by the pairs of characters making up the record length, address, and the code/data fields.

Each record may be terminated with a CR/LF/NUL. Additionally, an S-record may have an initial field to accommodate other data such as line numbers generated by some time-sharing systems.

Accuracy of transmission is ensured by the record length (byte count) and checksum fields.

S-RECORD TYPES

Eight types of S-records have been defined to accommodate the several needs of the encoding, transportation, and decoding functions. The various Motorola upload, download, and other record transportation control programs, as well as cross assemblers, linkers, and other file-creating or debugging programs, utilize only those S-records which serve the purpose of the program. For specific information on which S-records are supported by a particular program, the user's manual for that program must be consulted.

An S-record-format module may contain S-records of the following types:

S0 The header record for each block of S-records. The code/data field may contain any descriptive information identifying the following block of S-records. Under VERSAdos, the resident linker's IDENT command can be used to designate module name, version number, revision number, and description information which will make up the header record. The address field is normally zeros.

S1 A record containing code/data and the 2-byte address at which the code/data is to reside.

S2 A record containing code/data and the 3-byte address at which the code/data is to reside.

S3 A record containing code/data and the 4-byte address at which the code/data is to reside.

S5 A record containing the number of S1, S2, and S3 records transmitted in a particular block. This count appears in the address field. There is no code/data field.

S7 A termination record for a block of S3 records. The address field may optionally contain the 4-byte address of the instruction to which control is to be passed. There is no code/data field.

S8 A termination record for a block of S2 records. The address field may optionally contain the 3-byte address of the instruction to which control is to be passed. There is no code/data field.

S9 A termination record for a block of S1 records. The address field may optionally contain the 2-byte address of the instruction to which control is to be passed. Under VERSAdos, the resident linker's ENTRY command can be used to specify this address. If not specified, the first entry point specification encountered in the object module input will be used. There is no code/data field.

Only one termination record is used for each block of S-records. S7 and S8 records are usually used only when control is to be passed to a 3- or 4-byte address. Otherwise, an S9 record is used for termination. Normally, only one header record is used, although it is possible for multiple header records to occur.

CREATION OF S-RECORDS

S-record-format programs may be produced by several dump utilities, debuggers, the VERSAdos resident linkage editor, or several cross assemblers or cross linkers. On EXORmacs and VME/10, the Build Load Module (MBLM) utility allows an executable load module to be built from S-records. MBLM has a counterpart utility in BUILDS, which allows an S-record file to be created from a load module on any VERSAdos system.

Programs are available for downloading or uploading a file in S-record format from a host system to an 8-bit microprocessor-based or a 16-bit microprocessor-based system.

EXAMPLE

Shown below is a typical S-record-format module, as printed or displayed:

```
S00600004844521B
S1130000285F245F2212226A000424290008237C2A
S11300100002000800082629001853812341001813
S113002041E900084E42234300182342000824A952
S107003000144ED492
S9030000FC
```

The module consists of one S0 record, four S1 records, and an S9 record.

The S0 record is comprised of the following character pairs:

S0	S-record type S0, indicating that it is a header record.
06	Hexadecimal 06 (decimal 6), indicating that six character pairs (or ASCII bytes) follow.
00 00	A 4-character, 2-byte address field; zeros in this example.
48 44 52	ASCII H, D, and R - "HDR".
1B	The checksum.

The first S1 record is explained as follows:

S1	S-record type S1, indicating that it is a code/data record to be loaded/verified at a 2-byte address.
13	Hexadecimal 13 (decimal 19), indicating that 19 character pairs, representing 19 bytes of binary data, follow.
00 00	A 4-character, 2-byte address field; hexadecimal address 0000; where the data which follows is to be loaded.

The next 16 character pairs of the first S1 record are the ASCII bytes of the actual program code/data. In this assembly language example, the hexadecimal opcodes of the program are written in sequence in the code/data fields of the S1 records:

OPCODE	INSTRUCTION	
285F	MOVE.L	(A7)+,A4
245F	MOVE.L	(A7)+,A2
2212	MOVE.L	(A2),D1
226A0004	MOVE.L	4(A2),A1
24290008	MOVE.L	FUNCTION(A1),D2
237C	MOVE.L	#FORCEFUNC,FUNCTION(A1)

. (The balance of this code is continued in the
. code/data fields of the remaining S1 records,
. and stored in memory location 0010, etc.)

2A The checksum of the first S1 record.

The second and third S1 records each also contain \$13 (19) character pairs and are ended with checksums 13 and 52, respectively. The fourth S1 record contains 07 character pairs and has a checksum of 92.

The S9 record is explained as follows:

S9 S-record type S9, indicating that it is a termination record.

03 Hexadecimal 03, indicating that three character pairs (3 bytes) follow.

00 }
00 } The address field, zeros.

FC The checksum of the S9 record.

Each printable character in an S-record is encoded in hexadecimal (ASCII in this example) representation of the binary bits which are actually transmitted. For example, the first S1 record above is sent as:

type				length				address								code/data									checksum			
S		1		1		3		0		0		0		0		2		8		5		F		...	2		A	
5	3	3	1	3	1	3	3	3	0	3	0	3	0	3	0	3	2	3	8	3	5	4	6	...	3	2	4	1
0101	0011	0011	0001	0011	0001	0011	0011	0011	0000	0011	0000	0011	0000	0011	0000	0011	0010	0011	1000	0011	0101	0100	0110	...	0011	0010	0100	0001

SUGGESTION/PROBLEM REPORT

Motorola welcomes your comments on its products and publications. Please use this form.

To: Motorola Inc.
Microsystems
2900 S. Diablo Way
Tempe, Arizona 85282
Attention: Publications Manager
Maildrop DW164

Product: ______________________ Manual: ______________________

COMMENTS:

____

____

____

____

____

____

____

____

____

Please Print

Name	Title	
Company	Division	
Street	Mail Drop	Phone Number
City	State	Zip

Field Service Support: (800) 528-1908
(602) 829-3100

16347 PRINTED IN USA (9/83) MPS 3M