MOTOROLA

M68KSYSGEN/D8

System Generation Facility User's Manual

MICROSYSTEMS

QUALITY • PEOPLE • PERFORMANCE

MOTOROLA

M68KSYSGEN/D8

NOVEMBER 1985

SYSTEM GENERATION FACILITY USER'S MANUAL

Any addendums to previous revisions of this manual have been incorporated in this revision.

Eighth Edition

Copyright 1985 by Motorola Inc.

Seventh Edition March 1985

MICROSYSTEMS

REVISION RECORD

M68KSYSGEN/D7 -- March 1985. Reflects the following software levels: VERSAdos 4.4 and Link 1.8. Adds support of the MC68020, VM04, MVME120, MVME121, MVME122, and MVME123. New features: SYSGEN is now done under catalog name of system to be made using one generic SYSGEN command file, &.VERSADOS.CD. SYSGEN listing only lists drivers that were included. It is easier to configure system for drivers and to add independent drivers. All information is derived from system and driver configuration INCLUDE files. The SGSYMBL.LO is an optional pass two processor that generates a readable cross reference listing file (SYMBOLS.LS) of the parameters used during a SYSGEN.

M68KSYSGEN/D8 -- November 1985. Adds support of the MVME117, MVME130, and MVME131 VMEmodules. Makes minor corrections to the manual, incorporates boldface type in input/output examples, and adds a keyword index.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

GENERAL INFORMATION

1.1 INTRODUCTION TO SYSGEN

The VERSAdos System Generation Facility (**SYSGEN**) allows the user to custom-generate an operating system to suit a particular application. An operating system can be built around the RMS68K kernel, or the standard VERSAdos system furnished for a computer system can be customized. To generate a new operating system, a **SYSGEN** command file is edited by the user, using **SYSGEN** commands that establish parameters for the desired system configuration. A usable operating system boot file is created by performing the **SYSGEN** boot file process on the command file. To do a **SYSGEN**, the minimum memory requirement is 384Kb; however, 512Kb is required for EXORmacs and may be required for VMEmodule-based and VM04-based systems because of the additional disk controllers, drivers, and/or page sizes for Memory Management Unit (MMU) operation.

To enable **SYSGEN** to run in 384Kb of memory, Device Control Blocks/Channel Data Blocks are assembled individually and linker/merged dynamically as required. The "VERSAPT" patch files are also built dynamically as modules are used.

Each product or system type (refer to Table 1-1) has four **SYSGEN** support files furnished:

a. A copy file named <system>.COPYSGEN.CF

b. A switch file of boards for the system named <system>.CNFGDRVR.CI

c. A conditional **INCLUDE** file (based on the switch file) named <system>.IFDRVR.CI

d. A system dependent file named <system>.SYSTEM.CI

There are several generic files also furnished:

a.	&.VERSADOS.CD	Contains the **SYSGEN** commands and parameters and **INCLUDE** statements. Note that the ampersand (&) represents a null catalog.
b.	STD.SYSGEN.CF	Invokes the &.SYSGEN.CF file with the default arguments and creates all listing files.
c.	NOLIST.SYSGEN.CF	Invokes the &.SYSGEN.CF file with the default arguments but does not create the SYSASML.LS listing file.
d.	&.SYSGEN.CF	Executes the **SYSGEN** process with specified arguments (user defined or default).

TABLE 1-1. Command and Chain Filenames

PRODUCT TYPES	USER NUMBER	CATALOG <system>	SYSTEM INFORMATION	CONFIGURATION INFORMATION	CONDITIONAL
EXORmacs	9998	EXORMACS	SYSTEM.CI	CNFGDRVR.CI	IFDRVR.CI
VME/10	9998	VMES10	SYSTEM.CI	CNFGDRVR.CI	IFDRVR.CI
VM01	9998	VM01	SYSTEM.CI	CNFGDRVR.CI	IFDRVR.CI
VM02	9998	VM02	SYSTEM.CI	CNFGDRVR.CI	IFDRVR.CI
VM03	9998	VM03	SYSTEM.CI	CNFGDRVR.CI	IFDRVR.CI
VM04	9998	VM04	SYSTEM.CI	CNFGDRVR.CI	IFDRVR.CI
VMC 68/2	9998	VM02	SYSTEM.CI	CNFGDRVR.CI	IFDRVR.CI
MVME101	9998	VME101	SYSTEM.CI	CNFGDRVR.CI	IFDRVR.CI
MVME110	9998	VME110	SYSTEM.CI	CNFGDRVR.CI	IFDRVR.CI
MVME117	9998	VME117	SYSTEM.CI	CNFGDRVR.CI	IFDRVR.CI
MVME120/121	9998	VME120	SYSTEM.CI	CNFGDRVR.CI	IFDRVR.CI
MVME122/123	9998	VME122	SYSTEM.CI	CNFGDRVR.CI	IFDRVR.CI
MVME130	9998	VME130	SYSTEM.CI	CNFGDRVR.CI	IFDRVR.CI
MVME131	9998	VME131	SYSTEM.CI	CNFGDRVR.CI	IFDRVR.CI

```
NOTE:  All systems use command file                &.VERSADOS.CD.
       All systems use copy file          <system>.COPYSGEN.CF.
       All systems generate bootable file <system>.VERSADOS.SY.
```

If all parameters in the furnished VERSAdos (as listed in the appropriate <system>.CNFGDRVR.CI, <system>.SYSTEM.CI, and &.VERSADOS.CD files), agree with the user system configuration, the bootable <system>.VERSADOS.SY file may be used as is -- with no modification or **SYSGEN**. The appropriate configuration file (<system>.CNFGDRVR.CI) can be listed and examined to determine which boards/drivers are online in the corresponding standard bootable VERSADOS.SY file, and whether changes are necessary. The command, chain, and **INCLUDE** files for each target system are identified by their catalog name <system>. (Refer to Table 1-1.)

To perform a **SYSGEN**, run the <system>.COPYSGEN.CF chainfile to copy all the required files (including equate, macro, and UTILIB.RO), for that system into a user defined account. After the completion of COPYSGEN.CF, run the STD.SYSGEN.CF to invoke the **SYSGEN** process. **SYSGEN** must run under a catalog equal to the system type; e.g., to run an MVME120 **SYSGEN** in user account 9100 on hard disk volume SYS, enter the command:

```
USE SYS:9100.VME120
```

The file STD.SYSGEN.CF produces three listing files:

a. <system>.SYSLIST.LS	Lists the output of the **SYSGEN** process.
b. <system>.SYSASML.LS	Contains the listings from the assemblies and links that were done during the **SYSGEN** process.
c. <system>.SYMBOLS.LS	Contains a listing and cross-reference of all symbols and parameters with their values.

If the user does not want the SYSASML.LS file produced, run the file NOLIST.SYSGEN.CF. Using NOLIST.SYSGEN.CF reduces the time used by **SYSGEN**.

Chapter 2 presents the **SYSGEN** command syntax, both for the furnished chain and command files and for invoking **SYSGEN** directly on a user-written command file. Chapter 3 describes the **SYSGEN** command set. Chapter 4 describes the **SYSGEN** ROM capability.

Appendix A discusses control blocks and tasks. Appendix B describes hardware and software configuration. Appendix C contains listings of typical Motorola-furnished command files (for a VME/10 system both before and after **SYSGEN** is performed) and shows a typical **SYSGEN**-generated system map. Appendix D contains definitions of the **SYSGEN** parameters. Application **INCLUDE** file examples for both an assembly task and a Pascal task are shown in Appendix E. Appendix F contains an extract from a system **SYSGEN** listing.

1.2 DESCRIPTION

Some system attributes that can be tailored using the **SYSGEN** commands include:

a. Type and number of devices.

b. Number of users.

c. Number of logical units per user.

d. Amount of memory space for Global Segment Table (GST), Trace Table (TT), and Device Connection Queue (DCQ).

e. Number of files.

Since a file of commands is required as input, **SYSGEN** operation is similar to chain mode. However, all commands in the input file must be from the **SYSGEN** command set. The commands can reference source files, relocatable modules, and loadable modules that can contain tasks or code for execution in the supervisor mode. Adjusting the **SYSGEN** process to tailor the resulting file for a particular system configuration is done by modifying **SYSGEN** commands in the input command file. Source files may also require modification. Figure 1-1 shows a typical example for some **SYSGEN** commands including the main inputs and outputs.

SYSGEN handles module streams. A module is either a process or a task. A process includes code that runs in supervisor mode. A task includes code that runs in user mode and has Task Control Blocks (TCBs) and Task Segment Tables (TSTs), built by **SYSGEN**, associated with it. During the **SYSGEN** procedure, all process and task load module files are merged into a single output file suitable for boot loading.

SYSGEN is a text processor/substituter that controls the assembly and link of modules required for VERSAdos. Figure 1-2 shows the system flow of how **SYSGEN** controls these modules.

1.3 NOTATION

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

< >	Angle brackets enclose a "syntactic variable" that is to be replaced in a command line by one of a class of items it represents.
boldface strings	A boldface string is a literal, such as a command or a program name, and is to be typed just as it appears.
|	This symbol indicates that a choice is to be made. One of several items, separated by this symbol, should be selected.
[ ]	Square brackets enclose an item that is optional. The enclosed item may occur zero or one time.
[ ]...	Square brackets followed by periods enclose an item that is optional/repetitive. The item may appear zero or more times.

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

1.4 RELATED DOCUMENTATION

The following publications may provide additional helpful information. If not shipped with this product, they may be obtained from Motorola's Literature Distribution Center, 616 West 24th Street, Tempe, Az 85282; telephone (602) 994-6561.

DOCUMENT TITLE	MOTOROLA PUBLICATION NUMBER
M68000 Family CRT Text Editor User's Manual	M68KEDIT
M68000 Family Real-Time Multitasking Software User's Manual	M68KRMS68K
VERSAdos to VME Hardware and Software Configuration User's Manual	MVMEVDOS
VERSAdos Data Management Services and Program Loader User's Manual	RMS68KIO
M68000 Family VERSAdos System Facilities Reference Manual	M68KVSF
M68000 Family Resident Structured Assembler Reference Manual	M68KMASM

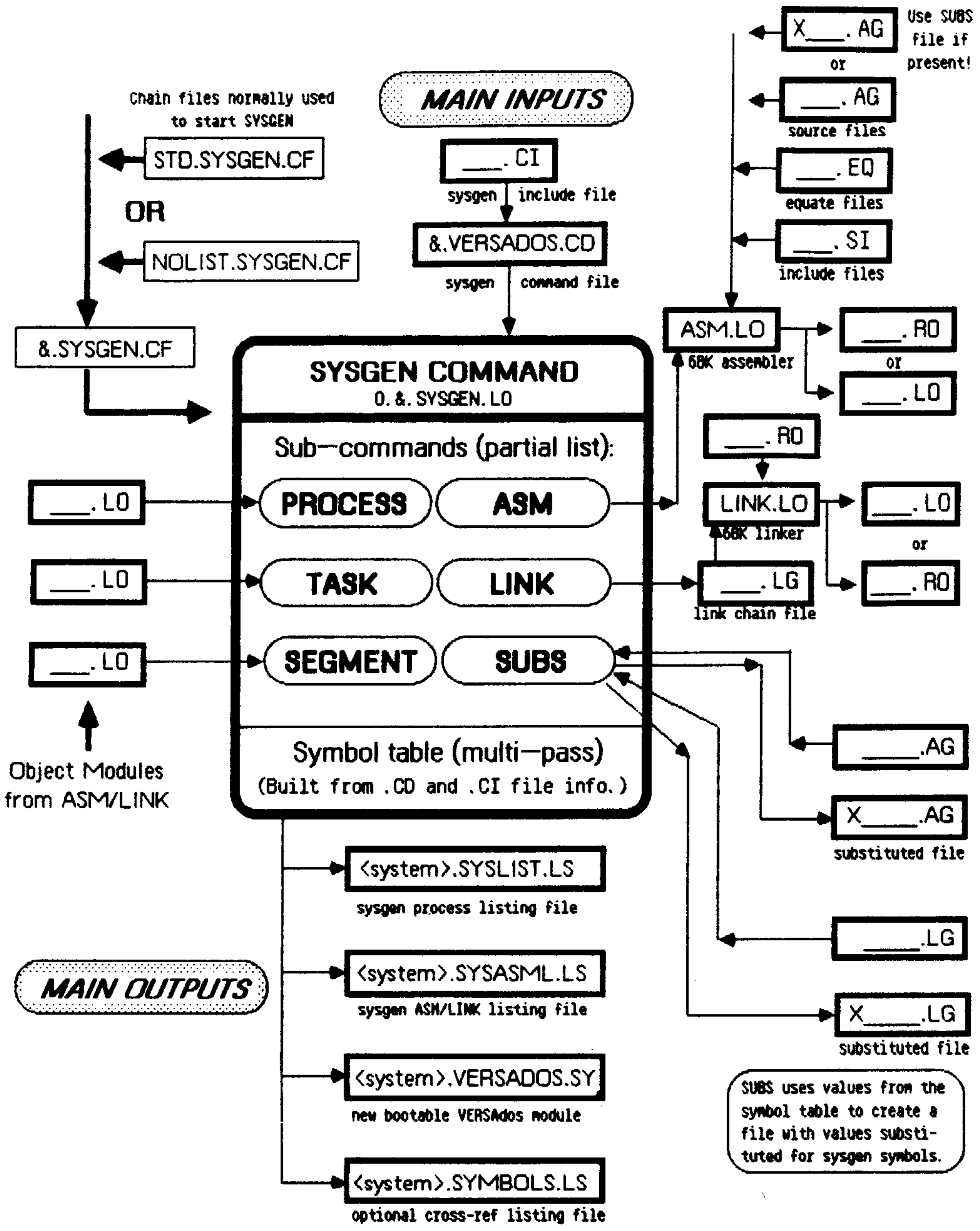

FIGURE 1-1. SYSGEN Command Pictorial

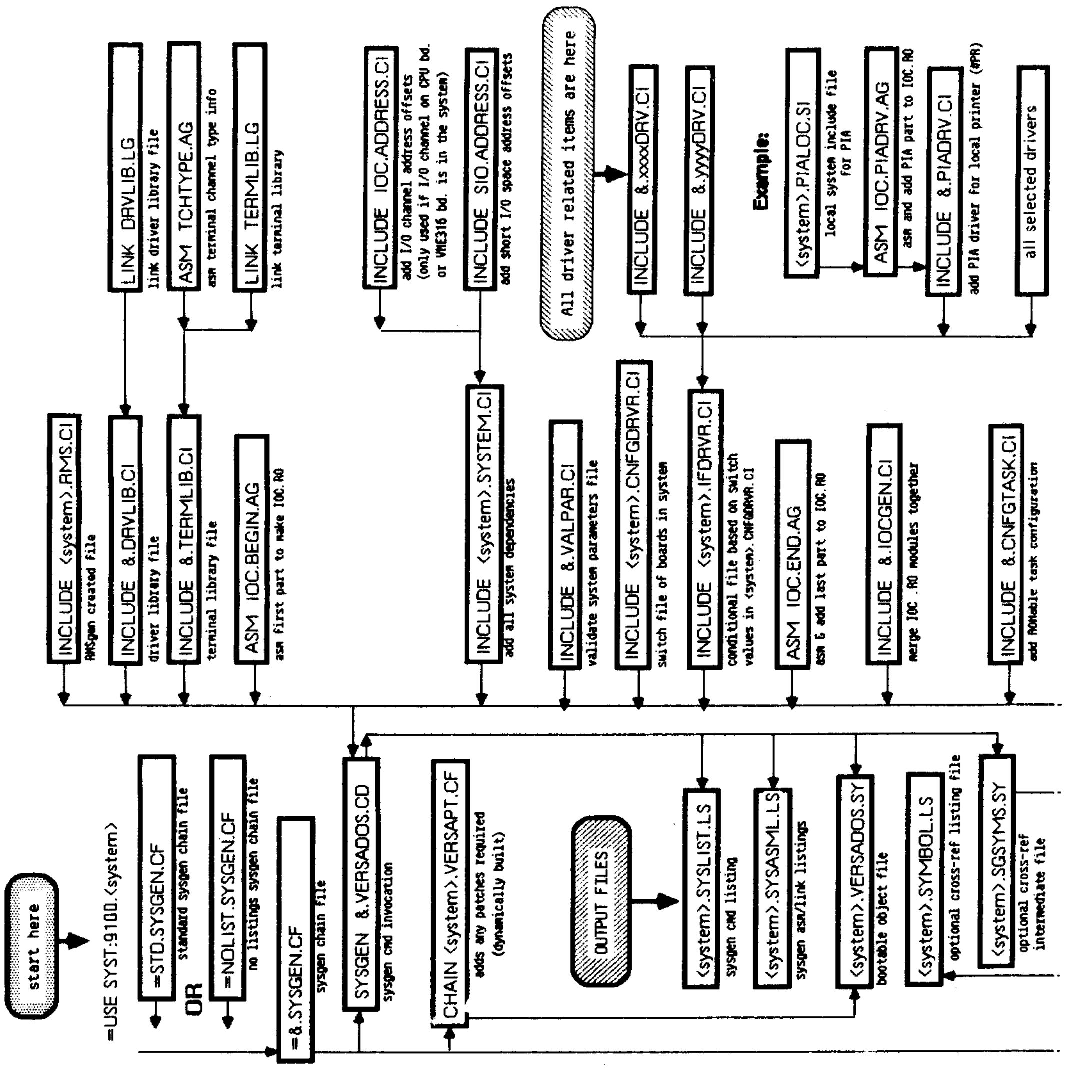
start here
=USE SYST:9100.<system>
=STD.SYSGEN.CF
standard sysgen chain file
OR
=NOLIST.SYSGEN.CF
no listings sysgen chain file
=&.SYSGEN.CF
sysgen chain file
SYSGEN &.VERSADOS.CD
sysgen cmd invocation
CHAIN <system>.VERSAPT.CF
adds any patches required (dynamically built)
OUTPUT FILES
<system>.SYSLIST.LS
sysgen cmd listing
<system>.SYSASML.LS
sysgen asm/link listings
<system>.VERSADOS.SY
bootable object file
<system>.SYMBOL.LS
optional cross-ref listing file
<system>.SGSYMS.SY
optional cross-ref intermediate file
INCLUDE <system>.RMS.CI
RMSgen created file
INCLUDE &.DRVLIB.CI
driver library file
INCLUDE &.TERMLIB.CI
terminal library file
ASM IOC.BEGIN.AG
asm first part to make IOC.RO
INCLUDE <system>.SYSTEM.CI
add all system dependencies
INCLUDE &.VALPAR.CI
validate system parameters file
INCLUDE <system>.CNFGDRVR.CI
switch file of boards in system
INCLUDE <system>.IFDRVR.CI
conditional file based on switch values in <system>.CNFGDRVR.CI
ASM IOC.END.AG
asm & add last part to IOC.RO
INCLUDE &.IOCGEN.CI
merge IOC .RO modules together
INCLUDE &.CNFGTASK.CI
add ROMable task configuration
LINK DRVLIB.LG
link driver library file
ASM TCHTYPE.AG
asm terminal channel type info
LINK TERMLIB.LG
link terminal library
INCLUDE IOC.ADDRESS.CI
add I/O channel address offsets (only used if I/O channel on CPU bd. or VME316 bd. is in the system)
INCLUDE SIO.ADDRESS.CI
add short I/O space address offsets
All driver related items are here
INCLUDE &.xxxxDRV.CI
INCLUDE &.yyyyDRV.CI
Example:
<system>.PIALOC.SI
local system include file for PIA
ASM IOC.PIADRV.AG
asm and add PIA part to IOC.RO
INCLUDE &.PIADRV.CI
add PIA driver for local printer (#PR)
all selected drivers

FIGURE 1-2. SYSGEN Overview (Sheet 1 of 2)

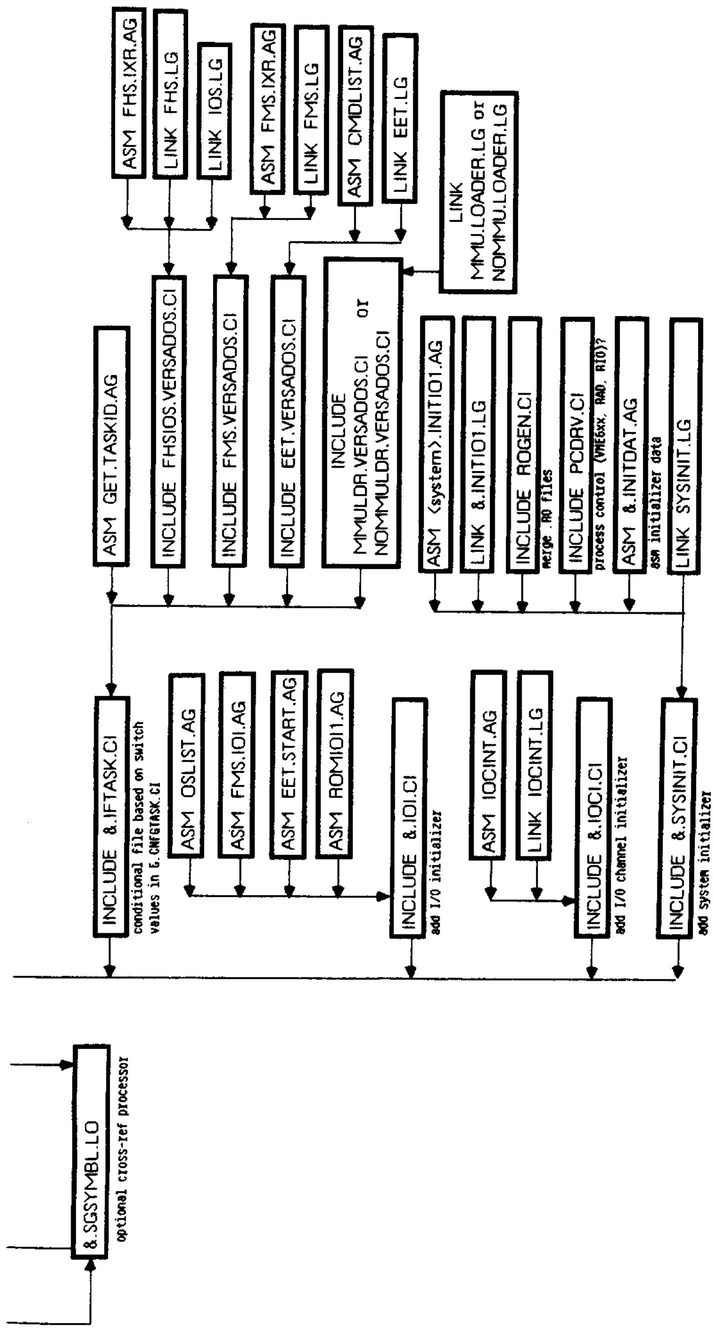

FIGURE 1-2. SYSGEN Overview (Sheet 2 of 2)

THIS PAGE INTENTIONALLY LEFT BLANK.

CHAPTER 2

INVOKING SYSGEN

2.1 GENERATING AN OPERATING SYSTEM USING SYSGEN

After the configuration and/or system files have been written or modified for the user configuration using the VERSAdos Editor, the **SYSGEN** facility must be called to create the new bootable VERSAdos load module. **SYSGEN** may be called directly or by executing either of the furnished chainfiles, STD.SYSGEN.CF, NOLIST.SYSGEN.CF, or &.SYSGEN.CF, to execute on the furnished command file, &.VERSADOS.CD. The configuration file contains descriptions of the user-changeable parameters and is the place where most changes are made. The system file also contains some user-changeable parameters. Specific driver items are found in the specific driver-related files.

Paragraph 2.2 contains instructions and the command line syntax for invoking **SYSGEN** on the furnished chainfiles, which then automatically operate on the furnished command files. To invoke **SYSGEN** directly on a command file, use the command line syntax in paragraph 2.3.

2.2 USING THE FURNISHED SYSGEN FILES

The procedures below apply to the VERSAdos releases for multi-user hard disk and single-user hard disk.

2.2.1 SYSGEN Steps

SYSGEN is performed on a hard disk. The steps required are:

a. Log on to the system default volume as user number 9100. This user number is the one normally reserved for **SYSGEN** processing; however, the user may elect to log on with a different user number. User numbers 9500-9999 are reserved for system use and should not be used as SYSGEN account numbers. Do not use user number 0 because the VERSADOS.SY file created will overwrite the existing VERSADOS.SY file under user number 0, and the system may not be rebootable.

b. To set up the volume, user number, and catalog (system type), issue a **USE** command where the **SYSGEN** will be done (for example, **USE SYS:9100.EXORMACS**).

c. To copy all the necessary files into 9100, including equate, macro, and UTILIB.RO files, invoke the appropriate copy chainfile (e.g., :9998.EXORMACS.COPYSYSGEN.CF). The chainfile contains documentation of the syntax for running the chainfile. The <system>.COPYSYSGEN.CF chainfile automatically sets the catalog to the correct system.

   NOTE: The MVME600-series drivers will not be copied (default). Refer to the chainfile documentation for details.

d. Modify the <system>.CNFGDRVR.CI file to select the board configuration for your system. Normally, this is all that needs to be done for most systems. You may optionally need to modify the <system>.SYSTEM.CI file, the &.VERSADOS.CD file, or specific driver files for non-standard media units. These files will have names such as &.M320DRV.CI and IOC.M320DRV.AG.

e. Start the **SYSGEN** process by invoking STD.SYSGEN.CF or NOLIST.SYSGEN.CF while under the catalog of the system being **SYSGEN**ed, or use &.SYSGEN.CF directly. Refer to these chainfiles for details on their usage. STD.SYSGEN.CF creates three listing files (.LS) and a bootable VERSAdos file (<system>.VERSADOS.SY) in the default account number/catalog. The first listing file is called <system>.SYSLIST.LS and contains all **SYSGEN** commands. The second listing file is called <system>.SYSASML.LS and contains all **SYSGEN** assembly and link listings. The third listing file is called <system>.SYMBOLS.LS and contains a cross-reference listing of the system symbols used. NOLIST.SYSGEN.CF creates two listing files (<system>.SYSLIST.LS and <system>.SYMBOLS.LS) and a bootable VERSAdos file (<system>. VERSADOS.SY) in the account number/catalog (<system>.SYSASML.LS is not produced).

f. Log off the system.

g. Log on as user 0 and save the current version of VERSADOS.SY by renaming it PRIOR.VERSADOS.SY. Then copy the new VERSADOS.SY from your **SYSGEN** account number to user 0.

h. Boot the new VERSAdos operating system by the usual method.

2.2.2 Driver Configuration File

The **SYSGEN** driver configuration file is <system>.CNFGDRVR.CI, where <system> is the name of the target system. This file consists of flags for each possible board type allowed in the system and any parameters required for the boards. This file is normally the only one that needs to be modified for most SYSGENs performed to add or remove standard device drivers. All other parameters are derived from these board switch parameters.

The as-delivered setting is for the standard VERSAdos operating system for that particular system.

2.2.3 System Dependent File

The system dependent file is <system>.SYSTEM.CI, where <system> is the name of the target system. This file contains parameters for the system's CPU board and the overall system itself. The user should not have to modify this file for normal **SYSGEN**s.

2.2.4 Conditional Driver INCLUDE File

The **SYSGEN** conditional **INCLUDE** driver file is <system>.IFDRVR.CI, where <system> is the name of the target system. This file contains conditionals based on the board switches in <system>.CNFGDRVR.CI to include only those drivers necessary to support the boards that have been configured into the system. The user should never have to modify the conditional **INCLUDE** driver file unless device drivers are being written for custom boards not supported by standard VERSAdos. (Refer to the Guide to Writing Device Drivers for VERSAdos, M68KDRVGD.)

2.2.5 SYSGEN Execution

When all **SYSGEN** files have been modified to reflect the desired configuration, **SYSGEN** may be executed. This is usually done by using one of the furnished chainfiles. After logging on to the appropriate volume as user 9100 with catalog equal to the system type of this **SYSGEN**, the user should type in the following command line:

```
=STD.SYSGEN.CF
```

This starts a standard **SYSGEN** with two listing files (.LS), one boot file (.SY), and one cross-reference file (.SY) to be generated. This chainfile does a **NOARG** and calls &.SYSGEN.CF to set the default conditions. You may optionally use &.SYSGEN.CF directly by entering the command line:

```
=&.SYSGEN.CF  <arg1>,<arg2>,<arg3>,<arg4>,<arg5>
```

where:

<catalog>	is set to the system type and <arg-n> are arguments. **NOTE:** Entry of all arguments is not necessary because each has a default value. Separating commas are required, however, whether defaults are used or not.
<arg1>	Command filename. Default: &.VERSADOS.CD
<arg2>	<temporary volume>:<user number>/<boot file>. Default: Logon volume name and user number. Refer to paragraph 2.3 for description of <temporary volume> and <boot file>.
<arg3>	List device or filename for **SYSGEN** messages. Default: <system>.SYSLIST.LS.
<arg4>	Reserved for future use.

<arg5>	Listing device or file. Default: <system>.SYSASML.LS. Legal devices are #, #PR, #PR1, #PR2, #PR3, #NULL, or a filename. All assembly listings and link maps are directed to the specified device or file. If <arg5> is a file, SYSGEN.TF is used as a temporary listing file to hold assembly and link listings before being copied to the desired output file, using the **COPY** utility with the append (;**A**) option. If <arg5> is #NULL, no assembly and link listings file is created. Due to restrictions in the **SYSGEN** literal substitution process, <arg5> cannot contain more than 30 characters.

The resulting boot file, <system>.VERSADOS.SY, is built under the user default volume and catalog.

The user can run the NOLIST.SYSGEN.CF file to do a **SYSGEN** with only the map and cross-reference listings. This reduces the time used by **SYSGEN**.

&.SGSYMBL.LO is an optional part (pass 2) of &.SYSGEN.LO that is run to create a readable output listing file from the resulting cross reference file, SGSYMS.SY, built under the user default volume and catalog.

NOTE

If the **SYSGEN** is ended prematurely, clear the arguments with "**=NOARG**" before reinvoking the **SYSGEN** chainfile.

EXAMPLE

```
=NOARG
=USE 9100.EXORMACS
=&.SYSGEN.CF  ,,SYSLIST.LS,,SYSASML.LS
```

This command executes the furnished chainfile &.SYSGEN.CF. The defaults used are:

```
<arg1>= &.VERSADOS.CD
<arg2>= logon volume and user number.
<arg4>= null
```

2.3 INVOKING SYSGEN DIRECTLY

The following command line invokes **SYSGEN** directly. Note that the supplied **SYSGEN** command files require additional arguments to be defined in the **ARG** command. These additional arguments are defined automatically when using the appropriate SYSGEN.CF chainfile. Thus, it is recommended to invoke **SYSGEN** through the chainfile instead of directly.

```
  =SYSGEN <command file>[,<temp vol>|,<temp vol>/<boot file>|,/<boot file>|,]
         [,<list device>][;<options>]
```

where:

SYSGEN	is the command mnemonic.
<command file>	is the name of the file containing the **SYSGEN** commands. The extension defaults to CD. Volume ID, user number, and catalog default to user defaults. Command filenames may be concatenated with a slash (/) between them (e.g., SYSCMD1/SYSCMD2/SYSCMD3). **SYSGEN** processes the three files as if they were one continuous file.
<temp vol>	is the volume name and user number for the temporary output file(s) created by the utility. Defaults to the logon volume name and user number if not specified.
<boot file>	is the name of the boot file created by the utility. If not specified, the volume ID, user number, and catalog default to the user defaults, and the filename and extension default to VERSADOS.SY. An existing file is overwritten.
<list device>	is the name of the device or file where all messages are sent. (Default is the logon device.) If a filename is specified, the extension default is LS; volume ID, user number, and catalog defaults are the user defaults. An existing file is overwritten.

The output listing from the **SYSGEN** process contains all the **SYSGEN** command lines as they appeared in the **SYSGEN** command file. All processed statements are preceded with the two-character string ". ". If the statement was read and not processed, two blanks precede it. Error messages are preceded by "%%" and auxiliary information is preceded by "- ".

<options> is one or more of the following:

R Causes the operating system to be configured for a Read Only Memory (ROM) environment.

When the **R** option is selected, the TCB is appended to the end of the boot file in an abbreviated format to conserve ROM space. After booting the operating system, this abbreviated TCB is expanded and written to RAM by the system initializer. The format for this compressed or "mini" TCB is:

```
MTCB       DC.L    '!TCB'
MTCBNAME   DC.L    <taskname>
MTCBSESSN  DC.L    <task session>

MTCBMON    DC.L    <monitor taskname>
           DC.L    <monitor task session>
MTCBUSER   DC.W    <user number>
MTCBLPRI   DC.B    <priority>
           DC.B    <reserved>
MTCBSTATE  DC.W    <state>
MTCBATTR   DC.W    <attributes>
MTCBENTRY  DC.L    <entry address>
MTSTMMU    DS.B    32 Task Segment Table MMU information
MTSATTR    DS.B    32 Task Segment Table attributes
```

The total length of this compressed TCB is 96 bytes.

The TCBs are built contiguously on disk, with the end of the list determined by a binary zero taskname.

P Causes the operating system to be configured with physical addresses only, i.e., configured to run on a system having no MMUs. For each memory management segment in the boot file, a physical starting address is assigned which matches the logical starting address obtained from the Loader Information Block (LIB) of the segment.

S Allows execution of a **SYSGEN** command file to be restarted at the beginning -- skipping execution of **ASM**, **COPY**, **DEL**, **LINK**, **SUBS**, and **PAUSE** commands -- until a prescribed point is reached, after which full processing is resumed. The purpose of the option is to save time and is normally used when a **SYSGEN** command file is changed or when **SYSGEN** execution has stopped prematurely.

The point where full execution is to resume is specified as a character string taken from the appropriate **SYSGEN** command file line. If there is a premature stop, look at the **SYSGEN** listing and specify the next to the last **SYSGEN** command executed as the restart point. The **SYSGEN** program searches for the string, resuming full execution when a match is found.

CAUTION

THE RESTART OPTION MUST NOT BE USED AT OR AFTER THE ASSEMBLY OF IOC.BEGIN.AG AND BEFORE THE PARAMETER MEMBEG =* IN THE &.VERSADOS.CD FILE. THE IOC.RO MODULE IS BUILT DYNAMICALLY AND THE ENTIRE PROCESS MUST BE EXECUTED CONTIGUOUSLY OR THE IOC.RO MODULE WILL NOT BE BUILT CORRECTLY. THIS MEANS YOU CANNOT RESTART AT ANY OF THE DRIVER FILES, OR WITHIN CNFGDRVR.CI, IFDRVR.CI, SYSTEM.CI, OR &.VALPAR.CI.

No restrictions are placed on length or content of the string, which is specified during the following dialog after execution of the **SYSGEN** command line is initiated:

```
ENTER CHARACTER MATCHING PATTERN FOR RESTART
```

After the entry of the desired string and a carriage return, the program responds with:

```
RESTART OPTION SPECIFIED - MATCH PATTERN IS: xxxxxx
```

where xxxxxx is the entered string. The entry of a carriage return initiates the abbreviated processing of the **SYSGEN** command file until it matches the specified string. Then the program displays:

```
----- NORMAL PROCESS RESUMING FROM RESTART
```

Resumes full **SYSGEN** processing and continues to the end of the command file.

T=n Allows the user to specify the number of user-defined symbols in the symbol table, thus either increasing or decreasing the amount of memory to be allocated for the symbol table. For example:

```
=SYSGEN SYSCMD,,SYSLIST.LS;T=350
```

causes execution of **SYSGEN**, using SYSCMD as the command file, and **SYSGEN** sets aside enough memory to accommodate 350 symbols in the symbol table. (Default=170.)

If **SYSGEN** is unable to allocate enough memory, it displays the message:

```
"WAITING FOR PHYSICAL MEMORY TO BECOME AVAILABLE!!"
```

When **SYSGEN** has allocated enough memory for the symbol table, the message:

"PHYSICAL MEMORY HAS NOW BEEN ALLOCATED!!"

displays and **SYSGEN** automatically continues operation.

C Causes **SYSGEN** to create a cross-reference file, SGSYMS.SY (default), under the user's current defaults, which contains information about the **SYSGEN** defined parameters and the user-defined parameters. The information in this file is the parameter name, the value for the parameter, the file(s) that define the parameter, and the file(s) that reference the parameter. The **SGSYMBL** utility is run to create a readable output listing file, SYMBOLS.LS, from the SGSYMS.SY file (refer to paragraph 3.20).

CHAPTER 3

SYSGEN UTILITY COMMANDS

3.1 THE SYSGEN UTILITY COMMAND LIST

This paragraph presents a brief description of each **SYSGEN** utility command. Detailed descriptions of each command follow this paragraph. A **SYSGEN** command line with an "*" as the first non-blank character is treated as a comment; the line is listed but no processing takes place. Any utility program not requiring interactive dialog may be invoked from within the **SYSGEN** command file by placing an "=" as the first non-blank character, followed by the command line.

<parameter>	is the name of a **SYSGEN** parameter followed by its value. The value is in effect throughout the remainder of the **SYSGEN** process and cannot be redefined.
=<progname> [<legal args>]	invokes a utility program (where <progname> is the name of the utility and <legal args> represents any command line input that is allowable for that utility). The utility cannot carry on an interactive dialog. Using this capability in the **SYSGEN** command file invokes the **COPY** utility with the append option to produce a single listing of all assemblies and links.
ABORT	forces **SYSGEN** to abort.
ASM	specifies an assembler command line that causes **ASM** to be invoked. Using **ASM** causes a search for a file with the same name, but preceded with an **X** (a **SUBS**ed file). If found, this "substituted" file will be assembled. This will not occur if **=ASM** is used, which invokes the assembler directly.
END	ends previous task or process.
ENDC	terminates the conditional processing associated with its associated **IF**xx directive.
EXCLUDE	specifies a segment of a process or task that is not loaded with the process or task.
IFxx	(where xx is **EQ**, **NE**, **GT**, **LT**, **GE**, or **LE**), initiates conditional processing.
INCLUDE	defines a file to be included in **SYSGEN** processing.
LINK	specifies a source file that contains input to the linkage editor and invokes **LINK**.

MSG	causes an operator message to be displayed at the relevant terminal.
PAUSE	halts **SYSGEN** execution until any character key is pressed.
PC	adjusts the location counter maintained during **SYSGEN** execution.
PROCESS	defines the beginning of a process stream of the type that results in a process being included in the output file. Also marks the end of the previous task or process if not completed by an **END** statement.
SEGMENT	defines the beginning of a segment stream of the type that results in a process being included in the output file. Also marks the end of the previous task or process if not completed by an **END** statement.
SUBS	indicates source file(s) where the values of **SYSGEN** defined parameters are substituted for the parameter names.
TASK	defines the beginning of a task stream of the type that results in a task being included in the output file. Also marks the end of the previous task or process if not completed by an **END** statement.

3.2 NOTES ON SUBSTITUTION PROCESS

SYSGEN provides flexibility through a two-step substitution process.

3.2.1 SYSGEN Command File

SYSGEN performs inline substitution for each command read from the command file. First the substitution is performed from the list of substitution parameters previously defined with the parameter command. After all the substitutions are made for that particular command, another pass is made on the command for substitutions from the **ARG** list previously defined from the **ARG** session control command. The command file itself is never changed; all substitutions are made on the memory image of the record after it is read from the file. After all substitutions are made, the command is processed.

3.2.2 Filename Appearing on SUBS Command Line

Here, substitutions are made in the records read from the file from the **SYSGEN**-maintained parameter list (built from previous parameter commands). Refer to paragraph 3.17 on the **SUBS** command for more detail on this process. No substitutions are made from the session control **ARG** list.

3.3 PARAMETER COMMAND

A parameter remains in effect from the point it is defined until the end of **SYSGEN** execution. Only select parameters may be defined. Redefinable parameters are restricted to those specifying TCB information (covered later) and those parameters having an ampersand (&) as their first character. A redefinable parameter is one that can be defined at two or more different places within the **SYSGEN** command file. The value used is the one associated with the most recent definition.

3.3.1 Parameter Command Syntax

```
    <parameter name>=<value>[<space><comment>]

    <parameter name>=*[+<expression>][<space><comment>]
```

where:

<parameter name>	is the name by which the parameter is known and referenced. Maximum of eight alphanumeric characters plus ampersand (&), dollar sign ($), and period (.). One-character parameter names are invalid to avoid conflict with session **ARG** parameters. A parameter name that begins with an ampersand character is a redefinable parameter, i.e., it can appear more than once on the left side of a parameter command statement. It is similar in concept to the **SET** directive in the M68000-family assembler. The ampersand is part of the parameter name and counts as one character in the name.
=	is a required delimiter.
<value>	is an <expression>, a <string>, or a <literal>.
*	is the current value of the **SYSGEN** location counter.
+ or -	is required if an <expression> follows.
<expression>	is an arithmetic expression involving hexadecimal and/or decimal constants and/or previously defined hexadecimal parameters. All arithmetic calculations are performed in integer mode. The operator precedence has multiplication and division as the highest priority, followed by addition and subtraction. Precedence of the same priority is from left to right. Use parentheses to alter the order of operations.

EXAMPLE: **$12*(4/(TAG1+TAG2))-2**

A dollar sign (**$**) preceding a constant indicates hex value. In the above expression example, if TAG1 were previously defined as 0 and TAG2 as 2, the expression value would be 34. This value is saved internally as the hexadecimal constant $22. Negative hex expressions are invalid, e.g., +-$23 and --$23 are invalid.

<string> is a string of up to 30 characters enclosed by single quotes. To encode a single quote in the middle of the literal, use two adjacent single quotes. The delimiting single quotes are saved as part of the symbol.

EXAMPLES: **'ABC'**
'DON''T'

<literal> is a string of up to 30 characters enclosed by double quotes (", or hex code $24). The 30 characters do not include the delimiting double quotes. The double quotes are not saved as part of the literal string. To encode a double quote in the middle of the string, use two adjacent double quotes.

EXAMPLES: **"LITSTRING"**
"AB""CDEF"

The backslash (\, or hex code $5C) cannot appear in any string because it is assumed to be a substitution sentinel and substitution is performed before processing by the parameter routine.

<space> represents a required blank space.

<comment> is a string of characters (maximum of 80, less preceding characters on that command line).

The value specified by a parameter command can be substituted in any source file that contains a parameter that matches <parameter name>. (Refer to the **SUBS** command.)

EXAMPLE

```
XX=10
YY=$F0
SS='ABC'
```

3.3.2 Special Parameters

The **SYSGEN** utility reserves a few parameter names for specifying TCB information. Although these parameters have the same format, they are restricted to certain values. The **SUBS** command does not do value substitution for these parameter names. The five redefinable special parameters are:

PARAMETER	MEANING	VALID VALUES	INITIAL VALUE
USER	Task user number	2 byte <number>	0
SESSION	Task session number	4 byte <number>	0
PRIORITY	Task initial priority	1 byte <number>	0
STATE	Task initial state	<string> with a value of: 'READ' (ready) 'WAIT' 'DORM' (dormant) 'SUSP' (suspended)	READ
ATTRIB	Task attributes	<string> with a value of: 'SYST' System task 'USER' User task 'RTIM' Real-time task 'CRIT' Task is critical to operating system. If this attribute and the 'SYST' attribute are in effect, the system will crash whenever the associated task aborts. 'NOCR' Task is not critical to operating system.	'SYST' 'CRIT'

All task attributes and state settings revert to a default value whenever a new **TASK** command is encountered. The default values are: system task, critical and ready. To minimize the chance of setting the state or attributes wrong, use **ATTRIB** and **STATE** commands associated with the task immediately following the **TASK** command. A single **ATTRIB** command affects only one bit in the attributes word. Issuing multiple **ATTRIB** commands for the same task can affect more than one bit.

NOTE: The **STATE** command, when issued for a particular task, will overwrite any of the information set up by a previous **STATE** command. Thus, only issue one **STATE** command for each task. However, **SYSGEN** does not enforce this rule.

The utility completely maintains six additional special parameters. The user cannot redefine these parameters. Only value substitution is allowed using the **SUBS** command. The **DATE** and **TIME** parameters are initialized from the system date and time when starting SYSGEN.

PARAMETER	MEANING
$TCBLST	Pointer used to maintain the list of TCBs. Each task is linked into this list and **$TCBLST** always contains the TCB address of the last task processed.
$TCBRDY	Same as above but includes only tasks whose initial state is ready.
$DATE	Six-character ASCII date stored as a literal, yymmdd, where yy is the last two digits of the year, mm is the month, and dd is the day of the month.
$TIME	Four-character ASCII time stored as a literal, hhmm, where hh is the hour and mm is the minutes into the hour.
$RA	16-bit binary abort code register. When a utility such as **ASM** or **LINK** aborts, **$RA** contains the abort code, obtained from the lower half of A0 when the task aborts.
$RD	16-bit binary diagnostic pseudo register. **$RD** contains the value that was in the upper 16 bits of D0 when a utility such as **ASM**, **LINK**, or **COPY** terminates. This information normally reflects error and warning information similar to the RD pseudo register used in chainfile processing. This register can be tested within the **SYSGEN** command file.

EXAMPLE

```
ASM DRIVER,,DRIVER
IFLE $C000-\$RD
    PAUSE - ERRORS IN ASM
ENDC
```

In the above example, if the assembler terminated normally, the pause is not executed. If **ASM** had aborted, then the pause would have executed. Refer to the VERSAdos System Facilities Reference Manual, under the discussion of the **CHAIN** utility, for additional information on the RA and RD pseudo registers.

3.4 ABORT COMMAND

The **ABORT** command forces the **SYSGEN** utility to abort.

Syntax

ABORT [comment]

3.5 ASM (ASSEMBLE) COMMAND

The **ASM** command allows one or more source files to be assembled and a relocatable module to be generated. Although no check is made, this feature is only useful if at least one source file had parameter substitution before the assemble request. The command should appear exactly as is required by the assembler.

Syntax

ASM <source file>,<object file>,<listing file>[;<options>]

where:

<source file>, <object file>, <listing file>, and <options> are as described in the M68000 Family Resident Structured Assembler Reference Manual.

Before invoking the **ASM** command, **SYSGEN** searches for source file(s) with the specified name(s) preceded by **X**. If found, it uses the substitution file instead of the corresponding source file. This will not occur if you use the direct assembler command, **=ASM**.

EXAMPLE

ASM FMSREF,FMSREF,#PR1;R

This command looks for the substituted filename of <system>.XFMSREF.SA first, and if found, assembles it (**SYSGEN** runs under catalog equal to the system type).

3.6 END COMMAND

The **END** command causes the processing of the previous task or process to be completed, or marks the end of the file.

Syntax

END [<comment>]

3.7 ENDC COMMAND

The **ENDC** command must be paired with a previous **IFxx** command. Together, these commands define a block of commands that may or may not be processed, depending on the results of the **IFxx** command.

Refer to the **IFxx** command description for an example.

3.8 EXCLUDE COMMAND

A segment name from the Loader Information Block (LIB) of a task or process can be specified on an **EXCLUDE** command line and can then be omitted from the output file portion for that task or process.

Syntax

```
    EXCLUDE <segment name>[<space><comment>]
```

where:

<segment name> is the name of a segment to omit from the output file.

Valid segment names are two, three, or four alphanumeric characters, and allow ampersand (**&**), period (**.**), and dollar sign (**$**).

For one task or process, a maximum of four **EXCLUDE** commands can be specified. These must follow the corresponding **TASK** or **PROCESS** command and precede the terminating statement for that task or process.

EXAMPLE

EXCLUDE DSEG	Exclude segment DSEG from the current process or task.

3.9 IFxx COMMAND

The **IFxx** command allows conditional processing of **SYSGEN** commands and must be paired with an **ENDC** command.

Syntax

```
    IFxx <expression>[<space><comment>]
```

where:

xx	is one of the following two-character strings:
	EQ, NE, GT, LT, GE, LE.

<expression>	is a numeric expression consisting of **SYSGEN** substitution parameters, **ARG** substitution parameters, and/or hex and decimal constants. A carriage return or a blank terminates the expression.

The expression is evaluated in a simple left-to-right scan, and obtains a 32-bit number. This number is compared to zero and then the xx-specified test is applied. If the test is true, processing continues in a normal manner. If the test is false, later command lines are not processed until the terminating **ENDC** command is found. Conditional tests may be nested to any depth. An expression may also be two strings or literals separated by a comma or a space. A comparison is made on a character basis between the two strings.

EXAMPLE

```
HDUDCO=4
FDUDCO=2
IFGT     \HDUDCO+\FDUDCO
   TASK IPC,.IPC
   PRIORITY=$D8
      .
      .
      .
ENDC
SYSTYPE="EXORMACS"
IFNE     "\SYSTYPE", "VM02"
   MSG  These commands would be processed
      .
      .
      .
ENDC
```

Here, it would process all the statements between the **IFNE** and **ENDC** commands. If the expression contains parameter substitution sentinels, then the associated parameter must have been previously defined.

3.10 INCLUDE COMMAND

The **INCLUDE** command allows the user to include a secondary file in the source input stream.

Syntax

```
INCLUDE <filename>
```

The next source stream images are taken from the file named on the **INCLUDE** statement. When that file is exhausted, processing resumes with the statement following the **INCLUDE**. Nesting of **INCLUDE** files is allowed (maximum of four).

3.11 LINK COMMAND

The **LINK** command allows one or more relocatable modules to be linked together to produce a loadable module.

Syntax

```
LINK <filename>
```

where:

<filename>	is the name of a chainfile containing the **=LINK** command line followed by any linkage editor command input. The default extension is CF and the data in <filename> can be in the same format as a chainfile. The file is not actually processed as a chainfile, but merely passed to the linker as input. No commands other than comments beginning with **=/*** may appear in the link file before the **=LINK** command.

If parameter substitution in <filename> is desired, precede the **LINK** command line with a **SUBS** <filename> command line. **SYSGEN** uses the substitution file (<filename> preceded by **X**) if one exists.

EXAMPLE

SYSGEN command file: Task stream name and start.

```
TASK  FMS
FMSSTR= *          (* represents the current value of the SYSGEN location
                   counter.)
SUBS  FMSLNK.CF    Make parameter substitution into FMSLNK.CF.
LINK  FMSLNK       Link in module FMSLNK.
```

Contents of the FMSLNK.CF file referenced on the file **SUBS** command line (before substitution):

```
=/* COMMENTS MAY GO HERE (optional)
=LINK  ,FMS,#PR;IXHM     (= is optional)
 SEGMENT 0:0  \FMSSTR
 INPUT FMS,FMSX
 END
=END                     (optional line)
```

Assuming the **SYSGEN** location counter had the value of $9C00 at the time FMSSTR was specified, the XFMSLNK file would be as follows after the **SUBS** is processed:

```
=/* COMMENTS MAY GO HERE
LINK, FMS, #PR;IXHM
SEGMENT 0:0  $9C00
INPUT FMS,FMSX
END
=END
```

3.12 MSG (MESSAGE) COMMAND

The **MSG** command outputs text to the logon device.

Syntax

MSG <content>

where:

<content> is a string of characters (maximum of 75).

EXAMPLES

```
MSG  REMOVE VOL 1

MSG  MOUNT  VOL 2

MSG  DEPRESS RETURN WHEN READY
```

3.13 PAUSE COMMAND

The **PAUSE** command temporarily halts **SYSGEN**. Execution continues when any character key (other than **BREAK**) is pressed.

Syntax

PAUSE [<comment>]

3.14 PC COMMAND

The **PC** command allows the user to alter the location counter maintained by the utility. This counter determines the destination memory locations in the output file.

Syntax

```
PC =   *[<space><comment>]

PC =   *+<expression>[<space><comment>]

PC =   <expression>[<space><comment>]
```

where:

*	is the current value of the **SYSGEN** location counter.

+	causes the following value to be added to the value of the current location counter.
<expression>	is a string of hex or decimal digits, up to a maximum of 10 characters, including **$**. If specified, it must be on a page (256 byte) boundary.

The new value of the location counter must be equal to or greater than the old value. The new **PC** value is sent to the list device. On startup, the counter value is initialized to zero and updated as tasks and processes are written to the output file. If the user changes the location counter value, **SYSGEN** writes zeros to the output file from the old value up to the new value unless the **PC** command precedes any **TASK**, **PROCESS**, or **SEGMENT** commands. In that case, zeros are not written to the output file. Instead, the new **PC** value becomes the starting address of the data in the boot file.

3.15 PROCESS COMMAND

A process stream causes a load module to be processed. After **SYSGEN** tailoring, the result is a module that is included in the output file. The module is transformed into supervisor mode suitable for executive or driver type application.

Syntax

PROCESS <filename>[<space><comment>]

where:

<filename>	is the name of the file containing the load module.

All commands between the **PROCESS** command and the next **END**, **PROCESS**, **SEGMENT**, or **TASK** command or end of file constitute a process stream.

Since the process runs in supervisor mode and the MMU is not available, a process must be written to the output file at the address specified in its Loader Information Block (LIB). Therefore, if **SYSGEN**'s location counter is less than the starting address of the process, zeros are written to the output file until the location counter equals the starting address. An error occurs if the location counter is greater than the process starting address.

The load address of the last process in the **SYSGEN** command file is inserted in the restart vector (offset 4) of the boot file.

EXAMPLE

PROCESS EXEC	Process the load module **EXEC** in supervisor mode.

3.16 SEGMENT COMMAND

The **SEGMENT** command allows a segment of a load module to be specified for inclusion in the boot file as a process.

Syntax

SEGMENT <filename>,<segment name>[<space><comment>]

where:

<filename>	is the descriptor of the load module file containing <segment name>. The minimum required is the filename field. The default extension is LO.
<segment name>	is the name of the segment to be included in the boot file. Valid segment names are two, three, or four alphanumeric characters, with ampersand (&), period (.), and dollar sign ($) allowed.

All commands between the **SEGMENT** command and the next **END**, **SEGMENT**, **PROCESS**, or **TASK** command constitute the segment stream.

The address of the specified segment in the LIB of the load module is ignored. Instead, the segment is included in the boot file at the address provided by the **SYSGEN** location counter when the **SEGMENT** command begins execution. A process generated in this way (by **SEGMENT** command execution) is not used as a startup address.

3.17 SUBS (SUBSTITUTION) COMMAND

The **SUBS** command allows the values of parameters to be substituted in a source file referenced by the **SYSGEN** command file.

Syntax

SUBS <filename1>[,<filename2>...,<filenamen>]

where:

<filenamen>	is the name of a source file where parameters are to be substituted. Default extension is SA. Multiple filenames, separated by commas, can be specified.

The **SUBS** command reads the specified source file, examining each record. Whenever a backslash is encountered, the characters that follow are examined to determine if they match the name of a **SYSGEN** parameter. (A backslash followed by just one of the characters **0-9** and **A-Z** is ignored.) If no parameter by that name exists, an error is logged. The utility copies the source file records with any indicated substitutions made to an output file.

The new file is given the name of the source file preceded by an **X**. The filename portion of <filenamen> is a maximum of seven characters, instead of the usual eight.

EXAMPLE

SYSGEN command file contains:

```
        NODISK=4
                         Parameter command lines
        DEFVOL='BOOT'

        SUBS FMSREF
```

FMSREF contains:

```
        * SET UP SPACE BASED ON NO. OF DISKS IN SYSTEM

                DS.B   \NODISK*50

          SYSDEF  EQU  \DEFVOL    SYSTEM DEFAULT VOLUME
```

The resulting source file XFMSREF contains:

```
        * SET UP SPACE BASED ON NO. OF DISKS IN SYSTEM

                DS.B    4*50

          SYSDEF  EQU  'BOOT'    SYSTEM DEFAULT VOLUME
```

The **ASM** command assembles the new file (refer to paragraph 3.13).

3.18 TASK COMMAND

A task stream causes a load module to be processed. After **SYSGEN** tailoring, the result is normally a task that is included in the output file.

Syntax

```
    TASK <filename>[,<taskname>][<space><comment>]
```

where:

<filename>	is the name of the file containing the load module. The default extension is LO.

<taskname>	is the name of the task. Overrides the name generated by the linkage editor. Valid tasknames include two to four alphanumeric characters, with ampersand (&), period (.), and dollar sign ($) allowed.

All commands between the **TASK** command and the subsequent **END**, **TASK**, **SEGMENT**, or **PROCESS** command or end of file constitute a task stream. The task starts at the address indicated by the current **SYSGEN** location counter unless the **P** option was specified. If the **P** option was specified on the **SYSGEN** command line, there is no MMU, and the task is located in the output file at the address specified in the loader information block of the task. Therefore, if the **SYSGEN** location counter value is less than the starting address of the task, zeros are written to the output file until the **SYSGEN** location counter value is equal to the starting address of the task. An error occurs if the location counter is greater than the task starting address.

EXAMPLES

TASK FMS,.FMS	Process the load module FMS.LO and rename the task to .FMS.
TASK PROG.LO	Process the load module PROG.LO. The taskname remains that as defined by the linker.

3.19 OTHER EXECUTABLE COMMANDS

By coding a command line preceded by "=" within the command file, any non-interactive utility program may be invoked from within **SYSGEN**. The utility runs the same as if it had been invoked from within a chainfile. (Refer to paragraph 3.1.)

3.20 SYMBOL UTILITY (SGSYMBL)

The **SGSYMBL SYSGEN** pass two processor generates a cross-reference listing file of the parameters used during a **SYSGEN**. The input for this utility is the file SGSYMS.SY under the user's current defaults. The output is the file SYMBOLS.LS under the user's current defaults. The file SGSYMS.SY must have been created by the **SYSGEN** utility with the **C** option specified at **SYSGEN** time or by the STD.SYSGEN.CF chainfile. The information in the SGSYMS.SY file is:

a. The parameter name
b. The value for the parameter
c. The file(s) that define the parameter
d. The file(s) that reference the parameter

See Figure 3-1 for an example of the listing. During its operation, **SGSYMBL** displays the number of the symbol that is currently processing. In the "Referenced in file(s)" column there may be filenames followed by a number in brackets, which is the number of times that symbol was referenced from that file, or the filename may print more than once.

Syntax

```
=SGSYMBL
```

```
SGSYMBL version 011485 4       03/01/85 17:12:40

SYSGEN was performed 03/01/85 16:23:00

         Value      Value        Value
Symbol   (hex)      (decimal)    (ASCII) ASCII string                  Defined in file(s)   Referenced in file(s)
-------- ---------  ------------ ------- ----------------------------- -------------------- --------------------

&CRTDV   $434E3033  &1129197619  'CN03'                                &.MPSC400.CI         &.MPSC400.CI
                                                                       VMES10.DRVS10.CI     IOC.MPSCDRV.AG
                                                                       &.VERSADOS.CD        VMES10.DRVS10.CI
                                                                                            IOC.DRVS10.AG

&DEFVOL  $6         &6           '....'                                &.VERSADOS.CD        &.VERSADOS.CD [3]

&DSKDV   $2F00      &12032       '../.'                                &.VERSADOS.CD

&FILENAM                                 &.XPIA410.SI                  &.PIA410.CI          IOC.PIADRV.AG
                                                                       &.MPSC400.CI         IOC.MPSCDRV.AG

&IOCBASE $F1C000    &15843328    '....'                                VMES10.SYSTEM.CI     &.VERSADOS.CD

&M420FLG $0         &0           '....'                                &.VERSADOS.CD

&MPSCFLG $1         &1           '....'                                &.MPSCDRV.CI         &.MPSCDRV.CI
                                                                       &.VERSADOS.CD

&PCDRV   $0         &0           '....'                                &.VERSADOS.CD        &.VERSADOS.CD

&PIAFLAG $1         &1           '....'                                &.PIADRV.CI          &.PIADRV.CI
                                                                       &.VERSADOS.CD

&PRTDV   $50523120  &1347563808  'PR1 '                                &.PIA410.CI          &.PIA410.CI [2]
                                                                       &.PIA410.CI          IOC.PIADRV.AG
                                                                       &.VERSADOS.CD

&SDRVADD $BB00      &47872       '....'                                &.MPSCDRV.CI         IOC.MPSCDRV.AG

&SDRVR                                   MSPR                          &.MPSC400.CI         IOC.MPSCDRV.AG [2]
                                                                                            &.MPSCDRV.CI [2]

&SERFLAG $1         &1           '....'                                &.MPSCDRV.CI         &.VERSADOS.CD
                                                                       &.VERSADOS.CD        &.MPSCDRV.CI

&SPRFLAG $1         &1           '....'                                &.MPSCDRV.CI         &.MPSCDRV.CI
                                                                       &.VERSADOS.CD

&SUPFLAG $0         &0           '....'                                &.VERSADOS.CD        &.MPSCDRV.CI

&TOTDSK  $2         &2           '....'                                &.RWINDRV.CI         &.VERSADOS.CD
                                                                       &.VERSADOS.CD        &.RWINDRV.CI
```

FIGURE 3-1. Excerpt from SYMBOLS.LS File

THIS PAGE INTENTIONALLY LEFT BLANK.

CHAPTER 4

SYSGEN ROM CAPABILITY

The ROM capability associated with the **SYSGEN** process allows the user to build a ROMed system that includes the RMS68K kernel, user-written applications programs, and that portion of VERSAdos functionality that allows device assignment and access. File management, the loader, and session control are not included in the ROM capability. The user can write applications in assembly language or Pascal language (modified version 2.3). Multiple Pascal programs in the same ROM system can share the same run-time library code, minimizing ROM space requirements.

In user number 9990 a chainfile exists, ROM.EXAMPLE.CF, that will create a ROM product using a Pascal task that can be executed on an MVME110 system. The file contains detailed instructions, including logon and invocation procedures.

The general steps for creating a ROM system discussed in detail in this chapter are:

a. Create an RMS at the desired ROM start address.

b. Invoke the appropriate COPYSGEN chainfile to accumulate the **SYSGEN** files into the desired user number.

c. Modify the files related to the ROM capability.

d. Create and insert an application **INCLUDE** file.

e. Select, via switch settings, the required drivers for the user application.

f. Start the **SYSGEN**.

4.1 GENERAL ROM SYSGEN CONSIDERATIONS

This **SYSGEN** example is for an MVME110 system and assumes that the files have been modified for the configuration of the user's system. The values used are not binding. The catalog field typically represents the <system> (VME110, VME120, VM02, etc.), for the target **SYSGEN**.

1. Modify the VME110.RMS.CD file value "RMS" to the desired user starting ROM address. This dynamically defines the value "ROMSADDR" which is the ROM start address. The user should ensure that there is no conflict between RAM and ROM partitioning in the command **INCLUDE** file VME110.SYSTEM.CI (refer to step 3 c.).

```
BEFORE:   RMS  =   $40000    Address where RMS68K starts.

AFTER:    RMS  =   $300000   Address where RMS68K starts.
```

Now start an RMS **SYSGEN** in user number 9999, to create the ROMable RMS at the selected ROM start address. The files generated by the RMS **SYSGEN** that will be used in the product **SYSGEN** are VME110.RMS.LO, VME110.RMS.LL, and VME110.RMS.CI. These files are automatically copied into the target user number where the system **SYSGEN** will occur when the system COPYSGEN is started. Invocation of an RMS **SYSGEN** is done by the command line **RMSGEN.CF** <system>. Thus, for an MVME110 system, the user would input **RMSGEN.CF VME110.**

2. Copy, using the VME110.COPYSGEN.CF file, those files that are used to create the VME110.VERSADOS.SY module. The parameters required to start the VME110.COPYSGEN.CF file are documented in the file. It is the user's responsibility to copy the user-written **SYSGEN** application-related files because this file will not copy them.

3. Modification of the following files must occur before initiating the **SYSGEN** for the target system:

   a. &.SYSGEN.CF

      This change requests the **SYSGEN** process to create a ROMable product.

      ```
      BEFORE:   =SYSGEN \5,\6,\7;CT=350

      AFTER:    =SYSGEN \5,\6,\7;CT350,R
      ```

   b. &.CNFGTASK.CI

      This change removes the File Management System (FMS), the Exit/Entry Task (EET), and the Loader (LDR) from the **SYSGEN** product.

      BEFORE:

      ```
      FMS$ = 1 Set =0 for skip FMS module, not =0 to include it
      EET$ = 1 Set =0 for skip EET module, not =0 to include it
      LDR$ = 1 Set =0 for skip LDR module, not =0 to include it
      ```

      AFTER:

      ```
      FMS$ = 0 Set =0 for skip FMS module, not =0 to include it
      EET$ = 0 Set =0 for skip EET module, not =0 to include it
      LDR$ = 0 Set =0 for skip LDR module, not =0 to include it
      ```

   c. VME110.SYSTEM.CI

      This change establishes the ROM ending address. The ROM ending address should be at least large enough to accommodate the ROM product being produced.

The user should make sure that the ROM start/end addresses do not conflict with the "MEMEND" addresses that are in this file. The ROM start address was established in step 1. The "MEMEND" addresses for the MVME110 are:

BEFORE:

ROMEADDR = 0	ROM end address defined by the user for a ROMable system. (The ROM start address (ROMSADDR) is defined in the VME110.RMS.CI file and has a value equal to the initial program counter.)
MEMEND1 = $200000	Ending address for onboard memory must be lower than (<) this.
MEMEND2 = $00000	Starting address for offboard memory must be greater than or equal to (>=) this. (Not applicable for an MVME110.)
MEMEND3 = $00000	Ceiling address for offboard memory must be lower than (<) this.

AFTER:

ROMEADDR = $313200	ROM end address defined by the user for a ROMable system. (The ROM start address (ROMSADDR) is defined in the VME110.RMS.CI file and has a value equal to the initial program counter.)
MEMEND1 = $200000	Ending address for onboard memory must be lower than (<) this.
MEMEND2 = $00000	Starting address for offboard memory must be greater than or equal to (>=) this. (Not applicable for an MVME110.)
MEMEND3 = $00000	Ceiling address for offboard memory must be lower than (<) this.

d. &.VERSADOS.CD

The application **INCLUDE** file created by the user should be inserted (example follows). Refer to Appendix E for an example of application **INCLUDE** files for an assembly task and a Pascal task. Application tasks should have a priority less than the File Handling Services (FHS) or the Input/Output Services (IOS). This is done by using the **PRIORITY SYSGEN** command. If the application **INCLUDE** file is for an assembler source, the application task should issue a Relinquish directive before soliciting the services of FHS and/or IOS to ensure the completion of their initialization. The Pascal initializer does this function for Pascal source.

```
    BEFORE:

  IFNE  \FHS$IOS$
    MSG
    MSG          ***************************************************
    MSG          **  System I/O Initializer
    MSG          ***************************************************
    INCLUDE &.IOI.CI

    AFTER:

    INCLUDE &.APLICATN.CI
  IFNE  \FHS$IOS$
    MSG
    MSG          **************************************************
    MSG          **  System I/O Initializer
    MSG          **************************************************
    INCLUDE &.IOI.CI
```

e. VME110.CNFGDRVR.CI

   This file must be modified by the user to select, via switch settings, the drivers required for the user application. With each <system>.CNFGDRVR.CI file there are certain default driver configurations. If the user does not modify this file, drivers not applicable to the application are included in the ROM product and never used.

4. Copy the user-created application-related files to the target **SYSGEN** user number.

5. After examining the file STD.SYSGEN.CF, establish the appropriate parameter and invoke the **SYSGEN** process using this chainfile.

6. The <system>.VERSADOS.SY created by the **SYSGEN** process can now be burned into ROM using the selected ROM chips and module.

7. Invocation procedures for the ROM product are:

   a. Press the **RESET** button.

   b. Set the program counter to the SYSTEM STARTUP ADDRESS established at the end of the system listing generated by **SYSGEN** (refer to Appendix F).

   c. Enter **G** followed by a carriage return.

   d. The application is now set up and ready to run.

This procedure works on any system. However, the user may want control transferred directly to the user application when the **RESET** button is pressed. This is done by replacing the bug PROMs with a set of dummy PROMs that contain two long words. The two long words contain the starting system stack address and the SYSTEM STARTUP ADDRESS, respectively. The starting system stack address is found in the VME110.RMS.CI file.

4.2 PASCAL ROM CONSIDERATIONS

Minor modifications were made to the Pascal version 2.3 run-time library routines to support the ROM capability. The standard Pascal run-time library will not support ROM. The modified modules are identified by the catalog of "RROM".

4.2.1 Pascal Initializer

The Pascal initializer was removed from the library and is now linked with the Pascal task at **SYSGEN** time. It is no longer position independent and forces the data segment associated with the Pascal task to be dynamically obtained. A Relinquish directive is issued to ensure that FHS and IOS have completed their initialization before accessing their services. Access of a module is also done by the initializer to assign logical units 5 and 6 as required by the Pascal task.

4.2.2 ROM Libraries

Two ROM shareable libraries have been created; one that supports floating point processing (RROM.RLIBFP.RO), and one with no floating point processing (RROM.RLIBNFP.RO). The library that contains floating point is about 30Kb in length, while the library without floating point is about 10Kb in length. The chainfiles RROM.RLIBFP.CF and RROM.RLIBNFP.CF can be used to build the floating point and non-floating point libraries.

4.2.3 Module RROM.ASSIGNLU.SA

4.2.3.1 Logical Units 5 and 6 Assignment. The Pascal compiler treats "INPUT" and "OUTPUT" references on the PROGRAM statement in a special way. Typically "INPUT" refers to the terminal keyboard and "OUTPUT" refers to the terminal screen. These logical unit assignments are normally handled by the Exit/Entry Task (EET). EET has been removed from the system and the module RROM.ASSIGNLU.SA has been created to do this assignment.

This module contains instructions as well as examples on how to modify the command line for terminal assignments.

4.2.3.2 Pascal Command Line. The RROM.ASSIGNLU.SA module also has a minimum command line that is ";Z=1" followed by a carriage return. This command line forces Pascal to obtain a new data segment dynamically for the Pascal task. It can also be used to specify device names or other options to allow run-time flexibility. The following example shows a Pascal task using the minimum command line:

<u>UNALTERED COMMAND LINE USAGE</u>

```
PROGRAM ptask(output);
VAR i       :  integer;
VAR a,b,c   :  real;
devpr       :  text;

BEGIN
rewrite (devpr,'#PR');
i := 1;

WHILE i <> 10  DO
  BEGIN
  writeln (devpr,' MOTOROLA MOTOROLA MOTOROLA ');
  i := i+1;
  END;

i := 1;
a := 3.14159;
b := 2.71828;

WHILE 1=1  DO
  BEGIN
  writeln ( i, '  MOTOROLA MOTOROLA MOTOROLA ',c: 10:5);
  i := i+1;
  b := b+0.035;
  c := b*a;
  END;

END.
```

An altered command line using "#PR,;Z=1" followed by a carriage return changes the "PROGRAM" line and the "REWRITE" line. An example of a Pascal task using this task is:

<u>ALTERED COMMAND LINE USAGE</u>

```
PROGRAM ptask(output,devpr);
VAR i       :  integer;
VAR a,b,c   :  real;
devpr       :  text;

BEGIN
rewrite (devpr);
i := 1;

WHILE i <> 10  DO
  BEGIN
  writeln (devpr,' MOTOROLA MOTOROLA MOTOROLA ');
  i := i+1;
  END;

i := 1;
a := 3.14159;
b := 2.71828;

WHILE 1=1  DO
  BEGIN
  writeln ( i, '  MOTOROLA MOTOROLA MOTOROLA ',c: 10:5);
  i := i+1;
  b := b+0.035;
  c := b*a;
  END;

END.
```

The RROM.ASSIGNLU.SA source module has a detailed description of how these functions can be used.

4.2.4 Shareable Run-Time ROM Library

The library link modules establish the shareable run-time ROM library. Both a floating point and a non-floating point version of easily modified standard link chainfiles are provided. The floating point modules are RROM.TASKFP.LG and RROM.RLIBFP.LG while the non-floating point modules are RROM.TASKNFP.LG and RROM.RLIBNFP.LG. The task link modules include the selected shareable run-time ROM library so that external references can be satisfied, but the segment containing the ROM library has been excluded from the task link at **SYSGEN** time. Refer to Appendix E for the segment exclusion. See Figure 4-1 for the standard link modules.

RROM.RLIBFP.LG

```
=/*
=/*
=/*     RROM.RLIBFP.LG  Chainfile to link globally shareable Pascal
=/*                     run-time routines.  These  run-time routines
=/*                     INCLUDE floating point.
=/*
=/*
=LINK ,RROM.RLIBFP.LO,\LINKLS;HAMIXSZ=100
SEG SEG0(RG):8  \GSPLSTR
INPUT RROM.RLIBFP.RO
END
```

RROM.TASKFP.LG

```
=/*
=/*      RROM.TASKFP.LG  Chainfile to  link  globally shareable Pascal
=/*                      run-time routines to a user task.  The run-time
=/*                      routines INCLUDE floating point.
=/*
=/*      The following SYSGEN link file can be used by the user by
=/*      changing only the name of the applications task '????' where
=/*      referenced.  NOTE that the library modules are 'INCLUDEd',
=/*      not 'LIBed'.  This is necessary to properly satisfy external
=/*      references with a shared library.
=/*
=/*
=LINK ,&.????.LO,\LINKLS;HAMIXSZ=100
SEG PROG(R):9 \PC
SEG SEG0(RG):8 \GSPLSTR
SEG SEG2(R):15
IN      &.????.RO
IN   RROM.INIT.RO
IN   RROM.ASSIGNLU.RO
IN   RROM.RLIBFP.RO
END
=END
```

RROM.RLIBNFP.LG

```
=/*
=/*
=/*      RROM.RLIBNFP.LG  chainfile to link globally shareable Pascal
=/*                       run-time routines.  These run-time routines
=/*                       DO NOT INCLUDE floating point routines.
=/*
=/*
=LINK ,RROM.RLIBNFP.LO,\LINKLS;HAMIXSZ=100
SEG SEG0(RG):8  \GSPLSTR
INPUT RROM.RLIBNFP.RO
END
```

FIGURE 4-1. Pascal Task ROM-Related Link Files

```
RROM.TASKNFP.LG

=/*
=/*      RROM.TASKNFP.LG  chainfile to link globally shareable Pascal
=/*                       run-time routines to a user task.  The run-
=/*                       time routines DO NOT INCLUDE floating
=/*                       point.
=/*
=/*      The following SYSGEN link file can be used by the user by
=/*      changing only the name of the applications task '????' where
=/*      referenced.  NOTE that the library modules are 'INCLUDEd',
=/*      not 'LIBed'.  This is necessary to properly satisfy external
=/*      references with a shared library.
=/*
=/*
=LINK ,&.????.LO,\LINKLS;HAMIXSZ=100
SEG PROG(R):9 \PC
SEG SEGO (RG):8 \GSPLSTR
SEG SEG2(R):15
IN      &.????.RO
IN   RROM.INIT.RO
IN   RROM.ASSIGNLU.RO
IN   RROM.RLIBNFP.RO
END
=END
```

FIGURE 4-1. Pascal Task ROM-Related Link Files (cont'd)

4.2.5 Floating Point Modules

Floating point modules have been modified to remove references to external definitions defined at compile time. For the non-floating point library, there is a reference to an external definition in a floating point module. To satisfy this reference, the dummy module RROM.FPPOINT.SA was created. This module, which should not be referenced because the user application does not use floating point processing, contains the required external definition and an illegal instruction for diagnostic purposes.

4.3 ADDITIONAL PROCEDURES FOR THE PASCAL USER

In addition to the general ROM procedures, the Pascal user should also:

a. Copy RROM.ASSIGNLU.SA and RROM.ASSIGNLU.AF to the **SYSGEN** target user number. Modify as required and assemble RROM.ASIGNLU.SA, using the RROM.ASSIGNLU.AF chainfile.

b. Decide which run-time shareable ROM library to use, floating point or non-floating point. If floating point is selected, copy RROM.RLIBFP.RO and RROM.TASKFP.LG to the target **SYSGEN** user number. If non-floating point is selected, copy RROM.RLIBNFP.RO and RROM.TASKNFP.LG. to the target **SYSGEN** user number. The user should modify the ".LG" file by changing "????" to the Pascal taskname (see Figure 4-1).

c. Build the Pascal application **INCLUDE** file into the **SYSGEN** target user number using the selected floating point or non-floating point link files. The application **INCLUDE** file in Appendix E can be used as a guide.

d. Copy RROM.INIT.RO to the target **SYSGEN** user number.

e. Copy the Pascal application-related files to the target **SYSGEN** user number.

APPENDIX A

VERSAdos - I/O RELATED CONTROL BLOCKS AND TASKS

A.1 USING THE SYSTEM MAP

The last thing in the **SYSGEN** print file is a map of the generated system. It shows the various tasks and processes present in the system when booting is complete, and the initial address where execution is to start. Appendix C shows a typical **SYSGEN** map. Referring to that example, the column "TASK" shows the various tasks in the system at the time it is booted. The TCB column shows the address in memory of the TCB for the task. After the system is booted, a task's state and priority is controlled by the **SYSGEN STATE** and **PRIORITY** parameters.

Each task can be made up of one or more segments, whose names are shown in the "SEG" column and whose addresses are shown in the "ADDR" column. The segment names are those given by the linkage editor when building the load module used to create the task.

Some load modules create processes instead of tasks. A process is a collection of instructions and data that is not represented by a TCB. Two processes are shown -- RMS and SYSINIT. RMS is RMS68K, which serves as the VERSAdos kernel. SYSINIT is system initialization, entered as soon as the entire system is booted. Note that the startup address is the beginning of SYSINIT. When SYSINIT is completed, it JUMPs to the dispatcher to start system processing. The memory used by SYSINIT is not allocated to any task and becomes available for system use as soon as SYSINIT is completed.

The addresses shown in the "ADDR" column are the actual addresses where the system is loaded as long as WHERLOAD is set to zero. For a VM01 system, WHERLOAD is set to load the system into offboard RAM. However, the first thing SYSINIT does is to relocate the loaded system into onboard RAM, making the addresses match unless the **R** option (ROMable) was specified on the **SYSGEN** command line. SYSINIT then continues in the relocated code.

A.2 THE .IOI TASK

When INT completes, and turns control over to the dispatcher, the highest priority ready task is .IOI, the I/O initializer. It is made up of two segments: .IOI and IOSG. Entry is into module IOI (part of segment .IOI), which sets the I/O system in motion. Its processing is broken down into four steps:

a. Allocate all channels. This uses the Channel Data Blocks (CDBs) assembled in SECTION 1 of the IOC assembly, and included as part of the .IOI segment. The macro CDB generates the Channel Control Blocks (CCBs). CDBs are only needed during I/O initialization.

b. Calculate, and save for step d, various data lengths. These include the data segment for FMS for its stack, Volume Descriptor Tables (VDTs), File Control Blocks (FCBs), and File Assignment Tables (FATs).

c. Declare segment IOSG shareable.

d. For each I/O system task:

   1. Grant shared access of segment IOSG to the task.

   2. Allocate an Asynchronous Service Queue (ASQ) for the task if requested.

   3. If a data segment is requested, get a data segment for the task, clear it, declare it shared, and transfer it to the task.

   4. Start the I/O system task according to the startup priority task (make it ready to run).

Task .IOI then terminates. Segment .IOI disappears with the task termination, but IOSG remains because it has been made shareable and is now being used by the various I/O tasks.

A.3 THE IOSG SEGMENT

IOSG is section 0 of assembly XIOC (or XVMIOC). It starts out as part of task .IOI, but is given to all the I/O tasks (as described in the previous subsection) before .IOI completes. At label IOCOMS in the assembly is a table of pointers and values. This table is at the beginning of the module and is called the System Value Table (SVT). Two sets of three pointers are of interest here:

OFFSET	POINTER
$10	Start of Logical Unit Table (LUT) space
$14	End of LUT space
$18	First LUT in chain of active LUTs
$1C	Start of Device Control Block (DCB) space
$20	End of DCB space
$24	First DCB in chain of active DCBs

To verify that it is the right memory location, check offset $2C. This should contain VERSADOSREV.

A.4 THE LOGICAL UNIT TABLE

Each task has an associated LUT. Active LUT entries are in a chain whose head is at the SVT+$18. **SYSGEN** reserves enough space for a LUT for each task (NOTASKS), with each table having room for information about one more than the maximum number of logical units available to each task (MAXLU). Logical unit zero is reserved for system use.

Each LUT consists of a 16-byte long header followed by multiple 8-byte entries. Each 8-byte entry corresponds to one possible logical unit. Figure 1 shows the format of the LUT.

SYMBOL	OFFSET	LENGTH	FIELD
LUTPTR	0 ($0)	4	Pointer to next table
LUTTID	4 ($4)	4	Taskname
LUTSES	8 ($8)	4	Task session
LUTMLU	12 ($C)	1	Maximum number of LU entries
LUTCAS	13 ($D	1	Number of current assignments
LUTUNM	14 ($E)	2	User number
LUTBEG	16 ($10)	0	Start of LU entries
			LU entry first
LUTCAP	16 ($10)	1	Current access permission
LUTCSF	17 ($11)	1	Current status flag
LUTATT	18 ($12)	2	Attributes of device/file
LUTDCB	20 ($14)	4	Address of connected DCB/FCB
.	. .	.	.
.	. .	.	.
.	. .	.	.

- Current access permission (LUTCAP)

Symbol	Value	Meaning
FOPPR	0	Public read
FOPER	1	Exclusive read
FOPPW	2	Public write
FOPEW	3	Exclusive write
FOPPRPW	4	Public read-write
FOPPREW	5	Public read, exclusive write
FOPERPW	6	Exclusive read, public write
FOPEREW	7	Exclusive read-write

- Current status flag (LUTCSF)

Symbol	Bit	Meaning
LUSFAC	0	Active LU entry
LUSFIO	1	I/O pending
LUSFCP	2	Close pending
LUSFAS	3	Assign pending
LUSFCW	4	Connection wait
LUSFDV	7	Device assignment

- Attributes of device/file (LUTATT) - same as DCBATT

FIGURE 1. Format of the Logical Unit Table

The LUT for a particular session and task can be found by following the chain of active LUTs, starting with the first one (pointed to by SVT+$18), and continuing by using the link pointer at offset $0 in the LUT. Look for the proper taskname and session number. When it is found, the entries for each logical unit can be examined. Unassigned logical units contain zero in the LUTDCB field. Also, bit LUSFAC of byte LUTCSF is zero. The current status of other active entries can be checked in byte LUTCSF.

Bit LUSFDV indicates whether the LUT represents a file assignment or a device assignment. If it is on, it represents a device assignment. The field LUTDCB points to a DCB for a device assignment and to a File Control Block (FCB) for a file assignment.

A.5 COMMUNICATION BETWEEN USER TASKS AND THE I/O SYSTEM

User tasks request service from the I/O system by using either a TRAP #2 or a TRAP #3. TRAP #3 requests service from File Handling Services (FHS), which runs as task .FHS. FHS handles file and device manipulation, such as allocation, assignment, and renaming. Refer to the VERSAdos Data Management Services and Program Loader User's Manual for details on the types of requests available.

TRAP #2 requests I/O operations on files or devices. Input/Output Services (IOS), which runs as task .IOS, handles these requests. Refer to the VERSAdos Data Management Services and Program Loader User's Manual for details on types of requests available.

These two traps are passed to FHS and IOS because the tasks declare themselves handlers of the traps when execution starts. After initialization, all execution in FHS and IOS occurs when issuance of the appropriate trap causes RMS68K to place a user/server event (code 7) in their event buffer.

APPENDIX B

HARDWARE AND SOFTWARE CONFIGURATION

B.1 DESCRIPTION OF DEVICE CONFIGURATIONS

In most instances, it is sufficient for the user to alter the parameters in the **SYSGEN** configuration switch file. However, occasionally it may be necessary to make changes to the Device Control Blocks (DCBs) or Channel Data Blocks (CDBs). A DCB is provided for each device generated in the system and contains device dependent information about each device. CDBs contain information regarding each channel that is used to build the Channel Control Block (CCB) when the channel is allocated. Information regarding the DCBs and CDBs is found in the files IOC.<driver name>.AG, where the driver name is MxxxDRV for VME boards (e.g., M320DRV) or <chip_name>DRV for chip drivers (e.g., ACIADRV). A **DIR IOC.*.AG** command lists all drivers available in the **SYSGEN** account number. The DCBs and CDBs are configured automatically according to the **SYSGEN** parameters and become part of a common I/O segment accessible to all I/O tasks.

It is also possible to reconfigure certain device parameters and attributes without re**SYSGEN**ing the system by using the configuration utility, **CONFIG**. The types of devices that may be reconfigured are:

a. Terminals
b. Magnetic tape drives
c. Printers

The number of each type of device that may be configured by **CONFIG** is limited by the number allowed by the current VERSADOS.SY file (these quantities are determined by examining the <system>.CNFGDRVR.CI file from which VERSADOS.SY was **SYSGEN**ed). If more devices are to be added to the system, the <system>.CNFGDRVR.CI file must be altered and a **SYSGEN** performed to create a new VERSADOS.SY. Refer to the M68000 Family VERSAdos System Facilities Reference Manual and the VERSAdos to VME Hardware and Software Configuration User's Manual for more details.

Assumptions made about hardware configurations and device mnemonics are:

a. If there are any Multi-Channel Communications Modules (MCCMs) in the system, the physical address of the first board is assumed to be $FF1000, and each additional MCCM is assigned to physical address $200 bytes greater than the previous. The device mnemonics corresponding to the terminals on the first MCCM are CN10, CN11, CN12, and CN13. Mnemonics for terminals on the second MCCM are CN2x, etc.

b. Remote terminals interfaced through the MVME400 module and printers interfaced through the MVME410 module have dynamic addresses via the jumper capability. The general format of these addresses is $F80JJA and $F80JJB where "J" implies that these digits depend on the jumper placement and the digits "A" and "B" represent Ports A and B, respectively. The addresses for all I/O Channel devices are defined in IOC.ADDRESS.CI as offsets from the I/O Channel Base Address (IOCBASE) as defined in <system>.SYSTEM.CI. The addresses for the boards on the short I/O address space are defined in SIO.ADDRESS.CI as offsets from the Short I/O Base Address (SIOBASE). They are defined as channel addresses rather than bus addresses so that there is a direct correspondence to the addressing information in the board manual.

c. Device mnemonics for printers are dependent on the location of printers specified at **SYSGEN**. The names generated are:

PR, PR1, PR2, PR3, and PR4

The order in which names are defined is:

EXORmacs

Local printer	=	PR
1st MCCM	=	PR1
2nd MCCM	=	PR2
3rd MCCM	=	PR3
4th MCCM	=	PR4

VMC 68/2

1st MVME410 parallel port	=	PR
2nd MVME410 parallel port	=	PR1
1st MCCM	=	PR2

VME/10

1st MVME410 parallel port	=	PR
2nd MVME410 parallel port	=	PR1

If **SYSGEN** specifies printers, the default device mnemonics are PR, PR1, PR2, PR3, PR4, respectively, and are assigned as needed (refer to the table above).

d. All printers are software-configured as low-speed printers. If the spooler task is running in the system, data directed to a printer is directed to a spooler file first and then output to the printer.

e. The device address of the first Floppy Disk Controller (FDC), Universal Disk Controller (UDC), or VM22 Intelligent Peripheral Controller (IPC) is assumed to be $FF0000. Each subsequent FDC, UDC, or VM22 IPC has an address $200 bytes greater than the previous. If only floppy disk drives on FDCs are configured in the system, they should be configured at addresses $FF0000 and $FF0200. The same configuration would be used if only UDCs or VM22 IPCs are used. However, for a combination of FDCs, UDCs, and/or VM22 IPCs the boards should be alternated, with a UDC being assigned to $FF0000.

   The device mnemonics for disk drives are determined as follows:

   1. The first two characters are:

      HD (if hard disk).
      FD (if floppy disk).

   2. The third character indicates the FDC or UDC controller board number (0-3).

   3. The fourth character indicates the device number within the FDC (0-3) or UDC (0-7), or VM22 IPC (0-B).

   Floppy disk drives configured on a UDC begin with a device number of four, regardless of the number of hard disk drives also configured on the UDC. Thus, if a system were configured with four hard disks on a UDC and four floppy disks on an FDC, the device address of the hard disk would be $FF0000 and that of the floppy would be $FF0200. The device mnemonics would be HD00, HD01, HD02, HD03, FD10, FD11, FD12, and FD13. If all disk drives were configured on a UDC, the device address would be $FF0000, and the device mnemonics would be HD00, HD01, HD02, HD03, FD04, FD05, FD06, and FD07. Floppy disk drives configured on a VM22 IPC begin with a device number of 8, regardless of the number of hard disk drives also configured on the VM22 IPC. Thus, if a system were configured with four hard disks and four floppy disks on a VM22 IPC, the device address would be $FF0000, and the device mnemonics would be HD00, HD01, HD02, HD03, FD08, FD09, FD0A and FD0B.

f. The device address of the first MVME315 is assumed to be $FF0000. Each subsequent MVME315 has an address of $200 bytes greater than the previous.

   The device mnemonics for disk drives are determined as follows:

   1. The first two characters are:

      HD (if hard disk).
      FD (if floppy disk).

g. The device addresses of RWIN1 and MVME420 are determined by jumper selection on board.

The device mnemonics for disk drives configured on the RWIN1 and MVME420 are determined as follows:

1. The first two characters are:

   HD (if hard disk).
   FD (if floppy disk).

2. The third character indicates the controller type.

3. The fourth character indicates the device number.

Hard disks always begin with a device number of 0, and floppy disks always begin with a device number of 2. Thus, if a system were configured with two hard disks and two floppy disks on a RWIN1, and one hard disk and two floppy disks on a MVME420, the device mnemonics would be HD00, HD01, FD02, FD03, HD20, and FD22.

h. The device address of MVME435A is determined by jumper selection on the board.

The device mnemonics for tape drives are determined as follows:

1. The first two characters are MT.

2. The third character indicates the board number (0, 1, 2, ...).

3. The fourth character indicates the device number on that board (0, 1, 2, 3, 4, 5, 6, 7) (see following figure).

```
  +-------------------+
  | MVME110           |
  +-------------------+
         |
+---------+
|         |           +---------------------------------+
|   +-------------+   |   +-----------+                 |
|   |  First Mag  |   |   |           |                 |                +--+
|   |    Tape     |---|---| Formatter |-----+       +---|--//----+    +--|  |
|   |   Adapter   |   |   |    #0     |     |       |   |        |    |  +--+
|   |   MVME435A  |   | +-|           |   +-----------+ |    +-----------+
|   +-------------+   | | +-----------+   | Transport | |    | Transport |
|        MTA0         | |                 |    #0     | |    |    #3     |
|                     | |                 +-----------+ |    +-----------+
|                     | |    #MT00                      |         #MT03
|                     +-|-------------------------------+
|                       |
|                     +-|-------------------------------+
|                     | | +-----------+                 |
|                     | | |           |                 |                +--+
|                     | +-| Formatter |-----+       +---|--//----+    +--|  |
|                     |   |    #1     |     |       |   |        |    |  +--+
|                     |   |           |   +-----------+ |    +-----------+
|                     |   +-----------+   | Transport | |    | Transport |
|                     |                   |    #0     | |    |    #3     |
|                     |                   +-----------+ |    +-----------+
|                     |    #MT04                      |         #MT07
|                     +---------------------------------+
+---------+
          |           +---------------------------------+
    +-------------+   |   +-----------+                 |
    |  Second Mag |   |   |           |                 |                +--+
    |    Tape     |---|---| Formatter |-----+       +---|--//----+    +--|  |
    |   Adapter   |   |   |    #0     |     |       |   |        |    |  +--+
    |   MVME435A  |   | +-|           |   +-----------+ |    +-----------+
    +-------------+   | | +-----------+   | Transport | |    | Transport |
         MTA1         | |                 |    #0     | |    |    #3     |
                      | |                 +-----------+ |    +-----------+
                      | |    #MT10                      |         #MT13
                      +-|-------------------------------+
                        |
                      +-|-------------------------------+
                      | | +-----------+                 |
                      | | |           |                 |                +--+
                      | +-| Formatter |-----+       +---|--//----+    +--|  |
                      |   |    #1     |     |       |   |        |    |  +--+
                      |   |           |   +-----------+ |    +-----------+
                      |   +-----------+   | Transport | |    | Transport |
                      |                   |    #0     | |    |    #3     |
                      |                   +-----------+ |    +-----------+
                      |    #MT14                      |         #MT17
                      +---------------------------------+
```

MAGNETIC TAPE DRIVE HARDWARE CONFIGURATION

B.2 DEFINITION OF DCBs AND CDBs

To modify the standard device configuration, an understanding of the DCBs' and CDBs' formats is necessary. Macros in the files MACRO.*.* have been defined to build the DCBs and CDBs. The DCBs are built using macros that contain device-independent data followed by the device-dependent data. The macro defining the device-independent data is common to all DCBs. The CDB macro defines a channel data block that is used to allocate channels.

The information required by the macro to build the appropriate DCB or CDB is:

DIPDCB MACRO	This macro defines the device-independent portion of a DCB and is used by the disk macro DSKDCB, the terminal macro CRTDCB, the mag tape macro MTADCB, and the printer macro PRTDCB.

PARAMETER NUMBER	LENGTH IN BYTES	DESCRIPTION
\1	4	This parameter is set to the length of the DCB being generated. It is used to build the address of the next DCB in the linked list. The DCB length is represented respectively by the equate CDCBLN, DDCBLN, PDCBLN for the terminal, disk, and printer. A value of zero implies the end of the linked list.
\2	4	This field contains the device mnemonic for this DCB (ASCII).
\3	4	This field contains the taskname of the driver to which this DCB belongs.
\4	4	This field contains the session number of the driver to which this DCB belongs.
\5	2	This field contains attributes of the device associated with this DCB.

BIT	ATTRIBUTE
0	Supports Read
1	Supports Write
2	Supports Binary
3	Supports Random
4	Supports Image
5	Supports Halt-I/O
6	Supports Position Record
7	Supports Filemark
8	Interactive Device
9	Printer Device
10	Supports Spooling
11	Supports Write with Cyclic Redundancy Check (CRC)
12-15	Reserved

PARAMETER NUMBER	LENGTH IN BYTES	DESCRIPTION
\6	1	This field contains a decimal code identifying the type of device and is maintained for backward compatibility only.

Hex	Code	Device
$1E	30	Interactive terminal on IPC interface (Motorola EXORterm 155 or VME/10)
$1F	31	Interactive terminal on IPC interface (Non-EXORterm 155 or non-VME/10)
$23	35	Interactive terminal on local driver (Motorola EXORterm 155 or VME/10)
$24	36	Interactive terminal on local driver (Non-EXORterm 155 or non-VME/10)
$3C	60	Magnetic tape
$5A	90	Low speed line printer on IPC
$5B	91	High speed line printer on IPC
$5F	95	Low speed line printer on local driver

To get information about a device, do a CONFIGURE/STATUS REQUEST (TRAP #2), which returns a channel type code field in the configuration status block. The device type code field displays whether the device is a terminal, printer, tape, floppy disk, or hard disk.

The attributes word and parameters provide detailed information about the device configuration. Refer to the VERSAdos Data Management Services and Program Loader User's Manual for more information.

PARAMETER NUMBER	LENGTH IN BYTES	DESCRIPTION
\7	1	This field contains the current device status.

BIT	MEANING
0	0 --> Device offline
	1 --> Device online
1	0 --> Device not write protected
	1 --> Device write protected
2	1 --> Device status has changed
3	1 --> Device busy for initialization
4	1 --> Device busy for configuration.
5	0 --> No timer to be cancelled for this device.
	1 --> Timer to be cancelled for this device.
6	1 --> Ignore timer event for this device.

PARAMETER NUMBER	LENGTH IN BYTES	DESCRIPTION
\8	4	This field contains the ASCII channel identifier associated with this DCB.
\9	1	This field contains the device number associated with the channel for this DCB.
\A	4	Address of supervisor DCB or session number if this is a supervisor DCB (for MVME300 GPIB drivers only).

CRTDCB MACRO This macro defines the DCB for a terminal.

PARAMETER NUMBER	LENGTH IN BYTES	DESCRIPTION
		Define the device-independent portion of the DCB.
DIPDCB		CDCBLN,\1,\2,\3,\4,\5,\6,\7,\8,0 (Calls DIPDCB macro with these ten arguments, \1 through \A.)
\9	2	This field contains the attributes mask for the device configuration.
\A	2	This field contains the parameters mask for the device configuration.
\B	2	This field contains the attributes word for the device configuration.
\C	2	This field contains the number of characters per line.
\D	4	This field contains the number of lines per page.
\E	4	This field contains the write time-out value for this device (0=no time-out).
\F	4	This field contains the read time-out value for this device (0=no time-out).
\G	1	This field contains the value for the XOFF character.
\H	1	This field contains the value for the XON character.
\I	1	This field contains the value for the BREAK equivalent character.
\J	1	This field contains the value for the discard output character.
\K	1	This field contains the value for the reprint line character.
\L	1	This field contains the value for the cancel line character.
\M	4	This field contains the read I/O terminators for this device.
\N	4	This field contains the end-of-line string for this device.
\O	1	This field contains the baud rate code for this device.

PARAMETER NUMBER	LENGTH IN BYTES	DESCRIPTION
\P	1	This field contains the number of NUL characters to use for padding.
\Q	1	This field contains parameters to terminate read I/Os for a class of characters.
\R	1	This field contains the terminal type code.

Code	Terminal Type
0	EXORterm 155, direct connect.
\$1-\$7F	Reserved for future use. Use a value in this range for non-EXORterm 155 or modem.

DSKDCB MACRO This macro defines the DCB for a disk.

PARAMETER NUMBER	LENGTH IN BYTES	DESCRIPTION
		Define the device-independent portion of the DCB.
DIPDCB		DDCBLN,\1,\2,\3,\4,\5,\6,\7,\8,0 (Calls DIPDCB macro with these ten arguments, \1 through \A.)
\9	2	This field contains the attributes mask for the device configuration.
\A	2	This field contains the parameters mask for the device configuration.
\B	2	This field contains the attributes word for the device configuration.
\C	2	This field contains the number of bytes per sector.
\D	4	This field will contain the total number of sectors for this device.
\E	4	This field contains the write time-out value for this device (0=no time-out).
\F	4	This field contains the read time-out value for this device (0=no time-out).
\G	1	This field contains the number of sectors per track.
\H	1	This field contains the number of heads for this device.

PARAMETER NUMBER	LENGTH IN BYTES	DESCRIPTION
\I	2	This field contains the number of tracks for this device.
\J	1	This field contains the value for the interleave factor.
\K	1	This field contains the spiral offset value in sectors.
\L	1	This field contains the physical sector size of the media.
\M	1	This field contains the starting head number for the drive.
\N	1	This field contains the number of cylinders on the drive.
\O	1	This field contains the precompensation cylinder number for the drive.
\P	1	This field contains the physical sectors per track on the drive.
\Q	1	This field contains the stepping rate code of the head on the drive.
\R	1	This field contains the reduced write current cylinder number for the drive.
\S	1	This field contains the Error Correction Code (ECC) data burst length for the drive.

PRTDCB MACRO — This macro defines the DCB for a printer.

PARAMETER NUMBER	LENGTH IN BYTES	DESCRIPTION
		Define the device-independent portion of the DCB.
DIPDCB		DDCBLN,\1,\2,\3,\4,\5,\6,\7,\8,0 (Calls DIPDCB macro with these ten arguments, \1 through \A.)
\9	2	This field contains the attributes mask for the device configuration.
\A	2	This field contains the parameters mask for the device configuration.
\B	2	This field contains the attributes word for the device configuration.
\C	2	This field contains the number of characters per line.
\D	4	This field contains the number of lines per page.
\E	4	This field contains the write time-out value for this device (0=no time-out).
\F	2	This field contains the logical line length.
\G	1	This field contains the end-of-line character.
\L-\S		(Refer to IODM.AG source.)

MTADCB MACRO This macro defines the DCB for a magnetic tape drive.

PARAMETER NUMBER	LENGTH IN BYTES	DESCRIPTION
		Define the device-independent portion of the DCB.
DIPDCB		MDCBLN,\1,\2,\3,\4,\5,\6,\7,\8,0 (Calls DIPDCB macro with these ten arguments, \1 through \A.)
	4	Space for status fields.
\9	2	Attributes mask.
\A	2	Parameters mask.
\B	2	Attributes word.
	2	Number of bytes/VERSAdos logical sector (not used).
	4	Total number of VERSAdos sectors on media (not used).
\C	4	Write time-out (system).
\D	4	Read time-out (system).
\E	1	Requested density for write from loadpoint.
\F	1	Number of read tries before error message.
\G	1	Number of write tries before erasing.
\H	1	Number of erasures before error message.
\I	4	Time-out for tape read.
\J	4	Time-out for space forward or space reverse.
\K	4	Time-out for rewind.
\L	4	Time-out for search forward or reverse for filemark.
	16	16 bytes reserved for future use.

CDB MACRO — This macro defines the channel data block macro inputs that are used to build the CCB.

PARAMETER NUMBER	LENGTH IN BYTES	DESCRIPTION
*+CDBLN		This field contains the address of the next CDB in the linked list.
\1	4	This field contains options for the Allocate command.
\2	4	This field contains the ASCII channel mnemonic.
\3	4	This field contains the channel type (refer to the VERSAdos to VME Hardware and Software Configuration User's Manual for the table of channel types).
\4	4	This field defines the maximum number of consecutive commands that can be interpreted with interrupts masked during a single I/O call (applies only to byte mode (nonIPC) channels).
\5	4	This field contains the physical address of the driver associated with the CCB to be created from this CDB.
\6	4	This field contains the supervisor channel mnemonic and is applicable only if bit 3 of the options is set.
\7	4	This field contains the physical address of the device in memory mapped I/O space.
\8	4	This field contains the number of bytes the device occupies in memory mapped I/O space.
\9	4	This field contains an auto vector (25-31) or a user vector (64-255) number.
\A	4	This field contains the hardware polling priority in the range of 1 to 7.
\B	4	This field contains the software priority in the range of 0-255, where 255 represents the highest priority.
\C	4	This field represents the segment count.
\D	4	This field contains the zero relative offset from the memory mapped I/O space where the first polling byte resides.
\E	4	This field is used with the polling test value to determine if the device caused the interrupt.

PARAMETER NUMBER	LENGTH IN BYTES	DESCRIPTION
\F	4	This field contains the polling test value.
\G	4	This field contains the zero relative offset from the base of memory mapped I/O where the reset byte resides.
\H	4	This field contains the value to reset, clear the interrupt, for the particular device.
\I	4	This field contains the zero relative offset from the memory mapped I/O space where the second polling byte resides.
\J	4	Reference E above.
\K	4	Reference F above.
\L	4	Reference G above.
\M	4	Reference H above.

B.3 MACRO EXAMPLES

Terminal DCB Example

The following input parameters:

```
    \1 CN10            \B TCP$ATW         \L $18
    \2 IOSID           \C $50             \M $DDE0000
    \3 IOSESS          \D $18             \N $D0A0000
    \4 $133            \E $DBBA0          \O $E
    \5 30              \F $DBBA0          \P $0
    \6 1               \G $13             \Q $0
    \7 CHAN_ID         \H $0              \R $0
    \8 0               \I $3
    \9 $06FE           \J $F
    \A $5FB3           \K $1A
```

```
IFGE    NTV30$1-1
CRTDCB  'CN10',IOSID,IOSESS,$133,30,1,'CHAN_ID',0,$06FE,$5FB3,TCP$ATW,$50,$18,
        $DBBA0,$DBBA0,$13,$0,$3,$F,$1A,$18,$DDE0000,$D0A0000,$E,$0,$0,$0
```

were used to create the following terminal DCB:

```
SECTION  0              === DCB SECTION ===
DIPDCB   CDCBLN,'CN10',IOSID,IOSESS,$133,30,1,'CHAN_ID',0,0
DC.L     *+CDCBLN       Address of next DCB in linked list.
DC.L     'CN10'         ASCII identification for this DCB.
DC.L     0              Address of Device Connection Queue (DCQ) entry.
DC.L     IOSID          Name of task making the request.
DC.L     IOSESS         Session of task making the request.
DC.L     0              Address of LUT.
DC.L     $133           Attributes of device associated with this DCB.
DC.W     0              Write/Read protect codes.
DC.W     0              'Device in use' flag.
DC.L     0              Write/Read counts.
DC.B     30             Device flag (device code).
DC.B     1              Device flag (device status).
DC.L     'CHAN_ID'      Channel identification.
DC.B     0              Device number associated with the channel.
DC.B     0              Task priority.
DC.L     0              Current record number.
DS.B     IOSBLN         Storage area for the Input/Output Control Block (IOCB)
                        being processed.
DC.L     0              Logical address of IOCB in user's address space.
DC.B     0              Configuration coordination flag (0 --> at defaults).
DC.B     0              Break count.
DC.L     0              Address of break service LUT.
DC.L     0              Break service address.
DC.L     0              Event claimer -- taskname.
DC.L     0              Event claimer -- session number.
DC.L     0              Address of supervisor DCB or session.
DC.L     0              Supervisor/subordinate DCB open count.
DC.L     0,0,0,0          Device-independent/dependent buffer zone.
DC.B     0,0,0,0          Space for status fields.
DC.W     $06FE            Attributes  mask.   Attributes mask is logically ANDed
                          with the attributes word to yield bits looked at.
DC.W     $5FB3            Parameters mask.
DC.W     TCP$ATW          Attributes word.
DC.W     $50              Number of characters/line.
DC.L     $18              Number of lines/page.
DC.L     $DBBA0           Write time-out (0=no time-out).
DC.L     $DBBA0           Read time-out (0=no time-out).
DC.B     $13              XOFF character (not applicable to EXORmacs).
DC.B     $0               XON character (not applicable to EXORmacs).
DC.B     $3               BREAK   equivalent   character   (not   applicable  to
                          EXORmacs).
DC.B     $F               Discard output character.
DC.B     $1A              Reprint line character.
DC.B     $18              Cancel line character.
DC.L     $DDE0000         Read terminators.
DC.L     $D0A0000         End-of-line string.
DC.B     $E               Baud  rate code ($E=9600).  The following codes may be
                          used to indicate the desired baud rate:
```

Code	Rate	Code	Rate
$00	50	$0A	2400
$01	75	$0B	3600
$02	110	$0C	4800
$03	134.5	$0D	7200
$04	150	$0E	9600
$05	300	$0F	19200
$06	600	$10-FF	Reserved
$07	1200		
$08	1800		
$09	2000		

```
DC.B     $0               NUL padding.
DC.B     $0               Terminator class.
DC.B     $0               Terminal type (0=EXORterm 155, Ø=any other type).
DC.B     0,0,0,0,0,       Internal use only.
         0,0,0,0,0
DC.B     0,0,0,0,0,0      Reserved.
ENDC
```

Printer DCB Example

The following input parameters:

```
    \1 PRTDV              \9 $0003
    \2 IOSID              \A $0023
    \3 IOSESS             \B PCP$ATW
    \4 $632               \C $84
    \5 91                 \D $42
    \6 1                  \E $1D4C0
    \7 CHAN_ID            \F $84
    \8 4                  \G $D
```

```
IFGE     NPV30$1-1
SET      $0            (PCP$ATW)
PRTDCB   PRTDV,IOSID,IOSESS,$632,91,1,'CHAN_ID',4,$0003,$0023,PCP$ATW,$84,
         $42,$1D4C0,$84,$D
```

were used to create the following printer DCB:

```
SECTION  0                      === DCB SECTION ===
DIPDCB   PDCBLN,PRTDV,IOSID,IOSESS,$632,91,1,'CHAN_ID',4,0
DC.L     *+PDCBLN               Address of next DCB in linked list.
DC.L     PRTDV                  ASCII identification for this DCB.
DC.L     0                      Address of DCQ entry.
DC.L     IOSID                  Name of task making the request.
DC.L     IOSESS                 Session of task making the request.
DC.L     0                      Address of LUT.
DC.L     $632                   Attributes of device associated with this DCB.
DC.W     0                      Write/read protect codes.
DC.W     0                      'Device in use' flag.
DC.L     0                      Write/read counts.
DC.B     91                     Device flag (device code).
DC.B     1                      Device flag (device status).
DC.L     'CHAN_ID'              Channel identification.
DC.B     4                      Device number associated with the channel.
DC.B     0                      Task priority.
DC.L     0                      Current record number.
DS.B     IOSBLN                 Storage area for the IOCB being processed.
DC.L     0                      Logical address of IOCB in user's address space.
DC.B     0                      Configuration coordination flag (0 --> at
                                defaults).
DC.B     0                      Break count.
DC.L     0                      Address of break service LUT.
DC.L     0                      Break service address.
DC.L     0                      Event claimer -- taskname.
DC.L     0                      Event claimer -- session number.
DC.L     0                      Address of supervisor DCB or session.
DC.L     0                      Supervisor/subordinate DCB open count.
DC.L     0,0,0,0                Device independent/dependent buffer zone.
DC.B     0,0,0,0                Space for status fields.
DC.W     $0003               Attributes mask.
DC.W     $0023               Parameters mask.
DC.W     PCP$ATW             Attributes word.
DC.W     $84                 Number of characters/line.
DC.L     $42                 Number of lines/page.
DC.L     $1D4C0              Write time-out.
DC.L     0                   Read time-out.
DC.W     $84                 Logical line length.
DC.B     $D                  End-of-line character.
DCB.B    15,0                Reserved space (15 bytes).
ENDC
```

Disk DCB Example

The following input parameters:

```
    \1 DSKNM              \B ATT_WORD          \L OFF
    \2 IOSID              \C $100              \M BYTES_PER_SECTOR
    \3 IOSESS             \D 0                 \N START_HEAD_NUM
    \4 DEV_ATT            \E $0                \O CYL_DRIVE
    \5 DEV_CODE           \F $0                \P PRE_COMP
    \6 DEV_STAT           \G SECT_PER_TRK      \Q SECT_DRIVE
    \7 CHAN_ID            \H NUM_HEADS         \R STEP_RATE
    \8 DSKNM&$F           \I CYL_DISK          \S R_W_PRE_COMP
    \9 ATT_MASK           \J INTERLEAVE        \T ECC_LEN
    \A PAR_MASK           \K SPIRAL

DSKDCB     DSKNM,IOSID,IOSESS,DEV_ATT,DEV_CODE,DEV_STAT,CHAN_ID,DSKNM&$F,
           ATT_MASK,PAR_MASK,ATT_WORD,$100,0,$0,$0,SECT_PER_TRK,NUM_HEADS,
           CYL_DISK,INTERLEAVE,SPIRAL_OFF,BYTES_PER_SECTOR,START_HEAD_NUM,
           CYL_DRIVE,PRE_COMP,SECT_DRIVE,STEP_RATE,R_W_PRE_COMP,ECC_LEN
```

were used to create the following disk DCB:

```
SECTION  0              === DCB SECTION ===
DIPDCB   DDCBLN,DSKNM,IOSID,IOSESS,DEV_ATT,DEV_CODE,DEV_STAT,CHAN_ID,
         DSKNM&$F,0
DC.L     *+DDCBLN         Address of next DCB in linked list.
DC.L     DSKNM            ASCII identification for this DCB.
DC.L     0                Address of DCQ entry.
DC.L     IOSID            Name of task making the request.
DC.L     IOSESS           Session of task making the request.
DC.L     0                Address of LUT.
DC.L     DEV_ATT          Attributes of device associated with this DCB.
DC.W     0                Write/Read protect codes.
DC.W     0                'Device in use' flag.
DC.L     0                Write/Read counts.
DC.B     DEV_CODE         Device flag (device code).
DC.B     DEV_STAT         Device flag (device status).
DC.L     'CHAN_ID'        Channel identification.
DC.B     DSKNM&$F         Device number associated with the channel.
DC.B     0                Task priority.
DC.L     0                Current record number.
DS.B     IOSBLN           Storage area for the IOCB being processed.
DC.L     0                Logical address of IOCB in user's address space.
DC.B     0                Configuration coordination flag (0 --> at defaults).
DC.B     0                Break count.
DC.L     0                Address of break service LUT.
DC.L     0                Break service address.
DC.L     0                Event claimer --taskname.
DC.L     0                Event claimer --session number.
DC.L     0                Address of supervisor DCB or session.
DC.L     0                Supervisor/subordinate DCB open count.
DC.L     0,0,0,0          Device-independent/dependent buffer zone.
DC.B     0,0,0,0           Space for status fields.
DC.W     ATT_MASK          Attributes mask.
DC.W     PAR_MASK          Parameters mask.
DC.W     ATT_WORD          Attributes word.
DC.W     $100              Number of bytes/sector.
DC.L     0                 Total number of sectors -- returned information.
DC.L     $0                Write time-out.
DC.L     $0                Read time-out.
DC.B     SECT_PER_TRK      Number of sectors/track.
DC.B     NUM_HEADS         Number of heads.
DC.W     CYL_DISK          Number of cylinders on media.
DC.B     INTERLEAVE        Interleave factor.
DC.B     SPIRAL_OFF        Spiral offset (in sectors).
DC.W     BYTES_PER_SECTOR  Physical sector size of media.
DC.W     START_HEAD_NUM    Starting head number on drive.
DC.W     CYL_DRIVE         Number of cylinders on drive.
DC.W     PRE_COMP          Precompensation cylinder number.
DC.B     SECT_DRIVE        Physical sectors per track on drive.
DC.B     STEP_RATE         Stepping rate.
DC.W     R_W_PRE_COMP      Reduced write current cylinder number.
DC.W     ECC_LEN           ECC data burst length.
DC.B     0,0               2 bytes reserved as offset to another parameter
                           block.
ENDC
```

CDB MACRO Example

The following input parameters:

```
    \1 $0000           \9 VECTNBR          \H 3
    \2 CNAME           \A VECTLVL          \I 0
    \3 CTYPE1          \B PRIORTY2         \J 0
    \4 254             \C 1                \K 0
    \5 ACIA            \D 0                \L 0
    \6 0               \E $80              \M 0
    \7 LTDA$02         \F $80
    \8 3               \G 0
```

```
CDB     $0000,CNAME,CTYPE1,254,ACIA,0,LTDA$02,3,VECTNBR,VECTLVL,PRIORTY2,
        1,0,$80,$80,0,3,0,0,0,0,0
```

were used to create the following CDB:

```
SECTION  1              === CDB SECTION ===
DC.L     *+CDBLN        Pointer to next CDB in list.
DC.W     $0000          Options for the Allocate command.
DC.L     CNAME          Channel mnemonic.
DC.B     CTYPE1         Channel type.
DC.B     254            Masked interrupt maximum instruction count.
DC.L     ACIA           Physical address of driver.
DC.L     0              Supervisor channel's mnemonic (only if bit 3 of
                        options is set).
DC.L     LTDA$02        Physical address of device in memory-mapped I/O space.
DC.W     3              Number of bytes device occupies in memory-mapped I/O
                        space.
DC.B     VECTNBR        Vector number.
DC.B     VECTLVL        Polling priority.
DC.B     PRIORTY2       Software priority.
DC.B     1              Segment count.
DC.W     0              Polling byte offset. -- [#1] --
DC.B     $80            Polling mask.
DC.B     $80            Polling test value.
DC.W     0              Offset from physical device address for reset.
DC.B     3              Value for reset.
DC.B     0              Reserved.
DC.W     0              Polling byte offset. -- [#2] --
DC.B     0              Polling mask.
DC.B     0              Polling test value.
DC.W     0              Offset from physical device address for reset.
DC.B     0              Value for reset.
DC.B     0              Reserved.

SET      DVCODE+1
ENDC
```

THIS PAGE INTENTIONALLY LEFT BLANK.

APPENDIX C

TYPICAL SYSGEN COMMAND FILES

Following is a listing of the **SYSGEN** command source files <system>.SYSTEM.CI, <system>.CNFGDRVR.CI, and <system>.IFDRVR.CI, as furnished for a VME/10 system with VERSAdos. Furnished **SYSGEN** files can be edited by the user to change parameters to reflect the desired configuration. The VMES10.CNFGDRVR.CI and VMES10.SYSTEM.CI files are designed to be executed with the chainfile &.SYSGEN.CF. **SYSGEN** is normally invoked from a chainfile. The following example is invoked from STD.SYSGEN.CF. Refer to the &.SYSGEN.CF chainfile for detailed instructions to invoke the **SYSGEN** process.

```
*
*        VMES10.SYSTEM.CI
*
*******************************************************************************
*  This file contains all board/system dependencies for VME/10.               *
*  It is called from "&.VERSADOS.CD".                                         *
*                                                                             *
*  The user should not have to modify this file except under extreme          *
*  circumstances.                                                             *
*******************************************************************************

*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*
*       ON BOARD RAM ADDRESSES
*
*       VMEbus drivers require the following target system dependent
*       variables to be defined when the macro file DUALPORT.MC is used:
*       Note that the ONBD$HI parameter may vary on the VME110 and
*       and VME101 systems depending on the size and number of RAM
*       chips used on the processor card.  The values shown are the
*       minimum requirements. Modify that parameter to match your own
*       system if different.
*
*
ONBD$LO    = $0           Low  value for on-board ram (as seen by driver)
ONBD$HI    = $5FFFF       High value for on-board ram (as seen by driver)
RAM$SG     = $D00000      Difference between on-board ram address as seen by
*                         driver and device.
OFFBD$LO   = \ONBD$LO+\RAM$SG Low  value for on-board ram (as seen by device)
OFFBD$HI   = \ONBD$HI+\RAM$SG High value for on-board ram (as seen by device)

MSG        ********************************************************************
MSG        **        Addresses of timer, etc.
MSG        ********************************************************************

TIMER      = $F1A081      Address of timer.
CLOCKFRQ   = 0            Number of clock ticks per millisecond
PANEL      = $0           Address of front panel.
TRCFLAG    = 0            Trace flag.  Zero implies don't trace.  The
*                         setting of bits in the TRCFLAG parameter will
*                         control which events cause an entry to be built
*                         in the trace table.
*                         Bit # in TRCFLAG      Event
*                            15                 TRAP #1
*                            14                 I/O interrupt not claimed
*                                                 by user task.
*                            13                 Timer interrupt.
*                            12                 User trap (2-15)
*                            11                 Exception
*                            10                 Dispatch
*                             9                 I/O interrupt claimed by
*                                                 user task
*                             8                 Return from LOADMMU
*                             7                 Simulated interrupt
*                             6                 SYSFAIL interrupt.
*
```

```
SYSFAIL    = 0              Determines whether or not the operating system will
*                           be interrupted when SYSFAIL is asserted on the bus.
*                           Some intelligent boards will assert SYSFAIL when
*                           they experience a failure of some kind.  If you
*                           have such boards in the system, AND THE DRIVERS FOR
*                           THESE BOARDS HAVE SYSFAIL HANDLERS, then you will
*                           probably want SYSFAIL interrupts enabled.  If the
*                           appropriate SYSFAIL handlers are not written, then
*                           taking a SYSFAIL interrupt will hang up the system,
*                           so you would want SYSFAIL interrupts disabled.
*                             0 = disable SYSFAIL interrupts.
*                             1 =  enable SYSFAIL interrupts.

MSG        ****************************************************************
MSG        **   Local terminal/printer device addresses
MSG        **   Short I/O space base address
MSG        **   Short I/O space address offsets
MSG        ****************************************************************

KBDOVRD    = 0              Keyboard override option:
*                           0 = keyswitch on front panel is enabled
*                           1 = keyswitch override - if key is in locked
*                               position the keyboard is still enabled.
*                           NOTE:  This parameter is only valid with VME/10
*                                  units with the panel keyswitch.

SIOBASE    = $FF0000        Base address of Short I/O space
INCLUDE    SIO.ADDRESS.CI get short I/O space addresses offsets

MSG
MSG        ****************************************************************
MSG        ** I/O Channel is on this CPU board.
MSG        ****************************************************************

CMULT      = 2              I/O channel multiplier (see IOC.ADDRESS.CI):
*                               = 1 for IOC run directly on the MC68020 or
*                                   MC68008 address bus of a user's custom CPU
*                                   board (because of byte addressing)
*                               = 2 for all others where byte address is not
*                                   available (only odd addresses are used) or
*                                   if running the IOC from a VME316 board
*
&IOCBASE   = $F1C000        Base address of I/O channel
INCLUDE    IOC.ADDRESS.CI get I/O channel addresses offsets

*************************************************************
**          Vector numbers                                 **
*************************************************************
*
IOCVEC1    = $41            I/O channel interrupt 1 vector.
IOCVEC2    = $43            I/O channel interrupt 2 vector.
IOCVEC3    = $44            I/O channel interrupt 3 vector.
IOCVEC4    = $45            I/O channel interrupt 4 vector.
*
IOCLVL1    = 2              I/O channel interrupt 1 level.
IOCLVL2    = 4              I/O channel interrupt 2 level.
IOCLVL3    = 5              I/O channel interrupt 3 level.
```

```
IOCLVL4    = 6             I/O channel interrupt 4 level.

MSG         ******************************************************************
MSG         **     Descriptive info about this operating system
MSG         ******************************************************************

TIMSLIC    = 2             Number of timer interrupts per time slice.
TIMINTV    = 16            Number of milliseconds between timer interrupts.
*                          (Not used on VME/10.  It is included for documentation
*                          only.  The actual value used for the VME/10 is
*                          15.625 msec.)
ROMEADDR   = $0            ROM end address -- defined by user for a ROMmable
*                              system.  Set to $0 if not a ROMable system.
*                              The ROM start address (ROMSADDR) is defined
*                              in the <system>.RMS.CI file and has a value equal
*                              to the initial program counter.
MEMEND1    = $2FF00        Ending addr for on-board  memory must be < this.
MEMEND2    = $2FF00        Starting  addr for off-board memory must
*                          be >= this.  Not applicable for a VME110
MEMEND3    = $280000       Ceiling addr for off-board memory (must be < this).
WHERLOAD   = $0            Memory address where boot file will be loaded
*                          Nonzero on VM01 only.  If nonzero, VERSAdos will
*                          be moved at initialization time.
CACHEF     = 0             Not used by VMES10 (needed by &.INITDAT.AG)
NOTNT      = 0             Not used by VMES10 (needed by EET.VERSADOS.CI)

MSG         ******************************************************************
MSG         **  Parameters about table sizes, etc.
MSG         ******************************************************************

PAGESIZE   = 256           Size in bytes of a page of memory.
*
ASN        = 127           # of address spaces

MSG         ******************************************************************
MSG         *   Copy C80C.SYSPAR.RO into &.SYSPAR.RO to make it generic
MSG         *   for .LG files and possible for sysgening more than 1 system
MSG         *   per account #
MSG         ******************************************************************

=COPY       CADE.SYSPAR.RO,&.SYSPAR.RO;Y
```

```
*
*         VMES10.CNFGDRVR.CI
*
* Configuration file for device drivers
*-------------------------------------------------------------------------------
* This file sets up the flags used by the "VMES10.IFDRVR.CI" file to
* conditionally include device drivers.
*
* The user should only have to modify this file to include/exclude drivers.
* If you add more boards/devices, you may have to increase the sysgen
* command T option for more symbols in the "&.SYSGEN.CF" file.
*
* To modify specific items of a driver, edit the corresponding driver file,
* "&.xxxxDRV.CI" except as noted where one driver handles multiple boards.
*-------------------------------------------------------------------------------

********************************************************************************
*---  BOARD/SYSTEM DEPENDENCIES are included from the &.VERSADOS.CD file.  ---
* VMES10.SYSTEM.CI   add processor board/system dependencies,
*                        including local terminals and printers
* &.CNFGTASK.CI      add O/S task configuration for ROM/RAM
********************************************************************************

*-------------------------------------------------------------------------------
NOLTERM   = 1          # of terminals on VMES10 serial ports (TERM); max= 1

*-------------------------------------------------------------------------------
NORWIN    = 1          # of RWIN1    Winchester controller boards
*
IFGT        \NORWIN
   CONTWIN1   = "0"          1st RWIN1 is controller 0
   NHRWIN$1   = 1            # of hard   disk drives on 1st RWIN1; max= 2
   NFRWIN$1   = 1            # of floppy disk drives on 1st RWIN1; max= 2
   RWIN0$1    = "'H5WIN15'"  Type of 1st hard   disk on 1st RWIN1, drive 0
   RWIN1$1    = "'H5WIN15'"  Type of 2nd hard   disk on 1st RWIN1, drive 1
   RWIN2$1    = "'F5DDDSI'"  Type of 1st floppy disk on 1st RWIN1, drive 2
   RWIN3$1    = "'F5DDDSI'"  Type of 2nd floppy disk on 1st RWIN1, drive 3

   CONTWIN2   = "1"          2nd RWIN1 is controller 1
   NHRWIN$2   = 0            # of hard   disk drives on 2nd RWIN1; max= 2
   NFRWIN$2   = 0            # of floppy disk drives on 2nd RWIN1; max= 2
   RWIN0$2    = "'H5WIN15'"  Type of 1st hard   disk on 2nd RWIN1, drive 0
   RWIN1$2    = "'H5WIN15'"  Type of 2nd hard   disk on 2nd RWIN1, drive 1
   RWIN2$2    = "'F5DDDSI'"  Type of 1st floppy disk on 2nd RWIN1, drive 2
   RWIN3$2    = "'F5DDDSI'"  Type of 2nd floppy disk on 2nd RWIN1, drive 3
   *
   *     NOTE:  You can not mix 5-1/4" and 8" floppies.  Pick one or the other.
   *
ENDC

*-------------------------------------------------------------------------------
NVME050   = 0          # of VME050   System Controller
IFGT        \NVME050
   NT050$1    = 0            # of terminals on the VME050 board; max= 2
   NP050$1    = 0            # of printers  on the VME050 board; max= 1
ENDC
```

```
*-----------------------------------------------------------------------------
NVME300   = 0            # of MVME300  IEEE 488 GPIB controller boards

*-----------------------------------------------------------------------------
NVME315   = 0            # of MVME315  winchester/floppy disk controller boards
IFGT        \NVME315
   CONT3151  = "3"          1st MVME315 is controller 3
   NH315$1   = 0            # of hard   disk drives on 1st VME315; max= 2
   NF315$1   = 0            # of floppy disk drives on 1st VME315; max= 4

   M3150$1   = "'H5WIN15'"  Type of 1st hard   disk on 1st MVME315 board
   M3151$1   = "'H5WIN15'"  Type of 2nd hard   disk on 1st MVME315 board
   M3154$1   = "'F8SDDSM'"  Type of 1st floppy disk on 1st MVME315 board
   M3155$1   = "'F8SDDSM'"  Type of 2nd floppy disk on 1st MVME315 board
   M3156$1   = "'F5DDDSI'"  Type of 3rd floppy disk on 1st MVME315 board
   M3157$1   = "'F5DDDSI'"  Type of 4th floppy disk on 1st MVME315 board

   CONT3152  = "4"          2nd MVME315 is controller 4
   NH315$2   = 0            # of hard   disk drives on 2nd VME315; max= 2
   NF315$2   = 0            # of floppy disk drives on 2nd VME315; max= 4

   M3150$2   = "'H5WIN15'"  Type of 1st hard   disk on 2nd MVME315 board
   M3151$2   = "'H5WIN15'"  Type of 2nd hard   disk on 2nd MVME315 board
   M3154$2   = "'F8SDDSM'"  Type of 1st floppy disk on 2nd MVME315 board
   M3155$2   = "'F8SDDSM'"  Type of 2nd floppy disk on 2nd MVME315 board
   M3156$2   = "'F5DDDSI'"  Type of 3rd floppy disk on 2nd MVME315 board
   M3157$2   = "'F5DDDSI'"  Type of 4th floppy disk on 2nd MVME315 board
ENDC

*-----------------------------------------------------------------------------
NVME316   = 0            # of MVME316  VMEbus to I/O channel interface boards;
*                        max= 1
**** NOTE:  Do not use VME316 in this system, as I/O channel is already on
****        the CPU board!  See "&.DRV316.CI" for details.

*-----------------------------------------------------------------------------
NVME320   = 0            # of MVME320  Winchester/floppy controller boards
IFGT        \NVME320
   CONT320   = "2"          MVME320 is controller 2
   NH320$1   = 0            # of hard   disk drives on VME320; max= 2
   NF320$1   = 0            # of floppy disk drives on VME320; max= 2

   M3200$1   = "'H5WIN15'"  Type of 1st hard   disk on 1st MVME320 board
   M3201$1   = "'H5WIN15'"  Type of 2nd hard   disk on 1st MVME320 board
   M3202$1   = "'F5DDDSI'"  Type of 1st floppy disk on 1st MVME320 board
   M3203$1   = "'F5DDDSI'"  Type of 2nd floppy disk on 1st MVME320 board
ENDC

*-----------------------------------------------------------------------------
NVME400   = 1            # of MVME400  dual 7201 serial port boards
IFGT        \NVME400
   NU400$1   = 2            # of ports/users on VME400 bd. #1; max= 2/bd.
   NU400$2   = 0            # of ports/users on VME400 bd. #2; max= 2/bd.
ENDC

*-----------------------------------------------------------------------------
NVME410   = 1            # of MVME410  dual 16-bit parallel port boards
```

```
IFGT         \NVME410
   NU410$1     = 1          # of printers in use on VME410 board #1; max= 2
   NU410$2     = 0          # of printers in use on VME410 board #2; max= 2
ENDC

*------------------------------------------------------------------------------
NVME435    = 0          # of MVME435 mag tape controller boards; max= 2
IFGT         \NVME435
   N435$1      = 0          # of tape drives on first  MVME435 board; max= 8
   N435$2      = 0          # of tape drives on second MVME435 board; max= 8
ENDC

*------------------------------------------------------------------------------
NVME600    = 0          # of MVME600  analog input  controller boards

*------------------------------------------------------------------------------
NVME605    = 0          # of MVME605  analog output controller boards
IFGT         \NVME605
   NU605       = 0          Number of users (total) for the MVME605 boards
ENDC

*------------------------------------------------------------------------------
NVME610    = 0          # of MVME610     AC  input  controller boards
IFGT         \NVME610
   M610QSIZ    = 128    Minimum number of entries in Interrupt Processing Queue
ENDC

*------------------------------------------------------------------------------
NVME615    = 0          # of MVME615/616 AC  output controller boards
IFGT         \NVME615
   NU615       = 0          Number of users (total) for the MVME615 boards
ENDC

*------------------------------------------------------------------------------
NVME620    = 0          # of MVME620     DC  input  controller boards

*------------------------------------------------------------------------------
NVME625    = 0          # of MVME625     DC  output controller boards
IFGT         \NVME625
   NU625       = 0          Number of users (total) for the MVME625 boards
ENDC

*------------------------------------------------------------------------------
NRAD       = 0          # of RAD      Remote A/D boards
IFGT         \NRAD
   NURAD       = 0          # of RAD users
ENDC

*------------------------------------------------------------------------------
NRIO       = 0          # of RIO      Remote I/O boards
IFGT         \NRIO
   NRIOINT     = 0          # of interrupt levels per I/O module
ENDC

*------------------------------------------------------------------------------
```

```
*
*        VMES10.IFDRVR.CI
*
* Conditional file for VMES10 device drivers
*-----------------------------------------------------------------------------------
* This file uses flags setup in the VMES10.CNFGDRVR.CI and VMES10.SYSTEM.CI
* files to conditionally include device drivers.
*
* The user should not have to modify this file to include/exclude drivers.
*
*    ******************************************************************
*    **  NOTICE:  The following conditionals are order dependent.    **
*    **           Do not change the order!  Local drivers 1st.       **
*    **           The order determines device number (HD00,PR, etc)  **
*    **           for most items.                                    **
*    ******************************************************************
*
*
*
IFNE        \NVME316
   *  The VME316 has no driver.  It has an initialization module that is
   *  merged into SYSINIT.RO.  The include file below defines the I/O channel.
   *
   INCLUDE     &.M316DEF.CI
ENDC
*
IFNE        \NORWIN
   INCLUDE     &.RWINDRV.CI
ENDC
*
IFNE        \NVME320
   INCLUDE     &.M320DRV.CI
ENDC
*
IFNE        \NVME315
   INCLUDE     &.M315DRV.CI
ENDC
*
IFNE        \NOLTERM
   INCLUDE     DRVS10.CI
ENDC
*
IFNE        \NVME400
   INCLUDE     &.MPSC400.CI
ENDC
*
IFNE        \NVME410
   INCLUDE     &.PIA410.CI
ENDC
*
IFNE        \NVME050
   INCLUDE     &.M050DRV.CI
ENDC
*
IFNE        \NVME300
   INCLUDE     &.M300DRV.CI
ENDC
```

```
IFNE          \NVME435
    INCLUDE       &.M435DRV.CI
ENDC
*
IFNE          \NVME600
    INCLUDE       &.M600DRV.CI
ENDC
*
IFNE          \NVME605
    INCLUDE       &.M605DRV.CI
ENDC
*
IFNE          \NVME610+\NVME620
    INCLUDE       &.M610DRV.CI
ENDC
*
IFNE          \NVME615
    INCLUDE       &.M615DRV.CI
ENDC
*
IFNE          \NVME625
    INCLUDE       &.M625DRV.CI
ENDC
*
IFNE          \NRAD
    INCLUDE       &.RADDRV.CI
ENDC
*
IFNE          \NRIO
    INCLUDE       &.RIODRV.CI
ENDC
```

THIS PAGE INTENTIONALLY LEFT BLANK.

APPENDIX D

DEFINITION OF SYSGEN PARAMETERS

This appendix contains an alphabetical listing of the **SYSGEN** parameters and their definitions. A copy is also on the released media under filename 0.&.SYSGEN.NW (& = null catalog) for online help capability using the CRT Text Editor.

D E F I N I T I O N   O F   S Y S G E N   P A R A M E T E R S

Parameter	Definition
&ADDRESS	Address of VME420 board
&CRTDV	Starting CRT Terminal Device Code
&DEFVOL	The number of default volumes for the system. Minimum is 4. (See NODEFVOL)
&DEVADD	Starting address of boards on Short I/O address space
&DSKDV	Starting Disk Device Code (1 less that __0_) (gets built into HD00, HD01,..., FD14, FD15,...)
&FILENAM	Parameter used to pass a .AG file a filename to include for the assembly.
&IOCBASE	Base address of I/O Channel
&M4200	Type of disk on drive 0 of VME420 board
&M4201	Type of disk on drive 1 of VME420 board
&M4202	Type of disk on drive 2 of VME420 board
&M4203	Type of disk on drive 3 of VME420 board
&M420FLG	Flag used to include M420DRV.RO If zero then M420DRV has not been included If non-zero then M420DRV has already been included
&MPSCFLG	Flag used to include MPSCDRV.RO If zero then MPSCDRV has not been included If non-zero then MPSCDRV has already been included
&NF420	The number of floppy disks on the VME420 board.
&NH420	The number of hard disks on the VME420 board.
&PCDRV	The number of Process Control Drivers This redefinable parameter is built as Process Control Drivers are included. (VME6xx, RAD, RIO)
&PIAFLAG	Flag used to include PIADRV.RO If zero then PIADRV has not been included If non-zero then PIADRV has already been included

&PRTDV	Starting Printer Device Code
&SDRVADD	Address of the supervisor driver for MPSC
&SDRVR	Name of the supervisor driver for MPSC
&SERFLAG	Flag used to define the number of serial port drivers This flag is used to determine if we need to have DRVLIB & TERMLIB. If zero then DRVLIB & TERMLIB do not need to be included If non-zero then DRVLIB & TERMLIB do need to be included
&SPRFLAG	Flag used to include MPSCSPR.RO If zero then MPSCSPR has not been included If non-zero then MPSCSPR has already been included
&SUPFLAG	Flag used to include MPSCSUP.RO If zero then MPSCSUP has not been included If non-zero then MPSCSUP has already been included
&T	Temporary parameter used to hold the value to determine whether the cache will be flushed or not.
&TOTDSK	Total number of disks (floppy & hard) This redefinable parameter is built as disks are included. Each volume defined requires approximately 2.25 bytes of memory.
&TOTPRT	Total number of printers This redefinable parameter is built as printers are included.
&TOTTERM	Total number of terminals This redefinable parameter is built as terminals are included.
&VECTNO	Beginning vector number for boards that are on the Short I/O address space.
ACIADRV	Address of the ACIA driver
ASMLS	ASM listing work file/device
ASMLSW	Assembly listing file switch 0 = ASMLS is a file 1 = ASMLS is a device
ASN	The number of address spaces
AUTOLOGN	Auto break/logon flag Bit 0: 0 = Auto break inactive 1 = Auto break active Bit 1: 0 = Auto logon inactive 1 = Auto logon active
AUTOTERM	Terminal ID of device autologon is to occur on.
BATCHPGE	Number of pages for batch job queueing. Each page

	accomodates 32 entries. In addition, there is space for 31 entries minus the number of terminals in the system.
BATDLY	Delay in milliseconds in batch between reload attempts
BCLRV	Bus Clear vector number
CACHE020	Determines whether the 68020's on-chip instruction cache will be used. 0 = don't use it 1 = use it
CACHEF	Variable for flushing cache $F80006 = flush bank Dummy value = don't flush bank
CACHESD	Cache supervisor data accesses 0 = don't cache it 1 = cache it
CACHESI	Cache supervisor instruction access 0 = don't cache it 1 = cache it
CACHEUD	Cache user data accesses 0 = don't cache it 1 = cache it
CACHEUI	Cache user instruction access 0 = don't cache it 1 = cache it
CHAINBAT	Switch to indicate if chain and batch processing are supported. Value of zero excludes batch and chain; non-zero includes them. This package requires approximately 3-1/2K of memory.
CLOCKFRQ	Number of clock ticks per millisecond
CMULT	I/O Channel multiplier 1 = for IOC run directly on the MC68020 or MC68008 address bus of a user's custom board (because of byte addressing) 2 = for all others where byte address is not available (only odd addresses are used) or if running the IOC from a VME316 board.
CONBATCH	Number of concurrent batch jobs that can be running. Cannot be more than NOTASKS.
CONT3151	Controller number for the 1st VME315 board
CONT3152	Controller number for the 2nd VME315 board
CONT320	Controller number for the VME320 board

D

CONT4205	Controller number for the SASI 5 board
CONT4208	Controller number for the SASI 8 board
CONTWIN1	Controller number for the 1st RWIN board
CONTWIN2	Controller number for the 2nd RWIN board
DARTDRV	Address of DART driver
DARTSPR	Address of supervisor DART driver
DCP$RTO	Read timeout for disks
DCP$VSS	VERSAdos sector size in bytes
DCP$WTO	Write timeout for disks
DCQPGE	The number of pages of memory for the device connection queue (DCQ). Minimum of 1 page, maximum of 10 pages. The DCQ is used to save concurrent requests to the same file or device. Each page of the DCQ can accomodate approximately 9 entries.
DEFAULT	System default volume:usernumber.catalog
DEFDAT	Default Data block length in sectors (256 bytes per sector). Used by file handler when no data block size is given at file allocation time. Minimum size is 4. Maximum size is 255.
DEFFAB	Default File Allocation Block (FAB) length is sectors (256 bytes per sector). Used by file handler when no FAB size is given. Minimum size is 1. Maximum size is 20.
DPRVAO	Dual ported RAM offset
DRVL	Address of DRVLIB
EET$	Flag to include EET module 0 = don't include EET module 1 = include EET module
EETSTR	Address of EET
EPCIDRV	Address of EPCI driver
FAIL	Board fail interrupt vector number
FHS$IOS$	Flag to include FHS/IOS module 0 = don't include FHS/IOS module 1 = include FHS/IOS module
FHSASR	FHS ASR entry point
FHSSTR	Address of FHS

FMS$	Flag to include FMS module 0 = don't include FMS module 1 = include FMS module
FMSASR	FMS ASR entry point
FMSSTR	Address of FMS
FOUR	Starting disk number in ASCII for drive 4
G0A$UCL	Bus 0 dev A UCL
G0B$UCL	Bus 0 dev B UCL
G0C$UCL	Bus 0 dev C UCL
G0D$UCL	Bus 0 dev D UCL
G0E$UCL	Bus 0 dev E UCL
G0F$UCL	Bus 0 dev F UCL
G0G$UCL	Bus 0 dev G UCL
G0H$UCL	Bus 0 dev H UCL
G0I$UCL	Bus 0 dev I UCL
G0J$UCL	Bus 0 dev J UCL
G0K$UCL	Bus 0 dev K UCL
G0L$UCL	Bus 0 dev L UCL
G0M$UCL	Bus 0 dev M UCL
G0N$UCL	Bus 0 dev N UCL
G1A$UCL	Bus 1 dev A UCL
G1B$UCL	Bus 1 dev B UCL
G1C$UCL	Bus 1 dev C UCL
G1D$UCL	Bus 1 dev D UCL
G1E$UCL	Bus 1 dev E UCL
G1F$UCL	Bus 1 dev F UCL
G1G$UCL	Bus 1 dev G UCL
G1H$UCL	Bus 1 dev H UCL
G1I$UCL	Bus 1 dev I UCL
G1J$UCL	Bus 1 dev J UCL

G1K$UCL	Bus 1 dev K UCL
G1L$UCL	Bus 1 dev L UCL
G1M$UCL	Bus 1 dev M UCL
G1N$UCL	Bus 1 dev N UCL
GB0$UCL	Bus 0 User Configuration Length (UCL)
GB1$UCL	Bus 1 User Configuration Length (UCL)
GST	The number of pages in global segment table. Minimum of 1, maximum of 10. Each page can accomodate approximately 14 entries.
HOGMODE	Specifies whether or not you want the VM02 to hog the VERSAbus (which allows it to run faster when accessing the VERSAbus). This may ONLY be used if there are no other cards in the system capable of becoming bus master. If in doubt, use 0. 0 = don't hog the bus 1 = hog the bus (no other intelligent board.)
INTSTR	Address of the initializer
IOCBASE	I/O Channel base address
IOCLVL1	I/O Channel interrupt 1 level
IOCLVL2	I/O Channel interrupt 2 level
IOCLVL3	I/O Channel interrupt 3 level
IOCLVL4	I/O Channel interrupt 4 level
IOCM	
IOCSTR	
IOCVEC1	I/O Channel interrupt 1 vector
IOCVEC2	I/O Channel interrupt 2 vector
IOCVEC3	I/O Channel interrupt 3 vector
IOCVEC4	I/O Channel interrupt 4 vector
IOSASR	IOS ASR entry point
IOSSTR	Address of IOS module
IOV	The number of pages in the I/O vector table. Minimum of 1. Each page can accomodate approximately 12 entries. For each vector claimed by a task using the CISR directive (Configure Interrupt Service Routine) a seperate entry is

	made into this table. The system imposes no maximum size for this parameter. For efficient use of system space, however, the formula for computing the table size should be: 1+(C/12) where 'C' is the number of different vectors that the user expects to claim via CISR directive.
IPCDRV	Address of IPC driver
KBDOVRD	Keyboard override flag 0 = keyswitch on front panel is enabled 1 = keyswitch override - if key is in locked position the keyboard is still enabled.
KILVECT	Vector number which forces system crash
L050$01	Address of 1st VME050 board
L300$01	Address of 1st VME300 board
L300$02	Address of 2nd VME300 board
L315$01	Address of 1st VME315 board
L315$02	Address of 2nd VME315 board
L320$01	Address of 1st VME320 board
L320$02	Address of 2nd VME320 board
L331$01	Address of 1st VME331 board
L331$02	Address of 2nd VME331 board
L331$03	Address of 3rd VME331 board
L331$04	Address of 4th VME331 board
L331$05	Address of 5th VME331 board
L331$06	Address of 6th VME331 board
L333$01	Address of 1st VME333 board
L333$02	Address of 2nd VME333 board
L333$03	Address of 3rd VME333 board
L333$04	Address of 4th VME333 board
L333$05	Address of 5th VME333 board
L333$06	Address of 6th VME333 board
L400$01	Address of 1st VME400 board
L400$02	Address of 2nd VME400 board

L410$01	Address of 1st VME410 board
L410$02	Address of 2nd VME410 board
L420$01	Address of 1st VME420 board
L420$02	Address of 2nd VME420 board
L435$01	Address of 1st VME435 board
L435$02	Address of 2nd VME435 board
L600$01	Address of 1st VME600 board
L600$02	Address of 2nd VME600 board
L605$01	Address of 1st VME605 board
L610$01	Address of 1st VME610 board
L610$02	Address of 2nd VME610 board
L615$01	Address of 1st VME615 board
L625$01	Address of 1st VME625 board
LDR$	Flag to include LDR module
LDRSTR	Address of LDR
LINKLS	Link listing work file
LINKLSW	Link listing file switch 0 = LINKLS is a file 1 = LINKLS is a device
LOGMSG1	Logon message part 1
LPDA$01	Local printer device address #1
LPDA$02	Local printer device address #2
LRAD$01	Address of 1st RAD board
LRIO$01	Address of 1st RIO board
LTDA$01	Local terminal device address #1
LTDA$02	Local terminal device address #2
LUMAX	Logical unit number maximum. LUMAX is a temporary symbol used to set the value of MAXLU, which is the maximum logical unit number that can be assigned for each task in the system. Restrictions are: 8 <= LUMAX <= 31
LV30$01	Address of 1st VM30 board

LV30$02	Address of 2nd VM30 board
LV30$03	Address of 3rd VM30 board
LV30$04	Address of 4th VM30 board
LWIN$01	Address of 1st RWIN board
LWIN$02	Address of 2nd RWIN board
M3150$1	Type of disk on drive 0 of 1st VME315 board
M3151$1	Type of disk on drive 1 of 1st VME315 board
M3152$1	Type of disk on drive 2 of 1st VME315 board
M3153$1	Type of disk on drive 3 of 1st VME315 board
M3154$1	Type of disk on drive 4 of 1st VME315 board
M3155$1	Type of disk on drive 5 of 1st VME315 board
M3156$1	Type of disk on drive 6 of 1st VME315 board
M3157$1	Type of disk on drive 7 of 1st VME315 board
M315DRV	Address of the VME315 driver
M320$DD	Post-data gap double-density floppy format
M320$HD	Post-data gap hard disk format
M320$LT5	Head load time 5-1/4" floppy drive in milliseconds
M320$LT8	Head load time 8" floppy drive in milliseconds
M320$LTH	Head load time hard disk drive in milliseconds
M320$SD	Post-data gap single-density floppy format
M320$ST5	Head settling time 5-1/4" floppy drive in milliseconds
M320$ST8	Head settling time 8" floppy drive in milliseconds
M320$STH	Head settling time hard disk drive in milliseconds
M3200$1	Type of disk on drive 0 of 1st VME320 board
M3201$1	Type of disk on drive 1 of 1st VME320 board
M3202$1	Type of disk on drive 2 of 1st VME320 board
M3203$1	Type of disk on drive 3 of 1st VME320 board
M3204$1	Type of disk on drive 4 of 1st VME320 board

M320DRV	Address of the VME320 driver
M420DRV	Address of the VME420 driver
M435DRV	Address of the VME435 driver
M600DRV	Address of the VME600 driver
M605DRV	Address of the VME605 driver
M610DRV	Address of the VME610/VME620 driver
M615DRV	Address of the VME615 driver
M625DRV	Address of the VME625 driver
MAXLU	MAXLU is kept to the minimum value to allow our standard sysgens to work in 384K. If you want more, change the "+3" to add whatever additional amount you need, or assign it to a specific value. (The "+3" was arbitrarily chosen.) MAXLU and NOTASK determine the amount of memory required for the logical unit table (LUT). The algorithm for determining the LUT size is as follows: LUT = 16+NOTASKS+8*NOTASKS*(MAXLU+1) There must be one LU for each disk volume (FMS assigns a different logical unit for each disk). This means that MAXLU must be greater that or equal to TOTDSK.
MCP$ATM	Bit 1 is recognized for a configure command
MCP$AW	Bit 0 = Reserved Bit 1 = 1 means user requests a density for write Bit 1 = 0 means user does not request a density
MCP$DEN	Density selected for write from loadpoint 0 - 1600 bpi (PE density) 1 - 800 bpi (NRZI density)
MCP$ERT	Number of times to erase before error message
MCP$PM	DEN, RDT, WRT, ERT fields are recognized for a configure command. RDTO, RWTO fields are recognized for a configure command. SPTO, SRTO fields are recognized for a configure command.
MCP$RDT	Number of read tries before error message
MCP$RDTO	Read timeout
MCP$RTO	Read Timeout (6 minutes to read 4K bytes, to search for a file mark, to read a blank tape to the end of tape, to rewind)
MCP$RWTO	Rewind timeout
MCP$SPTO	Space forward or reverse timeout

MCP$SRTO	Search forward or reverse for filemark timeout
MCP$WRT	Number of write tries before erasing
MCP$WTO	Write Timeout (5 seconds to write 4K bytes)
MEMBEG	Beginning of available memory
MEMEND1	Ending address for on-board memory; must be < this
MEMEND2	Starting address for off-board memory; must be >= this
MEMEND3	Ceiling address for off-board memory; must be < this
MFPDRV	Address of MFP driver
MMU	Address of MMU
MPCCDRV	Address of MPCC driver
MPSCDRV	Address of MPSC driver
MPSCSPR	Address of MPSCSPR driver
MPSCSUP	Address of MPSCSUP driver
MTA0LVL	Interrupt level
MTA1LVL	Interrupt level
N435$1	Number of tape drives on 1st VME435 board
N435$2	Number of tape drives on 2nd VME435 board
NF315$1	Number of floppy disks on 1st VME315 board
NF315$2	Number of floppy disks on 2nd VME315 board
NF320$1	Number of floppy disks on 1st VME320 board
NF4205$1	Number of 5-1/4" floppy disks on SASI 5" board
NF4208$1	Number of 8" floppy disks on SASI 8" board
NFRWIN$1	Number of floppy disks on 1st RWIN board
NFRWIN$2	Number of floppy disks on 2nd RWIN board
NH315$1	Number of hard disks on 1st VME315 board
NH315$2	Number of hard disks on 2nd VME315 board
NH320$1	Number of hard disks on 1st VME320 board
NH4205$1	Number of hard disks on SASI 5" board
NH4208$1	Number of hard disks on SASI 8" board

NHRWIN$1	Number of hard disks on 1st RWIN board
NHRWIN$2	Number of hard disks on 2nd RWIN board
NFV20$1	Number of floppy disks on 1st VM20 board
NFV20$2	Number of floppy disks on 2nd VM20 board
NFV21$1	Number of floppy disks on 1st VM21 board
NFV21$2	Number of floppy disks on 2nd VM21 board
NFV22$1	Number of floppy disks on 1st VM22 board
NHV21$1	Number of hard disks on 1st VM21 board
NHV21$2	Number of hard disks on 2nd VM21 board
NHV22$1	Number of hard disks on 1st VM22 board
NODEFVOL	The maximum number of default volumes that can be defined. Cannot be greater that NOTASKS+3.
NODIFFIL	The maximum number of different files that can be opened at one time. Cannot be greater than NOFILES. For every three different files, approximately 1K of memory is required. A ratio of 5 files for each terminal accomodates most requests.
NOFILES	The maximum number of files that can be opened in the system at one time. Limit of 200.
NOLOGON	Maximum number of invalid logon attempts before being rejected.
NOLOGONS	Number of terminals allowed to logon in the system.
NOLPRT	Number of local printers
NOLTERM	Number of local terminals
NORWIN	Number of RWIN board in the system
NOTASKS	The maximum number of tasks in the system at one time. Minimum of 1. VERSAdos contains a maximum of 6 resident tasks. Allowing for that, plus three for each terminal will accomodate most requests.
NOTNT	Number of Transparent Network Terminals
NOVM20	Number of VM20 boards in the system
NOVM21	Number of VM21 boards in the system
NOVM22	Number of VM22 boards in the system

NOVM30	Number of VM30 boards in the system
NP050$1	Number of printers on 1st VME050 board
NPV30$1	Number of printers on 1st VM30 board
NPV30$2	Number of printers on 2nd VM30 board
NPV30$3	Number of printers on 3rd VM30 board
NPV30$4	Number of printers on 4th VM30 board
NRAD	Number of RAD boards in the system
NRIO	Number of RIO boards in the system
NRIOINT	Number of interrupt levels per I/O module
NT050$1	Number of terminals on 1st VME050 board
NTV30$1	Number of terminals on 1st VM30 board
NTV30$2	Number of terminals on 2nd VM30 board
NTV30$3	Number of terminals on 3rd VM30 board
NTV30$4	Number of terminals on 4th VM30 board
NU400$1	Number of users/printers on the 1st VME400 board
NU400$2	Number of users/printers on the 2nd VME400 board
NU410$1	Number of users/printers on the 1st VME410 board
NU410$2	Number of users/printers on the 2nd VME410 board
NU605	Number of users (total) for the VME605 boards
NU615	Number of users (total) for the VME615 boards
NU625	Number of users (total) for the VME625 boards
NURAD	Number of users on RAD boards
NVME050	Number of VME050 boards in the system
NVME300	Number of VME300 boards in the system
NVME315	Number of VME315 boards in the system
NVME316	Number of VME316 boards in the system
NVME320	Number of VME320 boards in the system
NVME400	Number of VME400 boards in the system
NVME410	Number of VME410 boards in the system

NVME4205	Number of VME420 boards in the system connected to a SASI 5"
NVME4208	Number of VME420 boards in the system connected to a SASI 8"
NVME435	Number of VME435 boards in the system
NVME600	Number of VME600 boards in the system
NVME605	Number of VME605 boards in the system
NVME610	Number of VME610 boards in the system
NVME615	Number of VME615 boards in the system
NVME620	Number of VME620 boards in the system
NVME625	Number of VME625 boards in the system
OFFBD$HI	High value for off-board RAM (as seen by driver)
OFFBD$LO	Low value for off-board RAM (as seen by driver)
ONBD$HI	High value for on-board RAM (as seen by driver)
ONBD$LO	Low value for on-board RAM (as seen by driver)
P050DRV	Address of VME050 printer driver
P115DRV	Address of VME115 printer driver
PAGESIZE	Size in bytes of a page of memory
PANEL	Address of front panel
PAT	The number of pages in the periodic activation table. Each page can accomodate approximately 8 entries.
PCDRV	Number of Process Control Drivers
PCP$AFF	Auto form feed 0 = do not suppress auto form feed on assign 1 = suppress auto form feed on assign
PCP$ELC	End of line character
PCP$LNFD	Auto line feed 0 = printer does not support auto line feed 1 = printer does support auto line feed
PCP$LRL	Logical line length <= width of printer
PCP$REC	Width of printer (characters/physical print line)
PCP$RSZ	Depth of printer (lines/page)
PCP$TLRL	Wrap-around/truncate

	0 = wrap-around print if logical line length exceeded 1 = truncate print at logical line length
PCP$WTO	Number of milliseconds to allow before timing out a write
PIADRV	Address of the PIA driver
PTMVECT	Programmable timer vector number
PV01DRV	Address of VM01 printer driver
RADDRV	Address of the RAD driver
RIODRV	Address of the RIO driver
RAM$SQ	Difference between on-board RAM address as seen by driver and device.
REVNUMBR	Logon message part 2
ROMEADDR	ROM end address; defined by user for a ROMable system. Set to $0 if not a ROMable system.
ROMSADDR	ROM start address;
RWIN0$1	Type of disk on drive 0 of 1st RWIN board
RWIN0$2	Type of disk on drive 0 of 2nd RWIN board
RWIN1$1	Type of disk on drive 0 of 1st RWIN board
RWIN1$2	Type of disk on drive 0 of 2nd RWIN board
RWIN2$1	Type of disk on drive 0 of 1st RWIN board
RWIN2$2	Type of disk on drive 0 of 2nd RWIN board
RWIN3$1	Type of disk on drive 0 of 1st RWIN board
RWIN3$2	Type of disk on drive 0 of 2nd RWIN board
RWINDRV	Address of RWIN driver
SECURITY	Switch to indicate if security package is supported. Value of zero excludes package, nonzero includes it. This package requires approximately 1K of memory.
SERFLAG	
SERPTS	Serial port interrupt vector
SET1	A user definable parameter that can take on any one of the following values: "SYSTEM", "USER", or "DONT-CARE" If SET1 is set to "SYSTEM" then the cache bank should be set up as a supervisory cache. In this mode, RMS and the drivers will use the cache and RMS will not flush the cache

	on task switches. If SET1 is set to any other value, then RMS will flush the cache on all task switches that force a change in address space number.
SET2	A user definable parameter that can take on any one of the following values: "SYSTEM", "USER", or "DONT-CARE" If SET2 is set to "SYSTEM" then the cache bank should be set up as a supervisory cache. In this mode, RMS and the drivers will use the cache and RMS will not flush the cache on task switches. If SET2 is set to any other value, then RMS will flush the cache on all task switches that force a change in address space number.
SIOBASE	Base address of Short I/O address space
SIODRV	Address of SIO driver
SIX	Starting disk number in ASCII for drive 6
SPCCMD	Switch to indicate if the following user session management commands are supported: HELP, CLOSE, ASSIGN, NEWS. Value of zero excludes commands. This package requires approximately 1/2K of memory.
STACK	Address of stack
STARTRMS	
SWABRT	Software abort vector number
SYSFAIL	Determines whether or not the operating system will be interrupted when SYSFAIL is asserted on the bus. Some intelligent boards will assert SYSFAIL when they experience a failure of some kind. If you have such boards in the system, AND THE DRIVERS FOR THESE BOARDS HAVE SYSFAIL HANDLERS, then you will probably want SYSFAIL interrupts enabled. If the appropriate SYSFAIL handlers are not written, then taking a SYSFAIL interrupt will hang up the system, so you would want SYSFAIL interrupts disabled. 0 = disable SYSFAIL interrupts 1 = enable SYSFAIL interrupts
TCP$BITS	7/8 bits/char 0 = transmit/receive 8 bits/character 1 = transmit/receive 7 bits/character
TCP$BRC	Character to be interpreted like a break when received 0 = none
TCP$BRT	Baud rate code The following code may be used to set the desired baud rate.

Code	Rate	Code	Rate	Code	Rate	Code	Rate
$00	50	$05	300	$09	2000	$0E	9600
$01	75	$06	600	$0A	2400	$0F	19200
$02	110	$07	1200	$0B	4800	$10-$1FF	
$03	134.5	$08	1800	$0C	7200	Reserved	

TCP$CLC	Character which causes line to be deleted when received 0 = none
TCP$DOP	Character which causes output to be discarded when received 0 = none
TCP$ECHO	Echo characters 0 = driver should echo characters 1 = driver should not echo characters
TCP$EOL	End of line string
TCP$HCPY	Hardcopy device 0 = terminal is not a hard copy device it is a CRT 1 = terminal is a hard copy device not a CRT
TCP$MODM	Modem connect 0 = the port is not connected to a modem 1 = the port is connected to a modem
TCP$NLS	Number of ASCII null characters to send after each CR or LF
TCP$OFFH	Modem Offhook 0 = the port (if connected to modem) is not offhook 1 = the port (if connected to modem) is offhook
TCP$PNUL	Null characters 0 = Null characters are not considered data for image read 1 = Null characters should be considered data for image reads
TCP$PRTY	odd/even parity 0 = parity (if used) should be odd 1 = parity (if used) should be even
TCP$REC	Width of terminal (characters/line)
TCP$RLN	Character which causes line to be reprinted when received 0 = none
TCP$RSZ	Depth of terminal (lines/page)
TCP$RTO	Number of milliseconds to allow before timing out a read
TCP$RTV	Read terminators
TCP$STPB	1/2 stop bits 0 = follow each character with 1 stop bit 1 = follow each character with 2 stop bits
TCP$TAHD	Type ahead 0 = the type ahead feature is used 1 = the type ahead feature is not used
TCP$TFUL	Terminate the read of buffer full 0 = filling the buffer on a read will not term. the read

	1 = filling the buffer on a read should terminate the read
TCP$TRC	Terminator class $0X = none
TCP$TTP	Terminal type 0 = EXORterm 155
TCP$USEP	Parity check 0 = do not use parity 1 = parity should be checked and generated
TCP$WTO	Number of milliseconds to allow before timing out a write
TCP$XCTL	XON/XOFF control 0 = use CTS to control transmission 1 = use XON/XOFF characters to control transmission
TCP$XOF	XOFF character; when received, suspends transmission 0 = none
TCP$XON	XON character; when received, cancels a prior XOFF character 0 = any character
TERMDRV	Address of TERMDRV
TERMLIB	Address of TERMLIB
TERMOCNT	Number of terminal output timeouts before logoff
TIMER	Address of timer
TIMINTV	The number of milliseconds between timer interrupts
TIMSLIC	The number of timer interrupts per time slice
TOTDSK	Total number of disks
TOTTERM	Total number of terminals
TRACE	Number of pages in trace table. TRACE must be nonzero if TRCFLAG is nonzero. Each page can accomodate approximately 10 entries.
TRCFLAG	Trace flag 0 = don't trace The setting of bits in the TRCFLAG parameter will control which events cause an entry to be built in the trace table

Bit # in TRCFLAG	Event
15	TRAP #1
14	I/O interrupt not claimed by user task
13	Timer interrupt
12	User trap (2-15)
11	Exception
10	Dispatch
9	I/O interrupt claimed by user

		task
	8	Return from LOADMMU
	7	Simulated interrupt
	6	SYSFAIL interrupt

TWO	Starting disk number in ASCII for drive 2
UDR	Number of pages in user defined directive table. Minimum of 0. Maximum of 10. Each page can accomodate approximately 25 entries.
UST	The number of pages in the user semaphore table. Minimum of 1, maximum of 10. Each page can accomodate approximately 25 entries.
VM200$1	Type of disk on drive 0 on 1st VM20 board
VM200$2	Type of disk on drive 0 on 2nd VM20 board
VM201$1	Type of disk on drive 1 on 1st VM20 board
VM201$2	Type of disk on drive 1 on 2nd VM20 board
VM202$1	Type of disk on drive 2 on 1st VM20 board
VM202$2	Type of disk on drive 2 on 2nd VM20 board
VM203$1	Type of disk on drive 3 on 1st VM20 board
VM203$2	Type of disk on drive 3 on 2nd VM20 board
VM210$1	Type of disk on drive 0 on 1st VM21 board
VM210$2	Type of disk on drive 0 on 1st VM21 board
VM211$1	Type of disk on drive 1 on 1st VM21 board
VM211$2	Type of disk on drive 1 on 1st VM21 board
VM212$1	Type of disk on drive 2 on 1st VM21 board
VM212$2	Type of disk on drive 2 on 1st VM21 board
VM213$1	Type of disk on drive 3 on 1st VM21 board
VM213$2	Type of disk on drive 3 on 1st VM21 board
VM214$1	Type of disk on drive 4 on 1st VM21 board
VM214$2	Type of disk on drive 4 on 1st VM21 board
VM215$1	Type of disk on drive 5 on 1st VM21 board
VM215$2	Type of disk on drive 5 on 1st VM21 board
VM216$1	Type of disk on drive 6 on 1st VM21 board

D

VM216$2	Type of disk on drive 6 on 1st VM21 board
VM217$1	Type of disk on drive 7 on 1st VM21 board
VM217$2	Type of disk on drive 7 on 1st VM21 board
VM220$1	Type of disk on drive 0 on 1st VM22 board
VM221$1	Type of disk on drive 1 on 1st VM22 board
VM222$1	Type of disk on drive 2 on 1st VM22 board
VM223$1	Type of disk on drive 3 on 1st VM22 board
VM224$1	Type of disk on drive 4 on 1st VM22 board
VM225$1	Type of disk on drive 5 on 1st VM22 board
VM226$1	Type of disk on drive 6 on 1st VM22 board
VM227$1	Type of disk on drive 7 on 1st VM22 board
VM228$1	Type of disk on drive 8 on 1st VM22 board
VM229$1	Type of disk on drive 9 on 1st VM22 board
VM22A$1	Type of disk on drive A on 1st VM22 board
VM22B$1	Type of disk on drive B on 1st VM22 board
VM22DRV	Address of the VM22 driver
WHERLOAD	Memory address where boot file will be loaded. Non-zero for VM01 only. If nonzero, VERSAdos will be moved at initialization time.
WORKLS	Overall listing file/device
ZERO	Starting disk number in ASCII for drive 0

APPENDIX E

APPLICATION INCLUDE FILE EXAMPLES

ASSEMBLER APPLICATION EXAMPLE

The **DUMP** utility is modified for ROM application to allow the user the capability to access another terminal, the printer, and the disk as device assignments. The application **INCLUDE** file is .&.DUMP.CI and its associated file is .&.DUMP.LG.

.&.DUMP.CI

```
*****************
* DUMP.CI       *
************************************************************************
* DUMP utility modified to demonstrate ROMability
************************************************************************
*
TASK       &.DUMP.LO,.DMP
ATTRIB     = 'USER'
STATE      = 'READ'
SESSION    = 7
PRIORITY   = $42
SUBS       &.DUMP.LG
LINK       &.DUMP.LG
IFEQ       \LINKLSW
  =COPY       \LINKLS,\WORKLS;A
ENDC
END        DUMP
*
*
```

.&.DUMP.LG

```
 =LINK ,&.DUMP.LO,\LINKLS;HAMIX
 SEG SEG0:0,14      \PC
 INPUT &.DUMP.RO
 LIB   &.UTILIB.RO
 END
=END
```

PASCAL APPLICATION EXAMPLE

This Pascal task, which executes an infinite loop, is a simple task that writes to the printer and the terminal using the floating point capability. The application INCLUDE file is .&.PTASKFP.CI and its associated files are RROM.RLIBFP.LG and RROM.TASKFP.LG. If the user wants to use the non-floating point library, replace RROM.RLIBFP.LG with RROM.RLIBNFP.LG and RROM.TASKFP.LG with RROM.TASKNFP.LG.

.&.PTASKFP.CI

```
MSG      *****************************************************************
MSG      *                                                               *
MSG      *   LINK THE GLOBALLY SHAREABLE PASCAL RUN-TIME ROUTINES FOR    *
MSG      *   ROM USAGE                                                   *
MSG      *                                                               *
MSG      *****************************************************************

GSPLSTR = *
SUBS     RROM.RLIBFP.LG
LINK     RROM.RLIBFP.LG
 IFEQ        \LINKLSW
   =COPY        \LINKLS,\WORKLS;A
ENDC
PROCESS RROM.RLIBFP.LO

MSG        ***************************************************************
MSG        *                                                             *
MSG        *  LINK THE GLOBALLY SHAREABLE PASCAL RUN-TIME ROUTINES TO    *
MSG        *  THE USER TASK.  THIS IS NECESSARY TO SATISFY EXTERNAL      *
MSG        *  REFERENCES BUT SEGO WILL BE EXCLUDED AS SEGO               *
MSG        *  IS DEFINED ABOVE IN THE ROM LIBRARY AND SEG2 WILL          *
MSG        *  BE OBTAINED DYNAMICALLY BY THE INITIALIZER                 *
MSG        *                                                             *
MSG        ***************************************************************

TASK          &.PTASK.LO
ATTRIB      = 'USER'
STATE       = 'READ'
SESSION     = $100
PRIORITY    = $90
EXCLUDE       SEGO
EXCLUDE       SEG2
SUBS          RROM.TASKFP.LG
LINK          RROM.TASKFP.LG
 IFEQ       \LINKLSW
   =COPY        \LINKLS,\WORKLS;A
ENDC
END         PTASK
*
*
*
```

RROM.RLIBFP.LG

```
=/*
=/*
=/*      RROM.RLIBFP.LG  chainfile to link globally shareable Pascal
=/*                      run-time routines.  These run-time routines
=/*                      INCLUDE floating point.
=/*
=/*
=LINK ,RROM.RLIBFP.LO,\LINKLS;HAMIXSZ=100
SESG SEGO(RG):8  \GSPLSTR
INPUT RROM.RLIBFP.RO
END
```

RROM.TASKFP.LG

```
=/*
=/*      RROM.TASKFP.LG  chainfile to link globally shareable Pascal
=/*                      run-time routines to a user task.  The run-
=/*                      time routines INCLUDE floating point.
=/*
=/*      The following SYSGEN link file can be used by the user by
=/*      changing only the name of the applications task '????' where
=/*      referenced.  NOTE that the library modules are 'INCLUDEd',
=/*      not 'LIBed'.  This is necessary to properly satisfy external
=/*      references with a shared library.
=/*
=/*
=LINK ,&.PTASK.LO,\LINKLS;HAMIXSZ=100
SEG PROG (R):9 \PC
SEG SEGO(RG):8  \GSPLSTR
SEG SEG2(R):15
IN      &.PTASK.RO
IN   RROM.INIT.RO
IN   RROM.ASSIGNLU.RO
IN   RROM.RLIBFP.RO
END
=END
```

E

THIS PAGE INTENTIONALLY LEFT BLANK.

APPENDIX F

SYSTEM SYSGEN LISTING EXTRACT

The program counter with the startup address specified for the user **SYSGEN** system listing should be set. In this example, the program counter is set to $312600.

```
FILENAME      TASK     PROC     SEG      ADDRESS


RMS.LO                 RMS      RMS0     $300000
                                RMS2     $300100
DRVLIB.LO              DRVL     DRVL     $304E00
TERMLIB.LO             TERM     TERM     $305000
ACIADRV.LO             ACIA     ACIA     $306300
PIADRV.LO              PIAD     PIAD     $306600
FHS.LO        .FHS              .FHS     $306C00
IOS.LO        .IOS              .IOS     $308000
RLIBFP.LO              RLIB     SEG0     $309A00
PTASK.LO      PTAS              PROG     $311200
IOI.LO        .IOI              IOSF     $311900
                                .IOI     $312100
SYSINIT.LO             SYSI     .INT     $312600

- START-UP ADDRESS = $312600
TOTAL NUMBER OF USER DEFINED SYMBOLS = 204

0  ERRORS ENCOUNTERED
```

THIS PAGE INTENTIONALLY LEFT BLANK.

INDEX

INDEX

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.

19140 PRINTED IN USA (1/86) MESSENGER 3500