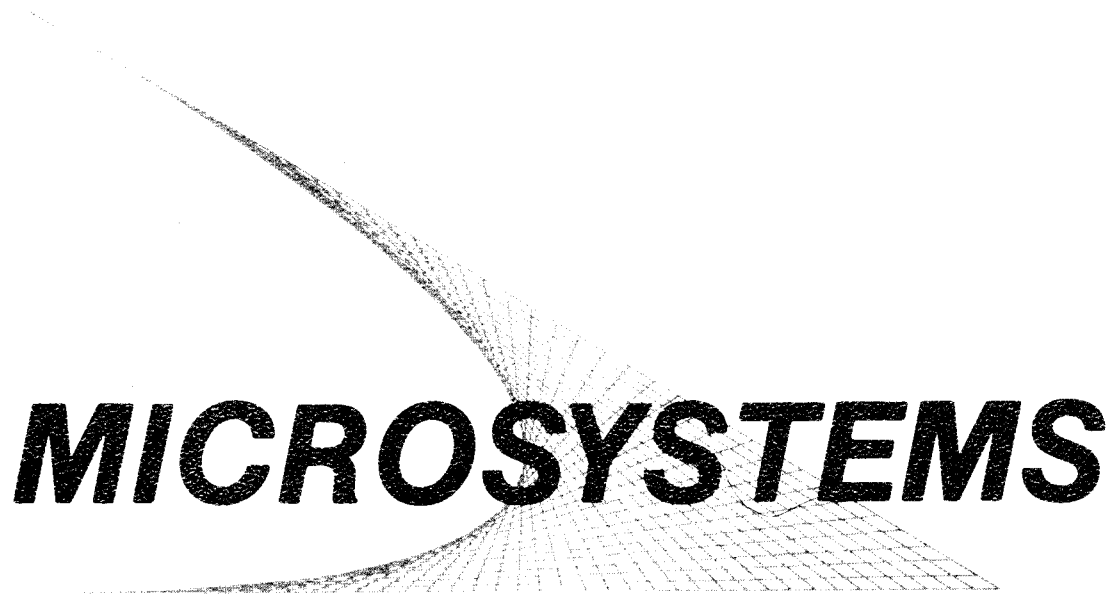
MOTOROLA

M68KVOVER/D6

VERSAdos Overview



QUALITY • PEOPLE • PERFORMANCE

MOTOROLA

M68KVOVER/D6

DECEMBER 1985

VERSAdos

OVERVIEW

Sixth Edition

Fifth Edition March 1985

MICROSYSTEMS

REVISION RECORD

M68KVOVER/D5 - March, 1985. Reflects the following software levels: VERSAdos 4.4, ASM 1.8, and LINK 1.8. Adds support of the MC68020, VM04, MVME120, MVME121, MVME122, and MVME123.

M68KVOVER/D6 -- November, 1985. Updated to include descriptions of a table-driven task initiator for RMS68K, a resident run-time library for Pascal, and a terminal-independent editor. Adds support for VMEmodules MVME117, MVME130, and MVME131. Adds a keyword index.

FORWARD

This manual is an overview or primer to the VERSAdos operating system. Brief discussions pertaining to various programming languages are also included to complete the system presentation.

Although this manual provides many details about the operating system, it cannot possibly contain all operating information pertinent to all portions of the system. Because of this, extensive references have been provided in this manual to aid you in locating these details. If conflicts are found between this manual and a specific reference manual, the information in the reference manual should take precedence.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

Page

LIST OF ILLUSTRATIONS

CHAPTER 1

VERSAdos OPERATING SYSTEM

1.1 INTRODUCTION

VERSAdos is a modular, multilayered operating system that provides a convenient interface between the user and system hardware. It is oriented to solving the general-purpose program generation requirements associated with the development of microprocessor-based systems, as well as the execution requirements of dedicated, real-time, multitasking/multi-user application systems. The modular nature of the operating system permits configuration of EXORmacs, VME/10, VMEmodule Monoboard Microprocessor-based systems (MVME101, MVME110, MVME117, MVME120/121, MVME122/123, and MVME130/131), and VERSAmodule 01, 02, 03, and 04 (VM01, VM02, VM03, and VM04) based systems that perform well in the real-time control system environment and also, for EXORmacs, in the multi-user environment. This flexibility reduces the costs and problems normally encountered during system integration by permitting extensive debugging to be performed on a compatible hardware/software configuration prior to the integration process. Many times, this will eliminate last minute software changes that are often required when the development system is not functionally compatible with the target system.

To fulfill its role, VERSAdos is responsible for accepting, checking, interpreting, and expediting user application requests. During execution of a task, the operating system may request assistance from various operating system support routines not directly accessible to the application program. These support routines assist in operator control, memory management, task segmentation, and input/output control for various hardware subsystems. This permits execution of more than one task at a time, thereby allowing several application programs to operate independently on the system. This also relieves the application program from the necessary chore of direct interaction with the system hardware. Instead, application programs communicate their input/output requests to the system via the operating system using a well-established and readily understood protocol.

The following paragraphs provide a detailed overview of the VERSAdos operating system: its features, structure, operation, and component parts. Explicit operational details concerning VERSAdos may be found in the following documentation.

M68000 Family VERSAdos System Facilities Reference Manual, M68KVSF

M68000 Family CRT Text Editor User's Manual, M68KEDIT

M68000 Family Real-Time Multitasking Software User's Manual, M68KRMS68K

VERSAdos Data Management Services and Program Loader User's Manual, RMS68KIO

1

System Generation Facility User's Manual, M68KSYSGEN

VERSAdos to VME Hardware and Software Configuration User's Manual, MVMEVDOS

These manuals and others referenced throughout this document may be ordered from Motorola's Literature Distribution Center, 616 West 24th Street, Tempe, Arizona 85252; telephone (602) 994-6561.

1.2 OPERATING SYSTEM FEATURES AND FUNCTIONS

The layered design of VERSAdos provides a degree of system flexibility, while maintaining a straightforward structure that is easy to understand and use. The layered structure also provides the unique ability of combining the normally diverse functions of time-sharing software development with real-time system control or else tailoring an operating system to the user's requirements. The high degree of modularity inherent with the major programs of VERSAdos permits each user to add specific functions for his individual requirements with a minimum of time and effort. With the incorporation of Intelligent Peripheral Controllers to optimize input/output (I/O) functions, Central Processing Unit (CPU) overhead is significantly reduced, thereby providing more computing power for each system user and permitting connection of higher data rate I/O devices. Several safeguards are provided to protect the integrity of users and tasks from other users, undebugged programs, and/or online control systems.

VERSAdos also offers system generation capabilities to permit various operating system functions to be added or deleted from the operating system, as required, during tailoring for various hardware configurations or special applications. The operating system has been optimized for use with hard disk-type storage devices. These and other features are available to each user. Refer to the System Generation Facility User's Manual.

Real-time I/O processing capabilities are provided that allow directly connected or processing-generated interrupts to be serviced. In addition, multiprogramming of real-time tasks can be accommodated. All tasks can be scheduled on a priority basis. Inter-task communications are included to pass parameters and/or control between tasks and/or the operating system. A special control, the semaphore, is used to provide task synchronization and to coordinate the use of shared resources.

Operating system I/O operations are device-independent; that is, they refer to the logical properties of operation instead of the physical characteristics or file formats. The File Management System (FMS) provides transfer control between memory and logical devices or files. Contiguous, sequential, and indexed sequential files are supported. For further information, refer to the VERSAdos Data Management Services and Program Loader User's Manual.

A powerful set of utility programs are provided in VERSAdos. The utilities are briefly described here, and more fully described in the VERSAdos System Facilities Reference Manual.

Session control commands to VERSAdos allow multiple users (on a multi-user system) or a single user to operate in an interactive online (foreground) mode, as well as in the batch (background) mode. Online mode allows entry of commands from both the console and via chainfile processing. Batch mode processing is made possible through a circular job queue. These functions are also described in the VERSAdos System Facilities Reference Manual.

1.3 OPERATING SYSTEM ORGANIZATION

The operating system is divided into four major layers, with each layer further subdivided into other layers. The four major layers are the Real-Time Multitasking Executive (RMS68K) layer, the I/O layer, the File Management layer, and the Session Management layer. These layers are shown in Figure 1-1 and described in the following paragraphs.

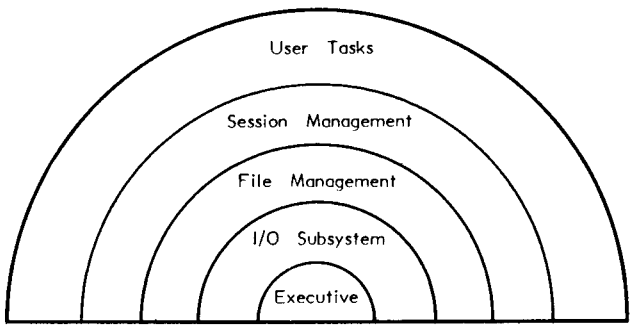


FIGURE 1-1. VERSAdos Structure

1.3.1 Real-Time Multitasking Executive Operation

RMS68K, the Executive, is the nucleus of the VERSAdos operating system and is available in either of two forms: as part of the complete operating system package or as a separate product -- RMS68K. RMS68K has responsibility for servicing all hardware and software generated interrupts and dispatching the interrupts to the proper tasks for processing. RMS68K also acts as the arbiter to resolve conflicts that result when competing tasks vie for processor time. Facilities that permit inter-task communications and task synchronization are also supported. RMS68K protects user applications while providing diagnostic feedback during error conditions.

1

RMS68K consists of an inner kernel (or nucleus) that supports the priority-driven, multitasking environment, and eight resource managers; each resource manager consists of data structures and about five to seven RMS68K directives, each providing a specific service from the resource manager. These resource managers are: Event Manager, Memory Manager, Task Manager, Time Manager, Semaphore Manager, Trap Server Manager, Exception Monitor Manager, and Exception Manager. (See Figure 1-2.)

RMS68K is structured in logical layers with each layer performing a particular range of functions. The layers may be viewed as internal, external, and channel management layers.

The functions provided by the internal layers are used by RMS68K to manage the processor, tasks, and physical devices such as timers. In addition, the internal layer performs work on behalf of requests from user tasks.

The functions provided by the external layer are directly available to user tasks through the use of directives. A directive (or request) contains all the information needed by RMS68K to perform the desired function.

The optional Channel Management Routines (CMRs) provide the channel-oriented physical I/O functions. All device drivers supplied with RMS68K and VERSAdos use the CMR mechanism for I/O.

Full details on the Executive are provided in the M68000 Family Real-Time Multitasking Software User's Manual, M68KRMS68K.

1.3.1.1 Task Structure. RMS68K and VERSAdos operations are task oriented. A task is a program, complete with its associated data area, that performs a functional unit of work. Application programs are performed as tasks and are executed according to their priorities, scheduling requirements, and availability of required resources.

Tasks fall into two general categories: user tasks and system tasks. Each task type can exist in two domains in the system, real-time or non-real-time. User tasks have the ability to control their own execution, resources, and interaction with other tasks. A user task may also, under certain conditions, control the execution and resources of another task. Through the use of Executive directives, a user task may perform the following functions:

Allocate/deallocate and otherwise manage memory
Communicate with other tasks
Perform inter-task coordination
Handle task related special events (such as faults)
Interface directly with physical I/O Channels

System tasks perform functions that provide services to user tasks. These tasks have access to all resources and may perform all operations without restriction. All tasks, whether user or system, are executed within the user mode of the MC68000, MC68010, or MC68020 microprocessor.

A real-time task is one that executes in a domain specifically designed for high performance; the real-time domain.

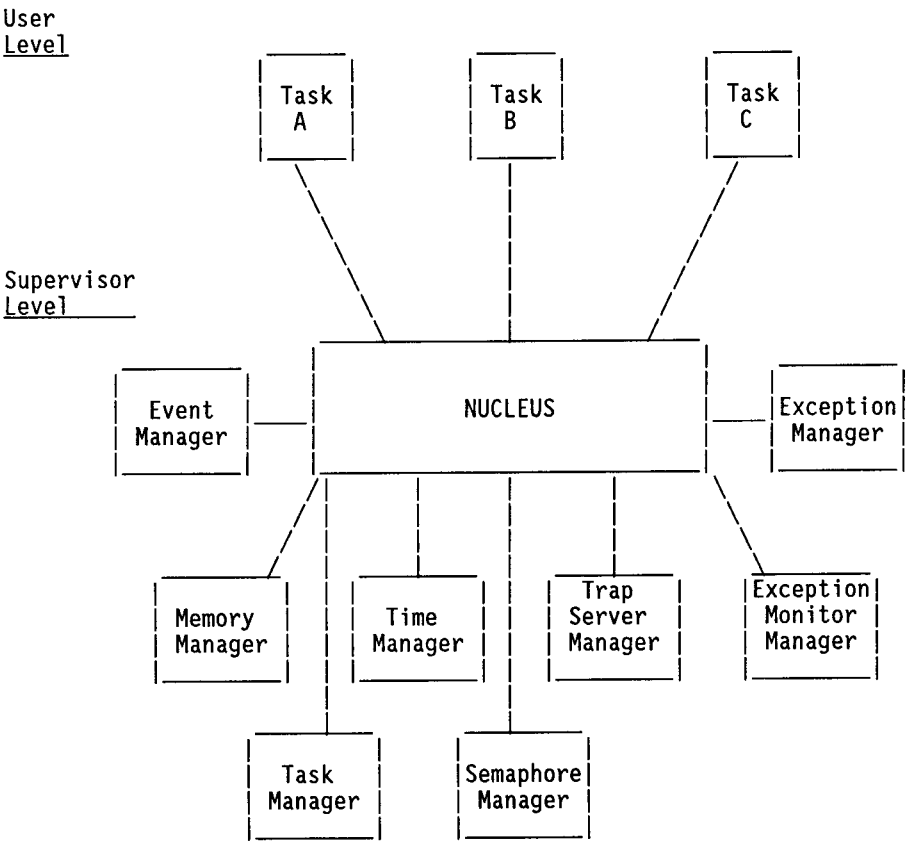


FIGURE 1-2. Real-Time Executive (RMS68K) Functional Partitioning

One constraint placed upon this domain is that the entire address space accessible by the task must be mapped, with the task's logical address identical to the system's physical address.

Another constraint is that the real-time task must use a newly defined 8-byte code, the task-ID, when referring to any other task within the system.

A non-real-time task is one that executes within the non-real-time domain.

This domain is not placed under the constraint that all logical task addresses be mapped to be equal to the corresponding physical system address.

Also, a task executing within the non-real-time domain must refer to any other task within the system via an 8-byte ASCII code representing the taskname and session number of the target task.

Both user and system tasks have names that identify the task, session numbers that group related tasks and protect the system from accidental or intentional tampering, and an internal 8-byte code generated by the Executive used as a task interface for a real-time task. User tasks can affect only those tasks having like session numbers, while system tasks can affect all tasks contained within the system.

Most RMS68K directives require specification of the taskname and the session number of the requesting task and/or target task. If these specifications are omitted, RMS68K will default to the taskname and session number of the requesting task.

The minimum task structure consists of a Task Control Block (TCB) and one program code segment. A task can have as many as four program code or data segments, each of which can be designated as a read only or read/write, plus one Asynchronous Service Queue (ASQ). The task's address space consists of the task's program code and data segments. A code segment may be restricted to a single task or may be shared with other tasks on a global basis (available to all tasks within the system) or on a local basis (available only to other tasks having the same session number). A shared segment may be marked as permanent to prevent deletion from the system when not being used by any task.

Program code segments contain instructions used during execution and are typically marked as read-only.

The main components of a program code segment are:

a. Main Code -- the basic element of a segment.

b. Trap Handling Code -- permits a task to respond to its own trap instructions.

c. Exception Handling Code -- permits a task to process its own trap exceptions.

d. Interrupt Service Routines (ISRs) -- handles external interrupts.

e. Asynchronous Service Routine (ASR) Code -- permits the passing of messages between tasks.

A task uses data segments for working storage, for passing bulk data to other tasks, and for sharing a common data area among two or more tasks. One data segment is usually allocated during task initiation, with additional segments allocated as required. However, both code and data may occupy the same segment if desired.

1.3.1.2 Task State Control Management. Directives are provided to control task execution by moving the affected task through various states. A task can control its own state transitions or those of another task. Functions performed by these directives include:

. Creating tasks
. Starting/stopping tasks
. Terminating/aborting tasks
. Wait/wakeup servicing
. Relinquishing execution control
. Setting priorities
. Requesting periodic activation
. Establishing logical connections
. Processing exception and trap instructions
. Delaying execution
. Changing ASQ/ASR status

Each task contains information about the task within a TCB. This TCB information permits RMS68K to maintain control of the task's execution, account for resource allocation to the task, and ensure task protection. The TCB remains associated with a task throughout the task's existence.

Task control is achieved by moving tasks through various task states. A task may make a transition from one state to another when any of the following actions occur:

a. The task issues a task control directive when in the running state.

b. An exception occurs while the task is in the running state.

c. RMS68K initiates special function handling (e.g., time-outs, semaphores, interrupts, requests of another task).

Since task execution is based on priority, with task priorities ranging from 0 to 255, a task receives an initial current priority and a limit priority when created. The current priority can be subsequently changed to any value less than or equal to the limit priority. The limit priority prevents one task from affecting another task having a greater priority.

RMS68K executes tasks from the READY state, starting with the task having the highest priority. If more than one task has the same priority, RMS68K selects the task that has been in the READY state the longest period of time. RMS68K enters the dispatch cycle to start execution whenever a task is removed from the RUN state. Some events that remove a task from the RUN state are:

a. A task relinquishes execution.

b. A task requests a delay.

c. A task requests service from another task.

d. A task chooses to wait for an external event.

e. A task executes a semaphore wait operation.

f. Task execution time exceeds the maximum permitted slice time (if this feature is installed).

g. A task is terminated.

Unless an ABORT or TERM (terminate) directive is responsible for removal from the RUN state, execution resumes immediately following the last instruction executed prior to removal. Executions initiated by a START directive begin at the task entry point.

Monitor and Sub-Tasks. A monitor task can be established that automatically receives notification at the termination of another task (referred to as a sub-task of the monitor task). A monitor task can monitor any number of sub-tasks without requiring that the monitor task have the same session number as its sub-task(s).

When a task is created or started, options specify which task, if any, is to be its monitor. The monitor can be assigned as the task requesting the creation or start, the requestor's monitor, or a third task. When a sub-task terminates, RMS68K places an event in the monitor task's ASQ, identifying the sub-task, the task initiating the termination, and a normal or abnormal termination indicator.

1.3.1.3 Message Passing Management. The ASQ holds events that are waiting for processing by a task. When a request is made by a task to pass a message to another task, or when a task needs to be notified of a particular occurrence in the system, an event is placed in a task's ASQ.

Task event control directives are provided to queue an event, to read an event, and to return to the point of interruption upon completion of ASR processing.

Task operation is affected by the occurrence of internally or externally generated events. When such an event occurs, RMS68K dispatches an event message to the task responsible for handling the event. An event can originate within RMS68K, within another task, or within the task currently executing. An event message contains the message length, an event code, and a message.

Each task capable of processing events must have an associated ASQ. The ASQ is a circular First In/First Out (FIFO) queue consisting of a queue control block, an event storage area, and an optional default receive buffer. RMS68K places an event into the ASQ when a message is sent to the task or when the task requires notification of an event occurrence within the system. Each incoming message to a task corresponds to one event entry in its ASQ. Entries are moved from the ASQ to the default receive buffer before the tasks are dispatched to the ASR, if the default receive buffer is enabled.

When a task wants to service an event, the task requests that the next event in its ASQ be placed into its event receiving area where the task can examine and process the event. A task can process events either synchronously or asynchronously.

Synchronous processing allows a task to inform the system when it is ready to process an event. If an event is not present, then the system puts the task into a wait state until an event arrives. When an event arrives the task is dispatched to the instruction after the Get-an-Event directive.

Asynchronous event processing consists of servicing events in an "interrupt-driven" manner using an ASR that is part of the task's program code and whose entry point is specified when the ASQ is allocated. Control is transferred to a task's ASR whenever the task's ASQ is not empty, the ASR is enabled, and the task is currently executing or has been dispatched because of an event occurrence. When an event "interrupt" occurs, the current system status is saved on the MC68000, MC68010, or MC68020 user stack. This permits control to be returned to the point of interruption upon completion of the ASR.

A task can also enable and disable the ASQ and ASR. RMS68K also disables the ASR upon entry of the ASR task to prevent another "interrupt" from occurring during the processing of the previous event. An ASR may also perform nested processing of interrupts by reenabling itself upon entry. Normally, the ASR would not reenable itself until process completion. When an ASQ is disabled, requests to queue events to the disabled ASQ are rejected. When an ASR is disabled, RMS68K will not enter it even though there are events in the associated ASQ. Although an ASR has one default entry point known to RMS68K, a task may request an alternate entry point if a special protocol has been previously established to define additional entry points.

1

1.3.1.4 Semaphore Management. Synchronization may be required to control activities or coordinate the use of shared resources. Tasks can accomplish synchronization and coordination by using a semaphore to indicate that a certain event has occurred. Task synchronization directives are used to control the semaphore, permit a task to create a semaphore, release all semaphores to which a task is attached, request (and wait, if necessary) for semaphore-controlled access or execution to be granted, and release a semaphore-controlled resource. VERSAdos supports three types of semaphores, with each applicable to a particular problem.

The Type 1 semaphore is used when several tasks require access to a single resource. The Type 2 semaphore controls execution of a sequence of tasks when a primary task must execute before dependent tasks. The Type 3 semaphore is used whenever one task controls a resource that other tasks desire to use.

1.3.1.5 Exceptions and Task Servers. Special function task control directives are also used to control server tasks and exception monitoring tasks. These directives permit a task to establish itself as a server task, acknowledge receipt or completion of a request, and initiate orderly shutdown of services. Several types of special function tasks are described in the following paragraphs.

Exception Monitoring. An exception monitor task may be implemented to provide execution control over a target task and to provide emergency processing when a task fails. Exception events that cause a target task to stop execution and cause the target task's exception monitor to be notified are specified in an exception monitor mask associated with the target task. An exception monitor task can also control the operation of a target task in a trace mode of operation. Directives are available to permit a task to become associated with an exception monitor task, to detach from an exception monitor task, and to specify events of interest to an exception monitor task. In a similar manner, directives permit an exception monitor task to control the current state of one of its target tasks.

Server Task Control. A server task is capable of receiving and processing requests from any task in the system. A trap instruction is executed by a user task to issue a request to a server task for processing. Thus, the server task appears to function as part of RMS68K.

Server tasks must have an ASQ to receive requests. Requests may then be processed either synchronously or asynchronously. If the request is processed asynchronously, the server must then include a processing start address. The user request appears as an event in the server task's ASQ.

A server may choose to serve any combination of trap instructions, and may also choose to receive an event in the ASQ when any task in the system terminates. Local processing of any trap instruction by a task overrides the server task handling of that trap instruction.

1.3.1.6 Interrupt Service Management. One or more ISRs can be included in a task. They are initiated by an external interrupt and execute at the priority level of the interrupt itself in the M68000 family microprocessor user hardware state. The ISR is useful, therefore, in creating I/O device drivers. ISRs can be configured by task directives to respond to a particular exception, to simulate an exception, and to return from exception execution.

1.3.1.7 Memory Allocation and Management Directives. Memory allocation and management directives are provided to allocate/deallocate segments, to share segments, to transfer segments, to allocate/deallocate an ASQ, and to move data from one task to another.

1.3.1.8 Physical Input/Output. CMR routines reside as part of the Executive, RMS68K. CMR logically manages channels and provides the link between memory mapped I/O space, interrupt vectors, and device drivers. CMR also provides the link between requestor commands and command service routines.

A channel is defined as a single contiguous portion of memory mapped I/O space associated with one or more polled identity conditions of interrupt.

A channel has a corresponding hardware interrupt vector number, a hardware priority level, and a software priority number. These three items are used to link devices into polling chains, which are used by a polling routine to determine which channel is associated with an incoming interrupt. When a channel is allocated, the CMR handler creates for that channel a Channel Control Block (CCB) that contains all of the information needed by the CMR handler to manage that channel. The CCB is then placed into the appropriate polling chain. There is a polling chain for every external interrupt vector. The CCBs are chained according to the software priority number -- those with higher software priority numbers are nearer to the head of the chain and thus serviced more rapidly when an interrupt occurs. See Figure 1-3 for a representation of polling chains.

When an interrupt occurs, control is passed through the first CCB (via a JSR) to the CMR interrupt handler routine. The CMR handler performs a minimum state save, resolves the CCB address, and calls the appropriate I/O handler. If the I/O handler returns without claiming the interrupt, CMR will call the handler of the next CCB chained for that vector. This continues until the chain is exhausted.

A code which indicates channel type is defined for a channel when that channel is allocated. Codes in the range of $10-$7F are reserved for standard VERSAdos channels. Values in the range of $01-$0F indicate a non-standard channel. Values in the range of $80-$8F indicate shared channels that can initiate I/O by more than one task. The value $FF indicates an interrupt only channel. Currently, no distinction is made by the CMR handler as to the particular value within the ranges.

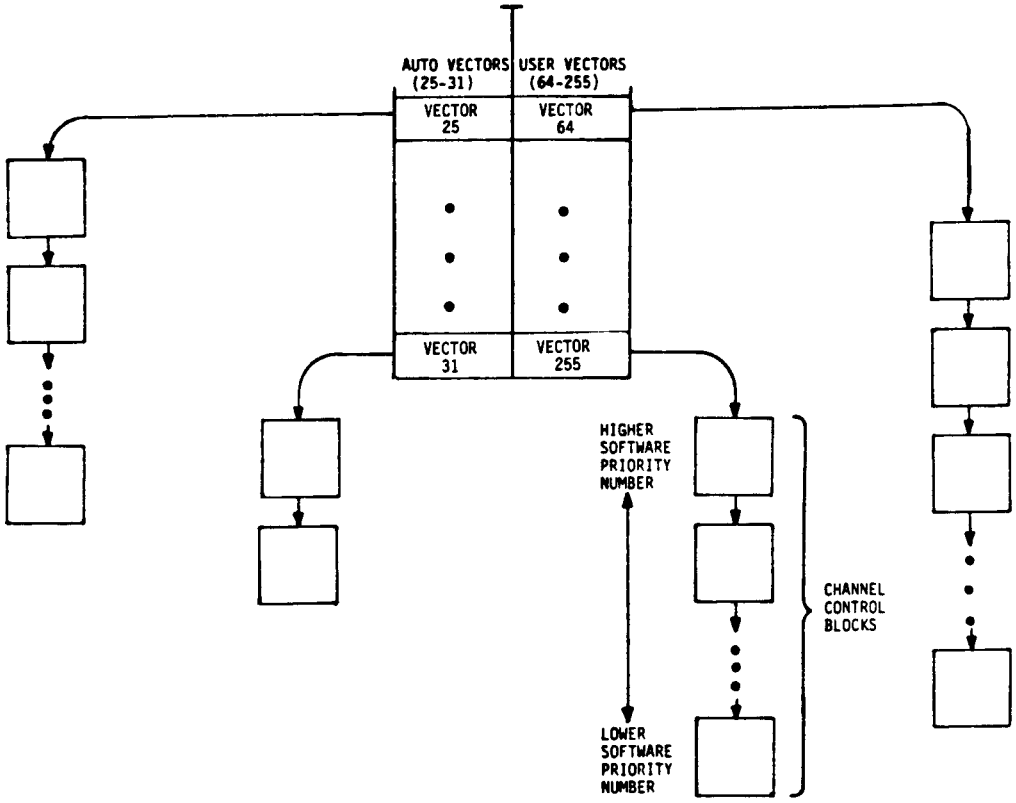


FIGURE 1-3. Representation of Vector Chains

1.3.2 I/O Subsystem

I/O operations within VERSAdos are essentially device-independent; i.e., they are based on logical properties rather than on device characteristics or file formats. Logical Unit Numbers (LUNs) are assigned to devices and files before I/O between programs and files/devices can occur.

In VERSAdos, all devices and files are treated as files. I/O is handled by two modules, Input/Output Services (IOS) and File Handling Services (FHS). IOS handles all data transfers, referring to task or user identification and LUN. FHS is called into service when creating disk files and their attributes, associating a LUN with each device or file.

Refer to the VERSAdos Data Management Services and Program Loader User's Manual for further information.

1.3.3 File Management System

The FMS Module handles contiguous, sequential, and indexed sequential files and respects the read/write access codes assigned to the files (refer to paragraph 1.2.4).

Contiguous files are fixed length and consist of physically contiguous records. Access may be random or sequential.

Sequential files and indexed sequential files may be fixed or variable-length groups of records. Access may be random or sequential (next record, current record, or prior record). Additionally, indexed sequential records may be accessed by key values. Files typically will not be physically contiguous. Disk space is allocated dynamically to contiguous blocks of records, called sectors, and various sectors are treated logically as segments. Each sequential or indexed sequential file has a File Access Block (FAB) describing the segments of the file for access by FMS.

VERSAdos also provides a program loader function, which creates new tasks, allocates memory segments, and reads the contents of each segment from the file into the allocated space.

Refer to the VERSAdos Data Management Services and Program Loader User's Manual for further information.

1.3.4 User Session Management

VERSAdos users identify themselves as valid users by logging onto the system with their user number, assigned by an individual referred to as the system administrator (user 0). The administrator has control responsibilities as well as certain privileges. The initial logon to the system is session 0001 and at this time defaults are established. Each subsequent logon by any user is assigned a sequential session number.

The administrator and other users have three levels of security capability.

User sessions may operate in the interactive (keyboard or chainfile command entry) or batch mode. Batch mode processing and "spooler" activity are under the overall control of the system administrator.

Refer to the M68000 Family VERSAdos System Facilities Reference Manual for additional information.

1

1.4 SYSTEM GENERATION (SYSGEN)

A System Generation (SYSGEN) capability is provided that permits users to produce a version of the operating system customized to specific needs. Relocatable and loadable modules of the user's choice can be processed using the SYSGEN command to produce an operating system file that reflects the user's system requirements. Some of the attributes of the operating system file that can be changed are:

Type and number of I/O devices

Number of users

Number of logical units per user

Amount of memory space allocated for the global segment table, trace table, and device connection queue

Number of files

A system which includes the RMS68K kernel, user applications, and that portion of VERSAdos functionality which allows device assignment and access can be put into Read Only Memory (ROM) using SYSGEN.

To perform a SYSGEN, the system must contain 512Kb RAM for systems based on VMEmodules or the VM04, or 384Kb RAM for all other systems.

VERSAdos includes chainfiles to perform the SYSGEN, tailored to specific systems. For further information on SYSGEN, refer to the System Generation Facility User's Manual.

1.5 TABLE-DRIVEN TASK INITIATOR

Included in VERSAdos are two utilities, TTGEN and TDTIGEN1, and chainfiles which can be used to create a Table-Driven Task Initiator (TDTI) system. A TDTI system is an operating system which includes some user-written tasks and some special code which is used to start the other tasks in the system in a controlled way. A TDTI system provides the following capabilities: Gives the user complete control over startup processing in the system; allows the user to make modifications to the system without having to re-SYSGEN every time; allows the user to define code to handle errors during startup processing; and simplifies the task of putting the end product into ROM. RMS68K, TDTI, VERSAdos I/O subsystem, drivers, and most VERSAdos components are directly ROMable. The VERSAdos file management subsystem, session control, and the loader are not.

CHAPTER 2

SYSTEM SOFTWARE

2.1 GENERAL

To complete the VERSAdos overview, the following paragraphs briefly describe the various development system software that is available. Each of these software packages, in addition to being compatible with VERSAdos, has been designed to help users in rapidly producing efficient programs for their 32-bit, 16-bit, and 8-bit microprocessor-based designs.

2.2 CRT TEXT EDITOR

The function of any editor is to facilitate entry, modification, or deletion of ASCII text files. This paragraph provides insight into the capabilities of the CRT Text Editor (E) and the minimum system configuration needed to support its operation. Further details concerning the use of this editor can be found in the M68000 Family CRT Text Editor User's Manual.

Two different modes of operation have been provided. The CRT mode has been designed to take full advantage of the functionality of the EXORterm 155 and VME/10 consoles, while the line mode is intended for use with any other type of TTY-compatible terminal. The line mode is also used for program editing under direction of an executing chainfile.

Three different levels of control are provided for source program entry and editing: page, command, and insert. The page and command levels are used with the CRT mode of operation, while the command and insert levels are used in the line mode.

Source program files produced with the editor and ASCII listing files produced by the various development system assemblers and compilers are in indexed sequential ASCII or sequential ASCII form. Both of these file forms can be edited by the CRT Text Editor. Files in sequential format are placed in a temporary scratch file for editing and saved only when the QUIT command is executed. However, indexed sequential files are edited directly, making it a much faster editing method. Future editing time can often be saved by choosing the indexed sequential format when producing new files and by reformatting old files to this format.

2.2.1 Minimum System Configuration for CRT Text Editor

The minimum configuration required for using the text editor is:

- One of the systems listed on page 1
- CRT/keyboard terminal
- EXORdisk III, LARK, CMD, or Winchester drives
- 256Kb RAM

2.2.2 CRT Text Editor Commands

This paragraph briefly describes each of the CRT Text Editor commands.

ADUP — Copies records from the file and places them in the XTRACT buffer, appending them to any records already in the buffer. The records also remain in the file.

AMOVE — Moves records from the file and places them in the XTRACT buffer, appending them to any records already in the buffer. The records moved are removed from the file.

CHANGE — Instructs the editor to search a portion of or an entire file for a character string and change it to the desired form.

COLM — Displays a ruler of column spacings (line mode only).

DELETE — Used to remove one or more lines (records) or portions of lines from the file. Specifying a vertical range (number of lines) permits a group of consecutive lines to be deleted from the file. If a horizontal range is specified in conjunction with the vertical range, the specified consecutive characters (rather than the entire line) can be deleted from each line within the vertical range.

DOWN — Moves the record pointer downwards.

DTAB — Deletes tab stops (line mode only).

DUPLICATE — Used to copy a line or group of consecutive lines from the file into the XTRACT buffer for later retrieval using the XTRACT command. Each use of the DUPLICATE command overwrites previous data in the XTRACT buffer.

EXTEND — Used to append data to the end of a line or group of consecutive lines within the file.

FIND — Used to position the editor file pointer to the start of the line in which the specified character, character string, or line number can be found. This command is very helpful in finding the exact location within the file to begin the edit session.

INSERT — Used from the command level (line mode only) to insert a line between two existing consecutive lines and position the editor file pointer at the start of the inserted line. Data may then be entered starting at this position. May also be used to insert at the top of a file and append at the end of a file.

LINE — Used to provide the line number corresponding to the current editor file pointer position.

LIST — Enables users operating in the line mode to list all or a specified portion of a file for display and inspection.

MERGE — Used to retrieve all or a consecutive number of lines from a specified file and insert them in the file being edited starting at the current position of the editor file pointer.

MOVE — Used in conjunction with the XTRACT command to relocate a line or group of consecutive lines within the edit file. An XTRACT buffer is established into which the specified line(s) are transferred for moving and later retrieval. The moved lines are then deleted from the file. Each use of the MOVE command overwrites previous data contained within the XTRACT buffer. The editor file pointer is then repositioned to the line following the last line deleted by the move operation .

PRINT — Used from the command line to list on the line printer any line or consecutive lines within the file being edited. The PRINT command does not alter the data being displayed on the screen.

QUIT — Used to end the edit session and return to the VERSAdos level. This command causes the file being edited to be closed in its then current state. Conversely, if the A option is specified on the command line for a sequential file, the original unedited contents of the file will be saved.

RANGE — Used to change the default vertical and horizontal range values of the FIND, CHANGE, PRINT, and SAVE commands.

SAVE — Used to create a new file having the name specified on the command line into which all or specified portions of the file being edited can be stored. This command proves very useful when editing to ensure minimal data loss in the event of system failure.

STAB — Specifies tab stops (line mode only).

TAB — Used to set or change the tab stops. This command permits up to 20 (maximum) tab stops to be set. These tab stops can be specified on a single command line or through multiple entries of the TAB command. Stops are specified by their character position within the line, separated by commas. By executing a TAB command having no parameters, tabs can be reset to the default format of every 10 characters or to the format corresponding to the option specified on the editor call command line. Entered tab values are sorted and saved in ascending order. Execution of the TAB command does not alter the screen display.

UP — Moves the record pointer upwards.

VERIFY — Permits users operating in the line mode to inspect changes to a file as they are being made. Following execution of this command, the editor outputs to the terminal any line affected by execution of subsequent commands. The default condition of the VERIFY command is off.

XTRACT Used to move data previously stored in the XTRACT buffer to the file being edited starting at the point indicated by the editor file pointer. The XTRACT command is used in conjunction with the MOVE and DUPLICATE commands to relocate lines of the file. An option is provided to permit the XTRACT buffer to be deallocated, thus regaining full use of this memory space.

2.3 TERMINAL-INDEPENDENT EDITOR

In version 4.5 and subsequent, VERSAdos furnishes a second editor program (TIE) that is terminal-independent. It has been designed to allow CRT mode editing on virtually any non-Motorola terminals. Simple modifications must first be made to one or more of the configuration files furnished, using the CRT Text Editor (E), and then TIE can be used with full functionality for CRT mode page and command level editing.

2.3.1 Page Editing

Page or display level allows editing directly on the screen by using the cursor positioning keys, labeled keys, and function keys that are standard on most keyboards. CTRL/key sequences can be used in place of function keys.

2.3.2 Command Editing

Command editing makes use of certain function keys or CTRL/key sequences that allow or require arguments. Commands supported are:

CHANGE	DUPE	MERGE
DELETE	FIND	MOVE
DTABS	JUMP	RANGE

The functionality of these commands is similar to that of the same commands listed in paragraph 2.2.1 for the E editor.

For further information on use of the TIE editor, refer to the VERSAdos Terminal-Independent Editor (TIE) User's Manual, M68KTIE.

2.4 STRUCTURED MACRO ASSEMBLER

The M68000 family macro assembler translates source programs written in M68000/M68010/M68020 assembly language into machine language capable of being executed by the MC68000, MC68010 or MC68020 microprocessor. The macro assembler also supports the MC68881 floating-point co-processor. Assembly language is a symbolic language consisting of a collection of mnemonics and symbols representing machine-instruction operation codes (opcodes) and directives (pseudo-ops, including macros), symbolic names (labels), operators, and special symbols. Directives are used to specify auxiliary actions to be performed by the macro assembler. This paragraph describes the macro assembler. For a more detailed description, refer to the M68000 Family Resident Structured Assembler Reference Manual, M68KMASM.

The macro assembler produces relocatable object code (code that is not affixed to a specific location in memory at the time of assembly), assigns storage locations to instructions and data, and performs auxiliary assembler actions designated by the programmer. The relocatable object modules produced by the assembler are compatible with the M68000 Family Linkage Editor which is used to link various relocatable object modules together to form a single loadable module that has absolute memory addresses assigned to the instructions and data. The assembler includes macro and conditional assembly capabilities and implements certain "structured" programming control constructs.

Assembly is a 2-pass process. During the first pass, the assembler develops a symbol table, associating user-defined labels with values and addresses. During the second pass, the translation from source language to machine language is performed, using the symbol table developed during pass 1. In pass 2, as each source line is processed in turn, the assembler generates appropriate object code and the assembly listing (if the list option is specified on the command line).

2.4.1 Minimum System Configuration for Macro Assembler

The minimum configuration required for the macro assembler is:

- One of the systems listed on page 1
- CRT/keyboard terminal
- EXORdisk III, LARK, CMD, or Winchester drives
- 384Kb RAM
- VERSAdos

2.4.2 Assembler Directives

The following list provides a brief description of the various assembler directives.

COMLINE — Identical to the DS.B (define storage in bytes) directive, except that it is passed on to the linkage editor as the location of the command line.

DC — Define a constant in memory. The DC directive may have one operand or multiple operands, each separated by a comma. The operand field may contain the actual value in decimal, hexadecimal, or ASCII. Alternatively, the operand may be a symbol or expression which can be assigned a numeric value by the assembler.

DCB — Instruct the assembler to allocate a block of bytes, words, or longwords, depending upon the size code specified.

DS — Reserve memory locations. The contents of the memory reserved is not initialized.

END — Indicate to the assembler that the source file is finished. Subsequent source statements are ignored. The END directive encountered at the end of the first pass through the source program causes the assembler to start the second pass.

EQU — Assign the value of the expression in the operand field to the symbol in the label field. The label and operand fields are both required and the label cannot be defined anywhere else in the program.

FAIL — Issue a warning or error message which identifies a programmer-generated error.

FEQU — Assign permanent floating-point value (MC68881 only).

FOPT — Assign floating-point options (MC68881 only).

FORMAT — Format the source listing, including column alignment and structured syntax indentation. This directive is selected by default.

IDNT — Specify an identification record to be passed to the linkage editor.

INCLUDE — Allow additional files to be included in the source input stream.

LIST — Print the assembly listing on the output device. This option is selected by default with the source text following the LIST directive being printed until an END or NOLIST directive is encountered.

LLEN	Specify the line length to be output to the printing device.
MASK2	Indicate that the source program is to be assembled to run on the MASK2 (R9M) chip. Specifying MASK2 causes the following changes in assembler processing: DCNT instruction replaces DBcc; STOP does not take an operand; and Bit operations are adjusted to the R9M format.
NOFORMAT	Do not format the source listing. The source listing will have the same format as the source input file.
NOLIST	Suppress the printing of the assembly listing until a LIST directive is encountered.
NOOBJ	Suppress the generation of object code.
NOPAGE	Suppress paging to the output device.
OFFSET	Define a table of offsets via the Define Storage (DS) directive without passing these storage definitions to the linkage editor (in effect, creating a dummy section). Symbols defined in an OFFSET table are maintained internally, but no code-producing instructions or directives may appear. SET, EQU, REG, XDEF, and XREF are allowed.
OPT	Indicate the assembler output options desired. The options are specified following the OPT directive.
ORG	Change the program counter to the value specified by the expression in the operand field. Subsequent statements are assigned absolute memory locations starting with the new program counter value.
PAGE	Advance printer paper to the top of the next page.
REG	Assign a value to a symbol in the label field that can be translated into the register list mask format used in the MOVEM instruction. The label cannot be redefined anywhere else in the program.
SECTION	Restore the program counter to the address following the last location allocated in the indicated section (or to zero if used for the first time).
SET	Assign the value of the expression in the operand field to the symbol in the label field. Although the EQU and SET directives appear to be the same, unlike the EQU directive, the SET directive permits the symbol in the label field to be redefined by other SET directives in the program. The label and operand fields are both required.
SPC	Output the specified number of blank lines on the assembly listing (if none are specified, make one blank line).

TTL — Print the title specified at the top of each of the following pages. A title consists of up to 60 characters. The same title will appear at the top of all successive pages until another TTL directive is encountered or until the end of the source file is reached.

XDEF — Specify symbols in the current module which are to be passed on to the linkage editor so that they may be referenced by other linked modules.

XREF — Specify symbols in the current module which are defined in other modules for identification by the linkage editor.

2.5 LINKAGE EDITOR

Programs written for MC68000-, MC68010-, or MC68020-based systems are processed by the various compilers or the assembler to create a relocatable object module. This object module (or group of object modules) is then relocated (and combined if more than one object module is specified) and linked to specific memory.

Instead of creating a load module, the linkage editor may optionally create a relocatable object module combining all of its input. Such a module may combine a group of interrelated object modules that have been completely debugged. Combining them all into a single module makes it easier for the user, because the new module can thereafter be referenced by a single file name. The linkage editor's task is also made easier and faster because any references between modules are resolved when the modules are combined.

A third type of linker output may optionally be selected. This is referred to as an "S-record" module, and is formatted to facilitate transferring files between computer systems.

The linkage editor requires two passes in order to create an output module. During the first pass, it builds a table of externally defined symbols and determines which sections are assigned (names, lengths, and starting addresses). It also determines which modules from the library (if any) are required. No attention is paid to the actual instructions and data in the relocatable object module during pass one.

Following pass one, if an S-record module or an absolute load module is being generated, the linkage editor assigns each section (e.g., program, data) to an absolute memory address. This address is the actual address at which the section will be loaded when the absolute load module is executed. This allocation of memory is a highly complex task that can be left totally up to the linkage editor or altered by various user commands.

If a relocatable object module is being produced, the linkage editor computes the total size of each section in use, opens the output file, and outputs the necessary information about each section and global symbol.

The linkage editor then proceeds to pass two, where the relocatable object modules read during pass one are reread in the same order. This time, however, the instructions and data in each module are relocated, linked (if necessary), and then written to the output file. If a relocatable object module is being produced, the input is not relocated, but any references between input modules are resolved (linked).

At the completion of pass two, the linkage editor outputs its final listings, the contents of which are determined by the option(s) specified on the invoking command line.

2.5.1 Minimum System Configuration for Linkage Editor

The minimum configuration required for the linkage editor is:

- One of the systems listed on page 1
- CRT/keyboard terminal
- EXORdisk III, LARK, CMD, or Winchester drives
- 384Kb RAM
- VERSAdos

2.6 PASCAL COMPILER

Pascal, first developed as a teaching tool, has gained wide acceptance as an applications and system programming language. Its structured nature and ease of maintenance have made it a favorable language in saving time and effort for users in program development/support. In recognition of this acceptance, Motorola has adopted Pascal as the high-level programming language for VERSAdos. Pascal is not furnished with VERSAdos but is available separately.

Motorola's Pascal is based on the language as defined in 1968 by Niklaus Wirth at the Eidgenossiche Technisch Hochschule in Zurich, Switzerland, with additions stimulated by Motorola's participation in the University of California at San Diego workshop and in the IEEE/ANSI standardization effort.

Pascal includes extensions for expressing certain embedded-control-type operations, an important consideration to a large class of microprocessor users. Other extensions are desirable to users who will implement business-oriented systems. Some of these extensions are:

Address specification for variables
Alphanumeric labels
Exit statement
External procedure and function declarations
1-, 2-, and 4-byte integers

This release also offers large structures, fast floating-point, and code optimization.

The Pascal compiler consists of three phases. Phase 1 processes a source program and produces a source listing and error messages as well as an intermediate code file. This intermediate code may optionally be input to Phase 1.5 to optimize the code, resulting in a possible reduction in the size of the code generated and increased execution speed. Either the output of Phase 1 or the optimized output of Phase 1.5 may then be input to Phase 2 to create a relocatable object file and its associated listing. The object file is combined with needed routines from the system library by the linkage editor, producing a load module that is ready to execute. Assembly language subroutines may also be linked into the load module. VERSAdos includes a resident run-time library of useful routines which may also be linked with Pascal programs.

For a more detailed description of Motorola's Pascal, refer to the M68000 Family Resident Pascal User's Manual, M68KPASC.

2.6.1 Minimum System Configuration for Pascal Compiler

- One of the systems listed on page 1
- CRT/keyboard terminal
- EXORdisk III, LARK, CMD, or Winchester drives
- 384Kb RAM
- VERSAdos

2.7 FORTRAN COMPILER

The FORTRAN compiler translates source programs into M68000 family machine language (object programs). Source programs are written using the CRT Text Editor or TIE editor. The object programs are relocatable, using the linkage editor to assign memory locations to one or a group of object programs and various routines contained in the FORTRAN library, and create executable load modules. Assembly language subroutines may also be linked into the load module. FORTRAN is not furnished with VERSAdos but is available separately.

Motorola's FORTRAN conforms to the ANSI FORTRAN 77 subset. Support for double precision variables and bit operations is also provided.

For further information, refer to the M68000 Resident FORTRAN Compiler User's Manual, M68KFORTRN.

2.7.1 Minimum System Configuration for FORTRAN Compiler

- One of the systems listed on page 1
- CRT/keyboard terminal
- EXORdisk III, LARK, CMD, or Winchester drives
- 384Kb RAM
- VERSAdos

2.8 DEbug

DEbug is a VERSAdos-resident monitor program used to debug other programs whose source code is written in assembly language for execution on the M68000/M68010. The language processor and the linkage editor supply information to the DEbug monitor.

DEbug allows the user to examine, insert, and modify program elements such as instructions, numeric values, and coded data.

Execution can be controlled by DEbug via the insertion of breakpoints into a program.

DEbug uses an extensive set of primitive commands for manipulation and examination of foreground tasks. A set of task-level commands may be used on foreground or background tasks and are applicable to both the single and multitasking modes of operation.

For complete details of DEbug functions, refer to the SYMbug/A and DEbug Monitors Reference Manual, M68KSYMBG.

2.8.1 Minimum System Configuration for DEbug

- One of the systems listed in page 1
- CRT/keyboard terminal
- EXORdisk III, LARK, CMD, or Winchester drives
- VERSAdos

2.9 SYMbug/A

SYMbug/A, referred to here as SYMbug, is a VERSAdos-resident multitasking utility that allows a user to debug application program(s) in terms close to the actual program itself. Unlike other debuggers that allow only absolute memory accesses, SYMbug generates information about the program that is available to the user during debug. Information is accumulated concerning assembler symbol names, module names, and section numbers. SYMbug automatically evaluates this type of symbolic information to absolute addresses. Now it is no longer necessary to reference a current link map to debug a program. Instead, knowledge of module names and symbols is sufficient to calculate relative offsets and debug the program by reference to an assembler listing. Without the overhead of user-responsible address resolution, the task of debugging a program becomes faster and easier with a reduction in the chance for error.

SYMbug is built around a multitasking kernel. It interfaces with the VERSAdos operating system to provide complete debug control to the user. User interface is via a powerful set of "primitive" commands. These commands allow the user to:

2

a. Examine/modify registers and absolute and program relative memory addresses specified in a number of ways:

   . Directly
   . In an expression
   . As an effective address
   . Symbolically
     (also allows control of display/modification formats)

b. Control program execution by allowing the user to:

   . Insert breakpoints into the program
   . Trace program execution
   . Monitor data changes

c. Direct multitasking functions by allowing the user to:

   . Modify task scheduling/information handling
   . Modify task attributes/status

d. Expand debugger functions through user generation of:

   . User "macros" built of a series of primitive commands
   . In line command/command block repeat functions
   . Default input/output format modifications

e. Access information outside of SYMbug so that the user may save and restore previously defined information:

   . Save and load program(s) to and from disk
   . Save and load symbolic information (macro names/local symbols) to and from disk
   . Generate debug session echo to printer

SYMbug is a self-documenting debugger. Errors are informative and precise and the user may also utilize the SYMbug HELP command to display a brief command syntax summary for all commands. This relieves the user of the trouble of scanning a reference manual for SYMbug information.

2.9.1 Minimum System Configuration for SYMbug

. One of the systems listed on page 1
. CRT/keyboard terminal
. EXORdisk III, LARK, CMD, or Winchester drives
. 256Kb RAM
. VERSAdos

2.9.2 SYMbug Primitive Command List

The SYMbug commands are separated into the following five groups:

Group 1: Execution

- Address Stop
- BReakpoint
- GO (Execute)
- TRace

Group 2: Modify

- Block Fill
- Block Move
- Memory Modify
- Memory Set

Group 3: Display

- Block Search
- Define Constant
- Display Formatted Registers
- Memory Display

Group 4: Session Control

- Command Repeat
- DEfaults
- DUmp Memory
- File Read
- File Save
- HELP
- MAcro Define
- QUIT Session
- Symbol Define
- MAcro Edit

Group 5: Task Control

- ATTAch task
- DETAch task
- EVENt definition
- LOAD (task)
- MASK exception
- STARt task(s)
- STOP task(s)
- STATus definition
- TASK notify
- TERMinate task
- WAIT task

2.10 VERSAdos UTILITIES

VERSAdos utilities are listed alphabetically, with descriptions of their usage, in the following paragraphs. Refer to the M68000 Family VERSAdos System Facilities Reference Manual, M68KVSF, for further information on these utilities and on various session control task commands.

ACCT -- The Account utility may be used by the system administrator to open a password file and an accounting file, and then to monitor individual and collective usage of the system. This utility probably will not be needed when the computer is used as a single-user system.

BACKUP -- BACKUP provides two methods of transferring data from one disk to another: track-by-track mode and file-by-file transfer mode. Because track-by-track mode requires that the source and destination disks be of the same type, the file transfer mode will be used for the standard configuration of VME/10. EXORmacs users will find track-by-track mode to be faster. File transfer mode must be specified with BACKUP options A or R. Options U, V, or B select track-by-track mode. Sub-options allow several variations in copying. 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). All destination disks must have been initialized previously with the INIT 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.

CONFIG -- The Configuration utility is a menu-driven program that enables the user to reconfigure certain device parameters and attributes temporarily or permanently without the necessity of reSYSGENing the operating system. Types of devices configurable are terminals, magnetic tape drives, and printers. Parameters and attributes changed with CONFIG may be put into a chainfile to be executed automatically at logon.

CONNECT -- The CONNECT utility allows the user of a VME/10 or other VERSAdos system with appropriate configuration 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 move any cables. When the L=n option is specified, CONNECT performs the following functions on the terminal from which it was invoked before connecting the terminal to the other port:

a. Resets the display screen.

b. Sets up the virtual screen (the area which scrolls while in CONNECT mode) as lines 1 through 1-n.

c. Display the message indicating successful connection.

COPY -- The COPY utility copies a file onto the same volume under a new filename, or onto another volume under the same or a new name. Options 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. Output can be sent to a printer if part of the system, or to the display terminal for a quick look at the contents of a file.

CREF -- The Cross Reference utility searches through multiple files for occurrences of specified symbols and displays messages as to where the symbols are found. A cross-reference listing can be directed to an output file.

DEL -- The Delete utility removes a file name from a disk directory and frees all space allocated to that file. Options allow a list of files or a "family" of files with like parameters (e.g., same catalog or same extension) to be deleted with one command, and/or to direct a list of files deleted (normally displayed on the CRT) to an output file or to a printer.

DIR -- Each VERSAdos disk contains a Volume Identification Block (VID), established when the disk was initialized. Information describing the disk space allocation, location, and attributes of each file contained on the disk is stored in this directory. Part or all of the information entered for each file can be obtained by using the DIR utility. Options provide greater detail.

DMT -- This utility, used in conjunction with the MT utility, enables non-IPC-controller systems such as the VME/10 to handle disks of unlike formats. DMT performs the complementary function of the MT utility. It forces VERSAdos to release control of a mounted floppy disk and to reject input/output requests to a new disk until the MT command has been reissued. Before using DMT the floppy must be offline -- i.e., the floppy drive door must have been opened.

DISPATCH -- The use of this utility is privileged; i.e., only logon user 0 may use it. It is used in conjunction with BATCH job processing, to change dynamically the number of batch jobs that are able to execute.

DUMP -- DUMP is a utility that allows examination and/or modification of one or more sectors of disk data. The basic command provides a display of the contents of a disk, a file, or a portion of a file, in hexadecimal; alternatively, the dump may be directed to a printer or into another file. Specifying the interactive option allows certain sectors of the disk or file to be read into a change buffer in memory; bytes may be individually examined, changed, and read back to the disk to replace the original version.

2

DUMPANAL -- DUMPANAL is an interactive utility used to analyze the contents of a system crash dump, if the data has been saved in a file by means of the firmware-resident monitor's DB command. DUMPANAL lists various system tables and memory locations as they appear in the dump file.

EMFGEN -- This utility allows the user to add error messages and/or alter existing messages in the error message file, ERRORMSG.SY, which is used by VERSAdos' error message handler to issue most system messages.

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 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.

INIT -- All blank diskettes or cartridge disks for use with VERSAdos must be formatted and initialized with the INIT utility before their first use. Formatting establishes a sector/track pattern on the diskette which is compatible with the processor and VERSAdos. Initializing creates a VID on the diskette which can be recognized by VERSAdos. The VID includes a user-supplied volume I.D., description, and ownership. A disk file directory is also created by INIT. If directed to do so, INIT will check the disk for bad sectors; if any are found, INIT will write their locations into the Sector Lockout Table (SLT) so data cannot be written to them.

Used diskettes can also be initialized with INIT to clear the file directory. (Disks containing wanted files should not be initialized, as their directory entries will be altered so as to be unrecognizable by VERSAdos, and new data will overwrite their contents.) The formatting function need not be performed when initializing a used VERSAdos disk. (NOTE: Formatting destroys all data on a disk.)

Two options are permitted. One of these permits the user to add bad block entries to the SLT while preserving the contents of the disk. The other allows specification of the address of the bootstrap file. The furnished VERSAdos bootstrap file is named SYS:0.<catalog>.IPL.SY, where <catalog> identifies the system type. The required address varies according to system type; these addresses are listed in the M68000 Family VERSAdos System Facilities Reference Manual.

LIB -- The Library utility makes useful software routines available for use by more than one program or more than once in a program. These routines, or program modules, are created in assembly or high-level language; put into a file using the editor; assembled or compiled; and combined into a "library" file or files with the LIB utility. These user-created library files, along with those supplied with the system and with optional high-level languages, can then be linked and made accessible to application programs. LIB offers several interactive commands to aid in manipulation of the modules while creating library files.

LIST -- Using the LIST utility, all or part of an ASCII disk file can be displayed, written to a separate file, or (if a printer is part of the system) printed. Selectable options allow specification of beginning and/or ending lines; numbering of lines; prompt for wider or narrower line length and longer or shorter page length specification; prompt for heading; and interactive mode. In interactive mode, if the heading prompt option or nonstandard length and width prompt option were specified, these parameters can be supplied. Lines to be listed can also be specified while in interactive mode.

MBLM -- Object files which were assembled using the M68000 Family Cross Macro Assembler are in S-record format. These files cannot be linked into load modules, but can be transported to the system and then converted to loadable and executable files by means of the MBLM utility.

MERGEOS -- Allows merging new modules into an operating system without the necessity of re-SYSGENing.

MIGR -- ASCII programs filed on MDOS-format diskettes can be converted to VERSAdos format on 8-inch floppy diskettes with the MIGR utility. MDOS is the resident operating system for Motorola's EXORciser computer.

MT -- MT allows VERSAdos to access disks of differing media formats on a non-IPC-controller system such as the VME/10. It must be used before performing I/O operations to a floppy diskette (except for the first diskette accessed after power-up). In turn, the DMT utility must be used after the diskette has been taken offline, to release the device. If the diskette is of VERSAdos format (contains a VERSAdos VID), entering the MT command and the device designation is all that is required. If the diskette is of foreign format, however, it may be accessed after mounting when configuration data has been supplied by the user during MT's interactive dialog.

NOVALID -- If system security level 2 or 3 is in effect, and a user password file exists, NOVALID is used to delete specified user number records from the file.

PATCH -- Changes can be made to executable load module files with the PATCH utility. Interactive subcommands allow the display and change of portions of a file after it has been read into memory. This makes it possible to make changes to a program without having to change the source and reassemble it. PATCH includes a one-line disassembler and one-line assembler.

PRTDMP -- The Print Dump utility, PRTDMP, allows dumping part or all of memory to a file after an abort of a load module. The file or a portion of it can then be displayed or routed to a printer for examination. To use this utility, the load module must have been linked with the linker's D option. Interactive commands vary the type of output.

RENAME -- This utility is used to change the name of a file and/or its catalog name. The system administrator (logon user 0) may also change a file's user number. User 0 or the volume owner may change a file's protection key.

REPAIR -- REPAIR is an interactive utility used to repair the various logical structures of disks and files if they have become damaged. These structures include:

VID	Volume Identification Block
SAT	Sector Allocation Table
CFGA	Configuration Area (media format)
SDB	Secondary Directory Block (catalog list)
SDE	Secondary Directory Entry (catalog entry)
PDB	Primary Directory Block (filename list)
PDE	Primary Directory Entry (filename entry)
FAB	File Access Block (list of Data Blocks)
DB	Data Block (list of sequential records)
HDR	Header
SLT	Sector Lockout Table
DTA	Diagnostic Test Area

Empty structures can be deleted using REPAIR. Bad blocks can be handled via SLT and alternate sector handling.

REPAIR can be used to recover a deleted file, if the file's DB and FAB have not been reallocated.

SCRATCH -- This utility quickly erases the VID of a used diskette so that it can be reused. Only the disk's owner or logon user 0 can SCRATCH a disk. The disk also may be reformatted with SCRATCH. After using this utility, the disk must be reinitialized by INIT.

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.

SNAPSHOT -- The SNAPSHOT utility, implemented only on the VME/10 or EXORterm 155, copies the display on the CRT screen to a file or to a printer.

SPL/SPOOL -- VERSAdos offers a spooling capability whereby a particular volume can be designated as storage media for a queue of files awaiting time-consuming background tasks such as batch and chain processing and printing. This frees the system for foreground operations. The operating system must be SYSGENed to add a printer or an auxiliary storage device. SPL must be installed in session 0001. SPOOL may then be accessed whenever needed in subsequent sessions. SPOOL includes a list of subcommands for initiating, monitoring, and cancelling spooling functions.

SRCCOM -- The Source Compare utility is used to compare two ASCII text files and list any differences.

SYSANAL -- SYSANAL is an interactive operating system debugging utility. It provides a means of examining system tables in RMS68K, the nucleus of VERSAdos, and at any part of memory while VERSAdos is running. Output is to the display screen or to a printer if one is available.

TRANSFER -- The ASCII file transfer utility allows uploading or downloading of files such as source code or S-records between the VERSAdos system and another system. The systems may be connected directly between serial ports, or by phone lines/modems. Both systems must be configured for the same baud rate and character makeup. TRANSFER uses two associated Pascal programs, ULOAD and DLOAD.

UPLOADS -- UPLOADS is used to migrate S-records from some external source to a VERSAdos system. The S-records must be received through an MVME400 dual port serial module or a Multi-Channel Communications Module (MCCM) I/O Channel which is connected to the source system via a direct RS-232C hardware configuration.

VALID -- VALID is used to control access to the system by user number. User numbers and password records are entered into, maintained by, and deleted from an "account" file, accessible only by the system administrator.

2.11 SYSTEM TEST AND DIAGNOSTICS

The self-test diagnostics packages consist of firmware routines (stored in ROM) and disk-resident module diagnostic programs. Brief descriptions of the diagnostics packages for EXORmacs and VME/10 systems are given here; they are more fully described in the following manuals:

EXORmacs Development System Maintenance Manual, M68KEMM
VME/10 Microcomputer System Diagnostics Manual, M68KVSDM.

2

EXORmacs

The power-up/restart test performs a brief test which exercises the EXORmacs DEbug and MPU Modules, as well as polls the status of the Intelligent Peripheral Controller (IPC) self-test. This test is always performed at power-up and after a system reset occurs.

The system module test performs the same test as the power-up/restart test, plus it exercises the memory modules and also initiates an extensive self-test on each IPC. This test, which is initiated by pressing two pushbuttons on the chassis front panel, verifies the minimum system requirements for MACSbug (the firmware monitor) and VERSAdos, and ensures that all contiguous RAM has been initialized. The test time duration is 5 to 40 seconds, depending upon memory size.

The disk-resident module diagnostics represent a complete system test of the EXORmacs system. Each module is tested using a single program. These programs are independently loaded and executed from MACSbug. Some of the diagnostics require operator input. All routines display fault information to the user via the display console CRT and/or the chassis STATUS display. This fault information can be used to isolate a faulty module or to establish a confidence level that the system is operational.

For further information on EXORmacs operation, refer to the EXORmacs Development System Operations Manual, M68KMACS.

VME/10

The power-up/reset test should be the first test executed in the system diagnostic package as it will verify that the basic functions of the system are operational. The first level of the test checks the basic functionality of the system, and the second level simulates the multitasked, asynchronous environment required for the operating system. The power-up/reset test is performed automatically at power-up, and can also be initiated at system reset.

The disk-resident module diagnostics are a complete test package for the VME/10 Microcomputer System. Each module is tested by a single program, which is independently loaded and executed from TENbug, the firmware monitor.

For further information on VME/10 operation, refer to the VME/10 Microcomputer System Overview Manual, M68KVSOM.

2.12 DRIVER SOFTWARE

The following driver software is furnished with VERSAdos and may be SYSGENed into the system if the applicable hardware is part of the system configuration:

DRIVER SOFTWARE	SYSTEM/MODULE
ACIADRV	EXORmacs, MVME110
DRVLIB	All
EPCIDRV	VM01, MVME101
IPCDRV	VM20, VM21, VM30
M300DRV	MVME300
M315DRV	MVME315
M320DRV	MVME320
M420DRV	MVME420
M435DRV	MVME435
M600DRV	MVME600
M605DRV	MVME605
M610DRV	MVME610
M615DRV	MVME615
M625DRV	MVME625
MFPDRV	MVME120
MPCCDRV	MVME050
MPSCDRV	VM02, VM03, MVME400
P050DRV	MVME050
P117DRV	MVME117
PIADRV	EXORmacs, MVME101, MVME410
PV01DRV	VM01
RADDRV	RAD1
RIODRV	RI01
RWINDRV	WIN1
SIODRV	VM04, MVME130, MVME131
TERMDRV	VME/10
TERMLIB	All
VM22DRV	VM22
ZIODRV	MVME117

Further information may be obtained by referring to the following manuals:

Guide to Writing Device Drivers for VERSAdos, M68KDRVGD
RAD1 Device Driver Software User's Manual, M68KRADDRV
RI01 Device Driver Software User's Manual, M68KRIODRV
MVME300 (GPIB Controller with DMA) I/O Driver Reference Manual, MVME3SW
MVME435ADRV Magnetic Tape Driver User's Manual, MVME435DRV
MVME605 Analog Output Module Driver User's Manual, MVME605DRV
MVME610/620 AC/DC Input Module Driver User's Manual, MVME610DRV
MVME615/MVME616 Driver Software User's Manual, MVME615DRV
MVME625 Driver Software User's Manual, MVME625DRV

Source code for the drivers is also furnished with VERSAdos. If a particular driver is to be modified by the user, the source can be altered and reassembled/compiled.

THIS PAGE INTENTIONALLY LEFT BLANK.

INDEX

THIS PAGE INTENTIONALLY LEFT BLANK.

SUGGESTION/PROBLEM REPORT

MICROSYSTEMS

QUALITY • PEOPLE • PERFORMANCE

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 _____

City _____ State _____ Zip _____

For Additional Motorola Publications
Literature Distribution Center
616 West 24th Street
Tempe, AZ 85282
(602) 994-6561

Four Phase/Motorola Customer Support, Tempe Operations
(800) 528-1908
(602) 438-3100

MOTOROLA

MOTOROLA ***Semiconductor Products Inc.***

P.O. BOX 20912 • PHOENIX, ARIZONA 85036 • A SUBSIDIARY OF MOTOROLA INC.

19193 PRINTED IN USA (1/86) MESSENGER 3500