SYSTEM V/68 OPERATOR'S GUIDE

PRODUCT CODE 72904
41967-00

MOTOROLA
Computer Systems

Product Code 72912
41975G01

SYSTEM V/68 DOCUMENTATION SET

VOLUME IV

SYSTEM V/68 PROGRAMMING GUIDE, 72908 (41971-00)

- Introduction
- An Introduction to Shell
- C Programming Language
- FORTRAN
- Curses and Terminfo Package
- Programming Language EFL

SYSTEM V/68 SUPPORT TOOLS GUIDE, 72909 (41972-000)

- Introduction
- Maintaining Computer Programs (MAKE)
- Augmented Version of MAKE
- The M4 Macro Processor
- The AWK Programming Language
- Desk Calculator Language (BC)
- Desk Calculator Program (DC)
- Lexical Analyzer Generator (LEX)
- Yet Another Compiler-Compiler (YACC)
- Common Object File Format

SYSTEM V/68 ASSEMBLER USER'S GUIDE, 72910 (41973-00)

- Introduction
- Warnings
- General Syntax Rules
- Segments, Location Counters, and Labels
- Types
- Expressions
- Pseudo-Operations
- Span-Dependent Optimization
- Address Mode Syntax
- Machine Instructions

SYSTEM V/68 COMMON LINK EDITOR REFERENCE MANUAL, 72911 (41974-00)

- Introduction
- Using the Link Editor
- Link Editor Command Language
- Notes and Special Procedures
- Error Messages
- Syntax Diagram for Input Directives

VOLUME V

SYSTEM V/68 DOCUMENT PROCESSING GUIDE, 72906 (41969-00)

- Introduction
- Advanced Editing
- Stream Editor
- Nroff and Troff User's Manual
- Table Formatting Program
- Mathematics Typesetting Program
- Memorandum Macros
- Viewgraphs and Slides Macros

SYSTEM V/68 ERROR MESSAGE MANUAL, 72902 (41965-00)

- Introduction
- Error Messages
- Index

SYSTEM V/68
OPERATOR'S GUIDE

Product Code 72904

Part Number 41967-00

Version 1

PREFACE

The *SYSTEM V/68 Operator's Guide* (Part Number 41967-00, Product Code 72904) describes release version FE81 of the SYSTEM V/68 operating system resident on the VME Series 20. The manual discusses SYSTEM V/68, getting started, and using SYSTEM V/68.

The *SYSTEM V/68 User's Reference Manual* and *SYSTEM V/68 Administrator's Manual* are referenced in this document. These manuals are organized as alphabetized entries within tabbed sections. The User's Reference Manual contains sections 1 through 6. The Administrator's Manual contains sections 1M, 7, and 8. References to these manuals are given as **name**(section). For example, **dinit**(1M) is a reference to the **dinit** entry in section 1M of the *SYSTEM V/68 Administrator's Manual.*

The following conventions are used in the command syntax, examples, and text in this manual:

boldface strings	A boldface string is a literal, such as a command or program name, and is to be typed just as it appears.
italic string	An italic string is a "syntactic variable" and is to be replaced by one of a class of items it represents.
[ ]	Square brackets enclose an item that is optional. The item may appear zero or one time.
[ ] . . .	Square brackets followed by an ellipsis (three dots) enclose an item that is optional/repetitive. The item may appear zero or more times.
[ ]	Boldface brackets are required characters.

Operator inputs are to be followed by a carriage return. The carriage return is shown as (CR) only if it is the only input required.

Unless otherwise specified, all address references are in hexadecimal throughout this manual.

CONTENTS

FIGURES

TABLES

CAUTION

Some information in this document is specific to the operation of Motorola EXORmacs, VM03, VME/10, or VMEsystem 1131 computers. If you are working with a VME Series 20 system, the *VME Series 20 Software Release Guide* will provide details specific to your system and should supersede similar information found in this document.

In addition, the *VME Series 20 Software Release Guide* describes setting system variables, generating a kernel, configuration planning, initializing disks, and reinstalling software.

1. SYSTEM OVERVIEW

1.1 SYSTEM V/68 OPERATING SYSTEM

This chapter is intended as a brief overview of the SYSTEM V/68 operating system. It includes a general description of the kernel, file system, I/O system, and shell. If you are already familiar with UNIX-derived operating systems, you can skip this chapter and begin with Chapter 2, "Getting Started."

1.1.1 Operating System Features

SYSTEM V/68 was derived from UNIX system V/68000, a jointly developed product of AT&T Technologies, Inc., and Motorola Inc. SYSTEM V/68 is a general purpose, multi-user, interactive operating system designed to make the user's computing environment simple, efficient, flexible, and productive. Features of the operating system include:

- Hierarchical, tree-structured file system
- Flexible, easy-to-use command language that can be tailored to meet specific user needs
- Sequential, asynchronous, and background processes
- Powerful text editors
- Document preparation and text processing systems
- Support of numerous programming languages, including:
  - C high-level programming language conducive to structured programming
  - Assembler - for MC68020 assembly language
  - FORTRAN 77 for numerical applications
- High-level optimizer for efficient code generation
- Support for the MC68881 floating point coprocessor
- Capability for formatting blank Winchester or floppy media
- Symbolic high-level language debugging
- System programming tools (for example, compiler-compiler)
- Sophisticated "desk calculator" packages
- Source Code Control System (SCCS)
- Communications facilities between CPUs running under a UNIX or SYSTEM V/68 operating system (uucp)

1.1.2 Functional Description

SYSTEM V/68 oversees the execution of many user programs which seem to execute simultaneously because of the system's ability to time-share the processor among all the programs. Actually, each program is scheduled to use the processor for a short period of time, to the exclusion of all other programs. This time-sharing makes it possible for many users to use the system simultaneously. It also makes accounting necessary to manage the system properly. The concept of a **process** was developed to allow the operating system to keep track of each program and its use of system resources. The process includes the program executing and information about the various internal parts of the processor affected by the program (such as memory registers, the name of the current directory, the status of open files, or information recorded at login time).

The operating system software includes the SYSTEM V/68 kernel, the Bourne **shell** command interpreter, the file system and various user and system commands. The kernel is the basic resident software on which the entire system relies. It is the only permanently resident part of the system. The kernel consists of system primitives, including various facilities to maintain the file system, support system calls, and manage system resources. The shell command interpreter, which is itself a user process executing under control of the kernel, allows the user to communicate with the operating system. A user invokes processes by issuing commands to the shell. Furthermore, a user can invoke another shell process using the shell command (**sh**(1)).

1.1.3 Kernel

1.1.3.1 Kernel and User Modes. The primary purpose of the kernel is to control system resources and user and system processes. In SYSTEM V/68, a user executes programs in an environment called a user process. When a system function is required, the user process enters the kernel by a processor trap. During this trap, there is a distinct switch of environment. Beforehand, the process is in the **user mode**; afterward, the process is said to be in a **kernel mode**. In the normal definition of processes, the user and kernel mode are different phases of the same process (they never execute simultaneously). Each process is a distinct entity, able to execute and terminate independently of all other processes. In fact, it is not always necessary for the user to be logged into the system while those processes are executing. From a strictly functional standpoint, it can be said there are no **system** processes; instead, there are simply processes in either a user or kernel mode. Each system process (or kernel mode of a user process) has its own stack (see paragraph 1.1.3.2).

When stored in primary memory, a user process occupies a specific address space. The address space associated with the process has certain access permissions for the user and the kernel. As a process changes mode from user to kernel and back, the access permissions to various structures in that address space change. Due to the different access permissions, those various structures can be visualized as being within either the user or kernel mode. These two modes can then be thought of as different entities, both associated with one process. Thus, certain structures are part of the kernel mode and certain structures are part of the user mode.

1.1.3.2 User Mode. The user mode consists of several structures maintained by the kernel mode. These structures (the text segment, the data segment, and the stack) consist of text, data, or information needed by the kernel mode. The data segment has two sections: initialized data and uninitialized data. The uninitialized data segment is called the bss segment and all bytes in it are set to zero when the process is created. Also associated with the user mode is a stack. The stack section, which can only grow, is managed by the kernel during the execution of subroutine linkage instructions. A process in user mode may execute from a read-only text segment, which is shared by all processes executing the same program. The memory savings and swapping efficiencies are significant when large, commonly used programs are shared.

1.1.3.3 Process Table. A process is the execution of an image; most SYSTEM V/68 commands execute as separate processes. Processes are created (or spawned) by the system primitive **fork**.

All the process structures are tied together by a process table, with one entry for each active process. As long as a process is in the system, it has an entry in the process table. Essentially, removing a process' entry in the table is the final act that terminates that process. Therefore, not all processes listed in the process table need to be running (for example, a process could be **sleeping**).

1.1.3.4 Interprocess Communication. The kernel provides several means by which processes can communicate with each other, including pipes, messages, shared memory, semaphores, and signals. Signals are the most frequently used means for a process to indicate the occurrence of an event that may affect another process. There are two specific system calls involved in interprocess signaling: the **kill**(2) system call used to send a signal and the **signal**(2) system call used to specify how the signal will be handled.

1.1.3.5 Scheduling. Process scheduling allows many processes sharing one CPU to be synchronized. It is accomplished with the sleep/wakeup mechanism. This mechanism allows coordination and optimization of the actions of a large number of processes. All active processes will be either **sleeping** (non-runnable and waiting for an event), on the run queue, or actually running (only one process at a time can run). Any process may be resident (in primary memory) or swapped (in secondary memory) at any point in time. The run status and residence status are maintained from the process table.

Which of the many possible processes is to run next? Associated with each process is a priority. Interactive terminal events have high priority, disk events are low, and alarm events (performed at a certain time of day) are very low. All user process priorities are lower than the lowest system priority. This means kernel processes will always run before any user process. The priority of kernel processes is not affected by CPU usage. Therefore, their priority remains constant unless a return to user mode is made. User process priorities are assigned by an algorithm based on the amount of recent compute time consumed by the process.

1.1.3.6 Swapping. One separate process in the kernel, the swapping process, swaps user processes in and out of primary memory. Swapping out means the image is copied to secondary memory and the primary memory it occupied is freed. Swapping in means allocating primary memory for a process and reading its segments into primary memory where the process will compete for the central processor with other loaded processes. Memory layout is shown in Figure 1-1.

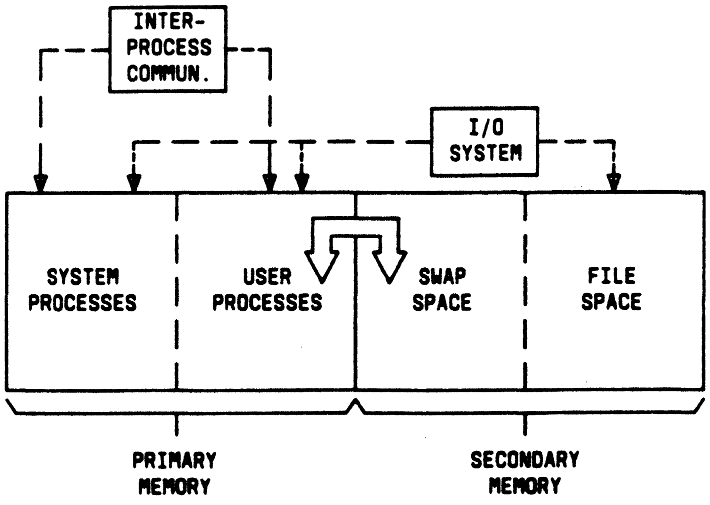

Figure 1-1. Memory Layout

A process terminates for one of two reasons: an explicit call to **exit**(2) or the default action of a signal. After finding a signal, a process looks for a handling routine. If none is found, the process is forced to call **exit**(2).

1.1.4 File System

1.1.4.1 Structure. The file system of SYSTEM V/68 consists of a highly uniform set of directories and files arranged in a tree-like structure (refer to Figure 1-2). Using this tree structure, files can be attached anywhere onto a hierarchy of directories. To the operating system, all files are physically the same (a one-dimensional array of bytes ending with EOF); however, the system keeps track of the file type in the file's i-node. Files are named by sequences of 14 or fewer characters (filenames).

1.1.4.2 Files. There are three types of files: ordinary files, directory files, and special files. In SYSTEM V/68 files normally reside on a disk. The kernel accesses all three types of files in the same way. The user and user application programs must interpret the files appropriately.

Because no particular structuring is expected by the system, an ordinary file contains whatever information the user places in it (for example, English text, source programs, or binary object programs). Any file that is not a directory or a special file is an ordinary file.

Directory files (also referred to as directories) provide the mapping (paths) between the names of files and the files themselves, and thus induce a maplike structure on the file system as a whole. Each user has a directory of files that are treated in like manner. Although a directory behaves exactly like an ordinary file, because it can only be written by the system, the system controls the structure of directories. The system also maintains several directories for its own use. One of these is the *root* directory (which may be considered the base directory). Any one of the files in the system can be found by tracing a path through a chain of directories until the desired file is reached.

Special files constitute the most unusual feature of the file system. Each supported I/O device is associated with at least one special file. Special files are read and written just like ordinary files, but requests to read or write result in activation of the associated device handler rather than the normal file access mechanism. An entry for each special file normally resides in some subdirectory of /dev.

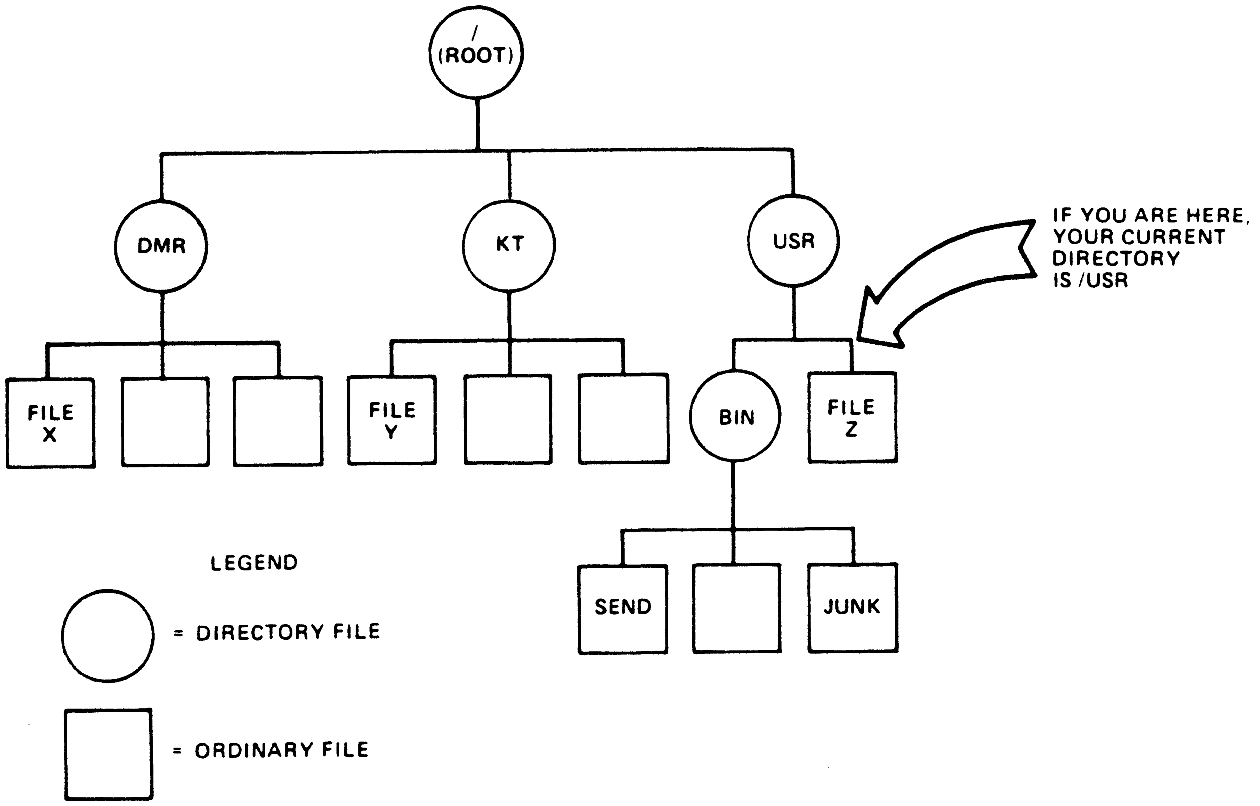

Figure 1-2. Hierarchical File System

1.1.4.3 Absolute Pathnames. When the name of a file is specified to the system, it may be specified as a pathname, which is a sequence of directory names, each separated by a slash (/) and ending with a filename. This sequence of directories preceding the filename is called a prefix. SYSTEM V/68 uses certain conventions when reading the prefix. If the prefix begins with a slash, the search begins in the root directory. This is called an absolute or full pathname. The pathname

/usr/bin/send

causes the system to search the root directory for directory *usr*, then to search *usr* for *bin*, and finally to find file *send* in *bin*. The file *send* may be an ordinary file, a directory, or a special file. As a limiting case, the name / refers to the root directory itself.

1.1.4.4 Relative Pathnames. A null prefix (or any prefix that does not begin with a slash) causes the system to begin the search in the current user directory. The simplest form of pathname (for example, *send*) refers to a file that is found in the current directory. This relative pathname allows a user to quickly specify a subdirectory without needing to know (or input) the full pathname. This is just one of several mechanisms built into the file system to alleviate the need to remember pathnames. For example, files can be linked across directories. Therefore, by linking a file to the current directory, the user need not supply a prefix when accessing the file. Also, the prefix ".." refers to the parent directory (the directory containing the current directory). When a process is created, a current directory and a root directory are associated with that process.

1.1.4.5 Access Permissions. Although the access (read, write, and execute) protection scheme is simple, it has some unusual features. Each user of the system is assigned a unique user identification (ID) number as well as a shared group identification number. When a file is created, it is marked with the user ID and group ID of its owner. Also given for new files is a set of protection bits that specify independent read, write, and execute permission for the owner of the file, for other members of the group, and for all other remaining users (refer to **chmod**(1)). The execute permission bit for a directory file is interpreted as **search** permission in that directory.

1.1.4.6 Disk Data Structure. The SYSTEM V/68 file system is a disk data structure accessed completely through the block I/O subsystem. A disk is considered a randomly addressable array of blocks. The operating system actually views memory physically as 512-byte sectors and converts the sector numbers to the logical block numbers. This is invisible to the user. The first sector, set aside for booting procedures, is unused by the file system. The second sector is the **superblock**.

The superblock contains a description of the file system (or volume) and includes the following:

- Size of the i-list and the entire volume
- Free block list and the number of free blocks
- Free i-node list and the number of free i-nodes

- Read-only status
- Mount device, pack name, and file system name
- File system type

Several file systems can be mounted simultaneously on different devices. Each mounted file system has a superblock and i-list. The i-list is after the superblock. It is nothing more than a list of file definitions. Each file definition is a multibyte structure called an i-node. The SYSTEM V/68 user accesses a file with a pathname, but the system itself uses the i-node to do the accessing.

The offset (or index number) for a particular i-node within the i-list is called its i-number. The combination of device name (major and minor) and i-number serves to uniquely name a particular file. To speed up the allocating of i-nodes, an i-node array and i-node list are maintained in primary memory in addition to the information contained in the i-node itself in secondary memory. The list and array work together to provide a buffer of up to 100 free i-nodes that can be allocated and freed quickly and efficiently.

1.1.4.7 I-nodes. An i-node contains a description of the file to which it points. This description includes the following information:

- The user and group ID of the file owner
- The file protection bits
- The physical disk address for the file contents
- The file size
- Time of creation, last use, and last modification
- The number of links to the file; that is, the number of times it appears in a directory under different names.

1.1.4.8 Mount Table. When another file system is mounted on the file system hierarchy, that system simply becomes an extension of the current file system. To allow mounting of other file systems (and easily unmounting them), a mount table is maintained. The mount table contains pairs of designated i-nodes and block devices. At each of the i-nodes listed, a file system is mounted (mounted on the device indicated).

1.1.5 I/O System

The Input/Output (I/O) system is divided into two separate subsystems: the block I/O subsystem (structured I/O) and the character I/O subsystem (unstructured I/O).

The user communicates with the peripheral devices by system calls. The system calls to input or output are designed to eliminate the differences between the various devices and styles of access. There is no distinction between **random** and **sequential** I/O, nor is any logical record size imposed by the system. SYSTEM V/68 also attempts to eliminate differences between ordinary disk files and I/O devices such as terminals, tape drives, and line printers.

An entry appears in the file system hierarchy for each supported device, so that the structure of device names is the same as that of filenames. Not only do the same read and write system calls apply to devices and disk files; the same protection mechanisms also apply. Figure 1-3 shows the functional layout of the I/O system.

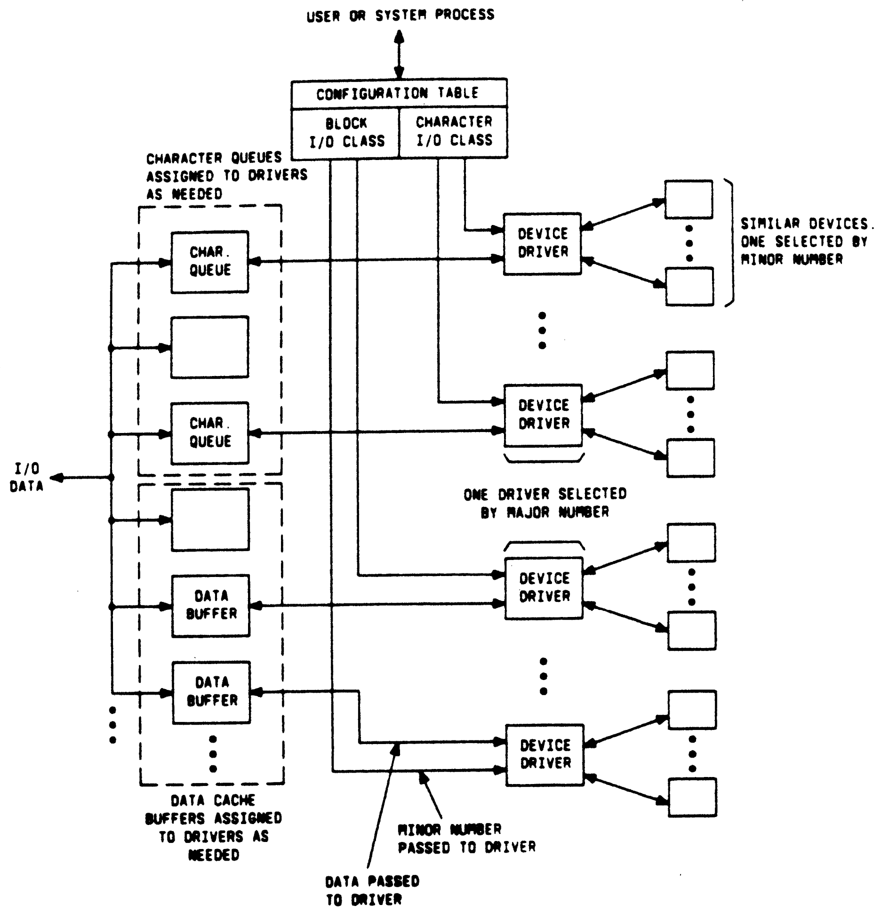
USER OR SYSTEM PROCESS
CONFIGURATION TABLE
BLOCK I/O CLASS
CHARACTER I/O CLASS
CHARACTER QUEUES ASSIGNED TO DRIVERS AS NEEDED
CHAR. QUEUE
CHAR. QUEUE
DATA BUFFER
DATA BUFFER
I/O DATA
DATA CACHE BUFFERS ASSIGNED TO DRIVERS AS NEEDED
DEVICE DRIVER
DEVICE DRIVER
DEVICE DRIVER
DEVICE DRIVER
SIMILAR DEVICES. ONE SELECTED BY MINOR NUMBER
ONE DRIVER SELECTED BY MAJOR NUMBER
MINOR NUMBER PASSED TO DRIVER
DATA PASSED TO DRIVER

Figure 1-3. I/O System

Each supported I/O device is associated with at least one special file and one device driver. Devices are characterized by a major device number, a minor device number, and a class (block or character). For each class, there is an array of entry points (configuration table) into the device drivers. The major device number is used to index the array when calling software for a particular device driver. The minor device number is passed to the device driver as an argument. The minor number has no significance other than that attributed to it by the driver. Usually, the driver uses the minor number to access one of several identical physical devices.

1.1.6 Shell

1.1.6.1 Functions. The user communicates with SYSTEM V/68 with the aid of a command programming language called the shell. The shell is a command-line interpreter; it reads lines entered by the user and interprets the lines as requests to execute other programs. The shell also provides conditional execution and flow control features. Thus, it is also a programming language (not dissimilar to the C language).

When a user logs into the system and before executing the shell, the login process sets up an execution environment. System default and user-specified profile files are read, and then the appropriate shell program is executed. The environment determines what commands the user has access to. Normally, a user will log into the system with one process associated with his or her user name. This process is the user's shell and will become the parent of all other processes the user creates. Although SYSTEM V/68 is set up to allow any binary program to be used as a shell, the actual shell program is normally used. The basic function of the shell from the operating system's point of view is simply to parent child processes, and a process need not be the shell program to do this. From the user's viewpoint, the shell should accept user input, execute commands, provide output, and access files.

1.1.6.2 Built-In Commands. There is a small subset of shell commands that are built in and differ from other commands primarily in that no separate process is created to execute these commands. They are part of the standard shell program and include all the execution and flow control constructs (refer to **sh**(1) for a description of individual built-in commands). All other commands are either functions, utility programs, or application programs.

1.1.6.3 File Descriptors. Programs executed by the shell begin with three open files with file descriptors 0, 1, and 2. When a program begins, file 1 is open for writing and is known as the standard output. Conversely, file 0 starts off open for reading and programs that must input messages entered by the user can read file 0, which is known as the standard input. File descriptor 2 starts off open for writing and is known as standard error. Files 0, 1, and 2 are normally attached to the user's terminal. Unless otherwise directed, input is taken from the terminal and output is sent to the terminal.

1.1.6.4 Pipes. An extension of the standard I/O notion is used to direct (or pass) the output from one command to the input of another without the use of temporary files maintained by the user. A sequence of commands separated by vertical bars (|) causes the shell to execute all the commands and to arrange that the standard output of each command be delivered (piped) to the standard input of the next command in the sequence.

A related shell feature, the &, allows asynchronous execution of commands and pipelines. If the pipeline is followed by &, the parent will not wait for the controlling process to exit before prompting the user for the next input. Also, the shell will output the process number of the controlling process to allow the user to track the progress of the command.

1.1.6.5 Shell Scripts. The shell can execute commands from a file. The commands are executed sequentially until an EOF is encountered or an exit command is executed. This feature allows the user to take advantage of the programming power of the shell.

For more information on the shell program, refer to **sh**(1) in the *SYSTEM V/68 User's Reference Manual* and the chapter "An Introduction to Shell" in the *SYSTEM V/68 Programming Guide.*

2. GETTING STARTED

2.1 SYSTEM DESCRIPTION

The VME Series 20 is a high performance, super-microcomputer system housed in a tower enclosure. It is configured using standard VMEbus modules making it an excellent system for the end user, as well as for the customer wishing to make hardware or software extensions for other applications.

SYSTEM V/68 software is furnished on the fixed hard disk. All of the object release software is already installed; however, should reinstallation be required, instructions for installing software are provided in Chapter 3 of this manual.

2.2 SYSTEM CHECKLIST

The front panel and back panel of the system tower are shown in Figures 2-1 and 2-2.

NOTE

This section is provided as introductory information. Do not attempt to power up until you reach paragraph 2.3, "System Power-Up."

2.2.1 Controls and Indicators

2.2.1.1 ON-OFF System Power Switch. The ON-OFF system power switch is located on the front panel. The ON-OFF system power switch is used to turn on power to the system. The "0" represents the OFF position; the "1" represents the ON position.

2.2.1.2 Hold-Run-Reset Key Switch. A spring-loaded reset switch is located directly below the ON-OFF Power Switch. For normal system operation turn the key switch to the RUN position. If it is necessary to reboot the system, turn the key to the RESET position. To prevent unauthorized use of the system, turn the key to the HOLD position and remove the key. The system can still be powered on or off in this position, but it cannot be used because it is locked in reset.

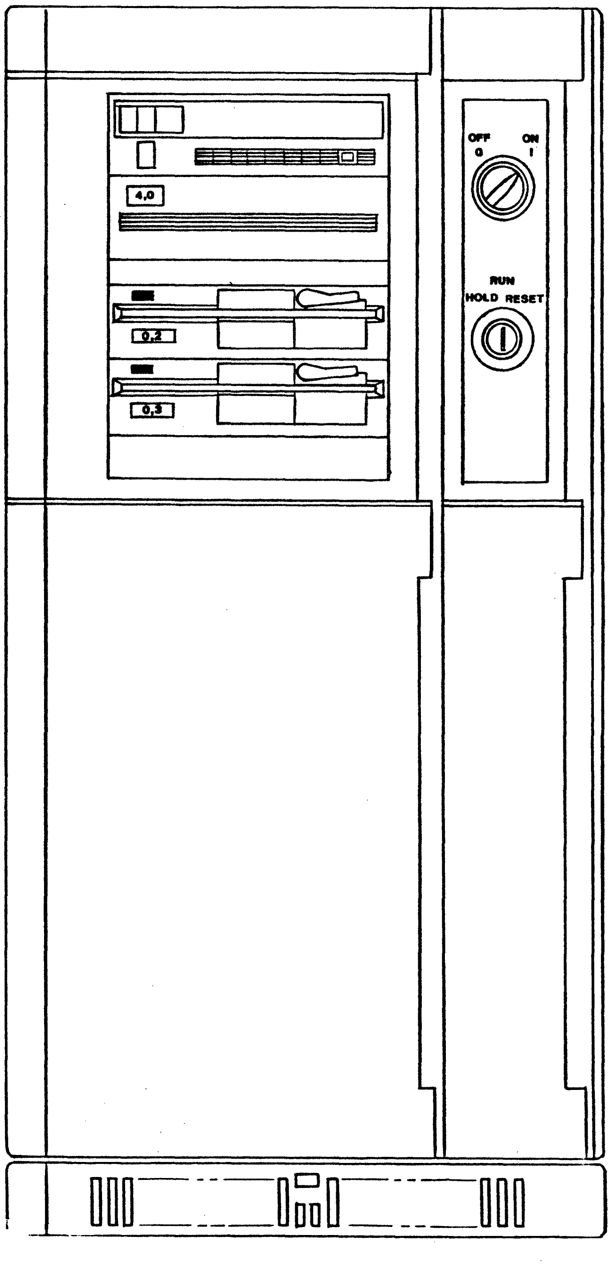

Figure 2-1. VME Series 20 Front Panel

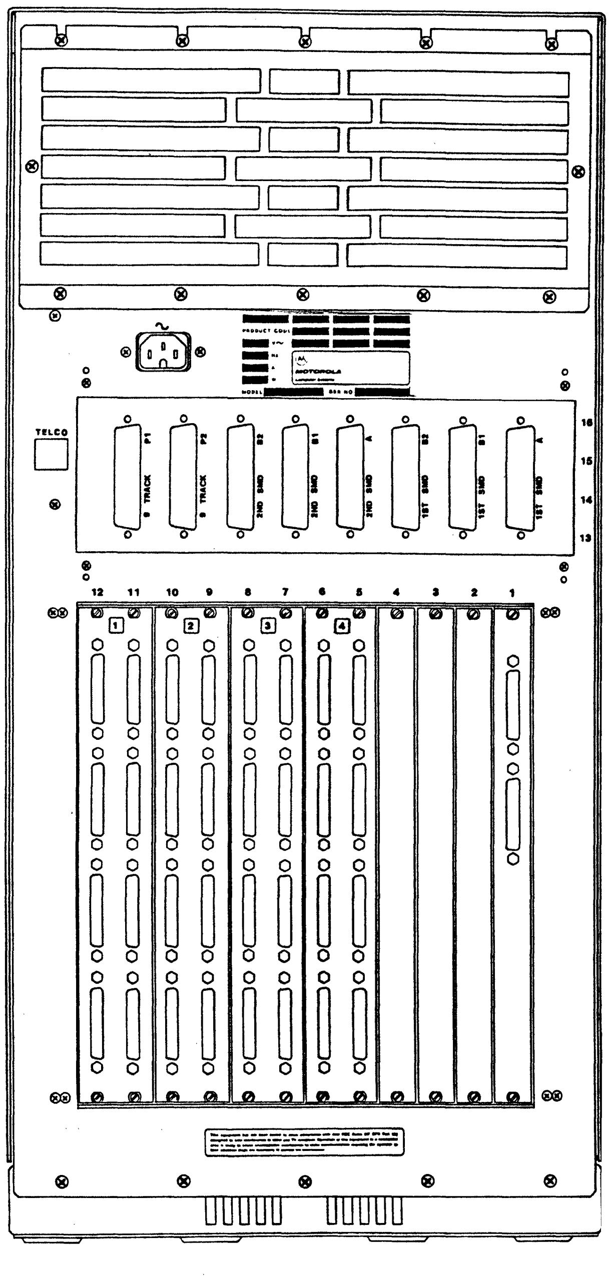

Figure 2-2. VME Series 20 Back Panel

2.2.2 Peripheral Equipment

The system is shipped ready for connection to terminals and printers.

2.2.2.1 Terminals. A standard TTY-compatible CRT/keyboard station with a 25-pin, RS232C, DTE to DTE, full-duplex cable should be connected to the serial port (CONSOLE) on the MVME707 module.

SYSTEM V/68 expects CONSOLE on the MVME707 module to be the system console. Port 2 of the MVME707 is reserved for the internal support modem. No external connection should be made. Additional terminals can be attached to the SMM1437 transition boards.

The system console terminal attached to CONSOLE should be configured as follows:

9600 baud
full-duplex
8 data bits
1 stop bit
no parity

For information on how to install terminals, refer to the system Installation/User Guide.

2.2.2.2 Printers. Serial interface printers can be attached to serial ports. For information on how to install printers, refer to the system Installation/User Guide.

2.3 SYSTEM POWER-UP

NOTE

If you want to use an alternate boot device, access the system debugger, initiate a call to the Motorola Customer Support Organization, or display system test errors, refer to paragraph 2.4, "System Power-up Menu."

Autoboot of the system is not yet possible if the **root** file system resides on an SMD drive, and there are no Winchester drives resident on the system. In this case, you MUST select an alternate boot device. Refer to paragraphs 2.4 and 2.4.2. See the *VME Series 20 Software Release Guide* for information on how to autoboot an SMD-based **root** file system when there *is* a Winchester configured as Controller 0, Drive 0.

To power up the system, turn the ON-OFF system power switch to the ON position and the key switch to the RUN position. If you have any external equipment, such as SMD drives, power them up AFTER powering up the main tower.

Immediately after power is turned on, an autoboot process performs minimal self tests. If these tests are successful, the following message is displayed on the system console (CONSOLE on the MVME707) within 10 seconds:

```
System Self Test   Rev. x.y
FPC passed test

(pause)
```

where *x.y* is the current revision of the System Self Test and FPC refers to floating point co-processor. The FPC message will display "passed test," "failed test," or "not present."

If a minimal system self test fails, an error message is displayed. In this case, or if no message at all is displayed within 10 seconds of powering on, turn the power switch to the OFF position and refer to paragraph 2.5, "Troubleshooting Hints."

After displaying the "System Self Test" message, the system continues with extended self tests. During testing, a display line will appear on the screen with the name of the system self test being performed followed by the word "PASSED" or "FAILED." This display line will change rapidly.

If all tests pass, the following message appears on the system console:

```
Testing Complete

Loading Operating System
```

When booting from an alternate boot device (see paragraph 2.4.2), the controller number, drive number, and file name (if changed) are displayed following the "Testing Complete" message.

SYSTEM V/68 is now loaded into system memory. The system reports the amount of available system memory:

```
IPL loaded at : $00050000

SYSTEM V/68 M68020 Version RxVy
01vme131
Copyright (c) 1985 AT&T
Copyright (c) 1986 Motorola, Inc.
All Rights Reserved

real mem = n
avail mem = n
```

where *x* refers to the software release number and *y* to the software version number. The value of *n* varies from system to system, depending on the amount of system memory.

The system as shipped moves automatically through **single-user mode** and comes up in **multi-user mode**. Note that single-user mode does not mean the operating mode when only one user is signed on. This mode might be more appropriately named "System Maintenance Mode" because its only appropriate use is to install software on the **root** device, back up and restore file systems, perform file system checks, and recover from crashes. All other system administrative functions can be performed in multi-user mode. If you need to enter single-user mode, use the shutdown procedure described in paragraph 2.7.

When the system comes up, the terminal will display:

```
Is the date day month date year correct? (y or n)
```

If the date is incorrect, type **n** and set the date and time. To enter the correct date and time, use the following format: *mmddhhmmyy*, where *mm*=month, *dd*=day, *hh*=hour, *mm*=minute, *yy*=year (refer to **date**(1) in the *SYSTEM V/68 User's Reference Manual*). For example,

```
0928073085
```

is the entry for September 28, 1985 at 7:30 a.m. Remember that the month, day, hour, minute, and year must each be a two-digit number with no blanks in the format; the entry for year is optional. In addition, the clock does not understand a.m. and p.m., so if it is 4:06 p.m., type the time in military time format, i.e., 1606.

If the date is correct, type **y** and the terminal will display:

```
Do you want to check the file systems? (y or n)
```

To diagnose and repair possible file system damage, a file system check is recommended. Type **y**.

During file system check you will see various messages relating to the check on the screen. You can ignore these messages. When the file system check is completed (or if you typed **n** to bypass the check), the following message will be displayed:

```
***SYSTEM MULTIUSER day month date time year***
```

The login prompt will then appear:

```
Console login:
```

2.4 SYSTEM POWER-UP MENU

Immediately after the system is powered up, an autoboot process performs minimal self tests. If these tests are successful, the following message is displayed on the system console.

```
System Self Test   Rev. x.y
FPC passed test

(pause)
```

where *x.y* is the current revision of the System Self Test and FPC refers to floating point co-processor. The FPC message will display "passed test," "failed test," or "not present."

After the "System Self Test" message is displayed, there is a five-second delay. During this time you can halt the start-up sequence by typing the letter **h** (halt). The following menu is then displayed:

```
1) Continue System Start-up              (refer to paragraph 2.4.1)
2) Select Alternate Boot Device          (refer to paragraph 2.4.2)
3) Go To System Debugger                 (refer to paragraph 2.4.3)
4) Initiate Call to CSO                  (refer to paragraph 2.4.4)
5) Display System Test Errors            (refer to paragraph 2.4.5)
Enter menu #:
```

2.4.1 Continue System Start-up

To continue system start-up, type **1** in response to the **halt** menu below.

```
1) Continue System Start-up
2) Select Alternate Boot Device
3) Go To System Debugger
4) Initiate Call to CSO
5) Display System Test Errors
Enter menu #:
```

This menu item is usually selected after completion of one of the other menu items. For example, after selecting an alternate boot device using menu item #2, the system returns the **halt** menu to the screen. You then type **1** to continue system start-up and the messages described in paragraph 2.3 are displayed on the screen. Selecting menu item #1 will always cause system start-up to begin from the first extended system self test.

2.4.2 Select an Alternate Boot Device

To select a boot device other than the default device (controller 0, drive 0), type **2** in response to the **halt** menu below.

```
1) Continue System Start-up
2) Select Alternate Boot Device
3) Go To System Debugger
4) Initiate Call to CSO
5) Display System Test Errors
Enter menu #:
```

The following prompt is then displayed:

Enter Alternate Boot Device (Controller, Drive, File): *x,y,z*

where *x* is the controller to be accessed, *y* is the drive, and *z* is the file to be loaded.

The system has an autoboot feature which defaults to the first MVME320A controller (0), the first Winchester drive (0), and the /stand/unix file. Use Table 2-1 to choose the alternate boot device configuration you want.

TABLE 2-1. Boot Device Configurations

CONTROLLER	DRIVE	FILE
	Winchester	
*0 = first MVME320A	*0 = first Winchester drive	Any file in the UNIX file
or	or	system partition 0 (directory "/").
1 = second MVME320A	1 = second Winchester drive	* /stand/unix
	Floppy	
	2 = first floppy drive	Any file in the UNIX file
*0 = first MVME320A	or	system partition 0 (directory "/").
	3 = second floppy drive	* /stand/unix
	Storage Module Device (SMD)	
2 = first MVME360	*0 = first SMD drive	Any file in the UNIX file
or	or	system partition 0 (directory "/").
3 = second MVME360	2 = second SMD drive	* /stand/unix

* = DEFAULT VALUE

Example: To boot diagnostics (contained in the file /diag) from the first floppy drive, type **0,2,diag** in response to the "Enter Alternate Boot Device" prompt. The **0** represents the first MVME320A controller, **2** is the first floppy drive, and **diag** is the file in the UNIX file system partition 0 (the "/" root directory).

Typing the default values is optional. In the previous example, typing **,2,/diag** has the same effect as **0,2,/diag.**

After typing the alternate boot device, the **halt** menu is displayed again. Type **1** to continue system start-up. Before booting from the alternate boot device, the controller number, drive number, and file name (if changed) are displayed following the "Testing Complete" message.

For the previous example, the following message would be displayed:

```
Controller 0, Drive 2
Loading File: diag
```

The messages described in paragraph 2.3 are also displayed.

2.4.3 Go to System Debugger

This feature should be used only by someone familiar with the 130Bug and 130Diag facilities.

To go to the system debugger, type **3** in response to the **halt** menu below.

```
1) Continue System Start-up
2) Select Alternate Boot Device
3) Go To System Debugger
4) Initiate Call to CSO
5) Display System Test Errors
Enter menu #:
```

The following prompt is then displayed:

```
130Diag>
```

While in the 130Bug or 130Diag, any error or exception process will cause control to be passed to the **halt** menu. Refer to the *MVME 130Bug Debugging Package* (MVME130BUG/D2).

To return to the **halt** menu, type **menu.**

2.4.4 Initiate a Call to CSO

The system has a remote maintenance facility that allows for test and diagnosis of system problems from a remote Customer Support Organization (CSO) Customer Service Center(CSC) via an internal modem. This facility can be used if extended System Self Test fails. After a System Self Test error is displayed the start-up sequence will stop and display the **halt** menu:

```
1) Continue System Start-up
2) Select Alternate Boot Device
3) Go To System Debugger
4) Initiate Call to CSO
5) Display System Test Errors
Enter menu #:
```

To initiate a service call, type **4**. The following message is displayed:

```
Initiating CSO Service Call

Internal modem test in progress. . . . . . . . . . . . . TEST PASS
```

The "TEST PASS" message is displayed after the internal modem tests are completed. Each dot following the "Internal modem test in progress" message represents one successfully completed test. The testing can take up to two minutes. The system then requests:

```
Enter SPR number:
Enter CSO service number:
```

Enter the SPR number for the system and the CSC telephone number supplied by the Customer Support Organization. The following messages will then be displayed:

```
Service call in Progress  -Connected
Error table dump in progress  -Complete
```

The system automatically dials the CSC and displays the "Connected" message.

If the call is unsuccessful, one of the following messages is displayed:

NO ABT	(Indicates no answer-back tone detected)
NO NUMBER STORED	(Phone number was not input by user)
BUSY	(Number dialed had busy signal)
NO ANSWER	(Call not answered after five rings)

Control will then be passed to the **halt** menu. You should try to place the call again after several minutes by typing the number **4**. If you are still unable to connect, telephone your local field representative.

If a connection is made, the system automatically sends an error log. After the error log dump is complete, a message from the CSC will be displayed advising further action. An operator at the CSC may request concurrent access to the system by displaying the following message:

```
Concurrent mode (y/n):
```

In the concurrent mode the operator at the CSC will be able to run tests and diagnose hardware and some software problems. (To stop the concurrent mode at any time, press and hold down the **Ctrl** key and then type the letter **a**. After the session is complete the system will hang up and display the following message:

```
Service call complete
```

After the call is complete, the **halt** menu will be displayed again.

2.4.5 Display System Test Errors

This feature should be used only by someone familiar with the 130Bug and 130Diag facilities.

To display system test errors, type **5** in response to the **halt** menu below.

```
1) Continue System Start-up
2) Select Alternate Boot Device
3) Go To System Debugger
4) Initiate Call to CSO
5) Display System Test Errors
Enter menu #:
```

All system self test errors that have been detected are then displayed in a table format. The table contains the major test category and number of failures within that category. Errors are cleared only at the beginning of the extended system self test sequence or manually from the debugger. Refer to the *MVME 130Bug Debugging Package* (MVME130BUG/D2).

After the table is displayed, the **halt** menu appears on the screen.

2.5 TROUBLESHOOTING HINTS

This paragraph describes initial steps to take if problems occur. If the solutions suggested below do not solve the problem, contact the Motorola Customer Support Organization.

SYMPTOM	POSSIBLE SOLUTION
No power; No sound of fans turning.	Plug in line cord. Note: There are no user-replaceable fuses nor a circuit breaker.
Cannot communicate with system.	Insure that RS232C cable is DTE to DTE, full duplex (3-wire cables will not work with this system).
	Make sure RS232C cable is plugged into the MVME707 in the bottom RS232C connector, which is CONSOLE.
	Make sure your terminal is on-line.
System will not boot.	Insure that RS232C cable is DTE to DTE, full-duplex (3-wire cables will not work with this system).

2.6 STARTUP PROCEDURES

2.6.1 Logging in as Root

Root is a very powerful user, appropriately called the **superuser**. The superuser is the only user who may perform all of the system maintenance tasks. Because the superuser, or root, is the only user with access to everything on the operating system, whoever is logged in as root is in a position to access, change, or remove anything on the system. You should reserve the powers of the superuser for only a limited number of users, perhaps solely for the system manager. (Refer to the *SYSTEM V/68 Administrator's Guide* for more information about the role of the system manager.) You may do this by giving root a password known only to those users to whom superuser powers are given.

Logging in is the process of identifying yourself to the system. To log in as root, type:

```
Console login: root
```

2.6.2 Creating Passwords

The following procedure can be used to create a root password and to change ordinary user passwords.

Type the **passwd**(1) command after the root prompt (#):

```
# passwd
```

The response from SYSTEM V/68 is

```
Changing password for root
New password:
```

Of course, in this case you change the password from none to one of your choice. Suppose you want to use the password "two4tea". Type:

```
New password: two4tea
```

when the system asks you for the new password. It must be at least six characters long; either uppercase or lowercase letters can be used, but the password must contain at least one number or special character and at least two alphabetic characters. It should be complicated enough so that it is impossible to guess, but relatively easy for you to remember. Note: The system does not insist on these conventions for the superuser password; however, it is good practice to follow them.

The word "two4tea" does not appear on your screen as you type it. However, you are asked to type it a second time to double check that you have typed it correctly. Simply type **two4tea** again.

```
Re-enter new password: two4tea
```

If you do not type it correctly, the following message appears:

```
They don't match; try again.
```

You must type the **passwd** command again and go through the process a second time. In the future, every time you try to log in as root you will be asked for this password.

Note that as long as you are logged on as root, the prompt is #. To log off, type:

```
# exit
```

or press the **Ctrl** key and type **d**.

2.6.3 Creating User Accounts

As shipped, the system has four user accounts already established. They are: *user1, user2, user3*, and *service*. This paragraph describes how to create additional accounts.

The superuser creates user accounts by means of the following four steps.

1. Edit the /etc/passwd file to create an entry for each new user. Fields in the entry are separated by colons. The seven fields provided are:

   login name
   encrypted password
   numerical user ID
   numerical group ID
   job number, box number, optional user ID
   initial working directory
   program to use as shell

   For example:

   ```
   susan::200:101::/usr/susan:/bin/sh
   ```

   The "Administrative Guidelines" chapter of the *SYSTEM V/68 Administrator's Guide* contains paragraphs describing the fields of the /etc/passwd file. As shown in the example above, the user ID and group ID numbers should be greater than 100; lower numbers are reserved by the system. Also note that the second field is blank initially. This is the field for the encrypted password. When Susan logs onto the system for the first time, she should use the **passwd**(1) command to enter a unique password. The **passwd** command will fill in the field in /etc/passwd.

   The command **pwck** can be used to verify additions to the file /etc/passwd (refer to **pwck**(1M) in the *SYSTEM V/68 Administrator's Manual*).

2. Edit the /etc/group file. For reasons of security or to restrict file access, the administrator assigns each user a group ID. Typically, this ID is based on the user's relationship to other users. Each entry in this file has four fields, separated by colons:

group name
encrypted password
group ID
comma-separated list of all users allowed in the group

For example:

```
pubs::200:susan,smith,jones,thomas
```

The format for this file is described in the "Administrative Guidelines" chapter of the *SYSTEM V/68 Administrator's Guide*. The third field of each line in /etc/group is a number which should be the same as the number in the third field of an entry in the file /etc/passwd.

Additions to /etc/group can be verified with the **grpck** command (refer to **grpck**(1M) in the *SYSTEM V/68 Administrator's Manual*).

3. Make a directory in /usr for the user being added. For example, if we are adding the user Susan to the system, type:

```
# mkdir /usr/susan
```

4. Change the ownership and the group identification of the directory from root to the new user. This is done with the **chown** and **chgrp** commands (refer to **chown**(1) and **chgrp**(1) in the *SYSTEM V/68 User's Reference Manual*). For example, typing the command:

```
# chown susan /usr/susan
```

changes ownership of the directory "susan" (the second susan in this example) to the user susan (the first susan in this example). Similarly, the command

```
# chgrp pubs /usr/susan
```

changes the group ID of the directory susan to pubs. Care should be taken that the home directory has the appropriate permissions. You can check the status of these permissions by using the **ls** command with the **-l** option. Before you change the ownership and group identification of susan's home directory, typing the command

```
# ls -l /usr
```

will show the line

```
drwxr-xr-x 2 root   root   32  Aug 30 13:03 susan
```

After the commands **chown** and **chgrp** have been performed, the line will read:

```
drwxr-xr-x 2 susan  pubs   32 Aug 30 14:01 susan
```

Note that the **chgrp** command will not work unless the file /etc/group has been edited as described above. The read-write-execute permissions shown in the example are those for normal operations in which only the owner has write permission and the group and others have read and execute permissions. Refer to **chmod**(1) in the *SYSTEM V/68 User's Reference Manual* for information on how to change file permissions.

2.6.4 Logging on as an Ordinary User

To log on, type:

```
Console login: logname
```

The system will respond with a $ prompt. Remember that $ is the default prompt for normal users; # is the root, or superuser, prompt. If you have any doubts about whether you are logged in as yourself or as root, you should be able to tell immediately by which system prompt appears on the screen. You can also use the **id**(1) command, which displays your user ID and group ID.

To create a user password, use the process shown above for the root password. The next time you log on to the system, you will enter this password in response to the logon prompts:

```
Console login:
Password:
```

If either the login name or the password is typed incorrectly, another "login:" prompt will be displayed.

2.6.5 Using the .profile File

If a file named **.profile** exists in your home directory, it is invoked to set up the environment and terminal variables when you log on. When the user accounts were created for the system (refer to paragraph 2.6.3), a **.profile** was placed in the home directory of each user. If you want information about some of the variables that have been set, use the **env**(1) command.

```
$ env
```

The terminal will display several lines of information in the form *name=value*, where *name* is a variable and *value* is the current assignment for that variable.

When additional user accounts are created, the root **.profile** that is shipped with the system can be copied to the user's home directory by typing:

```
$ cp /.profile /usr/logname/.profile
```

where *logname* is the name of the user's account.

Now the user should log off the system, by typing

```
$ exit
```

or pressing the **Ctrl** key and typing **d**; then log back on so that the **.profile** is invoked. This time the user will be prompted for the type of terminal being used. This information is used by the editors **vi**(1) and **ex**(1).

2.6.6 Command Syntax

In SYSTEM V/68, commands have the following structure:

command argument argument argument ...

White space is used to delimit command names and arguments. A carriage return (CR) enters the command. Arguments that contain spaces or tabs should be enclosed with double quotes.

A demonstration file is provided with the system to introduce common commands and procedures. This file is located in /mot/bin/getstarted and can be read by typing:

```
$ startup
```

2.7 SHUTTING DOWN THE SYSTEM

It is extremely important to use the correct procedure to shut down the system. Do not panic and use the RESET key switch, the ON/OFF system power switch, or the electrical plug to stop your system. If you feel you have made a mistake, this will only compound the problem.

Only the superuser may shut down the system. If you are logged on as a user, log off this account and log back on as **root**. When you see the root prompt (#), type:

```
# shutdown n
```

where *n* is the number of seconds that will elapse from the time users are notified to the time the system is shut down. The **shutdown**(1M) command terminates all currently running processes in an orderly and cautious manner. The superuser is asked whether a special message is to be used to broadcast the shutdown to users. Users are informed how many seconds will elapse before shutdown. The **shutdown**(1M) program then kills all active processes, unmounts file systems, and updates the file system superblocks (by means of the **sync**(1) command). The message: "Wait for INIT SINGLE USER MODE before halting" appears on the screen.

When the message "INIT SINGLE USER MODE" appears, type:

```
# sync
# sync
# sync
```

and wait 30 seconds after the root prompt (#) is displayed. The system can then be halted in two ways.

1. If you plan to power down for a period of time, turn off any external equipment, such as SMD drives; then turn the ON/OFF system power switch on the front panel to the OFF position.
2. If you plan to reboot the system immediately, turn the RESET key switch on the front panel to RESET.

2.8 BACKUP PROCEDURE

This procedure can be used to back up any double-sided, double-density, 5 1/4-inch diskette including the two "Minload" diskettes provided with the system.

NOTE

> Normally only the superuser is allowed to execute the utilities **dinit**(1M), **mkfs**(1M), **mount**(1M), and **umount**(1M). Included with this release is a set of similar utilities that allow a normal user to perform these activities. These utilities are: **fmt**(1M), **fs**(1M), **mnt**(1M), **umnt**(1M). The superuser makes entries in the file /mot/etc/perms to allow access to these utilities by normal users. For further information, refer to the *System V/68 Administrator's Manual.*

2.8.1 Format Diskettes

Format a blank diskette using the **dinit**(1M) or **fmt**(1M) utility. Only double-sided, double-density, 5 1/4-inch diskettes can be formatted.

To format a diskette, remove the write-protect tab from the diskette and insert the diskette into the drive.

If you have superuser permission, type the command:

```
# dinit -f -o m320dsdd5 /dev/r0xs7
```

where *x* specifies the correct physical drive number, that is, **2** for the first floppy drive (standard configuration) and **3** for the second floppy drive.

If you do not have superuser permission, you can use the command:

```
$ fmt flp
```

A bootable diskette can be created by typing:

```
$ fmt bflp
```

When a bootable floppy is made, a file system must be created on slice 0. To make a file system on the diskette, invoke **mkfs**(1M) on the diskette's block device entry, specifying either *1264* for slice 0 or *1276* for slice 7 as the blocks argument. For example:

```
# mkfs /dev/r02s7 1276
```

If you do not have superuser permission, you can create a 1264 block filesystem on slice 0 by typing:

```
$ fs flp
```

2.8.2 Copy Diskettes

Insert the diskette into the floppy drive. Use the **dd**(1) command to copy the contents of the diskette onto the hard disk.

```
dd if=/dev/r0x of=/backflop bs=32k
```

where *x* specifies the correct physical drive number, that is, **2** for the first floppy drive (standard configuration) and **3** for the second floppy drive.

Remove the diskette from the floppy drive, put the formatted blank diskette into the drive, and use the **dd**(1) command to copy the files back from the hard disk.

```
dd if=/backflop of=/dev/r0x bs=32k
```

2.9 RELATED DOCUMENTATION

The following publications may be obtained from Motorola Computer Systems, 3013 S. 52nd St., Tempe, Arizona 85282 or from Motorola Computer Systems Literature Stores, 10700 N. De Anza Blvd., Cupertino, California 95014.

Software manuals	Product Code
Complete set of the following SYSTEM V/68 documentation:	72912
Individual manuals:	
SYSTEM V/68 User's Reference Manual	72905
SYSTEM V/68 Administrator's Manual	72900
SYSTEM V/68 Administrator's Guide	72901
SYSTEM V/68 Error Message Manual	72902
SYSTEM V/68 User's Guide	72903
SYSTEM V/68 Operator's Guide	72904
SYSTEM V/68 Document Processing Guide	72906
SYSTEM V/68 Programming Guide	72908
SYSTEM V/68 Support Tools Guide	72909
SYSTEM V/68 Assembler User's Guide	72910
SYSTEM V/68 Common Link Editor Reference Manual	72911

The following publications provide additional helpful information. They may be obtained from Motorola's Literature Distribution Center, 616 West 24th St., Tempe, Arizona 85282.

Hardware and Debug manuals

MVME050 System Controller Module and MVME701/MVME701A I/O Transition Module	MVME050/D2
MVME130 VMEmodule 32-bit Monoboard Microcomputer Module Manual	MVME130/D3
MVME204-1/-2 Dual Ported Dynamic Memory Module User's Manual	MVME204/D2
MVME320A VMEbus Disk Controller Module User's Manual	MVME320A
MVME130Bug Debugging Package	MVME130BUG/D2
MVME707 RS-232C Serial Port Distribution Module User's Manual	MVME707/D1
M68KVMMB851 Memory Management Board User's Manual	M68KVMMB851/D1
High Level C Code Optimizer	M68KUNOPT/L1

Supplemental Documentation

VMEbus Specification Manual, Rev C.1	HB212/D
MVMX32bus Design Specification	MVMX32BS
UNIX SYSTEM V PRIMER (Waite, M., Martin, D. & Prata, S. Indianapolis, IN. Howard W. Sams & Co., Inc., 1984)	TB311
C PRIMER PLUS (Waite, M., Martin, D. & Prata, S. Indianapolis, IN. Howard W. Sams & Co., Inc., 1984)	TB310
Microprocessor Software Catalog	BR126/D

For SYSTEM V/68, release version FE81, the information on using VME Series 20 is provided in the *VME Series 20 Software Release Guide* shipped with your software. The chapter, Using VME Series 20, will be included in this manual in a future update.

APPENDIX A. SYSTEM SPECIFICATIONS

The VME Series 20 specifications are given in the table below. The specifications are subject to change without notice.

TABLE A-1. System Specifications

CHARACTERISTIC	SPECIFICATION
Temperature	
Operating	10 degrees C to 40 degrees C
Storage	-40 degrees C to 65 degrees C
Temperature gradient	
Operating	10 degrees C/hr
Non-operating	Below that which can cause condensation
Relative humidity	10% to 80% (non-condensing)
Altitude	
Operating	-200 to 7000 feet
Storage	-1000 to 50,000 feet
Physical dimensions	
Width	12.5"
Height	25.6"
Depth	28"
Weight	135 lbs. with typical full configuration
Power requirements	
Input voltage	90 - 132 VAC (labeled 110) or 180 - 264 VAC (labeled 220) internally strappable
Frequency	47 through 63 Hz
Power Usage	450 Watts typical with full configuration; yields 1500 BTUs heat output
	550 Watts maximum; yields 1900 BTUs heat output
Thermal shock	
Operating	2 degrees C/min
Storage	24 degrees C/hour
ESD	
Operating	12,000 V (except rear panel connectors)
Storage	20,000 V

APPENDIX B. ERROR MESSAGES FOR THE MVME320A

Error messages for the MVME320A disk controller and suggested actions are summarized in the following listing. Refer also to the **m320fmt**(1M) and **mvme320**(7) entries in the *SYSTEM V/68 Administrator's Manual.*

M320: drive #*n*, ctl #*n*, bad track data exceeds # buffer

DESCRIPTION: Open processing for the indicated drive and controller was unable to read the entire bad track table into a system buffer. This error also generated the "open found no format/config" message (Refer to the message that follows.) The bad track subsystem generates a "Bad table overflow" message when there is insufficient space available in its storage pool.

M320: drive #*n*, ctl #*n*, open found no format/config

DESCRIPTION: Device open was unable to read the disk, or failed to find the required volume label information on the disk. This message appears only for hard disks.

SUGGESTED ACTION: Format the disk. All other operations return an I/O error. Refer to **dinit**(1M) for information describing the formatting operation.

M320: invalid special command (2) #

DESCRIPTION: An internal error exists in the MVME320A disk controller driver. Contact your internal support organization. Your driver may have been modified incorrectly.

M320: ctl #, spurious interrupt

DESCRIPTION: An unexpected interrupt has been received from the indicated MVME320A disk controller. If the message appears immediately after bootload, it may be ignored. If the message continues to appear, the controller may be malfunctioning, or the boards may be jumpered incorrectly. Contact the Motorola Customer Support Organization.

M320: ctl #*n*, interrupt when not (logically) busy

DESCRIPTION: An interrupt from the indicated controller has been received but the driver software has no record of an outstanding request. This error could result from either a malfunctioning controller or an error in the driver.

M320: drive #*n*, ctl #*n*, mask 0*x*#, interrupt not on active unit

DESCRIPTION: An interrupt from the indicated controller shows request completion for some unit other than the expected drive. The mask is the hexadecimal value of controller register B. This error could result from either a malfunctioning controller or an error in the driver.

M320: drive #*n*, ctl #*n*, mask 0*x*#, extra interrupt

DESCRIPTION: An interrupt from the indicated controller shows request completion from some other unit, in addition to the expected drive. The mask is the hexadecimal value of controller register B. This error could result from either a malfunctioning controller or an error in the driver.

M320: drive #*n*, ctl #*n*, cmd#*n*, main #*n*, ext 0*x*#, i/o error

DESCRIPTION: This message describes an I/O error where drive #*n* and ctl #*n* refer to the MVME320A controller and drive where the error occurred, cmd #*n* defines the controller command code of the failed operation, main #*n* defines the main status as a number, and ext is the extended status expressed as a hex bit mask.

NOTE: "Sector id not found" (mask bit 0x08) may be erroneously reported when the actual source of the problem cannot be determined.

M320: dump device 0*x*# not available

DESCRIPTION: The VME12x debug monitor crash routine cannot execute because damage to the system is so severe that the driver software cannot function properly. Reboot the system or contact your internal support organization.

M320: dump truncated at end of logical device
#*n* blocks written

DESCRIPTION: Space on the dump device is exhausted. Refer the problem to your system administrator.

M320: dump I/O error, terminating

DESCRIPTION: I/O on the dump device has failed after five retries. Reboot the system.

M320: controller # timed-out, disabled

DESCRIPTION: The indicated controller started a request more than five seconds ago and has not yet generated a completion interrupt. As a result, the MVME320A driver has set internal flags that make it impossible for new requests to reach the controller. The error message reflects a serious controller malfunction. Reboot the system.

M320: controllers *#n* - *#n*, csr address conflict

DESCRIPTION: Device driver initialization has detected a fatal conflict in the assignment of control and status register (csr) addresses for the two controllers indicated. Both are unusable. The SYSTEM V/68 configuration, the MVME320A board strappings, or both, are incorrect. The csr address is set in Jumper JW6 on the board. Refer to the *SYSTEM V/68 Administrator's Manual.*

INDEX

In this index a page number indicates only where reference to a topic begins; the reference may extend to the following page or pages.

USER'S COMMENTS

SYSTEM V/68 OPERATOR'S GUIDE

Product Code 72904
Part Number 41967-00

Motorola welcomes your comments and suggestions. Please use this form.

- Does this manual provide the information you need? ☐ Yes ☐ No
  - What is missing?

- Is the manual accurate? ☐ Yes ☐ No
  - What is incorrect? (Be specific.)

- Is the manual written clearly? ☐ Yes ☐ No
  - What is unclear? (Be specific.)

- What other comments can you make about this manual?

- What do you like about this manual?

- Was this manual difficult to obtain? ☐ Yes ☐ No

Please include your name and address if you would like a reply.

Name____
Company______________________________________
Address_______________________________________
____

• What is your occupation?

☐ Programmer	☐ Operator	☐ Manager
☐ Systems Analyst	☐ Instructor	☐ Customer Engineer
☐ Engineer	☐ Student	☐ Other___________________

• How do you use this manual?

☐ Reference Manual	☐ Introduction to the Subject
☐ In a Class	☐ Introduction to the System
☐ Self Study	☐ Other_________________

fold **fold**

MOTOROLA COMPUTER SYSTEMS
3013 S. 52nd Street
Tempe, AZ 85282

Attention: Software Publications, X4

fold **fold**

Staple Here

MOTOROLA
Computer Systems

3013 S. 52nd St.
Tempe, AZ 85282