digital

INTEROFFICE MEMORANDUM

TO: LIST

DATE: 20 APRIL 1977 [illegible]
FROM: CARL GIBSON/FRANK HASSETT
DEPT: REAL TIME/COMPUTATION
EXT: 5517/3272
LOC/MAIL STOP: ML12-3/E13/ML5-5/E76

SUBJ: RSX-11M MULTIPROCESSING SOFTWARE FUNCTIONAL SPEC.

The enclosed specification is the result of the design efforts accomplished to date in planning for RSX-11M Multiprocessing.

The first 20 pages represent the functional specification and are organized in summary form. The appendices provide the technical back-up detail. It is worth noting that pages 20 through 177 take the form of a Reference Manual for RSX-11MP and specify the system at a second level of detail. Pages 178 through 207 provide internal design data at a yet lower level of technical detail.

A specification presentation and review will be scheduled in approximately two weeks to assimilate reviewer feedback into this design. Time and place TBA.

Funding and staffing for this project are presently being worked as part of the current Red Book process. A project plan will be proposed following resolution of these issues.

/cfc

Distribution:

Roger Allen	RG	John Garison	ML5/E40
Jega Arulpragasam	ML5-2/E56	Mark Gerhardt	ML5/E76
Eric Baatz	ML5/E76	Teri Gerrasimenko	ML5/E40
Bob Beck	ML21-3/E87	Carl Gibson	ML12/E13
Don Bennett	ML5/E45	John Gilbert	ML5/M40
Hal Berman	PK3/S44	Luke Gillespie	ML5/E97
Dave Best	ML3/E35	Debbie Girdler	ML5-2/E19
Paul Bezeredi	ML5/E19	Brad Glass	ML5/E76
Ken Blackett	MR1/M42	John Gorczyca	MR2/E79
Verell Boaen	ML21-1/E81	Donna Graves	MR2/E70
Jim Bokhari	HD	Ron Ham	ML5/E40
John Carlson	PK3-2/S17	Dave Harvey	RG
Ed Christiansen	MR1/E37	Frank Hassett	ML5/E76
Peter Christy	ML12-3/A62	Bill Heffner	ML5/E76
Pete Conklin	ML12-3/E80	Roger Heinen	ML3/E35
Karen Condon	PK3-1/F51	Stephen Heiser	ML5/E39
Dave Cutler	ML3/E88	Chuck Hess	ML5/E19
Bob Daley	ML21-4/E20	Brian Higgs	PK2/A35
Clark D'Elia	ML5/E76	James Hirni	NU
Tom Donovan	HD	John Holz	ML5-2/E50
Dick Eckhouse	ML3-2/E41	Steve Hort	GE
Dennis Fiore	HD	Dan Hubble	PK3/M56
Ken Follien	ML3/E67	Mike Huddart	RG
Kurt Friedrich	ML12/E13	Malcolm Johnston	ML21-1/E81
Frank Garabo	PK3/M21	Larry Jones	WM/P21
Ed Gardner	ML5/E39	Jo Ann Kasson	ML5/E76
		Martyn Kee	RG

Distribution:

Pete Kilbourn	PK3/M56	Ed Quiet	ML5/E40
Kim Kinnear	ML5/E76	Audrey Reith	ML5/E40
George LaMontagne	ML5/E45	Tom Rhodes	ML5/E97
Bruce Leavitt	ML5/E40	Paul Ritchie	RG
Howard Lev	ML5/E76	Rick Robinson	ML5/E39
Mike McGough	PK3/M21	Bob Rosenbaum	ML5/E40
Bruce McNaughton	NU/C10	Marcia Rough	ML5/E76
Paul Massiglia	PK3/M10	Jerry Rufener	PK3/M56
Jack Mileski	ML12/E13	Charles Samuelson	MR2/E70
Robin Miller	WM/P21	Wayne Saunders	PK2/A35
Tom Miller	ML5/E76	Benn Schreiber	ML5/E76
Bill Minty	HD	Dave Schroeder	PK3/M33
Warren Moncsko	ML5/E19	Minoo Scroff	PK3/M12
Chuck Monia	ML3/E35	Ellen Simich	ML5/E40
Russ Moore	ML21-1/E81	Iain Smith	RG
Kathleen Morse	ML3/E88	Gregg Thibodeau	NU
Dick Murphy	ML5/E39	Mark Uhrich	PK1/P84
Dave Nelson	ML3-4/E31	Donald Van Volkenburg	ML5/E45
Richard Newland	RG	Peter Wannheden	ML5/E76
Herb Nichols	ML5/E76	Bill Weiske	CSS-Nashua
Kathie Norris	ML5/M17	Bill Wessell	WA
Jackie Pakarinen	ML5-5/E40	Pat White	ML12/E51
Dave Palmer	ML12/A62		
Bob Paulson	ML5/E40		
Kiluba Pembamoto	ML21-1/E81		
Hilary Pierce	RG		
George Plowman	ML5/E97		

RSX-11M

Multiprocessing

Software Functional Specification

COMPANY CONFIDENTIAL

COPY NO.___________

For additional copies,
contact Pat Murphy
ML5-5/E76 Ext. 3258

digital equipment corporation · maynard, massachusetts

Digital Equipment Corporation COMPANY CONFIDENTIAL
RSX-11M Multiprocessing Software Functional Specification

Title: RSX-11M Multiprocessing Software Functional Specification

Specification Status: Draft

Architectural Status:

File: MPFUNC.MEM[301,305]

Retrieval Number: 130-951-061-00

PDM #: not used

Date: 11-Apr-77

Superseded Specs: none

Authors: Kim Kinnear, Chuck Monia

)ist: Yvonne Hicks

Reviewers: Eric Baatz
Bob Beck
Verell Boaen
John Carlson
Carl Gibson
Debbie Girdler
Frank Hassett
John Holz
Malcolm Johnston
Kim Kinnear
Howard Lev
Tom Miller
Chuck Monia
Russ Moore
Kiluba Pembamoto
Ed Quiet
Mark Uhrich
Bill Weiske

Abstract: This document is the functional design specification for the RSX-11M Multiprocessing Software. Topics that are addressed include guidelines for system availability and maintainability; system hardware configurations supported; program-assisted reconfiguration, and performance features. The major portion of this specification consists of a detailed and comprehensive RSX-11M Multiprocessing Reference Manual, in preliminary form.

vision History:

Description	Author	Revised Date
Technical update	K. Kinnear	18-Mar-77
Technical update	C. Monia	21-Mar-77
Technical update	K. Kinnear	26-Mar-77

CONTENTS

RSX-11M Multiprocessing Software Functional Specification

1.0 PRODUCT OVERVIEW OF RSX-11MP

1.1 Product Abstract

Description:

RSX-11mP is an operating system that functions in a multiprocessor hardware environment. It consists of the RSX-11M operating system, enhanced to support multiprocessors, together with the additional tasks that are required for multiprocessing support.

System Objectives:

- o Provide increased hardware availability by utilizing:
  - . Redundant functional units for increased systems reliability.
  - . The ability to reconfigure the system under operator control.
- o Improve system performance by:
  - . Providing the ability to execute tasks simultaneously.
  - . Efficient utilization of added resources for peak load handling.
- o Incorporate additional maintainability features in the system and associated tasks to ensure that:
  - . An adequate level of maintainability is retained.
  - . Maintenance resources required are not out of proportion to system cost.

System Users:

This system is intended for sophisticated and knowledgeable users who can exploit its performance and availability potential. When coupled with properly designed applications, multiprocessing can provide a significant reduction in cost penalties resulting from downtime, combined with substantial throughput improvements. The st-effectiveness of this system, when effectively utilized, is substantial.

Supported Hardware:

The hardware supported consists of the PDP-11/70 with a minimum of 128KB of multiport MOS memory and the following hardware features:

- o Cache flush (invalidation of all cache contents under program control)
- o Manual Port Controls for multiport memory
- o CPU modifications required to permit the MK11 CSRs to be addressed.
- o Cache bypass
- o ASRB instruction used for interlocking processes running on different CPUs.
- o An interprocessor interrupt and 'sanity timer' (one per CPU)
- o A 'time-of-day' clock (containing battery-backup)
- o Multiprocessor Bootstrap

Up to 4 processors are supported.

NOTE

> In anticipation of an increase in the number of simultaneously active tasks, 3 and 4-processor configurations will require a minimum of 256KB of memory to ensure satisfactory performance.

System Characteristics:

All other hardware requirements and support are as specified for RSX-11M V4.

The system will consist of multiport memory containing a single copy of the Executive and its data base. One processor may be connected to each memory port and all physical memory is addressable from each CPU.

Except for devices, which may be distributed among busses, the system is fully symmetrical. Any task can run on any processor and any processor can service a non-I/O Executive request.

uests for I/O and the servicing of interrupts must be performed by the processor to which the device controller is attached. Interprocessor interrupt hardware is provided to ensure that a CPU can be dispatched promptly to service these requests.

The use of RSX-11M as a nucleous for multiprocessing is based on the following considerations:

- The Executive is characterized by high performance and low overhead.
- The extensions for multiprocessing are fully transparent to all non-privileged tasks.
- Multiprocessor changes are localized, few in number, and readily implemented.
- The design of the Executive permits symmetric multiprocessing, thereby eliminating one processor as a single point of failure.
- The Executive is modular in structure and highly reliable.

As in the single-processor system, the RSX-11mP Executive arbitrates among tasks that are competing for system resources on a priority basis. To the mP Executive, each CPU represents a discrete resource that can be allocated to run a single task. The Executive 'doles' out processors to the highest priority tasks that are ready to run until all CPU's have been allocated.

<u>System Features</u>:

The following capabilities will be added to RSX-11M to produce a product that meets the availability and performance goals:

Availability/Maintainability features:

- Symmetric multiprocessing
- Support for dual port devices
- Support for mixed massbus configurations
- Support for DT03 FP Unibus switch
- Support for program-assisted reconfiguration of most functional units under operator control to allow powered-up maintenance
- Enhanced error logging capability
- System crash analyzer
- Ability to split system into independent threads
- Ability to load stand-alone diagnostics into an offline thread from a Files-11 volume

- o Ability to allow online processor diagnostics to be run by 'binding' a task to a CPU
- o Ability to allow the operating system to be bootstrapped from any Files-11 device

The RSX-11M error logging interface will be modified to provide:

- o Processor number on parity errors and other traps
- o Support for the MK11 multiport MOS memory error detection and reporting features
- o More detailed reports of concurrent system I/O activity on the occurrence of an I/O error including:
  - . The name and unit number of each active NPR device currently recognized by the error logger
  - . The type of activity in progress (read or write)
  - . The area of memory used for the data transfer
- o Support for multiport I/O, including notification of the port in use at the time an error occurred
- o The provision for recording a 'snapshot' of the system configuration at the time the error occurred

Dual port device support will require the following maintainability features so online diagnostics can be run:

- o The ability to suspend port switching (load balancing) when a diagnostic is run
- o The ability to command use of a specific port

In addition to the above, additional reconfiguration capability will be provided to allow concurrent operation of multiple thread configurations.

The provisions for mixed massbus support allow more efficient I/O topologies to be generated, since configurations such as dual port RP06's and magtapes (via the RH01 dual port adapter) can have full data path redundancy with half the number of controllers required in a non-mixed system.

Performance features:

- o Multiple parallel processes
- o Overlapped disk seeks

- o Disk seek optimization

Cache Support:

User access to the 11/70 cache bypass feature is provided to disable the cache when the processor is addressing a shared region. This support will require modifications to the memory management directives.

Cache invalidation will be performed automatically by the Executive and is transparent to the task.

Task/Procedure Modifications:

The following system tasks and procedures must be modified as indicated, to provide multiprocessor support:

ATL/ACT	- (Active Task List) - Display the CPU on which the task is running (Task-CPU Affinity).
SAVE	- Initiate the multiprocessor system.
DEVICES	- Display the device-bus configuration.
RUN	- Accept a CPU number for task-CPU affinity.
TAL	- Display task-CPU affinity; task-bus affinity.
OPEN	- Display addresses in memory that are local to each CPU. Open locations that are specific to each bus (I/O page locations).
LOAD	- Load new device drivers and databases.
UNLOAD	- Unload new device drivers. Recognize new data base structure.
INSTALL	- Set 'cache bypass' for a global common block accessed by a task. Detect conflicts in device common block references.
TIM	- Report the time as read from the TOD clock.

SET /MAIN
SET /SUB
	- Attach the 'cache bypass' attribute to a named common block. Allocate device partitions on multiple busses.

PAR	- Display: o Diagnostic partitions o Cache bypass partitions

- o Partition-bus mask (for device partitions)

TKTN	- Display task processor number.
PMD	- To display processor on which the task was running at termination.

The following new MCR functions must be provided:

REC	- Reconfigure functional units.
LOD	- Load stand-alone diagnostic from Files-11 device.
EXTPAR	- Extend physical size of partition.

The following VMR (virtual MCR) functions must be changed:

DEVICES
TASKLIST
LOAD
RUN
UNLOAD
INSTALL
SET /MAIN
SET /SUB
PAR

The following new VMR functions must be provided:

SET /(NO)EXEPAR	- Allocate/deallocate memory reserved for each processor.
REC	- Reconfigure functional units.
EXTPAR	- Extend physical size of partition.

1.2 References

11MR4.DOC - Preliminary Project Plan for RSX-11mP
11MR4.MEM - Cursory Project Plan for RSX-11M Release 4

1.3 Glossary

Availability	A measure of the capability of a system to perform a designated function.
Asymmetrical Systems	A multicomputer system in which

	there is a significant variation between the capabilities of the various processor units comprising the total system.
Bus Run	A path for unibus data and control signals that cannot be separated under program control.
Common-bus System	A system organization in which all functional units are connected to a single common interconnection bus. It refers to any organization, even those having multiple busses, if there are not sufficient paths available to permit all functional units to be active at the same time.
Directly Coupled Systems	A coupled system in which the common storage media used for data transfer is high-speed central memory.
Fault-tolerant Systems	Any system organization in which the motivation for the use of redundant functional units and the design of the interconnection system is to allow the system to overcome the effects of failures in either the units or the interconnect system. It often meets all of the criteria for being a multiprocessor; however, the motivations are not primarily those applicable to other multiprocessor systems.
Functional Units	The major operational units of a digital computer; for example, central memory, arithmetic and logic, control, input/output controller or processor.
Indirectly Coupled System	A coupled system in which the common storage media utilized for data transfer is an on-line device such as a tape, drum, or disk.
terprocessor Intereference	A reduction in the effective speed of a processor due to delays in accesses to central memory caused by the memory being in use by other

	processors or the I/O.
Lock	The protection of a data set by inhibiting access to it by all processes except the single current user.
Multi-access	The ability to control a functional unit from more than one controller.
Multicomputer System	A system having more than one processor but not meeting all of the criteria of multiprocessors.
Multiple-bus/Multiport Systems	A multiprocessor system in which interconnection is implemented by individual busses connecting each processor and input/output controller to a separate port on the central memory.
Multiport	The existence of more than one port used for the transfer of data to or from a functional unit.
Multiprocessor	(The definition given in this specification is quite explicit). Generally, a multiprocessor computer contains two or more processor units, each having the following identical capabilities: o Each unit has access to shared common central memory. o Each unit has common access to at least a portion of the I/O devices. o All units are controlled by one operating system that provides interaction as required between the processors and the programs they are executing at the task and hardware (interrupt) levels.
..fline	The resource is known to the operating system and one or more of the following are true:

	o The resource is not physically present. o A path for the transfer of control and data signals does not exist from the online system. o The resource has not been marked for online status by the reconfiguration software.
Online	The resource is known to the operating system and all of the following are true: o The resource is physically present. o A path for the transfer of data and control signals exists from the online system. o The resource has been marked for online status by the reconfiguration software.
Port	An interconnection point or interface to a functional unit for the transfer of data.
Private Memory	That portion of central memory usually accessible only to a single processor.
Reconfiguration	Changes in the system organization that may be achieved by either manual or automatic means.
Reliability	A measure of the ability to function without failure.
Satellite Computers	A multicomputer system in which the satellites act as slaves to the main processor.
Switched Bus Run	A unibus run that can be switched by a DT03 unibus switch.
Symmetrical System	Synonymous with multiprocesor system.

Thread	Set of functional units necessary to form an operable system.

1.4 Internal Interfaces Under ECO Control

None

1.5 Markets

This product is intended for sophisticated users in applications where availability and/or performance are the primary requirements.

As such, the principle markets are:

. As a base for in-house 11/70 applications such as TPS, DBMS, Typeset and Commercial end products.

. As a foundation for 11/70 based high-availability CSS or OEM products.

1.6 Competitive Analysis

Competitive products are the TANDEM multicomputer system, stressing high availability, and the Data General Dual Nova which can share peripherals but which does not utilize shared memory.

The 11/70 product utilizes resources more effectively than the dual NOVA while providing comparable availability and significantly improved performance.

TANDEM availability is superior, however 11/70 multiprocessing should win in the wider marketplace requiring enhanced availability coupled with performance for cost effectiveness. Other factors in favor of the 11/70 are:

. Design Maturity
. Available Software
. Service organization

Other known competitors who have announced or are offering similar architectures are Varian and Interdata. Assuming DEC's entry into the marketplace is timely, we should be able to compete effectively against them on our traditional strengths.

1.7 Product Audience

RSX-11mP will be visible primarily to programmers and designers catering to relatively unsophisticated end users. As such, the kernel and associated hardware form a foundation that must be augmented by carefully designed applications to fully realize the benefits of multiprocessing.

2.0 PRODUCT GOALS

2.1 Performance

When compared to a single processor system, peformance of the multiprocessor configuration is heavily dependent on the workload, use of shared regions and mix of tasks in the system.

A single task in a lightly loaded system may exhibit some thoughput degradation due to CPU switching via the Interprocessor Interrupt when I/O is performed. Multiple, I/O-bound, tasks will exhibit marginal rformance improvement due to the increases in total bus bandwidth (I/O is distributed among busses) and extra processors. The greatest improvement in system throughput results when multiple CPU-bound tasks are contending for available processors.

The effectiveness of the cache will be degraded in the multiprocessor environment due to the following factors:

- o The requirement to invalidate the cache contents whenever a processor enters the Executive.
- o The need to bypass the cache when a task references a shared region that contains volatile data.

In general, instruction execution will still benefit from the cache, since bypass is limited to that portion of the virtual address space that is used to map the shared region.

The user will be able to force the cache on when referencing common code or invariant shared data.

2.2 Support Objectives

To be defined later.

2.3 Environments

2.3.1 Minimum Configuration:

CPU:	11/70 with the following modifications: . Modified to access MK11 CSRs . Cache bypass . Cache invalidation . Lock instruction (ASRB)
Memory:	Multiport MK11 MOS memory, 128KB minimum. Must have DATIP cycle.
I/O:	All standard 11/70 devices available under RSX-11M V4 are supported. Dual-port device support is provided for the following: RP04/05/06 RM03 RK06/RK07 TU16/TE16 (via RH01 dual port adapter) TU45/TU78
Special Devices:	Interprocessor Interrupt and Sanity Timer, Time-of-Day clock, Manual MK11 port controls.
Bootstrap:	A special multiprocessor version of the M9301 is required.

2.4 RAMP Goals

See Appendix A.

2.5 Nongoals

- o Multiprocessing support in RSX-11S.
- o Support for non-cached multiprocessing as a fully tested, releasable, product.
- o Support for MA11 multiport memory (error logging and parity).
- o Device driver and data structure compatibility. All user and existing DEC-supplied drivers and data structures must be modified to run under RSX-11mP and RSX-11M V4.

3.0 FUNCTIONAL DEFINITION

3.1 Operational Description

See Appendix A.

3.2 Restrictions

See Appendix A.

4.0 COMPATIBILITY

RSX-11mP will be fully compatible with RSX-11M V4 as follows:

- o Nonprivileged tasks can be transported between systems without source code modifications and without relinking.
- o Privileged tasks can similarly be transported between systems without modification or relinking provided that the following is true:
    - . The task adheres to the established protocols for referencing the Executive data base.
    - . The task does not reference any data structures specific to multiprocessing.

Possible incompatibilities exist for tasks which now use running priority to block the execution of other tasks. User's will be warned of this condition in the RSX-11mP Multiprocessing Reference Manual.

RSX-11mP and RSX-11M V4 device drivers and data structures will not be compatible with Version 3. Driver changes will be of a clerical nature except for the following cases:

- o Extended functionality (such as overlapped seeks or dual port support) is incorporated.
- o The driver is called before the I/O packet is entered in the request queue (UC.QUE set).

We will fully document the procedures for converting existing drivers to run under Release 4.

DECNET V2 device drivers and associated code must include provisions for multiprocessing. RSX-11mP will not be transparent to DECNET.

5.0 INPUT/OUTPUT INTERACTION AND IMPACT

Communication between the multiprocessor Executive and the remainder of the system is in accordance with the provisions incorporated in RSX-11M V4.

6.0 ERROR REPORTS

See Appendix A.

7.0 PACKAGING AND SYSTEM GENERATION

See Appendix A.

8.0 DOCUMENTATION

e following new documentation will be produced:

RSX-11mP Multiprocessing Reference Manual - Derived from Appendix A.

The following documents must be modified:

RSX-11M Guide to Writing an I/O Driver

RSX-11M Task Builder Reference Manual

RSX-11M System Generation Manual

RSX-11M Executive Reference Manual

RSX-11M Operators Procedures Manual

RSX-11M I/O Drivers Reference Manual

9.0 USER EXAMPLES

See Appendix A.

APPENDIX A

RSX-11M MULTIPROCESSING REFERENCE MANUAL

APPENDIX A

RSX-11M MULTIPROCESSING REFERENCE MANUAL

CONTENTS

FIGURES

PREFACE

0.1 OBJECTIVES AND READER ASSUMPTIONS

This manual provides experienced implementors with the details required to design and implement systems that are based on the RSX-11M Multiprocessing architecture.

This manual is not self-contained. Previous experience with the RSX-11M Executive is essential to the understanding of concepts that are presented here. Basic to such understanding is familiarity with real-time, interrupt-driven systems, PDP-11 processors and related terminology.

0.2 STRUCTURE OF THE DOCUMENT

Chapter 1 provides a brief introduction to multiprocessing, discusses concepts that are fundamental to the implementation of multiprocessing under RSX-11M, and summarizes the capabilities provided by RSX-11mP.

..apter 2 describes system hardware configurations supported, along with guidelines for configuring the multiprocessor system.

Chapter 3 describes the Executive Services provided to support multiprocessing.

Chapter 4 describes the Executive I/O services, along with the related data structures and procedures for writing device drivers to run under RSX-11mP and RSX-11M V4.

Chapter 5 describes the provisions and procedures for error detection, online fault isolation, maintenance and repair of both hardware and software fault(s) contained in the Executive, and error logging system.

Chapter 6 describes the multiprocessing bootstrap and operating procedures.

Chapter 7 provides a description of the System Generation dialogue.

CHAPTER 1

MULTIPROCESSING CONCEPTS AND FACILITIES

1.1 DEFINITION OF MULTIPROCESSING

A multiprocessing system contains the following elements:

- o Two or more central processor units. Each unit has identical capabilities.
- o Main processor memory that is shared and accessible by all processors on the system.
- o A single operating system (the RSX-11mP Executive) in control of all processor and I/O resources.

All memory and I/O resources are equally available to all tasks, regardless of the processor on which the task is running.

The motivation for this architecture is the following:

- o To increase system availability by incorporating multiple units (CPU's memories and I/O devices) into a single integrated system. Substitution of good units for faulty ones by program-assisted reconfiguration allows system operation to continue after fault isolation and while repair is in progress.
- o To increase system performance by adding processors to achieve parallel task execution and by using additional resources to handle peak system loads.

2 OVERVIEW OF RSX-11MP

The multiprocessing system elements can be configured in a number of ways. The topology for the 11/70, shown in the simplified diagram in

Figure 1-1, was selected because it offers both increased performance and enhanced availability due to:

- o Multiple bus structure
- o Dual port device support
- o Switched bus configuration
- o Redundant system elements
- o I/O distributed among busses

The system requires the following basic hardware in addition to the minimum RSX-11M V4 11/70 configuration:

- o 128 KB of MK11 MOS memory with multiport memory controller
- o Manual port controls
- o 11/70 CPU with cache modifications, software lock instruction, and MK11 memory support.
- o Interprocessor Interrupt and Sanity Timer
- o Time-of-day clock
- o Multiprocessor bootstrap

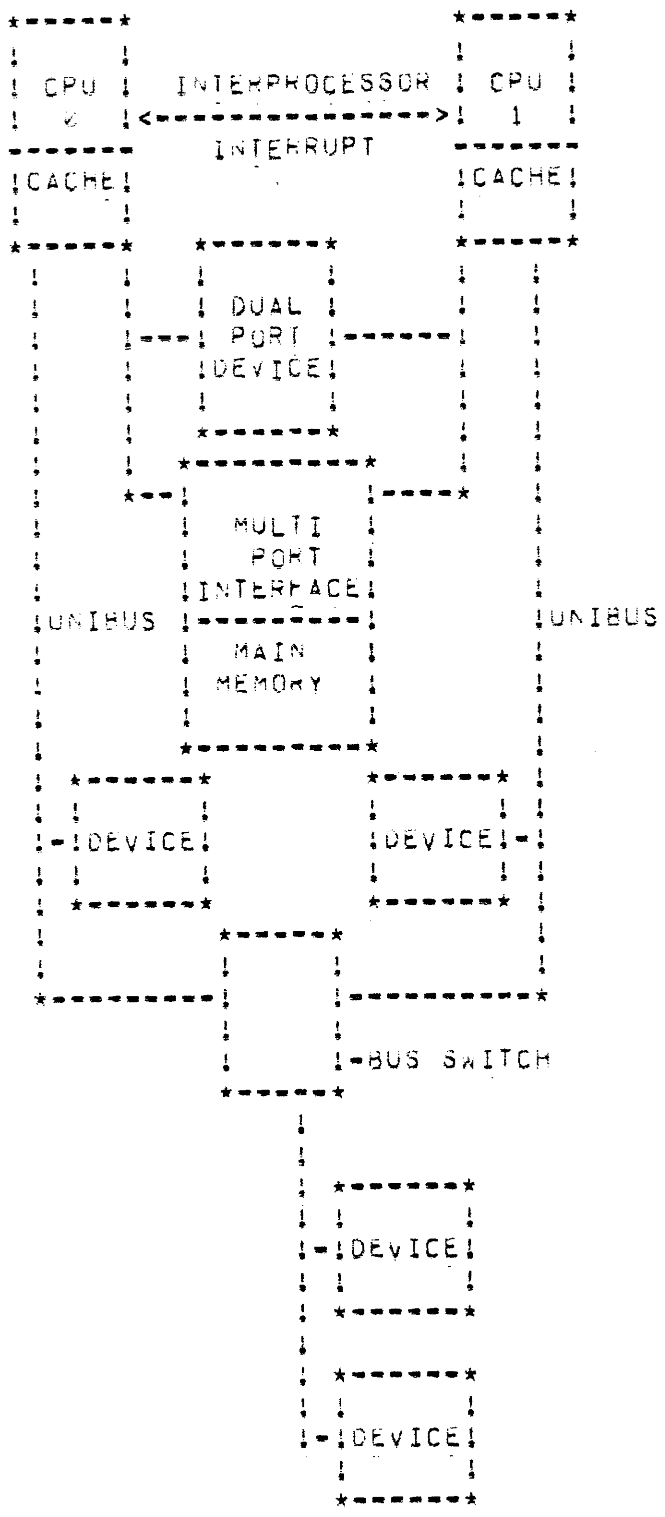


Figure 1-1
11/70 Multiprocessing Topology

1.2.1 Performance Features

Features that contribute to enhanced system performance are:

I/O and processor load sharing

Disk seek optimization*

Overlapped disk seeks*

I/O load sharing for dual port devices is accomplished by equalizing the data transfer load at each device port to reduce bus contention and increase effective bus bandwidth.

Shared access to main memory is achieved through a multiport memory controller that arbitrates among contending CPUs and devices. The characteristics of this controller permit instruction execution to be symmetrical and processor-independent. Program and memory-generated traps (parity errors, for instance) are only effected for the processor referencing memory when the trap occurred. As a result, any processor can service an executive request (EMT 377) or other synchronous traps. To prevent race conditions, concurrent access to Executive data structures is inhibited by a software locking chanism.

Unlike main memory, external page addresses are local to each processor. Thus I/O initiation and interrupt service can only occur on the processor having access to the device.

To allow the degree of I/O independence required for efficient resource utiliation, an interprocessor interrupt is provided to pre-empt execution of a CPU whenever access to a device on its unibus is required by the Executive. A task can request I/O service on any device regardless of the CPU on which the task is running. Except for the cases noted in Chapter 3 and 4, the system appears fully symmetric to the non-privileged task. The ability to distribute I/O while retaining symmetry at the task level offers the following advantages:

o Redundant data paths to dual port devices

o Interrupt processing distributed among all CPUs

o Effective I/O bandwidth is increased because fewer devices are contending for each bus.

Non-multiport unibus devices can be placed on the switchable bus. In the event of CPU failure, this bus can be transferred to another procesor for I/O service.

addition to I/O devices, the following elements are local to each

----

* Available without multiprocessor support.

processor:

- o General registers 0-7
- o Processor status (the PSW)
- o Memory management status
- o Cache memory

The state of all processors is completely independent. A processor that is halted, running with interrupts inhibited, or in system state in no way effects the operation of other CPUs.

In addition to increasing CPU speed, the effect of the private cache is to increase system throughput by reducing memory port contention. However, the caching effect can mask updates to data in shared memory that have been made by I/O devices and processes running concurrently on another CPU or by a single process that has switched CPUs (a task process or the Executive itself). The procedures for preventing 'stale' cache data are discussed in section 1.4.

1.2.2 Availability and Maintainability

RSX-11mP provides a number of features that contribute to enhanced availability, in addition to component redundancy. These include:

- o The ability to configure systems having different device types on the same massbus controller
- o Support for reconfiguration of most functional units under operator control to facilitate powered-up maintenance
- o Comprehensive error logging capability
- o The ability to decompose the system into two or more independent threads
- o The ability to load stand-alone diagnostics into an offline thread from a Files-11 volume
- o The ability to run online processor diagnostics by allowing a task to execute only on one processor.
- o The ability to specify explicite device-controller combinations when running online device diagnostics on dual port devices.

1.3 MULTIPROCESSOR OPERATION: NON-CACHED

The purpose of this section is to 'walk through' a simplified operational cycle (directive and I/O processing) in the multiprocessor Executive.

The system configuration is the dual processor 11/70 shown in Figure 1-1. One task is currently active on each processor (task A on processor 0, task B on processor 1). For the sake of simplicity, the effect of the cache is omitted in this description.

When task A issues a non-I/O directive, the EMT trap occurs on processor 0. This CPU enters kernel state and is vectored to the Executive code that handles synchronous traps. Since no I/O is performed, the Executive carries out the directive and returns control of the processor to task A. Execution of task B, on processor 1, is uninterrupted.

When task A issues a request for I/O, processor 0 enters system state as described. If the device can be accessed from CPU 0, the driver is called immediately to perform I/O. Task A can suspend execution until I/O completes, thus freeing the CPU for use by another task.

the device resides on processor 1, the Executive transfers control to this CPU (via the interprocessor interrupt) and releases CPU 0 for use by another task. The Executive, now running on processor 1, calls the driver to initiate the transfer.

As in the single processor system, the allocation of CPUs to tasks is interrupt driven. When I/O completes, the Executive will scan the list of active tasks, searching for the highest priority tasks that can run (including the null task). On each scan of the list, the Executive will start unblocked tasks until all processors are busy.

This simplified view of multiprocessing ignores the potential for race conditions that exists when two or more processors enter the Executive concurrently. Contention is resolved by a single software lock that blocks access to the Executive by all but one CPU.

Other processors assert lock requests until the owning processor opens the lock and returns to user state. Thus, contention is avoided by forcing all CPUs to access the executive data base in serial fashion.

Since Executive overhead is low, the time wasted in such collisions is small (typically 10% of CPU time under heavy processor load).

1.4 MULTIPROCESSOR OPERATION: CACHED

.e effect of the 11/70 private cache on multiprocessor operation is discussed in this section, along with the CPU modifications necessary to allow transparent cache operation.

The interposing of a cache between the processor and main memory can mask the CPU from data residing in backing store. In a multiprocessor environment, such masking can result in stale data being contained in one or more processor caches. The condition can occur in one of the following ways:

- o Task A and B, running in parallel and on different CPU's, access a global common block. A write to a location in this shared memory by task A causes the data in Task B cache to become invalid.

- o Task A is running on CPU 0, while I/O is being performed for it on CPU 1.

- o The Executive or a task, running serially on two processors creates stale data as follows:

    Read location X on CPU 0

    Write location X on CPU 1

    Read location X on CPU 0

ter the write on processor 1, the cache on CPU 0 contains stale data.

Another effect, caused partially by the current 11/70 memory system, is to invalidate the software locking mechanism described above. The absence of an uninterruptible read-modify-write cycle eliminates all test-and-set instructions, thereby requiring a more complex interlocking scheme.

To circumvent these problems, while retaining cache effectiveness, the following functional modifications to the 11/70 processor and memory system are required:

- o Provisions for software locking

- o An efficient cache bypass to allow shared data to be accessed by several parallel processes

- o A way to quickly invalidate the cache so a process can run sequentialy on several CPUs

Provisions for software locking have been implemented by:

- o Adding read-modify-write capability to the MK11 memory (DATIP cycle)

- o Modifying the ASRB instruction to bypass the cache and initiate the above memory cycle

Cache bypass is implemented as follows:

- o Bypass can be unrestricted-quickly turned on and off for short sections of code such as interrupt service routines.
- o Bypass can be restricted to parts of the task's virtual address space to allow non-cached references only for data shared between tasks.

Full cache bypass is used when short segments of code, such as interrupt service routines, must have guaranteed access to main memory. In this case, the degradation in performance is not significant. Bypass is, of course, unavoidable when two or more parallel processes are accessing a common data base. In this case, performance degradation is reduced by confining the virtual address space over which cache bypass is effected to the region containing shared data.

Total cache invalidation is necessary if a process is allowed to run serially on all CPUs. Because all context switching is performed by the Executive, the cache flush can occur without task intervention. The 'stale data' sequence described above is eliminated by the insertion of system-initiated cache invalidation as follows:

```
Read location X on CPU 0
(flush CPU 0 cache)

Write location X on CPU 1
(flush CPU 1 cache)

Read location X on CPU 0
```

The invalidation performed during the transition from CPU 0 to 1 ensures that the next reference to location X from processor 0 will correctly access main memory.

Extensive testing is planned to ensure that the extent of cache performance degradation is acceptable. The results of this testing will be used to determine if changes in the cache bypass and invalidation strategies are required in the Executive.

1.5 SUMMARY

The overview of multiprocessing presented in this chapter discussed the following elements and their counterparts in the PDP-11/70 multiprocessing system:

- o Two or more central processors, PDP-11/70 CPUs
- o A single operating system, RSX-11mP, in control of all processor and I/O resources
- o All memory and I/O resources available to all tasks, through

multiport MOS memory, the interprocessor interrupt, and multiport devices.

In the process of describing the essentials of the multiprocessing system, most details of the operating system and hardware have been omitted. The Executive is fully discussed in Chapter 3 and 4. Chapter 2 covers the specific hardware configurations that are supported and defines nomenclature that is used elsewhere in this manual.

CHAPTER 2

MULTIPROCESSOR SYSTEM CONFIGURATIONS

2.1 INTRODUCTION

In this chapter, conventions are established for identifying functional units. Minimum hardware configurations and restrictions are discussed and guidelines are presented for designing a ltiprocessor configuration.

2.2 NAMING CONVENTIONS

Logical device names, consisting only of device-mnemonic and logical unit number, are adequate when disks, magtapes and terminals are treated as anonymous resources for performing I/O. When processing device errors or issuing system reconfiguration commands, however, it is necessary to:

- o Identify both the device and its associated controller.
- o Identify all other functional units on the system (busses, memories, processors) that can be reconfigured under program control or that are implicated in errors reported to the error logger.

If the user is trying to isolate a fault on-line, he may want to disable I/O to one controller on a multiport device or one unit on a controller (by port). The procedure for naming functional units, described in the following paragraphs, allows the user to visualize the system as a collection of controller-port-device combinations for peripherals in a manner that facilitates system reconfiguration and unique identification of all major system components.

..e components to which these conventions apply are the following:

Processors

Bus Runs

Memory Boxes

Device Controllers

Physical Devices/units

Slave Units

2.2.1 Processors

Processors are identified as:

```
     CPnn   ,
```

where nn corresponds to the processor identification encoded in the Interprocessor Interrupt hardware or the wildcard symbol (*). If unspecified, a value of zero is assumed.

The wildcard symbol (*) refers to all processors on the system.

2.2.2 Memory

Memory boxes are identified as:

```
     MBnn   ,
```

where nn is a decimal number that corresponds to the relative position of the parity and control CSR registers for the box on the I/O page or the wildcard symbol (*). If nn is unspecified, a value of zero is assumed.

The wildcard symbol (*) refers to all memory boxes on the system.

Restrictions:

This convention only applies to systems containing all MK11 MOS memory.

2.2.3 Bus Run

A bus run electrically connects all controllers that are not separated by any bus reconfiguration devices such as the DT03 unibus switch.

Bus runs are of two types:

- o Processor run - A run that originates at a CPU
- o Switched run - A run that originates at a bus switching device.

Busses are identified as:

    Pnn

for a processor run or

    Snn

for a switched run

The bus number, nn, is a decimal value, or wildcard (*), that is determined as follows:

- o For an unshared bus, nn is the number of the processor to which the bus is attached.
- o For a shared bus, nn corresponds to the unit number of the DT03 that is used to switch the bus.

The wildcard refers to all bus runs of the specified type

Restrictions:

No attempt is made to distinguish between the unibus and 11/70 memory bus at this level. It is expected that both busses will have the same number in any architectural hardware labelling scheme.

The conventions described assume that there is one, and only one, CPU for each processor run.

2.2.4 Device Controllers

A device controller is identified as:

    <cn><id>   ,

where:

<cn>	is a two-character Ascii string specifying the controller name.
<id>	is a single letter (or wildcard (*)) specifying the controller identification. If unspecified, the letter A is assumed. The wildcard refers to all controllers of the specified type.

The two-character string corresponds to the standard device name (refer to Table 5-2) except for the following cases:

Mnemonic	Controller
RH	RH11/RH70 controller
YL	DL11-A-E and W single line interface when used as a terminal (TT:) interface
YH	DH11 16-line multiplexer when used as a terminal (TT:) interface
YM	DM11-BB modem control for the DH11.
YJ	DJ11 16-line multiplexer when used as a terminal (TT:) interface
YZ	DZ11 8-line multiplexer when used as a terminal (TT:) interface

e identification is determined by the order in which the controller s specified during system generation.

Examples:

The string:

RHA or RH

identifies the first RH70 controller specified during system generation. The second and third controllers are specified as:

RHB and RHC

The string:

DKA or DK

identifies the first RK11 controller specified during system generation. Subsequent controllers are identified as:

DKB, DKC, etc.

The string:

DBA

is undefined. All references to a massbus controller must use the mnemonic 'RH'.

2.2.5 Devices

A device is identified as:

    <dn> <id> <un>-<sl>:   ,

where:

<dn>	is a two-character Ascii string specifying the device name.
<id>	is a single letter (or wildcard (*)) specifying the identification of the controller to which the device is attached. The wildcard refers to all devices of the specified type.
<un>	is an octal number (or wildcard (*)) specifying the physical unit number of the device. This is the number us ed to address the device via the hardware registers. The wildcard refers to all units on the controller identified.
<sl>	is an octal number specifying the slave unit or the wildcard symbol (*). The numeric value corresponds to the number used to address the slave unit via the hardware registers. The wildcard symbol (*) refers to all slave units attached to the sub-controller.

All applicable elements of the name must be specified explicitly.

The slave unit specification is treated as follows:

- o a slave specification for an unslaved device is always in error. The string must be omitted.
- o The slave specification is required for slave devices.

The two-character string corresponds to the standard device name except for the following cases:

Memonic	Controller
YH	DH11 16-line multiplexer
YJ	DJ11 16-line multiplexer
YL	DL11 A-E single line interface
YZ	DZ11 8-line multiplexer

A device specification is distinguished from a controller by inclusion of the colon (:) delimiter in cases where ambiguity is possible.

The single letter I/D may correspond to the identification letter of the controller attached to any port of the device. In cases where the device is being referenced, I/D's may be used interchangeably. In cases where an explicit device-port combination is required or implied, the correct I/D must be used. These conditions will be specified in all commands and information displays described in this document.

Examples:

Unit 0 on RK11 controller A is identified as:

```
DKA0:
```

A single-ported RP04, attached to the second RH70 controller as unit 1, would be identified as:

```
DBB1:
```

A dual-ported RP04, attached as unit 1 to the first and second RH70 controllers, would be identified as:

```
DBA1:, or DBB1:
```

rU16 tape drive would be identified as:

```
MMB1-0:   ,
```

where:

B specifies the second RH70 controller

1 specifies the unit number of the TM02/TM03 formatter

0 specifies the slave unit number

A dual-ported TU16 connected to RH70 controllers C and D would be identified as:

```
MMC1-5:   or   MMD1-5:
```

The second line on DH11 multiplexer A is specified by the string:

```
YHA1:
```

To resolve ambiguities:

```
DKA   or   DK
```

r-fers to the first RK11 controller.

```
DKA:
```

is in error, since explicit unit specification must be given.

The following specification is illegal:

```
     DBA0-0:
```

the RP04/05/06 is not a slaved device.

2.3 SYSTEM CONFIGURATIONS SUPPORTED

2.3.1 Minimum Hardware Requirements

The minimum hardware needed for multiprocessor software support is the following (in addition to the basic RSX-11M V4 requirements):

- o 128KB of MK11 MOS memory. No other memory type is supported in the multiprocessor environment.
- o MK11 Manual Port Controls
- o Two PDP 11/70 processors (one CPU if in degraded mode) with the following modifications:

  Cache flush

  Cache bypass

  ASRB instruction modified for software locking

  MK11 MOS memory support
- o Interprocessor Interrupt and Sanity Timer
- o Time-of-day clock
- o Multiprocessor Hardware Bootstrap

If a programmable clock or line clock is on the system, one such device must reside on each processor.

NOTE

3 and 4-processor configurations will require a minimum of 256KB of memory

2.3.2 Maximum Hardware Configurations

The following are the maximum hardware configurations that can be supported by the RSX-11mP software:

Processors:	4 (one per memory port)
Memory Boxes:	8
Memory Size:	4.4MB addressable by the online system.
Bus Runs:	A maximum of 16 including all switched runs*

----

* An unswitched unibus run and memory bus attached to the same CPU count as one run.

Additional offline memory can be supported, provided the number of memory boxes does not exceed 8.

2.3.3 Device Support

In addition to the basic device support provided by RSX-11M V4, the following configuration features are available:

Dual port support for the following devices:

RM03

RK06/RK07

RP04/05/06

TU16/TE16/TM02/TM03 (with RH01 dual port massbus adapter)

Support for switched busses controlled by the DT03 FP bus switch

Support for different device types on the same RH70 controller (mixed massbus)

Except for the DT03 shared bus, multiprocessor support is not necessary in order to include the above features in the operating system.

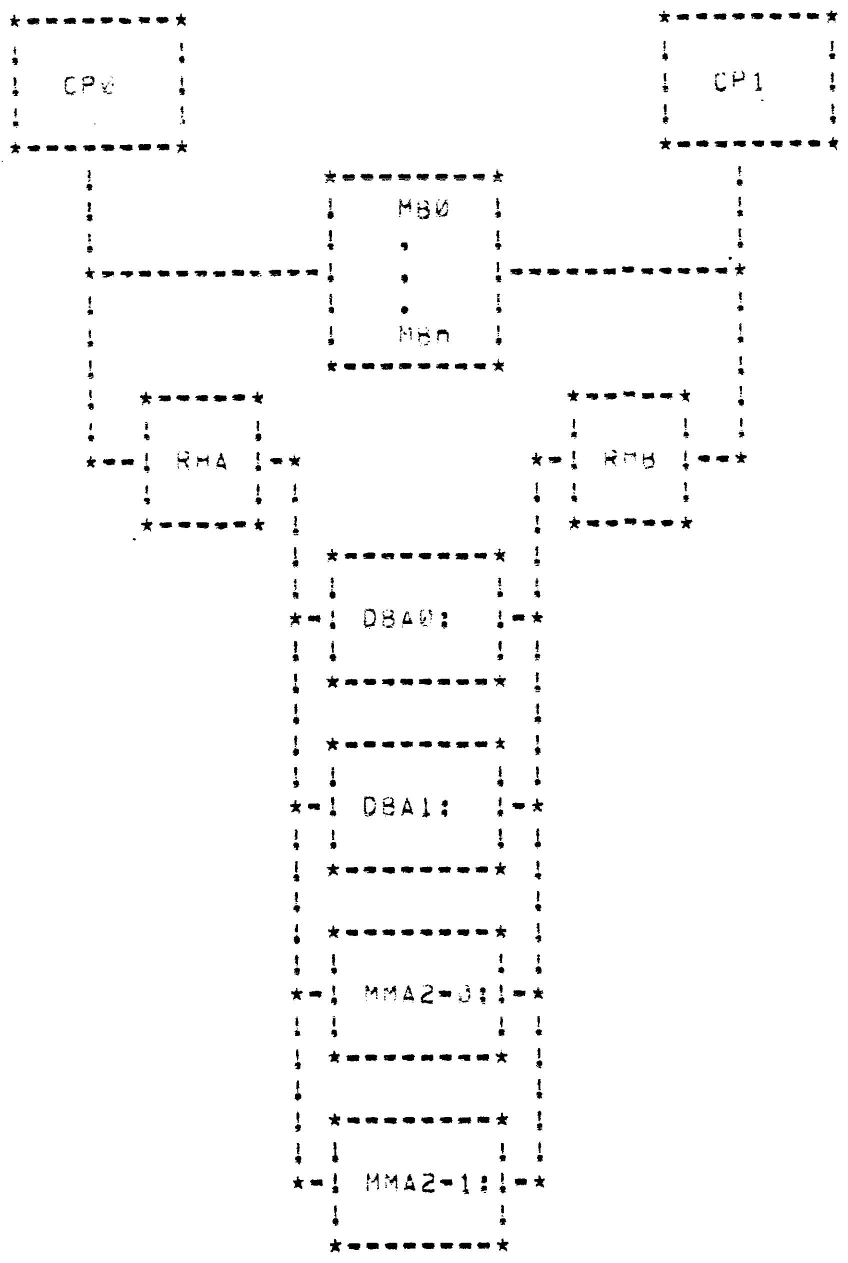
2.3.4 Mixed Massbus Configurations

Mixed massbus configurations provide full data path and device and controller redundancy with fewer controllers than an equivalent non-mixed configuration. The non-mixed dual-port configuration shown in Figure 2-1 requires 4 controllers to achieve full redundancy while the configuration shown in Figure 2-2 requires only two. In addition, more RH70 slots are available for expansion in the mixed massbus system.

```
*---------*                         *---------*
!         !                         !         !
!  CP0    !                         !  CP1    !
!         !                         !         !
*---------*                         *---------*
     !            *---------*            !
     !            !   MB0   !            !
     !            !    .    !            !
     *------------!    .    !------------*
     !            !    .    !            !
     !            !   MBn   !            !
     !            *---------*            !
     !  *-----*                 *-----*  !
     !  !     !                 !     !  !
     *--! RHA !-*             *-! RHC !--*
     !  !     ! !             ! !     !  !
     !  *-----* !             ! *-----*  !
     !          ! *---------* !          !
     !          ! !         ! !          !
     !          *-! DBA0:   !-*          !
     !          ! !         ! !          !
     !          ! *---------* !          !
     !          !             !          !
     !          ! *---------* !          !
     !          ! !         ! !          !
     !          *-! DBA1:   !-*          !
     !            !         !            !
     !            *---------*            !
     !                                   !
     !  *-----*                 *-----*  !
     !  !     !                 !     !  !
     *--! RHB !-*             *-! RHD !--*
        !     ! !             ! !     !
        *-----* !             ! *-----*
                !             !
                ! *---------* !
                ! !         ! !
                *-! MMB0-0: !-*
                ! !         ! !
                ! *---------* !
                !             !
                ! *---------* !
                ! !         ! !
                *-! MMB0-1: !-*
                  !         !
                  *---------*
```

Note: RH01 Dual Port adapter and TM02 Formatter are not shown.

Figure 2-1
Non-Mixed Massbus Configuration



Note: RH01 Dual Port adapter and TM02 Formatter are not shown.

Figure 2-2
Mixed Massbus Configuration

NOTE

In order to implement 11/70 mixed massbus support it is required that the RH70 bus address extension register (RHBAE) and status register 3 (RHCS3) be jumpered to appear at displacements of 50 and 52 (8) respectively from control and status register #1 (RHCS1) for all RH70 controllers in the system. This installation requirement is at variance from the practice of setting these registers adjacent to the set of device-dependent registers.

2.3.5 Compatibility

2.3.5.1 System Compatibility

cause of the register offsets and shared controllers described ove, mixed massbus configurations will be incompatible with all systems except RSX-11M V4. A user who is planning to run other operating systems on his hardware should not configure a system of this type.

We will supply procedures for converting Release 3 RSX-11M massbus drivers to be compatible with this register configuration so that existing V3 users can make the transition to RSX-11mP more easily. This procedure will allow these drivers to run with the new register offsets only. A single controller cannot be dynamically shared among different device types except under RSX-11M V4.

2.3.5.2 Task Compatibility

Existing tasks will be incompatible with the multiple bus structure under the following conditions:

- o The task references more than one device common block, and
- o Each device common block resides on a different bus.

Since I/O page address space is local to each CPU, it is possible that such tasks cannot be run in the multiprocessor system. The user must onfigure the system in advance to avoid such conflicts.

2.3.6 Multiple-Thread Systems

Under RSX-11mP, the user will be able to configure functional units into one or more fully decoupled systems consisting of:

- o An online system performing the primary application.
- o One or more offline systems running another version of the operating system or a stand-alone diagnostic.

The distribution of resources between threads is done through reconfiguration commands issued by the operator. The operating system will not automatically distribute or share resources among threads.

The components of the online system are defined as the functional units that are online when the system was SAVed and will usually constitute a subset of all available units. Upon startup, only those elements declared online will be allocated by the Executive. The operator (or the indirect command file referenced at startup) can add units to the online system as necessary to satisfy the anticipated workload.

3.7 Enhanced Availability Systems

In configuring a system for availability, the following design criteria must be considered:

- o The system configuration required to satisfy the steady-state workload requirements.
- o The number of added functional units required to achieve the availability goals.

If the capability for powered-up maintenance is desired, then the system should be separable into an online thread and an offline subsystem that can be used to run stand-alone diagnostics.

The minimum hardware requirements for stand-alone diagnostics, in addition to the device under test, are an offline thread consisting of:

- o One CPU
- o One terminal and interface
- o One Memory Box (at least 64KB capacity)
- o Optionally, one diagnostic load device.

The system will be capable of loading a stand-alone diagnostic from a Files-11 volume into the memory box dedicated to the offline thread.

In this case, a portion of the physical address space visible to the online system (the contents of one box) is used as a window into the diagnostic configuration.

2.3.8 Clock Configurations

In addition to the time-of-day clock, the system will also support the programmable clock (KW11-P) and the line clock (DL-11W or KW11). When either of these clocks is specified at system generation it becomes the preferred time standard for the system.

The clock configuration must be symmetrical; each processor must have an operable clock of the specified type (each set to the same frequency). The CSR and vector addresses must be identical for each processor.

2.4 ALLOCATION OF CSR AND INTERRUPT VECTOR ADDRESSES

In the multiprocessor system, each CPU has its own I/O page and interrupt vector addresses. It is possible, therefore, for devices to have the same CSR or vector addresses if the devices reside on different processor busses. Conversely, the CSR and vector address associated with a given device type can be different for each processor.

In general, we will allow the user maximum flexibility in the assignment of addresses, subject to the usual hardware constraints and the following restrictions:

- o The following devices must be connected to the same CSR and interrupt vector addresses on each processor:

  - . The MK11 port controller
  - . Each controller interfaced to a given DT03 FP bus switch
  - . The line clock or programmable clock
  - . The time-of-day clock and Interprocessor Interrupt

- o The RH70 bus address extension register (RHBAE) and status register 3 (RHCS3) must be jumpered to appear at offsets of 50 and 52(8) respectively from control and status register #1.

- o The same interrupt vector and CSR space must be reserved on each processor for all devices that are attached to a switched bus.

CHAPTER 3

EXECUTIVE FUNCTIONAL EXTENSIONS FOR MULTIPROCESSING

3.1 INTRODUCTION

This chapter describes the Executive functional extensions required for multiprocessing.

ɔ.2 GOALS

The modifications discussed in the following sections provide the following capabilities:

- o Allow a non-privileged task to run in a multiprocessor environment without re-linking and without modifications to the source code.
- o Provide a similar capability for privileged tasks that adhere to the current protocols for interfacing with the Executive and are not sensitive to changes pertaining to multiprocessing in the Executive data structures.
- o Provide an interface for non-privileged tasks that allows online processor diagnostics to be run.
- o Provide user-level access to the cache bypass modifications.

3.3 STRUCTURE OF THE MULTIPROCESSING EXECUTIVE

The multiprocessing Executive consists of the following elements:

ɿmmon memory containing:

- o Re-entrant code

- o Shared data and system dynamic core pool

Memory reserved for each processor, that contains:

- o Processor-specific impure data (current task, processor stack interrupt vectors, etc.) that is referenced by re-entrant executive modules or privileged tasks

- o Multiple copies of system trap processing and exception handling code

The allocation of physical and virtual memory is shown in figure 3-1. 8KB of reserved memory, mapped by Kernel APR0, is required for each processor. This memory is allocated from the multiport, shared memory available to the entire system. Since the mapping registers are local to every CPU, processor context is automatically maintained by setting each processor's kernel APR0 to the appropriate 8KB segment of main memory.

The Partition Control Block defining this allocation is hardwired into module SYSTB.MAC and is enabled whenever multiprocessor support is generated into the Executive.

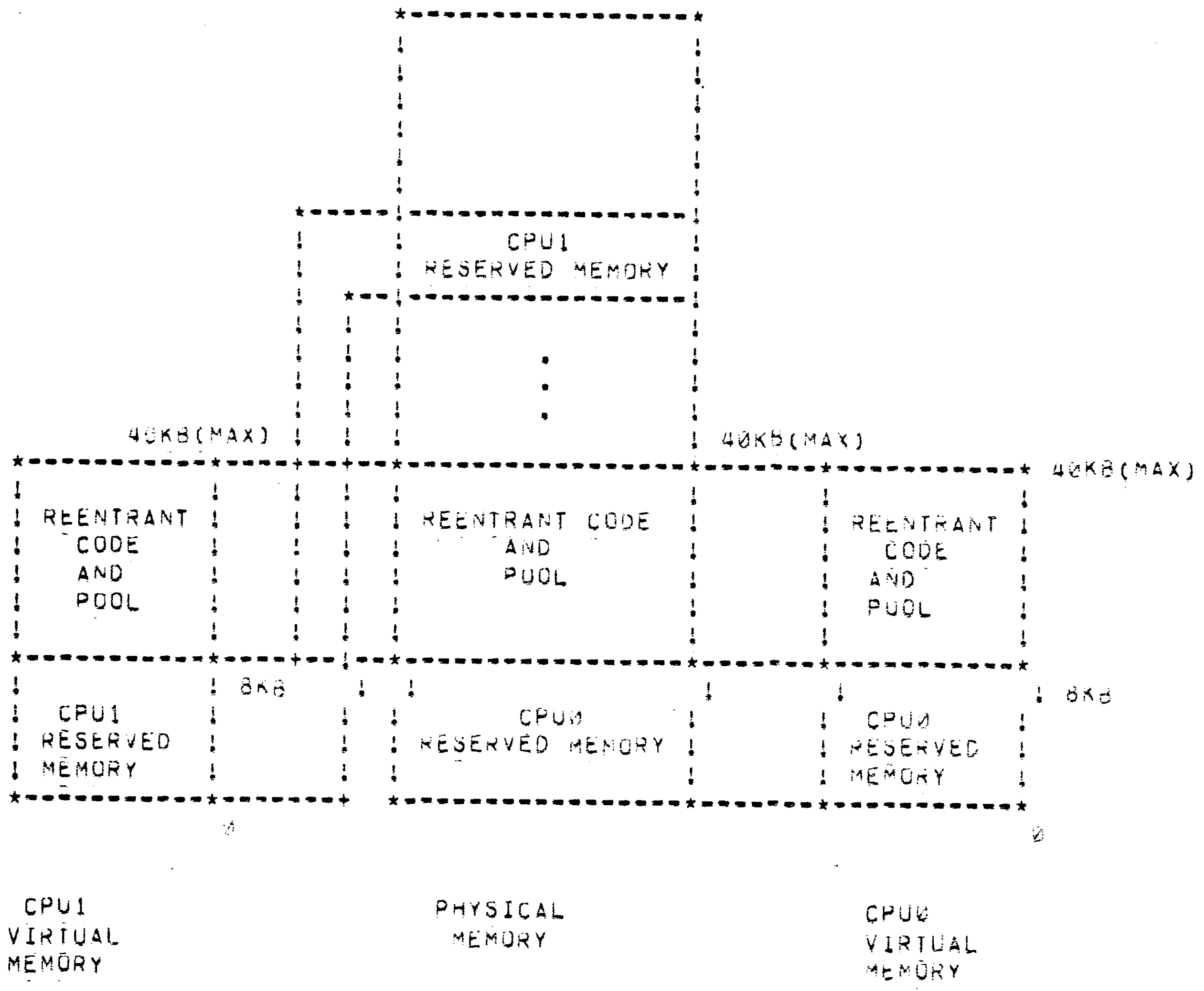


Figure 3-1
Executive Memory Allocation

Space for reserved memory is allocated as follows:

- o The area of physical memory from 0 to 8KB is reserved for the processor from which the system was booted.
- o One task-controlled partition that contains the impure data for all additional processors is allocated.

The necessary partition space and data structures are setup (or eliminated) by the VMR function:

```
SET /(NO)EXEPAR = base    ,
```

as described in Chapter 6. The command processor will fill in the appropriate portions of the preallocated partition control block, set the block busy, and thread it into the PCB list.

The partition name assembled into the PCB will contain an embedded blank to prevent it from being deallocated by MCR. The 'SET /(NO)EXEPAR' function will only exist in VMR.

3.4 MULTIPROCESSING FUNCTIONAL EXTENSIONS

3.4.1 External Modifications

This section describes the extensions to the Executive that effect the operating system interfaces used by both privileged and non-privileged tasks.

3.4.1.1 Task-CPU Affinity

Task-CPU affinity is provided to allow online processor diagnostics to be run, and to allow manual, application-specific, load balancing in the rare case where it is desirable. The effect of this attribute is to restrict task execution to one specific processor. Processor affinity is established in one of the following ways:

- o A 'Request' or 'Run' directive can be issued specifying the affinity attribute and processor on which the task is to execute.
- o The task can request or cancel processor affinity for itself.

The 'Request' and 'Run' directive modifications are as follows:

QUEST:

Fortran Call:

```
     CALL REQUES (tsk, [opt], [ids])
```

where:

tsk	= Real variable containing the task name in radix 50 format.
opt	= 4-word integer array
opt(1)	= Partition name first half (Radix-50)
opt(2)	= Partition name second half (Radix-50)
opt(3)	= Priority in bits 0-7. Bits 8-15 specify the affinity flag and processor number as follows: Bit 15 = 1 implies task-CPU affinity Bits 8-14 specify the processor number
opt(4)	= User identification code
ids	= Directive status

Macro Call

```
     RQAF$   tsk, [prt], [pri], [ugc], [umc], [aff]
```

where:

```
     tsk = task name (4)
     prt = partition name (4)
     pri = task priority (1)
     ugc = UIC group code (1)
     umc = UIC member code (1)
     aff = affinity flag and processor number (1)
```

Macro Expansion

```
     RQAF$   ALPHA,,,20,10,200
     .BYTE   11.,7    ;RQST$ MACRO DIC, DPB SIZE=7 WORDS
     .RAD50  /ALPHA/  ;TASK 'ALPHA'
     .WORD   0,0      ;PARTITION IGNORED
     .BYTE   0,200    ;PRIORITY IGNORED, CPU AFFINITY TO PROCESSOR 0
     .BYTE   10,20    ;UIC UNDER WHICH TO RUN TASK
```

Local Symbol Definitions:

```
     R.QSTN - Task name (4)
     R.QSPN - Partition name (2)
     R.QSPR - Priority (1)
     R.QSAF - Task affinity (1)
```

R.QSGC - UIC group (1)
R.QSPC - UIC member (1)

DSW Return Codes

IS.SUC	Successful completion
IE.UPN	Insufficient dynamic memory
IE.INS	Task is not installed
IE.ACT	Task is already active
IE.ADP	Part of the DPB is out of the issuing task's address space
IE.SDP	DIC or DPB size is invalid
IE.NPR	Nonexistent processor specified

Notes:

1. Task-CPU affinity is ignored in a non-multiprocessor system.

2. The remainder of the affinity byte is ignored when bit 7=0.

3. The changes described are an extension of the existing directive. The new macro is provided to allow addition of the affinity attribute without introducing source code incompatibilities.

RUN:

Fortran Call

```
     CALL RUN (tsk, [opt], [smg], snt, [rmg], [rnt], [ids])
```

where:

tsk = Real variable containing task name in radix-50 format

opt = 4-word integer array containing the following:

> opt(1) = First half of radix-50 partition name
>
> opt(2) = Second half of radix-50 partition name
>
> opt(3) = Priority in bits 0-7. Bits 8-15 specify the affinity flag and processor number as follows:
> Bit 15=1 implies task-CPU affinity
>
> Bits 8-14 specify the processor number
>
> opt(4) = User identification code

smg = Schedule delta magnitude

snt = Schedule delta unit

rmg = Reschedule interval magnitude

ids = Directive status

Macro Call:

```
     RNAF$ tsk, [prt], [pri], [ugc], [umc], [smg], snt, [rmg], [rnt],
           [aff]
```

where:

tsk = Radix-50 task name (4)

prt = Radix-50 partition name (4)

pri = Priority (1)

ugc = UIC group code (1)

umc = UIC member code (1)

smg = Schedule delta magnitude (2)

snt = Schedule delta unit (2)

rmg = Reschedule delta magnitude

rnt = Reschedule delta unit

aff = Processor affinity flag and CPU number (1)

Macro Expansion

```
RNAF$   ALPHA,,,20,10,20.,3,10.,3,201
.BYTE   17.,11   ;RUN$ MACRO DIC, DPB SIZE=11 WORDS
.RAD50  /ALPHA/ ;TASK 'ALPHA'
.WORD   0,0      ;PARTITION IGNORED
.BYTE   0,201    ;PRIORITY IGNORED, TASK AFFINITY WITH CPU 1
.BYTE   10,20    ;TASK UIC
.WORD   20       ;SCHEDULE MAGNITUDE=20.
.WORD   3        ;SCH. DELTA TIME UNIT=MINUTE(=3)
.WORD   10.      ;RESCHEDULE INTERVAL MAGNITUDE=1.0
.WORD   3        ;RESCHEDULE INTERVAL UNIT=MINUTE (=3)
```

Local Symbol Definitions:

R.UNTN - Task name (4)
R.UNPN - Partition name (4)
R.UNPR - Priority (1)
R.UNAF - Task affinity (1)
R.UNGC - UIC group code (1)
R.UNMC - UIC member code (1)
R.UNSM - Schedule magnitude (2)
R.UNSU - Schedule unit (2)
R.UNRM - Reschedule magnitude (2)
R.UNRU - Reschedule unit (2)

DSW Return Codes

IS.SUC -- Successful completion
IE.UPN -- Insufficient dynamic memory
IE.INS -- Task is not installed
IE.ITI -- Invalid time parameter
IE.ADP -- Part of the DPB is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid
IE.NPR -- Nonexistent processor specified

Notes:

1. Task-CPU affinity is ignored in a non-multiprocessor environment.

2. The remainder of the affinity byte is ignored when bit 7=0.

3. The changes described are an extension of the existing directive. The new macro is provided to allow addition of the affinity attribute without introducing source code

incompatibilities.

Compatibility:

The changes to the Run and Request directives do not cause incompatibilities with existing code. Under RSX-11D and IAS, the entire priority word is tested for validity (must be less than 251). This is equivalent to requesting the task in non-affinity mode.

Under Release 3 of RSX-11M, priority is not recognized, hence no check on this parameter is made. Tasks running only under RSX-11M V3 can possibly fail to execute in the multiprocessor environment. The potential for problems of this type will be documented. Anyone who currently uses DEC-supplied Macros to generate Executive directives will have no compatibility problems.

A task requesting affinity should be prepared to re-issue the directive with the affinity byte cleared when running under RSX-11D or IAS, since the directive will be rejected with the error code IE.PRI

Requesting and Cancelling Processor Affinity:

Processor affinity is requested (or cancelled) for the issuing task by means of the following new directive:

Set Processor Affinity:

Fortran Call

```
     CALL SPAF (iaff, [ids])
```

where:

iaff = Affinity flag (bit 7) and processor number (0-15).

ids = Directive status.

Macro Call:

```
     SPAF$S aff
```

where:

aff = Affinity (2)

Macro Expansion

```
     SPAF$S #201, ERR
     MOV     #201,-(SP)       ;REQUEST CPU 1 AFFINITY
     MOV     (PC)+,-(SP)      ;SPAF$S MACRO DIC, DPB SIZE=2 WORDS
     .BYTE   ???,2
     EMT     377              ;ISSUE DIRECTIVE
     BCS     ERR               ;BRANCH TO ERROR ROUTINE
```

Description:

This directive is used to request or cancel CPU affinity for the issuing task. If bit 7 is set, the remainder of the low byte is interpreted as the processor to which affinity is requested.

On successful completion, the task will be running on the specified CPU. Bit 7 is cleared to cancel affinity. In this case, the remainder of the affinity parameter is ignored.

Requesting affinity to an non-existent or offline CPU results in the directive being rejected.

Local Symbol Definitions:

S.PAF -- Affinity attribute (2)

DSW Return Codes:

IS.SUC -- Successful Completion

IE.SDP -- DIC or DPB size is invalid

IE.NPR -- Nonexistent processor specified

IE.OFL -- Requested processor is offline

Notes:

1. When the task is aborted, the affinity attribute is removed to allow normal termination processing.

2. An affinity task may be blocked indefinitely if the processor is placed in an offline state.

3.4.1.2 Task-Bus Affinity

To faciliate non-interrupt I/O, tasks may currently link to common blocks that reside on the I/O page, thus allowing the task to reference device registers without issuing a QIO. Since a device can reside on a switchable bus, task-bus affinity allows a referencing task to automatically migrate to the processor controlling the bus whenever the switch position is altered.

The affinity attribute is contained in the device common block PCB and automatically propagated to the task. There will be no directives ɔr modifying bus affinity.

The system will detect and prevent lock-up due to conflicting bus and

CPU affinity requirements. Bus affinity will override CPU affinity.

Task-Bus affinity provides a mechanism for implementing memory local to each CPU. This potential will not be exploited in the first release of RSX-11mP. Initially, task-CPU affinity will be implemented internally as task-bus affinity.

3.4.2 Powerfail Processing

A powerfail condition sensed by one processor will cause the entire online system to shut down. Operation will not resume until power recovery has been effected for all processors.

The Executive must perform power recovery for all switched busses (by calling the DT03 FP driver) before calling any drivers at their powerfail entry points.

A 1-second delay is required to ensure that each bus has had time to reconnect to a processor.

ɜ.4.3 Interprocessor Interrupt and Sanity Timer Support

The Interprocessor Interrupt is the vehicle for tranferring execution of the operating system from one CPU to another in a timely manner. This feature is used primarily for I/O processing. The Executive interface is discussed in Chapter 4.

The sanity timer will be serviced by the Executive on each processor in step with the system clock. The occurrence of a sanity timer interrupt on any processor will crash the online system.

3.4.4 System Timekeeping Functions

All multiprocessing systems must have a time of day (TOD) clock. Optionally, the programmable or line clock can also be specified at system generation.

The processor from which the system was booted is responsible for performing all timekeeping operations. All other processors must recognize clock interrupts in order to service the sanity timer.

To guarantee program compatibility, the line clock or programmable ˈock, if specified at system generation and present on the system, is ɔnsidered the primary time base. The TOD is used to set system time following a power outage or bootstrap.

The TOD becomes the time base when no other clock was defined at system generation or when the specified clock is not on the host system.

3.4.4.1 Time of Day Service

The TOD is used to provide the following services:

- o It establishes absolute system time whenever the system is booted or executes power recovery code.
- o It is the system time base when no other device has been specified.

NOTE

The following is dependent on the specific TOD implementation.

the former case, the time is measured in 'ticks' past the base year. Base year is permanently recorded in the system image through the VMR 'TIM' function.

In the latter case, the system accumulates time in response to the 10 ms TOD interrupts.

The absolute time registers are only referenced when setting or reporting TOD status or resetting system time under the conditions described above.

The MCR TIM function will always attempt to set the TOD when current time is specified. If the hardware is write-protected, the message:

```
TIM - TIME-OF-DAY CLOCK NOT UPDATED
```

will appear on the operators console as well as the requestor terminal. The time maintained by the operating system will reflect the new value. However, there will be a discrepancy between the hardware and operating system time.

If a change in 'base year' is implied in an MCR TIM command, the operator will be notified (the time is set as requested). VMR can then be invoked to update the system image.

A display of time as recorded by the TOD must be requested explicitly; herwise only the software-maintained values are shown. The changes [illegible] MCR syntax for reporting TOD status are described in Chapter 6.

3.4.4.2 Restrictions

The events on the system time queue will not be rescheduled following a power outage. All scheduled activities will slip by the time interval over which the system was inoperative.

3.4.5 Cache Bypass Support

User access to the cache bypass feature is required for shared regions accessed by tasks running on different processors. Bypass support will be provided in the memory management directives. In addition, the MCR SET /MAIN command will be modified to allow the bypass attribute to be attached to static common blocks. INSTALL will copy the attribute when building the header for referencing tasks.

3.4.5.1 Memory Management Directive Support for Cache Bypass

Tasks will be permitted to enable or disable cache bypass in the Create Region directive and directives that map the task into a gion, namely;

Create Address Window (CRAW$)

Map (MAP$)

Receive-by-reference (RREF$)

The Create Region directive will be modified to allow cache bypass to be specified explicitly as a region attribute. The bypass attribute is defined by flag RS.CON in the region descriptor flags word. This bit, when set, specifies that the cache will be enabled by default whenever a portion of the region is mapped into any referencing task. Conversely, the cache is bypassed when RS.CON is reset. The default is 'cache bypassed' (RS.CON reset).

The Attach Region and Send-by-Reference directives will be modified to reject access to a region that is attached to any task with write-access which does not have cache bypass enabled. In this case, the directive is rejected with the DSW code IE.PRI. Write-access is always permitted for the owning task.

A special case is the task region itself. It may be read/write and still sent-by-reference to another task. While the receiving task is mapped, both tasks will operate on the region in bypassed mode. When e task which receives it unmaps it, the task that is executing in .e task region will have its cache reenabled. For instruction fetches, a 50% minimum hit rate exists with the cache enabled (avg. around 90%), and does not exist when it is bypassed. For this reason, the user should minimize the time in which he executes with the cache bypassed.

Cache bypass in the mapping directives is controlled by flag bit WS.CON in the window descriptor flags word (W.NFLG). Setting this bit to 1 forces the cache to be enabled regardless of the region attribute. Setting the bit to 0 defaults the cache state to the condition specified in the Create-Region directive (or the MCR SET /MAIN command).

3.4.5.2 MCR Support For Cache Bypass

The MCR command SET /MAIN will be modified as discussed in Chapter 6 to attach the bypass attribute to common blocks. The default will be 'cache bypassed'.

Install will copy the attribute to the window block in the header of each referencing task.

5 CRASH PROCESSING

The operating system will implement the following provisions for containing the damage inflicted by a faulty processor or operating system malfunction.

- o Redundant copies of the crash processing software will reside in the private area of memory allocated to each CPU.
- o If more than one processor is online, the processor detecting a fault will halt, allowing the sanity timer to notify the remaining CPU's of an error condition. This will minimize reliance on bad functional units.

Sanity timer interrupts always terminate system operation.

Crash processing consists of dumping the system image to a mass storage device for analyses. A comprehensive error formatter will be written to assist in diagnosing software-related errors.

3.6 MEASUREMENTS REQUIRED

Certain specific times and events are critical to the performance of the system. Accurate measurement of these times is necessary to aid th in system design and potential adjustment to unforseen _rcumstances. Certain gross measurements may also be necessary for proper sales and marketing of these systems (e.g., "A four processor system has roughly the power of n.m single independent systems")

3.6.1 Specific Measurements

(To be supplied).

CHAPTER 4

I/O PROCESSING

4.1 INTRODUCTION

This chapter presents changes to the RSX-11M I/O structure required to support a number of new capabilities, and is directed toward a reader already familiar with RSX-11M I/O processing. More information on the rrent I/O algorithms and data structures may be found in the RSX-11M Guide to Writing an I/O Driver.

Sections 4.1 through 4.5 in this chapter give some preliminary terms and concepts necessary to a complete understanding of the chapter. Sections 4.1 and 4.2 provide a general overview of the changes. Sections 4.3 and 4.4 are the equivalent of a Design/Functional Specification for these changes.

4.1.1 Other Design Issues

This chapter does not yet address the following design issues:

1. Interprocessor interrupt (II) support and associated mailboxes.

2. Seek optimization.

3. Need for dynamic unibus mapping register (UMR) allocation.

4.1.2 Philosophy

general, the design presented here will make changes in a manner hat will impact existing code, both in-house and user-written, as little as possible. Changes will be necessary to all drivers, and will be extensive for those drivers intended to support new functionality. In addition, this design will require changes to all previously created I/O data bases.

Many of the changes presented are designed to bring the RSX-11M I/O data structures more closely into correspondence with the physical reality of the hardware configuration. One goal, for instance, is to have exactly one Controller Request Block (KRB) for each physical device controller. (The KRB is defined in Section 4.4.2).

4.2 DEFINITION OF TERMS

4.2.1 Unibus Run

A "Unibus Run" consists of a group of devices all of which are electrically connected to the same unibus, and which are not separated by any bus reconfiguration devices (such as the DT03 Unibus Switch). We will refer often to a "Unibus Run Mask", a 16-bit word that contains a bit for every unibus run present in the mask. Unibus runs are numbered from 0 to 15, and the system is restricted to a maximum 16 unibus runs. Bit zero corresponds to unibus run zero, bit one run one, etc.

Limiting the total number of unibus runs will probably not cause serious restrictions on the maximum size of multi-processor systems. A four-processor system would typically have eight unibus runs; four primary runs containing processors and four secondary runs consisting of the switched busses. There will be a limited number of places in the system where unibus runs will be referenced. These will be carefully documented and indexed to allow future expansion above the current maximum of 16, if this is required.

4.2.2 Multiple Access Paths to Devices

We distinguish between multiport and multiaccess devices. A multiport device is one which may transfer data via a number of different data paths, but which has only one control path. In terms of unibus peripherals, a data transfer path is an NPR unibus interface, and a control path is a Control and Status Register interface. Any device attached to an RH11 Massbus controller is automatically dualport since the RH11 has two data transfer paths and one control path.

A multiaccess device is one which may transfer data over a number of different data paths, and may also be controlled over multiple access ths. An RWP04-BA dual controller RP04 is an example of this type of vice, since each dual access unit may interface to one of two completely independent controllers.

In general, a multiport device is used for an increase in performance, while a multiaccess device is used to increase both performance and availability, since in a multiaccess scheme the controller is not a single point of failure.

4.2.3 Overlapped Seeks

Overlapped seek support is software support of a hardware feature found in most advanced disk drives. This feature allows a number of drives attached to the same controller to all execute a seek function simultaneously, although only one data transfer operation may occur at any one time. Some drives allow seeks to overlap data transfers, while others do not.

4.2.4 Data-Late Errors

Data-late errors, not due to faulty hardware, occur whenever an NPR device is unable to transfer a word to or from memory in time to prevent loss of information. This error may be due either to failure gain control of the unibus, or the inability of the memory to ansfer the word in time. Most device controllers have "silo" memories to increase their tolerance to occasional system bandwidth overloads. Devices that have higher transfer rates usually have larger silos to offset their increased vulnerability to data-late errors.

4.2.5 Seek Optimization

Seek optimization is a technique that attempts to increase disk throughput in a system where the average length of the disk input request queue is large. Whenever a new request is required, the input queue is examined to determine the request that will cause the shortest disk arm movement (seek). Obviously, any throughput increase is at the expense of an increase in variance of the wait time for an individual request.

4.3 FUNCTIONAL SPECIFICATION

Seven functions to be supported under Release 4 of RSX-11M and RSX-11mP are discussed in this section. While multiprocessing support is the motivating factor in some cases, support will usually be ailable under RSX-11M V4.

4.3.1 Distributed I/O

In a multiprocessor system, peripheral devices are in general accessible to only one unibus run. While multiaccess devices exist, not all devices have multiaccess capability, and those that do will not necessarily be accessible from every unibus. A method must therefore be found to ensure that when a driver accesses a controller, it is currently executing under control of a processor in whose I/O space the controller registers reside. We plan to allow processors to remain anonymous as far as task execution is concerned. Execution of a user task will not be restricted to the processor associated with a device to which the task directs I/O.

4.3.2 Multiaccess I/O

Devices exist that have multiple access paths for both control and data transfer. A single unit may only be accessed from one port at a time. Changes, therefore, must be made to the data structures to represent this situation, and must be made to the code to exploit the additional access path. We will support dual-access operation for those devices equipped with the necessary hardware functionality. rt switching will be completely dynamic and under control of the system.

4.3.3 Overlapped Seek I/O

Software support of overlapped seek capability is not necessarily an RSX-11mP goal. However, since it will be added to Release 4 of RSX-11M, the changes made to support distributed and multiaccess I/O should not preclude overlapped seek support. Furthermore, most devices that support dual access will also support overlapped seeks. It is felt that changes made to the I/O structures for multiprocessing support interact so heavily with changes necessary to support overlapped seeks that an integrated approach to implementation is appropriate. The major difficulity in implementing overlapped seek support involves the need to have multiple device units in execution at the same time, and the need to represent this situation within the currently existing I/O data structures. Additional space is needed to store the increased driver process context necessary to support multiple units in operation at the same time on the same controller.

4.3.4 Mixed Massbus Support

We plan to offer mixed massbus support as a standard feature in RSX-11M. Mixed massbus support refers to support of the RH controller with different device units attached to the massbus; for example, an RH70 with RP04s and TM02/TU16s both attached to its massbus.

The support will take a different form than the support currently available in RSX-11M V3 in order to be compatible with the redesign of the data structures. Mixed massbus support for the PDP-11/70 will require further modifications to overcome the problems created by the placement of the additional controller registers present on the RH70.

4.3.5 Seek Optimization

Seek optimization for moving-head disk storage units will be offered as an option of RSX-11M V4. It will only be available on devices for which overlapped seek support has been selected. It will always be active if the feature has been selected during SYSGEN.

Seek optimization will be transparent to the device drivers, and will involve modifications to the queuing and dequeuing of I/O packets for a particular unit.

The SSTF/SLTF (shortest seek time first/shortest latency time first) algorithm will be implemented, with an "age" count kept to allow us to implement a fairness doctrine. The purpose of the fairness doctrine is to place an upper bound on the maximum waiting time that a specific)O request will experience. To put it another way, we will prevent a particular request from being "locked out" by a sustained high frequency arrival of requests that are all to one area of the disk.

The priority structure of the current RSX-11M I/O subsystem will be maintained, with two limits set at SYSGEN, between which all requests will be considered of the same priority. The FIFO nature of requests for a specific file or device from a specific task will also be preserved.

4.3.6 Augumented Error Logging

New capabilities will be added to the error logger to provide what is called an activity report whenever an error occurs.

Changes will be necessary to the I/O processing structures to properly tag the I/O packets that are active (on devices that support error logging) when an error occurs on any device. These I/O packets will be retained in the system until the error logger has the opportunity to extract desired information from them. An upper limit on such "old" I/O packets will be set, and, when the limit is reached, packets will be thrown away.

.3.7 Dynamic UMR Allocation

Currently, the Unibus Mapping Registers (UMRs) on the 11/70 are statically allocated to a device at SYSGEN time. In a multiprocessor system, where a unibus NPR device may be connected to a switched bus, static allocation of UMRs is not practical. Therefore, dynamic allocation will be instituted. A driver will simply hang if it requests a UMR and none are available.

4.4 GENERAL I/O STRUCTURE CHANGES

To support the new capabilities discussed in the previous section, a number of changes have been made in the I/O data structures. This section will describe theses changes in general. Section 4.5 describes how each new capability will be implemented in the context of these new structures.

The key data structures that are of importance at the driver level are the UCB, SCB, and, to a lesser extent, the DCB. In our new scheme, the UCB and DCB do not change significantly, while the SCB is extensively reorganized.

4.1 The Overlapped Seek Problem

As currently implemented, the RSX-11M I/O processing mechanism is designed to allow one active unit per controller. Multiple controllers may be active concurrently, yet only one unit per controller may be active. This structure precludes any reasonable implementation of overlapped seek support, which has as its basic goal the maximumization of simultaneous operation of the units on one controller. It also makes it more difficult than necessary to do efficient implementations of other new capabilities.

The primary reason for this serialization is that the storage for the driver process context is contained in the SCB, which also contains the controller specific information. Since there is "one SCB for each controller in the system", there can be only one unit active per controller. The specific driver process context referred to includes the I/O packet address, the timeout counts, and the fork block.

4.4.2 Data Structure Changes for Overlapped Seek

It has been decided to divide the SCB into two parts -- one for driver process context storage, and one for controller/hardware specific information. The control block that contains the driver context will still be called the SCB, and still be pointed to by U.SCB. The new ntrol block that contains the controller specific information such the CSR and vector address will be called the Controller Request Block, or KRB.

The new SCB will have most of the fields that were referenced often by drivers from the old SCB, and will also contain a pointer to the appropriate KRB, in cell S.KRB. Both new control blocks will contain a number of fields not previously present, in order to allow increased functionality.

This structure is quite flexible -- there will be one SCB per controller (KRB) for those occasions where the old I/O structures were adequate, and multiple SCBs per KRB to support multiple units in operation in parallel on the same controller. All existing drivers must change. Most drivers will require only minimal changes. The drivers, however, that are to support increased functionality will need to change is direct proportion to that increase.

One very desirable result of this particular form of reorganization is that the Executive will not need to know whether the device has one SCB per controller or multiple SCBs per controller. The Executive will always find the I/O queue, timeout count, I/O packet address, and fork block at the appropriate offsets from the SCB pointer in the UCB at U.SCB, regardless of the number of SCBs per KRB. Refer to Addendum A for a complete description of the form of the SCB and KRB, and to Appendix B for diagrams of various data base structures associated with particular example configurations.

4.4.3 The Driver Process

To fully understand this I/O structure, the reader needs to grasp the concept of a "driver process". A driver process is created when an I/O request is queued onto an SCB that is not busy. Potentially, there exist as many driver processes as there are distinct SCBs in the system.

Once initiated, the driver process dequeues a request, initiates the proper I/O function, waits for a completion interrupt, posts I/O status, and then dequeues another request. This sequence of execution continues until the I/O queue is empty. The driver process then terminates.

It is essential that the reader understand the difference between a driver process and the driver code it executes. A driver process is an execution agent with its context stored, when it is not executing, in the SCB. It executes driver code for a particular device type on behalf of a specific unit. Each independent SCB on a KRB may have a driver process active, but there is only one set of driver code. Synchronization among driver processes on the same controller is accomplished via the Controller Request Block (KRB).

4.4.4 Compatibility with the Past

Since many I/O devices supported by RSX-11M do not require unit operation in parallel, we will make conversion of their device drivers to use the new SCB-KRB structure as simple as possible. The structure of the SCB and KRB is such that if only one SCB is required per controller (KRB), then the two control blocks may be allocated contiguously. (Refer to the pertinent figure in Appendix B.) In this case, all fields that used to be in the SCB and are now in the KRB may be referenced by their old offset names from the start of the SCB.

4.4.5 Driver Changes Required

Each driver that is to support new functionality will be changed. The specific changes to be made will depend , in part, on the particular device/functionality combinations that are desired.

In those systems that do not 'SYSGEN-in' the new functionality, the SCB and KRB will be allocated contiguously simply to save space.

The SCB cell S.KRB contains a pointer to the KRB. The KRB cell K.CSR ntains the CSR address for the controller, and the offset K.CSR will ways be zero, so that the pointer to the KRB will always point directly at the cell containing the CSR address.

4.5 SPECIFIC I/O STRUCTURE CHANGES

This section describes the design changes made to the RSX-11M I/O structure to support the features mentioned in Section 4.2.

4.5.1 Distributed I/O Changes

It is necessary that the Executive be able to determine to which processor each peripheral device in the system is attached. If we were to deal only with configurations that had no switchable buses, a processor number associated with each controller would be adequate. However, we must be able to deal with devices that change the processor to which they are attached while the system is running.

To provide this support, the unibus run mask defined in section 4.2.1 will be used instead of a processor number. Each unibus run is essentially a group of distinct devices that, because of the way they e physically attached to the unibus, will always be attached to the me processor. The unibus run is the smallest switchable fragment of a particular unibus.

Each KRB in the system contains a unibus run mask, and each processor control block will always contain a mask that represents the unibus runs to which it is currently attached. If a bus is switched from one processor to another, only the processor unibus run connection masks need be altered.

The methods for forcing particular operations onto specific processors are described in the sections that follow.

4.5.1.1 $CFORK and $FORK Changes

Basically, it is the device drivers that need to execute on specific processors. These drivers already have a process structure with the context stored in the SCB.

A cell containing the unibus run mask (URM) has been added to the Fork block , to define the placement for that device in the system. This mask will have no effect when queuing a fork block. The fork dequeuing routine will, however, only dequeue and process fork blocks whose unibus run masks intersect the current processor unibus run mask. Any URM that is zero will signify that it can be processed on any processor.

This creates a condition not present in previous versions of the Executive. The fork queue will not necessarily be empty when the Executive returns to a user task. To ensure that the fork blocks that remain in the queue will be speedily processed, the Executive will interrupt (using the interprocessor interrupt) a processor that has fork blocks queued waiting for processing.

A new Executive routine that is to be written will be designed to be used whenever the Executive wants to assure that certain driver code is processed on a particular processer. This is the conditional fork routine, $CFORK. It will return if the unibus run mask of the device whose UCB is in R5 shows that the driver may execute on this processor. It will perform a $FORK if execution must be continued on another processor.

NOTE

> $CFORK assumes that execution is currently at executive level. It is to be used solely to assure execution on the correct processor. $FORK will unconditionally fork, and should be used when the processor is not at executive level. It too will assure continued execution on the correct processor.

$FORK has been modified to be processor-specific so that each time a driver forks, it will automatically be continued on a compatible processor. Driver initiation on a particular processor will be handled by $DRQRQ, $GTPKT, and $CFORK. $DRQRQ performs basically the same function as in current systems. $GTPKT assures the driver of execution on the correct processor. Whenever $GTPKT finds an I/O packet that it would normally pass back to the driver, it performs a $CFORK to assure execution on the correct processor. If the current processor is the correct processor (often the case) $CFORK will simply return and $GTPKT will return the I/O packet to the driver (with Carry Clear). Should $CFORK find it necessary, it will perform a fork. $FORK will return with Carry Set (as we will have modified it to do), which will force the driver to dismiss itself. Eventually, the fork processor will restart the driver executing on the appropriate processor.

This approach has a number of advantages, in that all I/O functions processed by $GTPKT or passed off to the ACP will be performed without a $FORK to another processor, regardless of the device referenced. Only I/O packets that actually go to the driver must be processed by a driver executing on a particular processor.

An implication of this approach concerns drivers that have the UC.QUE bit set, signifying that they wish to be called without queueing the 'O packet. Often this is to allow use of the current task context to _p secondary I/O buffers, an operation that is not affected by the modifications described here as long as the driver then queues the I/O packet. The restriction concerns those drivers that used UC.QUE set to force $DRQIO to pass them I/O packets which they intend to process immediately. Certain restrictions now apply to this form of I/O processing. No driver should process an I/O packet that it has received directly from $DRQIO without performing a $CFORK first to guarantee execution on the correct processor.

NOTE

> Specifically, no driver should process an I/O packet that will cause it to access the device registers unless that packet was received from $GTPKT, or an intervening $CFORK has been performed. Packets received from $DRQIO may not be processed directly unless they cause no activity in the I/O page, and thereby need not be executed on a particular processor.

4.5.1.2 IOKIL Changes

Before IOKIL can call drivers at their CANCEL entry points, it must have them execute on the appropriate processor, and it must handle multiaccess devices properly.

IOKIL will create a separate $FORK process for every unit that is to be called but is not accessible on the current processor.

4.5.1.3 Powerfail

On powerup, the powerfail entry is only called for devices currently accessible from the processor that just powered-up.

4.5.1.4 Timeout

The Executive timeout processor will immediately call all driver processes that have timed out and are accessible from the current processor. For those that are not, processor-specific fork blocks will be created that will, when executed, call the driver process at its cancel entry point while executing on the correct processor.

4.5.2 Multiaccess Device Support

Two terms, "assignment" and "access", are used with a specific technical meaning in this section and in section 4.5.3. Controller "assignment" is the process of setting S.KRB in the SCB to point to an online KRB. Requesting "access" involves serializing driver access to a particular controller. This is done by queueing driver process fork blocks on the controller request queue listhead in the KRB. It is necessary to have "access" to the "assigned" controller prior to actually changing the registers in the I/O page.

To support multiaccess (multi-controller) devices, the I/O data structures must reflect the existence of alternate controllers. These controllers are to be represented by SCB's that have multiple pointers to KRB's. The pointers to all the possible KRB's are contained in the SCB, separately from S.KRB. The S.KRB field is still used to point to the KRB, but the KRB now pointed to is considered to be the currently assigned KRB. In this way, once KRB assignment has been made and S.KRB set to point to the appropriate KRB, all software executes without modification. The KRB format is shown in Addendum A.

5.3 Operation of Units in Parallel

Operation of multiple device units in parallel on a single controller has two primary advantages. It is necessary in order to support overlapped seek functions, and it allows delayed controller assignment for more efficient load sharing.

A number of ramifications exist concerning the operation of units in parallel.

- o Each independent unit must request access to its controller prior to executing functions that use it, and release it when it is through.
- o Devices that have sub-controllers (e.g. multiple units on a TM02) must request the sub-controller in addition to the main controller.
- o Interrupts may come from a unit that does not currently "own" the controller. Some method of fielding these interrupts is necessary.
- o A separate SCB is necessary for each unit that is to operate independently.

5.3.1 Request and Release Controller

Three routines exist for requesting and releasing controller access: Request Controller for Control Function ($RQCNC), Request Controller for Data Transfer ($RQCND), and Release Controller ($RLCN). Controller reassignment is made prior to actually requesting access to the controller whenever load sharing is enabl
ed on a multiaccess unit,
and $RQCNC, or $RQCND, is called.

These routines are also affected by three other factors. First, if load-sharing is not enabled on this unit (US.LDS=0), the current controller assignment is not altered. This permits static controller assignment, perhaps for diagnostic mode.

Second, no KRB will be assigned if it is offline (KS.OFL=1), and should no online KRB be found, $RQCNC and $RQCND will terminate the I/O packet with IE.OFL and return to the driver's initiator entry point to get a new packet.

The two-word controller request queue listhead in the KRB is used by $RQCNC or $RQCND to queue the fork blocks of driver processes that are waiting for access to the controller.

Whenever $RLCN is called, it will return to the next driver that is waiting. Priority is given to drivers waiting for control access over those waiting for data transfer, by queueing positioning requests onto the front of the queue, with data transfer requests added to the end of the queue.

When a driver/unit is given access to a controller, the controller status is set to busy, and the UCB address of the unit is placed in the owner word (K.OWN). A value of zero in K.OWN indicates a free controller. In this fashion, controller interrupts may be sent to the correct unit for processing.

4.5.3.2 Interrupts From Multiple Units

Interrupts from most controllers of magnetic media are of two types -- controller initiated or unit initiated. Most control functions to the drive (including the positioning commands SEEK and SEARCH) will terminate with unit interrupts and the controller will report the physical unit number of the interrupting unit.

The other type of interrupt is initiated by the controller to report termination of a function that busied out the controller (usually a data transfer command). The unit that caused this interrupt may be determined by the "owner" of the controller (KRB), since only one unit may issue such a command to a particular controller at any one time.

Interrupt handling of the following situations are discussed in the remainder of this section:

- o Interrupts that occur from units performing overlapped control functions (e.g., SEEKS).
- o Interrupts from mixed massbus devices, where multiple device drivers are involved.
- o Interrupts from subcontroller devices.

Overlapped Function Interrupts

To allow correct unit determination, a table is added to the KRB; this table contains UCB addresses for all units on that controller, indexed by unit number. This is done to allow correct UCB determination on unit specific interrupts, when the physical unit number is returned from the controller.

In RSX-11M V3, the CNTBL is always a table of UCB addresses indexed by controller number. This works because only one unit may be active on a controller at any one time (for SCB queued I/O devices).

In the future, however, we will use the CNTBL as a table of KRB, not UCB addresses, and the CNTBL will become a static table. When an interrupt is detected, the correct KRB is determined by indexing the ~~TBL with the controller number from the PS. From there, if the ntroller is interrupting (as opposed to a unit) then the unit that currently "owns" the KRB is the unit to process. If it is a unit that is interrupting, the driver must determine the physical unit number from the controller, and use it to index the UCB table in the KRB to determine the correct unit.

In order to properly dispatch interrupts for a device that supports overlapped operation, one must interact with the controller to determine whether the interrupt received was a "drive" interrupt or a "controller" interrupt.

Mixed Mass-bus Devices

The discussion in the previous section applies also to mixed mass-bus devices. In addition, the interrupt dispatch code must jump indirect through D.VINT in the driver dispatch table pointed to by the DCB of the resultant UCB. There is a section of common interrupt dispatch code for all massbus devices to avoid duplicating code in drivers.

4.5.3.3 Subcontroller Devices

Certain devices have two-level controllers. For these, a unit must quest and receive access to both controllers prior to executing any operations. Access to the subcontroller is controlled by the Request and Release Controller functions.

Every subcontroller on the system has a Subcontroller Request Block (KRB1), which is used by the request and release routines to serialize access to the subcontroller. The KRB1 is a subset of the KRB. The current owner UCB address is used as a busy flag, and each unit on the subcontroller points to its KRB1 from U.KRB1. This is a device dependent parameter in the UCB. It must be at the same location for all subcontroller devices. It is the responsibility of the request and release routines to ensure that access is granted correctly to the subcontroller. A special entry point for the Release Controller is available ($RLCN1) which only releases the device controller, not the subcontroller.

On subcontroller devices that permit overlapped execution of functions, it is the driver's responsibility to correctly determine the unit that caused the interrupt. The UCB addressed by the UCB table in the KRB is the first unit on the subcontroller, so that the pointer to the KRB1 may be retrieved.

NOTE

> The UCB returned is not necessarily the one associated with the unit that caused the interrupt. On a TM02, for example, the slave unit number is available from the controller, and it is used to index

into the table at the end of the KRB1 to
find the correct UCB.

Interrupts For Subcontroller Devices

In the most complex case, the subcontroller device supports overlapped control functions and is mixed mass-bus. Then, when the driver (finally) gets control, it has a UCB and an SCB. The UCB is not necessarily the one for the unit that interrupted. It is, however, one of the units on the correct subcontroller. The KRB1 can be found from this UCB and the slave unit number can be determined from the subcontroller. The KRB1 UCB table is indexed, using the slave unit number, to find the correct UCB address for the slave unit.

4.5.4 Data-Late Error Recovery*

The current recovery procedure for data-late errors is to immediately try the transfer some number of times until success is achieved or the system gives up.

Unfortunately, this method is inadequate for data-late conditions because the retry mechanism simply aggravates the bus-memory contention that caused the problem. Clearly then, some method is needed to smooth out these transients by spreading the I/O traffic over a longer time span.

The recovery scheme for data-late errors requires four princpal capabilities:

1. A method to store processes that are waiting their turn to retry their data tranfers.

2. A measure of system-wide NPR activity, which will allow a decision to be made on when the waiting driver processes will be allowed to continue.

3. Changes to all drivers capable of getting data-late errors to recognize them explicitly and call the recovery routine.

4. Some sort of timer that will prevent driver processes from waiting in the queue forever.

The driver processes will be queued up in one system-wide wait queue. usual, the driver fork block will be used to store the driver

----
* Not implemented in the first release of RSX-11mP.

process context. The routine $QDRVW (Queue Driver for Wait) will be called to enter the driver into the wait queue. The driver will not release the controller prior to calling $QDRVW.

The system-wide NPR I/O count will be incremented by $RQCND, and decremented by $RLCN. Clearly, then, the data-late recovery mechanism wll only work for systems where unit operation in parallel is supported. The actual operations on the NPR I/O count will be performed by a common routine. This routine, when called to decrement the NPR I/O count, will examine the NPR I/O threshold to see if any of the waiting drivers can be started. If any can, they will be entered into the system fork queue.

In diagnostic mode, the drivers will disable the data-late recovery mechanism.

Data-Late Error Prevention

A method to attempt to prevent data-late errors by sensing near saturation of the system NPR bandwidth, and delaying the initiation of transfers that would seriously exceed that bandwidth has been ~~ nsidered. At this time we feel that implementation of such a ~~ocedure without more specific information concerning the problem is unwise.

4.5.5 Seek Optimization

Seek optimization requires two new routines; $NQPKT to enqueue an I/O packet (in $DRQRQ), and $DQPKT which will dequeue a packet in $GTPKT. These routines will be used for all devices, regardless of whether they support seek optimization or n
ot. They will be driven by S2.SOP
in S.ST2, the seek optimization status bit.

$NQPKT will, for devices that support seek optimization, check two limits; $SKLOP and $SKHIP. These are the low and high priority limits, respectively, between which I/O packets will be considered of equal priority for the purposes of seek optimization. $NQPKT will set the priority byte in the I/O packet to $SKLOP if it falls within the range. Once this mapping is performed, it will perform an effective $QINSP with one addition. The seek optimization word of the I/O packet will be initialized to the vaue of U.SKA (the seek age count), and decremented once for each I/O packet passed over that is issued from the same task with the same priority.

NOTE

One word is required in the I/O packet for seek optimizaton use while the I/O packet is in the queue. Its location is to be defined.

$DQPKT, called from $GTPKT, will examine the I/O queue and scan the entire highest priority class for the I/O packet with the closest LBN to the current LBN for that unit (held in U.LBN and U.LBN+2). As it scans, it increments the age count for each packet; if it reaches zero this packet has aged and become the "closest" I/O request, irrespective of its actual LBN.

4.5.6 Error Logging Enhancement

The activity report added to the error log information for each device error will necessitate saving the I/O packets that are active on error-logging devices at the time of the error.

e I/O packets that are currently active will be kept on the I/O active queue ($IOACT). An I/O packet will be considered "active" for the purposes of error logging from the time they are dequeued from the I/O input queue for an error logging device until the time they would normally be deallocated by IOFIN.

Should an error occur, all I/O packets on the active list and currently waiting for an interrupt will be set with the error sequence number of the first error for which they were active. If more than one error occurs, the count field will be incremented.

At the point that an I/O packet would normally be deallocated it will, if it was active during an error, be kept in the error logging queue of I/O packets, $IOERR. The error logging task will have the responsibility to deallocate those packets.

NOTE

The exact location of the one word and one byte required in the I/O packet is to be supplied.

.5.7 Dynamic UMR Allocaton

(To be supplied)

4.5.8 Modules Changed

DREIF
DRQIO
ERROR
IOSUB (EXTENSIVE)
POWER
SYSXT (EXTENSIVE)
TDSCH
SYSTB (SYSGEN)

New conditional assembly symbols.

M$$PRO	- multiprocessor support (controls distributed I/O)
S$$UBD	- subcontroller device support (e.g. TM02 support)
M$$ACD	- multiaccess device support
MA$XY	- multiaccess support for device XY
M$$IXD	- mixed massbus device support (already present)
O$$LAP	- overlapped SEEK support
OV$XY	- overlapped seek support for device XY
R$$CON	- reconfiguration support
D$$LAT	- extended data-late error recovery support

4.6 I/O EXAMPLE

This section will trace the operation of the I/O system for a dual-access RP04 that supports overlapped seeks. To support these features, each RP04 unit has an independent SCB. In this example, load-sharing is enabled.

For information that will aid in following this example, refer to the figures in Appendix B.

Assume that initially DS0: is doing a data transfer. A user level task issues a QIO for DB4:, and $DRQIO gets control. The I/O packet is then queued for DB4: on its SCB, and the driver is called at its initiator entry point. It immediately calls $GTPKT to deliver the next I/O packet, and $GTPKT finds the unit not busy. Since this is a multi-access device, a controller is not currently assigned, so $GTPKT simply returns the I/O packet just queued to the driver.

The driver determines that this is a read function, and calculates the sector address of the transfer. A request is made to request control access to the controller, via $RQCNC. This routine assigns an acceptable controller and then determines that the controller is busy. In this case, however, the device supports control functions during data transfers; $RQCNC simply ensures that execution is on the correct processor (via $CFORK) and returns to the driver.

The driver is assured control of the device registers since they may be changed only at executive level. The driver initiates the SEEK function, sets its timeout, releases the controller, and returns.

NOTE

> Drivers for devices that support overlapped SEEKS may not change the device registers except at executive level -- never at device level. The only exception to this rule is that Attention Summary Registers and Interrupt Enables may be cleared. This should not affect the standard RSX-11M drivers.

This return will either go back to $DRQIO, or to the fork processor, depending on whether $CFORK found it necessary to perform a fork. If it didn't, return is to $DRQIO. If it did, return is to the fork processor.

All this time, DS0: has been performing data transfers. At some point, the seek on DB4: terminates, but this interrupt will be ferred until the current data transfer is complete.

When the data transfer on DS0: completes, and RDY is asserted by the RH controller, an interrupt will occur. The interrupt service routine (not part of any driver) will retrieve the controller number from the PS, and index the CNTBL with it to determine the KRB address of the interrupting controller. At this time it will examine the device registers (read only) to determine whether the interrupt was initiated by the controller or the drive. In this case, it was the controller, so the UCB address of the unit that currently "owns" the KRB is retrieved from K.OWN. A transfer through the Device Dispatch Table to the correct driver (DSDRV) is made, and this driver immediately $FORKS.

Another interrupt occurs immediately, since the Seek complete interrupt attention bit was not cleared. The interrupt service routine goes through the same procedure as described above, except that it determines from the controller that it was a drive-initiated interrupt. The drive physical unit number is determined from the Attention Summary Register, the attention bit is cleared, and the UCB table at the end of the KRB is indexed with the physical unit number to find the UCB address. The correct driver (DBDRV) is entered via a transfer through its DDT (driver dispatch table), and it immediately $FORKS.

All the interrupt conditions have been serviced at this point, and the rst fork to execute is the one from DSDRV. It gets control and xamines the device registers to see if the data transfer completed successfully, which it did. DSDRV then releases the controller and calls $IODON.

At some point, the fork for DBDRV gets processed. The driver determines that the seek completed successfully, and it issues a Release function to the drive. It then calls $RQCND. When $RQCND returns, the driver has control on the correct processor to initiate the data transfer. From this point processing is identical to the previous data transfer for DS0:.

4.7 DRIVER CONVERSION

Existing RSX-11M V3 drivers will not be compatible with RSX-11M V4. The body of this chapter has dealt with the changes to drivers and data structures that are necessary to incorporate added functionality. This section deals with modifications to existing drivers for compatibility only.

The changes required to update a driver are divided into two categories:

- o Modifications to a 'conventional' driver and the data structure requirements
- o Further modifications to drivers that process the I/O packet before it is entered in the queue

4.7.1 Converting a Conventional Driver

The changes described below apply in the following cases:

- o When the driver receives all requests via a call to $GTPKT
- o When the KRB is adjacent to the SCB
- o When no new functionality is desired in the drivers.

When these preconditions are met, the changes to an existing driver are as follows:

1. A table of interrupt service routine entry points must be appended to the driver dispatch table. Refer to Addendum A, "NEW DATA STRUCTURES", for the form of the extension to the driver dispatch table. A macro will be provided for this purpose.
2. The form of the call to INTSV$ and INTSE$ will change slightly.
3. The call to $GTPKT and subsequent operations on the CNTBL will be made into a macro.

NOTE

Except for those drivers with new functionality, such as the mass storage drivers, the Version 3.1 and Version 4 drivers may be maintained as a single source module.

4.7.2 Converting a Driver that Receives I/O Packets Directly

In addition to the conversion required for conventional drivers, drivers that receive I/O packets without calling $GTPKT must not access any device registers until a call to $CFORK has been executed.

If synchronous, non-interrupt-driven I/O is performed in this fashion, the issuing task now expects I/O to be complete upon regaining control. (Throughput is improved by eliminating event flag processing).

the multiprocessing system, the conditional fork required to access [illegible]e device registers may release the processor to the task before I/O is complete. To maintain compatibility, the driver must block the task before calling $CFORK. When execution resumes on the target processor, the task is unblocked; the requested I/O is performed and the standard Executive routine for I/O completion is invoked.

Additional changes will be necessary.

CHAPTER 5

ONLINE MAINTENANCE AND RECONFIGURATION FACILITIES

5.0 INTRODUCTION

This chapter describes the provisions for error detection, online fault isolation, maintenance and powered-up repair that are incorporated in RSX-11mP.

.ıe following features are incorporated in the multiprocessing system:

- o Program-assisted reconfiguration of functional units under operator control, i.e:
    - . Add memory boxes
    - . Add or remove processors, devices or controllers from the pool of system resources
    - . Display the status of all system resources
- o Reconfiguration of switchable busses under control of the DT03 FP bus switch
- o Improvements in error logging content*
- o Additions to the error logger for multiprocessor support
- o Diagnostic loader for loading a stand-alone diagnostic into an offline thread

----

* Also available in the single processor environment.

5.1 SYSTEM RECONFIGURATION

The function of the Reconfiguration subsystem is three fold:

- o It allows the operator to define the set of resources that belong to the online system.
- o It provides the capability to explicitly establish controller device combinations for online maintenance and troubleshooting
- o It allows functional units to be removed from the pool of resources allocated to the online system so that an offline thread can be configured for stand-alone diagnostics.

Reconfiguration of functional units is performed by commands issued to MCR or, offline, through VMR commands that operate on the disk image of the system.

All reconfiguration commands set or display the status of resources. A resource can have one of the following states:

NON-EXISTENT	The named resource is not recognized by the operating system.
OFFLINE	The resource is known to the operating system but cannot be allocated because one of the following conditions exist: . It was marked offline by a previous reconfiguration command. . It is not physically present on the target system. . A path for the transmission of data and control signals from the online system does not exist.
ONLINE	The resource is known to the operating system and all of the following are true: . It was marked online by a previous configuration command. . It is physically present in the target system. . A path for the transmission of data and control signals exists from the online system.

Reconfiguration commands can always be issued to mark a resource online or offline. The transition to the online state is not complete until all the remaining conditions noted above have been fulfilled.

A resource is always flagged 'not physically present' until a successful test is made to access it. Thus, on startup, devices and controllers attached to a processor or bus that is offline are marked 'not present'.

The online reconfiguration system will consist of two co-operating tasks; ...REC, which receives and parses operator keyboard input and converts it into appropriate QIO calls to REC..., which performs the actual reconfiguration, based on the QIO packets received from the keyboard processor (or other tasks).

Reconfiguration commands are issued via QIO's to psuedo-device RD:. The driver, resident in the Executive, will copy the reconfiguration parameters into dynamic storage and invoke task REC... to service the request. Each time REC... services a reconfiguration command it will initiate an entry into the error log, recording:

- o The time-of-day
- o The command performed

e count of system errors logged since reconfiguration will be cleared.

Other elements of the online reconfiguration subsystem are:

- o The DT03 FP bus switch driver (device SB:)
- o The MK11 programmable port control driver (device MB:)

Offline reconfiguration, through VMR, consists of commands that simply alter the executive data base on the disk image.

5.1.1 Operator Reconfiguration Commands

The basic form of all reconfiguration commands is as follows:

```
REC[ONFIGURE] <resource-name>[/switch] <state>
```

where:

<resource-name> is one of the following:

Switched Bus Runs
Processors
Memory Boxes
Device Controllers
Physical Devices/Units
Slave Units

NOTE

> Individual terminal lines (for device TT) cannot be controlled via reconfiguration commands.
>
> DECNET devices are not supported.

Optional items are enclosed by brackets.

<state> defines the reconfiguration activity. The semantics of this parameter are dependent on the resource being controlled as described in the following paragraphs. Alternative states that can be selected are separated by an exclamation point ! . The operator may select one of the items in the list.

5.1.1.1 Memory Reconfiguration

```
<resource-name>:=  <memory-list>

<memory-list>:  MBmm [, MBnn, ...]

<state>:  ON[LINE][/INT[ERLEAVE]]!  <null>
```

NOTE

> The wild card symbol * may not appear when more than one list element is present.

Semantics - Task ...REC

ONLINE	- The memory boxes are marked as available for use by the operating system. The memory boxes are assigned to physical addresses as follows:
non-interleaved	- The base address for each box is set in the order specified, starting at the initial tcp of memory and increasing in increments of the box size, as read from the MK11 control registers, until all addresses have been assigned.
interleaved	- The base address is set at the initial top of memory. The interleave sequence is in the order specified that is:

The first box responds to A00
The second box responds to A02
The third box responds to A04
The fourth box responds to A06

The top of memory is set to the highest physical address spanned by the interleaved configuration.

<null>	- The null string requests the display of information about the memory boxes listed. The following data is shown.

```
MBnn   state   base-address   int   size
```

where:

state	is ONLINE or OFFLINE
base-address	is the physical address of the box in units of 64 byte blocks. If the box is offline base-address is replaced by a string of 6 asterisks(*).
int	is one of the following:

<null>	box is not interleaved
2-0	two-way interleaved - A00
2-2	two-way interleaved - A02
4-0	four-way interleaved - A00
4-2	four-way interleaved - A02
4-4	four-way interleaved - A04
4-6	four-way interleaved - A06

Boxes that are interleaved together have the same base address.

size	is the actual box size (as read from the MK11 control registers) in units of 1024(10) word blocks. The value is expressed as a decimal quantity.

__Semantics__ - __Task VMR__

Memory box reconfiguration is not available in VMR.

NOTE

The operator must force reconfiguration upon bootstrap by placing the port control switches in manual mode as described in Chapter 6.

Restrictions:

- o Memory reconfiguration cannot be performed by VMR.
- o A memory box cannot be placed offline by the MCR reconfiguration task.
- o When interleaving boxes, the number of list elements must be equal to the desired interleaving factor (2 or 4).
- o Interleaved boxes must have identical memory capacity.
- o The operator does not have explicit control over placement of memory boxes in the physical address space when a program-assisted reconfiguration command is processed.
- o Wildcards (*) are not permitted in any command except a request for configuration status.

Example:

```
     REC   MB3,MB4   ON/INT
```

Result:

Assuming the current top address is 200000 (8) and the capacity of each box is 65KB (200000(8)) then, after reconfiguration:

Top address = 600000 (2-32KB boxes added).

Memory box 3 is configured as follows:

```
     Base address:  200000
     Interleave factor:  2-way
     Responds to:  Address A00
```

Memory box 4 is configured as follows:

```
     Base address:  200000
     Interleave factor:  2-way
     Responds to:  Address A02
```

Error Conditions:

The following error messages are issued by task ...REC.

```
     MEMORY BOX nn ALREADY ONLINE
```

Cause:

A reconfiguration command was issued for a memory box that is currently online.

Remedial Action:

Correct the box number, if in error, and reissue the command.

PHYSICAL MEMORY CAPACITY EXCEEDED

Cause:

The storage capacity of the box causes the 4.4 MB maximum size to be exceeded.

Remedial Action:

None, the system must be rebooted to alter the box configuration.

BOX CAPACITY NOT IDENTICAL

Cause:

A request to interleave boxes was rejected because the units do not have the same amount of memory.

Remedial Action:

Reissue the command specifying boxes with the same physical capacity or do not specify interleaving.

ILLEGAL REQUEST FOR INTERLEAVING

Cause:

The number of memory boxes specified is not equal to an interleave factor (2 or 4).

Corrective Action:

Reissue the command with the correct number of boxes specified or with no interleaving.

MEMORY BOX SPECIFIED MORE THAN ONCE

Cause:

A memory box appeared in the list more than once.

Remedial Action:

Correct the list specification and reissue the command.

CPnn PORT CONTROLLER IS IN MANUAL MODE FOR MBnn

Cause:

A memory reconfiguration request was rejected because the port controller for the indicated CPU was not under program control.

Remedial Action:

Place the port control in 'programmable mode' for the processor indicated. Verify that all remaining controllers for the boxes to be reconfigured are in automatic, then reissue the command.

UNABLE TO ALIGN MEMORY BOX BASE ADDRESS

Cause:

The requested memory reconfiguration could not be performed without creating a void in the physical address space. This problem results when the base address of the memory box to be placed online cannot be set to the current top address (usually occurs when interleaving was specified).

Remedial Action:

Reissue the reconfiguration command with no interleaving specified.

MEMORY BOX nn IS NOT PRESENT

Cause:

The MK11 CSR for the box number specified is non-existent.

Remedial Action

Verify that the correct box number was specified and reissue the command if necessary.

5.1.1.2 Switched Bus Reconfiguration

<resource-name>:= Snn

tate>:= CPnn [/MAN[UAL]] !OFF[LINE] [/MAN[UAL]]

<u>Semantics:</u> <u>Task</u> <u>REC...</u>

CPnn - The switched bus is connected to processor nn.

OFFLINE - The switched bus and all devices on it are removed from control of the operating system.

Appending the switch '/MANUAL' to the processor identification or OFFLINE command indicates that the operator will position the switch manually prior to operating the switch, the operator must wait for the message:

```
REC -- BUS Snn IS OFFLINE
```

When switching processors, the bus is not restored to service until a 'connect' interrupt is received on the target processor.

Switching the bus, or placing it offline causes it to be marked for offline status. The bus is not relinquished until all I/O on attached devices has completed. When this occurs, the 'offline' message described above will be issued.

Tasks linked to device common blocks associated with the bus will be ocked from execution until the bus is connected to an online ocessor.

<null> - A null string requests the display of bus status. The following is printed:

```
Snn     <status>     <CP-list>
```

Where:

<status> is one of the following:

CPnn - bus is online and connected to processor nn

OFFLINE - bus is offline

<CP-list> - is the set of processors to which the bus can be connected. A pound sign (#) will precede each processor that is offline.

<u>Semantics</u> - <u>Task VMR</u>

CPnn - The switched bus is marked for connection to processor nn upon bootstrap.

OFFLINE - The switched bus, and all devices on it, are removed from control of the operating system.

<null> - The bus state is displayed in the format described for MCR task ...REC.

Restrictions:

- A bus controlling mounted devices cannot be placed offline or switched to another processor.
- A QIO issued to a device on a bus that is being switched to another controller will be immediately rejected with the error code IE.OFL.
- A bus cannot be switched to an offline processor.
- Wild cards (*) are not permitted in any command except the request for bus status display.

Error Conditions:

The following error messages are issued by task REC... . The asterisk denotes those messages issued by VMR.

*SWITCHED BUS nn CANNOT BE CONNECTED TO PROCESSOR mm

Cause:

The specified DT03 FP has no port in the processor; the requested connection cannot be made.

Remedial Action:

Verify that the correct processor and/or shared bus was specified. A list of eligible processors can be obtained by requestng a display of bus status.

*PROCESSOR mm IS OFFLINE

Cause:

A request to connect the bus to an offline processor has been rejected. A connection cannot be made to a processor not under control of the operating system.

Remedial Action:

Place the bus offline and complete the connection manually.

CONNECTION REFUSED

Cause:

The bus could not be disconnected from an offline processor in order to complete the requested connection.

Remedial Action:

Have the offline system relinguish control of the shared bus, then repeat the request.

BUS MARKED FOR OFFLINE STATUS

Cause:

A request to place the bus offline was issued but the bus was already marked for offline status.

Remedial Action:

Wait for the offline notification issued by REC... .

*BUS ALREADY OFFLINE

Cause:

A request to place the bus offline was issued for a bus that is currently offline.

Remedial Action:

Verify that the correct bus was referenced and re-issue the command if necessary.

DEVICES MOUNTED

Cause:

An attempt to reconfigure the bus was rejected because one or more devices attached to the bus are mounted.

Remedial Action:

Use the MCR command DEV to identify the devices with volumes mounted, then issue the appropriate DMO commands and re-try the request.

NO RESPONSE FROM BUS SWITCH

Cause:

The bus switch failed to complete the reconfiguration command within 1 second while in programmable mode.

Remedial Action:

Faulty hardware is indicated. Field service should be informed of the problem. The operator can issue the command with the 'MANUAL' qualifier to alter the bus switch.

BUS POWERED DOWN

Cause:

The bus switch cannot be connected because there is no power on the switched bus.

Remedial Action:

Power up the bus and reissue the command.

BUS NOT PRESENT

The bus cannot be reconfigured because the controller does not reside on the target CPU.

Remedial Action:

If an erroneous bus or CPU number was used, reissue the command with the correct number.

5.1.1.3 Device Controllers

Resource-name
:= <Cn><id> (Refer to Section 2.2, "NAMING CONVENTIONS")

<state>:= ON[LINE]OFF[LINE]! <null>

Semantics - Task ...REC

ONLINE - The controller is marked as available for use by the operating system.

OFFLINE - The controller is marked for offline status. It is removed from the set of resources available to the operating system.

The controller is not relinquished by the operating system until all pending I/O has completed. When this occurs, the operator will be notified with the message:

```
REC -- CONTROLLER resource-name IS OFFLINE
```

The <null> string requests the display of controller status. The following information is shown:

```
resource-name   status   cpu   bus-name

device-list
```

where:

resource-name	is the name of the controller
status	is OFFLINE or ONLINE
cpu	is the name of the controlling processor
bus-name	is the bus to which the controller is attached

device-list	is a list of devices that can be accessed by the controller

The symbol '#' will precede the CPU, bus name or device if any of these resources is offline.

<u>Semantics - Task VMR</u>

ONLINE	- The controller is marked as available for use by the operating system.
OFFLINE	- The controller is marked for offline status.
<null>	- Controller status is displayed as described for task ...REC.

The status of a controller may be set independently of any busses or processors to which it is attached. If a controller is placed online but no path to it exists in the online system, I/O requests will be rejected with the error code IE.OFL.

Restrictions:

o A controller cannot be placed offline when:

  . One or more of its devices are mounted, and

  . There is no alternate path for control and data transfer from the online system.

o Wildcards(*) are illegal in all commands except for device status requests.

o The following controllers cannot be reconfigured:

  . All DECNET controllers

  . Controllers for the following devices:

LPS-11	DA11B
ICS/ICR-11	DMC11
UDC-11	DP11
DL11 A, B, C, D, E	DQ11
DJ11	DU11
DH11	DUP11
AR11	DZ11
VT11/VS60	
DSS/DRS11	

ror Conditions:

The following error messages are issued by task ...REC. The asterisk (*) denotes a message that may be issued by VMR in addition to REC.

*CONTROLLER resource-name ALREADY ONLINE

Cause:

The status of the specified controller was 'online' when the configuration request was processed.

Remedial Action:

Correct the controller identification, if in error, and reissue the command.

*CONTROLLER resource-name MARKED FOR OFFLINE STATUS

Cause:

The controller was marked for 'offline' status when the configuration request was received.

Remedial Action:

rrect the controller identification, if in error, and reissue the mmand.

DEVICES MOUNTED

Cause:

A request to configure the controller offline was rejected because:

- o One or more of its devices were mounted and,
- o No alternate path to the devices was available to the online system.

CONTROLLER NOT PRESENT

Cause:

While processing a request to place the controller online, the reconfiguration task was unable to access the CSRs to verify that the hardware was present on the system. The controller state is set to 'marked-online'.

Remedial Action:

tablish a path to the controller, if possible.

5.1.1.4 Device/Units and Slave Units

<resource-name>:= <dn><id>[-<sl>]: (Refer to Section 2.2, "NAMING CONVENTIONS").

<state>:= ONLINE!OFFLINE!*!<controller-name>!<null>

<u>Semantics - Task ...REC</u>

ONLINE	- The device is marked as available for use by the operating system. It can be accessed by all controllers to which it is attached.
OFFLINE	- The device is marked for offline status. It is not relinquished by the operating system until all pending I/O has completed. All new I/O requests for the device are rejected with status IE.OFL.
*	- The device can be serviced by all controllers to which it is attached.
<controller-name>	- Is a controller specification as described in Section 2.2, "NAMING CONVENTIONS". The device can be serviced only by the identified controller.
<null>	- The status of the device unit is displayed as follows:

```
     resource-name      state      controller-list
```

where:

resource-name	is the name of the device
state	is OFFLINE or ONLINE
controller-list	is a list of controllers that can access the device. A pound sign (#) precedes the name of any controller that is offline.

Placing a device offline prohibits further requests from being queued to it. The device is not actually offline until the message:

```
     REC  --  DEVICE <device-name> IS OFFLINE
```

i lisplayed on the console.

<u>Semantics - Task VMR</u>

ONLINE - The device is declared available for use by

	the operating system.
OFFLINE	- The device is declared unavailable for use by the operating system.
*	- The device can be serviced by all controllers to which it is attached.
<controller-name>	- The device is 'bound' to the specified controller. All I/O for the device will be processsed only by the named controller.

Restrictions:

- o A mounted device cannot be reconfigured.
- o No DECNET device can be reconfigured.
- o The following devices cannot be reconfigured:

LPS-11	VT-11/VS60
ICS/ICR-11	DRS/DSS-11
UDC-11	DA11B
DL-11 A-E	DMC11
DJ11	DP11
DH11	DQ11
DR11	DU11
DZ11	DUP11

Error Conditions:

The following error messages are issued by task ...REC. The asterisk (*) denotes VMR messages.

```
     *DEVICE <device-name> ALREADY ONLINE
```

Cause:

The status of the specified device was 'online' when the configuration request was processed.

Remedial Action:

Correct the device-name, if in error, and reissue the command.

```
     *DEVICE <device-name> MARKED FOR OFFLINE STATUS
```

.use:

The device was marked for offline status when the configuration request was received.

Remedial Action:

Correct the controller identification, if in error, and reissue the command.

DEVICE MOUNTED

Cause:

A request to reconfigure the mounted device was rejected because the online system would be unable to access it for I/O.

Remedial Action:

Dismount the device (via DMO) and reissue the configuration request.

DEVICE NOT PRESENT

Cause:

A request to reconfigure the device was rejected because a poll of ,ices at system startup (or during a previous reconfiguration command) indicated that the device was not present.

Remedial Action:

Correct the controller identification if in error and reissue the command.

5.1.1.5 Processors

<resource-name>:= CPnn (refer to Section 2.2, "NAMING CONVENTIONS")

<state>:= ONLINE!OFFLINE !<null>

Semantics - Task ...REC

ONLINE - The processor is marked as available for allocation by the operating system.. The interprocessor interrupt link to all other online processors is enabled.

OFFLINE - The processor is marked for offline status. The execution of the task in control of the processor is interrupted and all further scheduling of the CPU is inhibited. The interrupted task will usually resume execution on another CPU. However, if task-processor affinity was in effect, task execution will be suspended until the processor is returned to service.

When all I/O under control of the processor completes, the Interprocessor Interrupt link to all remaining online CPU's is disabled.

Before placing the CPU online the operator must:

- o Halt and reset the processor.

- o Set the manual port controls identically to those f
or the
online system.

Placing the processor offline prohibits further I/O requests from being queued to controller device combinations accessed by the CPU. The processor is not relinquished by the operating system until all pending requests have been serviced and the message:

```
REC -- PROCESSOR nn IS OFFLINE
```

appears at the issuing console.

<null>	the null string requests the display of processor status. The following information is printed:

```
<processor-name>     status
```

where:

processor-name is the requested CPU

status is OFFLINE or ONLINE

Semantics - Task VMR

ONLINE	- The processor is marked as available for allocation by the operating system.
OFFLINE	- The processor is marked for offline status.
<null>	- The processor status is displayed in the format described for task ...REC.

Restrictions:

- o A processor cannot be placed online until the system has been SAVed at least once.

- o At least one processor must be online.

- o A processor cannot be placed offline unless alternate access paths to mounted devices are available to the online system.

- o A processor cannot be placed offline while a bus switching operation is in progress.

- o Wild-cards are prohibited in all commands except a request for status (<null>).

Error Conditions:

The following error messages are issued by task ...REC. The asterisk (*) denotes VMR messages

*PROCESSOR <CPU-name> ALREADY ONLINE

Cause:

The status of the specified device was 'online' when the configuration request was processed.

Remedial Action:

Correct the cpu-name, if in error, and reissue the command.

*PROCESSOR <cpu-name> MARKED FOR OFFLINE STATUS

Cause:

The processor was marked for offline status when the OFFLINE configuration request was received.

Remedial Action:

Correct the cpu-name if in error and reissue the command.

*PROCESSOR <cpu-name> CANNOT BE PLACED OFFLINE

Cause:

An attempt was made to configure the only remaining CPU offline.

Remedial Action:

At least one processor must be allocated to the online system. Configure another CPU online, then reissue the reconfiguration request.

DEVICES MOUNTED

Cause:

A request to place the CPU offline was rejected because one or more mounted devices would be inaccessible to the online system.

Remedial Action:

Dismount the device or issue the reconfiguration commands necessary to give the online system an alternate access path, then reissue the original command.

If the device is attached to a switched bus, the device must be dismounted.

NO RESPONSE FROM PROCESSOR nn

Cause:

A request to place the specified processor online failed because the CPU did not answer the boot command issued through the interprocessor interrupt within 5 seconds.

Remedial Action:

Halt and reset the CPU to enable it to respond to the boot command, then reissue the request. A subsequent failure indicates a fault in the IIST hardware that should be reported to DEC Field Service.

SYSTEM NOT SAVED

Cause:

A request to place the processor online failed because the system image was not setup for multiprocessing by the MCR SAV command.

Remedial Action:

Execute the MCR SAV command, then reissue the request.

NOTE

> A processor can be placed online via VMR. However, the CPU is not activated on a new system until the image is SAVed.

5 1.1.6 General Error Messages

All error messages are preceded by the suffix 'REC - '. Those that apply to VMR are flagged with an asterisk (*).

*SYNTAX ERROR

Cause:

The command contained unrecognizable syntax.

Remedial Action:

Retype the command.

*UNKNOWN RESOURCE <resource-name>

Cause:

The operating system contains no provisions for recognizing, servicing or allocating the specified resource.

Remedial Action:

Retype the command using the correct resource name.

5.1.2 QIO Interface to Task REC...

Task REC... services requests issued via the QIO directive to pseudo device RD:

Its function is to reconfigure functional units in response to requests issued by one or more privileged tasks and to supply configuration status.

5.1.2.1 GET LUN Information Macro

Word 2 of the buffer filled by the GET LUN INFORMATION system directive (the first 'device' characteristic word) contains the following information. A bit set to 1 indicates that the described characteristic is true.

BIT	SETTING	MEANING
0	1	Record oriented device
1	0	Carriage control device
2	1	Terminal device
3	0	Directory device

4	0	Single-directory device
5	0	Sequential device
6-12	0	Reserved
13	0	Device mountable as communications channel
14	0	Device mountable as FILES-11 volume.
15	0	Device mountable

Words 3 and 4 of the buffer are undefined. Word 5 contains the size of the buffer filled by the new QIO functions READ SYSTEM CONFIGURATION (IO.RSC) and READ CONFIGURATION OF FUNCTIONAL UNITS (IO.RCF).

5.1.2.2 QIO Macro

Table 5-1 lists the QIO functions performed by the reconfiguration task.

Table 5-1
QIO Functions for Reconfiguration

Format	Function
QIO$C IO.RSC... <buff>	Read system configuration.
QIO$C IO.RCF... <stadd,buff>	Read configuration of functional units.
QIO$C IO.SCF... <stadd>	Set configuration of functional unit.
QIO$C IO.KIL...	Cancel I/O request.

where:

stadd: is the starting address of the configuration table (must be on a word boundary).

buff: is a buffer to receive configuration status (must be word aligned). The length must be at least that specified in the fourth characteristics word.

The configuration table is used to specify the information to be supplied by the reconfiguration task or to define the requested configuration of a functional unit. A table entry consists of 9 words in the format shown in Figure 5-1, "Configuration Table Entry Format".

When more than one type of functional unit is implicated in a request for configuration status, then the specified buffer is filled with a list of 9-word entries terminated by a zero.

```
OFFSET

              ----------------------------------
   0          ! RESOURCE NAME (ASCII)          !
              ----------------------------------
   2          ! UNIT NUMBER   ! CONTROLLER I/D !
              ----------------------------------
   4          ! STATE OR      ! SLAVE UNIT     !
              ! FUNCTION CODE ! NUMBER         !
              ----------------------------------
   6          ! BUS MASK                       !
              ----------------------------------
  10          ! RESERVED                       !
              ----------------------------------
  12          ! CONFIGURATION WORD 0           !
              ----------------------------------
  14          !         DATA WORD 1            !
              ----------------------------------
  16          !              WORD 2            !
              ----------------------------------
  20          !              WORD 3            !
              ----------------------------------
```

Figure 5-1
Configuration Table Entry Format

The entry is subdivided into two parts:

- o Words 0-4 contain the name, location, and state of the functional unit.
- o Words 5-8 contain device dependent configuration data.

The information in the first 5 words consists of the following:

word 0 - Resource name consisting of one or two ASCII characters. The first character is contained in the low byte. When the name contains one character, the high byte is zero (null).

word 1 - The low byte contains a 7 bit physical unit number and a controller flag in bit 7.

The physical unit number is the unit number by which the device is known to its controller. In the case where the controller flag is set to 1, the physical unit number is, instead, an ASCII character which denotes the controller ID.

The flag is set to 1 if the resource name defines a controller. A setting of 0 indicates that a switched bus, processor, memory box, or device is being referenced.

The high byte contains a binary logical unit number in the range 0-99. or a wildcard symbol (252).

word 2 - Slave unit number (low byte) and state or function code (high byte).

Slave unit number is a binary value in the range 0-99. or a wildcard specification (octal value 252).

State is returned by the reconfiguration task in response to a request for information and may assume one of the following values:

0 (online) 1 (marked for offline) 2 (offline)

Function is setup by the issuing task to request configuration of a functional unit (via function code IO.SCF). One of the following values is recognized:

0 - Place functional unit online.

1 - Mark functional unit for offline status.

rd 3 - Bus mask, a binary value containing a bit set to one for each bus run to which the functional unit is connected. The mapping between bus runs and mask bits is as follows:

Processor bus runs 0-n are represented by:

bits 0-n

Switched bus runs 0-m are represented by:

bits n+1 to n+m+1

n+m+1 must be less than 15.

word 4 - Reserved, must be zero.

The four words of configuration data are dependent on the functional unit and request type as defined in the following section.

5.1.2.3 QIO Functions to Read Configuration Status

QIO functions IO.RSC and IO.RCF return information on system and functional unit configurations and status to the specified buffer. The information will consist of one or more configuration entries rminated by a word of zero.

Data Returned:

For function code IO.RSC, the following status and data are returned:

Successful Completion:

I/O Status block:

Word 0:	IS.SUC
Word 1:	Number of functional units in the system.

The entries describing each functional unit are placed in the buffer in the following sequence:

```
Processors
Memory Boxes
Controller - type 1 - Port A
Controller - type 1 - Port B
Connected devices
        .
        .
        .
Controller - type 2 - Port A
Controller - type 2 - Port B
Connected devices
        .
        .
        .
Switched bus 0
        .
        .
        .
Switched bus n
```

Controllers and associated devices are listed together. The controllers having access to multiport devices are listed in order followed by each device accessed.

The contents of each buffer entry is as follows:

<u>Buffer Entry, Central Processors</u>:

Resource-name: CP

Controller ID and flag: 0

Unit Number: Processor number (0-4)

Slave Unit number: 0

State: 0 = online, 2 = offline

Bus Mask: A single bit is set representing the bus run to which the CPU is connected.

Bit 0 - CPU0, Bus Run 0
Bit 1 - CPU1, Bus Run 1
Bit 2 - CPU2, Bus Run 2
Bit 3 - CPU3, Bus Run 3

Configuration Words 0-3: Cleared (0).

<u>Buffer Entry, Memory Boxes</u>:

Resource-name: MB

Controller I/D and flag: 0

Unit number: Memory Box number (0-7)

Slave unit number: 0

State:	0 = online, 2 = offline
Bus Mask:	A single bit is set representing the bus run to which each port is connected. Bit 0 - Bus Run 0 Bit 1 - Bus Run 1 Bit 2 - Bus Run 2 Bit 3 - Bus Run 3
Configuration Word 0:	If online - the physical base address of the box. If offline - 0.
ıfiguration Word 1:	The box size, in units of 1024(10) word blocks, as read from the MK11 controller.
Configuration Word 2:	The interleave factor and address where:

The low byte represents the interleave address as follows:

Value	Address
0	A00
2	A02
4	A04
6	A06

The high byte represents the interleave factor as follows:

Value	Interleave Factor
0	not interleaved
2	2-way interleaving
4	4-way interleaving

Internal interleaving is always enabled.

ıfiguration Word 3: Cleared

NOTE

If the MK11 is disabled, a box size of 0
is returned.

Buffer Entry, Controllers:

Resource-name:	See Table 5-2.
Controller I/D:	An ASCII alphabetic character
Controller flag:	set to 1
Unit number:	0
Slave unit number:	0
State:	0 = online, 2 = offline
Bus Mask:	A single bit is set representing the bus run to which the controller is connected.
Configuration Words 0-3:	Cleared (0)

<u>Buffer Entry, Devices</u>:

Resource-name:	See Table 5-2.
Controller I/D:	A seven-bit number representing the device's physical unit number on its controller.
Controller flag:	0
Unit number:	A binary value representing the device's logical unit number.
Slave unit number:	A binary value representing the slave unit number (if any).
State:	0 = online, 2 = offline
Bus Mask:	Cleared (0)
Configuration Words 0:	Two ASCII characters representing the controller mnemonic of the device controller.
1:	A word with a bit set for each controller to which the device is capable of being connected.
2:	A word with a bit set for each controller, specified in word 1, that is currently online.

Table 5-2
Device Controller Names

Name	Device/Controller
AD	AD01-D Analog-to-Digital Converter
AF	AFC11 Analog-to-Digital Converter
AR**	AR11 Laboratory Peripheral System
CT	TA11/TU60 Tape Cassette
DB*	RP04, RP05, RP06 Pack Disk (RH)
DF	RF11/RS11 Fixed-Head Disk
DK	RK11/RK05 Cartridge Disk
DL	RL11/RL01 Cartridge Disk
DM	RK611/RK06/RK07 Cartridge Disk
DP	RP02/RP03 Pack Disk
DR	RM03 Pack Disk (RH)
DS*	RS03 and RS04 Fixed-Head Disks (RH)
DT	TC11/TU56 DECtape
DX	RX11/RX211/RX01/RX02 Flexible Disk
GR**	VT11/VS60 Graphics Systems
IC**	ICS/ICR Industrial Control Local and Remote Subsystems
IS**	DSS/DRS Digital Input and Output Subsystems
LP	LP11, LS11, and LV11 Line Printers
LS**	LPS11 Laboratory Peripheral System
MF*	TM78/TU78 Magnetic Tape (RH)
MM*	TM02/TU16/TE16 Magnetic Tape (RH)

Device mnemonic only.

** Not reconfigurable.

MS	TS11/TS04 Magnetic Tape
MT	TM11/TU10 or TS03 Magnetic Tape
PP	PC11 Paper Tape Punch
PR	PC11 or PR11 Paper Tape Reader
RH***	RH11/RH70 Massbus Controller
UD**	UDC11 Universal Digital Controller
SB	DT03 FP Unibus Switch
TT*	Terminal Devices
XB**	DA11-B Parallel Unibus Link
XL**	DL11-A-E Communication Line Interface
XM**	DMC11 Synchronous Communication Line Interface
XP**	DP11 Synchronous Communication Line Interface
**	DQ11 Synchronous Communication Line Interface
XU**	DU11 Synchronous Communication Line Interface
XW**	DUP11 Synchronous Communication Line Interface
YH***	DH11 Asynchronous Serial Line Multiplexer
YJ***	DJ11 Asynchronous Serial Line Multiplexer
YL***	DL11-A-E, W Asynchronous Single Line Interface
YM***	DM11-BB Modem Control Interface for DH11
YZ***	DZ11 Asynchronous Serial Line Multiplexer

----

* Logical device mnemonic only.

** Not reconfigurable.

 * Required for both device and controller specifications to reconfiguration task.

**** Not allowed as device mnemonic.

<u>Buffer</u> <u>Entry,</u> <u>Switched</u> <u>Busses</u>:

Resource-name: S <null>

Controller I/D: 0

Unit Number: Switched bus run number (unit number of DT03 FP)

Slave unit number: 0

State: 0 = online, 2 = offline

Bus Mask: Bus mask of the switched bus to which the DT03 FP is connected.

Configuration Word 0: Bus mask representing all bus runs to which the switched bus can be connected.

Configuration Word 1: Bus mask representing all online bus runs to which the switched bus can be connected.

ror Returns:

The following error codes are returned in the low byte of I/O status block word 0. Word 1 is always 0.

IE.SPC - Buffer is not within the virtual address space of the issuing task.

IE.BYT - Buffer is byte aligned.

Read Configuration of Functional Units

For function code IO.RCF, the task must supply the following:

A single 3-word entry describing the functional unit for which configuration data is required. Words 0-2 contain the folllowing information:

Word 0 - Resource name

Word 1 - Controller I/D and flag (low byte) Unit number (high byte)

Word 2 - Slave unit number (low byte); state of function code (high byte) is ignored.

A buffer, word aligned, whose size is equal to the value returned in word 5 of the GET LUN INFORMATION macro.

The contents of the 3-word entry are given, in text that follows, for each functional unit type. The data returned is in the format described for function IO.RSC.

e I/O status block is setup as follows, assuming successful completion:

Word 0: IS.SUC

Word 1: Count of functional unit entries placed in buffer.

<u>Functional Unit Description Words, Central Processor</u>:

Resource-name: CP

Controller I/D and flag: Ignored

Unit number: Wildcard (252) or CPU number (0-3)

Slave unit number: Ignored

State/Function code: Ignored

The wildcard (252) directs the reconfiguration task to search for and return information on all processors in the system.

Functional Unit Description Words, Memory Boxes:

Resource-name: MB

Controller I/D and flag: Must be 0

Unit number: Memory box number (0-7) or wildcard (252)

Slave unit number: Ignored

State/function code: Ignored

The wildcard (252) directs the reconfiguration task to search for and return information on all memory boxes in the system.

<u>Functional Unit Description Words, Controllers</u>:

Resource-name: See Table 5-2.

Controller I/D: One ASCII alphabetic character or wildcard (052)

Controller flag (bit 7): set to 1

Unit number: Ignored

Slave unit number: 0

State/function code: Ignored

The wildcard directs the reconfiguration task to search for and return information on all controllers in the system of the specified type.

The information is returned in the format described for function IO.RSC. The controller data is followed by entries describing each device accessed. For multiport devices the list of controllers precedes the set of devices.

<u>Functional Unit Description Words, Devices</u>:

Resource-name: See Table 5-2.

Control I/D: One ASCII alphabetic character or wildcard (052)

Controller flag (bit 7): set to 0

Unit number: Binary value in the range 0-99, or wildcard (252)

Slave unit number: Binary value in the range 0-99, or wildcard (252)

State/function: Ignored

The wildcard directs the reconfiguration task to search for and return information on all devices that fit the remaining elements of the specification.

<u>Functional Unit Description Words, Switched Busses</u>:

Resource-name: S <null>

Controller I/D: Ignored

Controller flag: Ignored

Unit number: Binary value in the range 0-16 specifying the number of the switched bus run or wildcard (252)

Slave unit number: Ignored

State/Function: Ignored

The wildcard directs the reconfiguration task to return information on all switched busses in the system.

Error Returns:

The following error codes are returned in the low byte of I/O status block word 0. Word 1 is always 0.

IE.SPC - Buffer is not within the virtual address space of the issuing task

IE.BYT - Buffer is byte aligned

IE.NSR - Non-existent resource specified

5.1.2.4 QIO Functions to Set System Configuration

Configuration of functional units is specified by QIO function code IO.SCF. This function requires a single parameter specifying the address of a 9-word entry containing the resource-dependent information defined in the following paragraphs.

Reconfiguration commands can only be issued by a privileged task.

<u>Set Configuration, Central Processors</u>:

Controller I/D and flag: 0

Unit number: Processor number (0-4)

Slave unit number: 0

State/function: 0 = online, 1 = mark for offline

Bus Mask: 0

Configuration Words 0-3: Cleared (0)

When an online request is issued, the reconfiguration task will transmit a 'boot' request to the processor followed by an interrupt request (no sooner than 1 second later).

The modified M9301 bootstrap will issue a reset, then monitor the interrupt-pending flag on its IIST interface. If the flag is set within 2 seconds, the bootstrap will enable interrupts. Warm bootstrap code, resident in the operation system, is then entered to ing the processor online.

The hardware bootstrap does not attempt to access the load device until it has polled the IIST unsuccesfully for at least 3 seconds.

Whenever a processor is returned to service the reconfiguration task must:

- o Transfer control to the CPU.
- o Test for the presence of all CSRs on the processor bus (including the MK11 and DT03 FP) and mark the controller data base accordingly.
- o Call the appropriate drivers at the power-recovery entry point.

If, as the result of te above, a switched bus is returned to online status, a CSR scan and power recovery sequence will be required for all devices on the switched bus.

When an offline request is issued, the reconfiguration task will first verify that alternate access paths are available to any mounted devices. If so it will 'mark' the processor offline to inhibit re-scheduling, and cause an interprocessor-interrupt to be posted so current task execution is interrupted (after declaring task-CPU -finity for itself on another CPU).

After all pending I/O requiring interrupt service has completed, all other online processors will be masked from recognizing interrupts and boot request from the designated CPU.

If the processor was being used to maintain system time, this function will be delegated to another online CPU (possibly by clearing a software lock).

Finally, the processor will execute a RESET instruction followed by a halt.

When the processor is offline, tasks having the processor or bus affinity attribute for the CPU may be permanently blocked.

Also, some device drivers, such as the LPS-11 and ICS/ICR, that perform I/O to a task for an indeterminate amount of time may have interrupt service terminated. Tasks that are triggered in response to such interrupts may remain dormant indefinitly.

NOTE

> To allow such tasks to be aborted, a special device flag will have to be set in the driver data base so a driver can be called unconditionally at its 'I/O rundown' entry point (without first executing a conditonal fork).

Status Returns:

The following values are returned to the low byte of I/O status block word 0. Word 1 always cleared.

IS.SUC - Request succeeded.

IE.NSR - Non-existent resource specified.

IE.FLN - Processor was already offline when offline request was issued.

IE.TMO - A request to place the CPU online failed because the processor did not respond to the boot request within 5 seconds.

IE.ONL - Processor was already online when the online request was issued.

IE.ILL - Processor cannot be placed offline because it is the only active CPU remaining.

IE.MOU - Processor cannot be placed offline because one or more devices under its control are mounted and cannot be accessed from another online processor.

IE.BAD - Invalid function code specified (not 0 or 1).

IE.PRI - Issuing task is not privileged.

IE.SPC - Entry is out of the address space of the issuing task.

IE.BYT - Entry is not word aligned.

IE.NMP - Multiprocessing is not enabled.

<u>Set Configuration, Memory Boxes</u>:

Resource Name: MB:

Contr
oller I/D and flag: 0

Unit Number: 0

Slave Unit number: 0

State/function: 0 = online

Bus Mask: 0

Configuration Words (0-3) Set as follows:

Word 0: High byte contains the interleave flag where:

0 = no interleaving

1 = interleaving

Low byte specifies the number of boxes (must be in the range 1-4).

Words 1-2: Each byte contains the number of a memory box to be placed online. No box number may appear more than once.

The sequence proceeds from right to left and from low to high addresses until the box count is reached.

When no interleaving is specified, each box is placed online and the physical address space is extended by the total memory capacity.

The programmable port controllers for each online processor are accessed to setup the physical address space.

When interleaving is requested, the interleave factor is determined by the box count in the low byte of word 0 (must be 2 or 4). The assignment of the least-significant bits is in the sequence in which each memory box appears in words 1 and 2 . That is:

The first box number is assigned to A00
The second box number is assigned to A02
The third box number is assigned to A04
The fourth box number is assigned to A06

Each interleaved box must have the same memory capacity.

atus Returns:

The following values are retured to the low byte of the I/O status block word 0. Unless otherwise specified, word 1 is zero.

IS.SUC -	Request succeeded. Word 1 contains the amount of physical memory online (in units of 1024 blocks).
IE.NSR -	Nonexistent resource specified.
IE.BAD -	Invalid function code specified (must be 0).
IE.ONL -	A memory box was already online when the online request was issued. Word 1 contains the box number.
IE.INT -	Memory boxes could not be interleaved because the boxes do not have the same capacity or the number of boxes is incorrect (must be 2 or 4).
IE.MAN -	Memory boxes could not be placed online because a CPU port controller was in manual mode. Word 1 contains the box number (low byte) and processor number (high byte).
IE.SPC -	Entry was out of the address space of the issuing task.
IE.BYT -	Entry was not word-aligned.
IE.ADR -	Request configuration could not be performed without creating a gap in the physical address space.
IE.OFL -	Memory box specified in word 1 does not exist in the target system.

Set Configuration, Switched Bus Run:

Resource Name: S

Controller I/D and flag: 0

Unit number: Bus number

Slave unit number: 0

State/function: 0 = online, 1 = mark for offline

Bus Mask: Bit 7 set indicates a manual operation when the requested function is 'ONLINE'. The bus mask (bits 0-3) is set to the processor bus run for which a connection is desired. The mask is set as follows:

Bus Run	Mask (binary)
P00	0001
P01	0010
P02	0100
P03	1000

Bus mask is ignored when the request is offline.

Bus switching operations are performed by the DT03 FP bus switch driver.

Prior to issuing a switch-bus request to the driver, it is the responsibility of the reconfiguration task to:

- o Ensure that there are no mounted devices on the bus.
- o 'Quiesce' the bus by blocking further QIO requests from being issued to devices on the bus and waiting for all I/O in progress to be complete.
- o Disable all tasks accessing device common blocks that reside on the bus.

The DT03 driver, on receipt of the command, will issue a reset to disable interrupts on all devices attached to the bus. On completion of the switching operation, the reconfiguration task must:

- o Redirect all interrupt vectors on the vacated processor to the 'nonsense' interrupt routine (if applicable).
- o Write the ISR address and priority into the vector of the processor that received control of the bus.
- o Scan the CSRs on the bus to determine which devices are present.

o Initiate power recovery for all devices that are present.

NOTE

> One or more unsolicited interrupts may be lost from the ICS/ICR, UDC, LPS and terminal drivers whenever the bus is switched.

Upon receipt of an offline request, the reconfiguration task will perform the checks described above, then issue an 'offline' request to the driver to disable interrupts from devices on the bus. All interrupt vectors will be redirected to the 'nonsense' interrupt.

Manual switching requests, issued with bit 7 set in the function and status byte, result in the same sequence of events as for an automatic 'offline' request except that the switching request issued to the driver remains pending until a 'connect' interrupt is received at the target processor. The reconfiguration task is notified of completion, via the normal I/O-done mechanism, and initiates the CSR scan and power-recovery sequence to re-enable device interrupts.

atus Returns:

The following values are returned to the low byte of the I/O status block word 0. Unless otherwise specified, word 1 is zero.

IS.SUC -	Request succeeded. For a manual request, this means that the bus is offline and can be switched by hand.
IE.NSR -	Specified switched memory bus does not exist.
IE.BAD -	Invalid function code or bus mask was specified.
IE.NLN -	The switched bus was already offline when the offline request was issued.
IE.DNR -	For an automatic switch-bus request, a 'connect' interrupt was not received from the target processor within 2 seconds after the switch command was issued.
IE.MOU -	One or more devices are mounted.
IE.PFL -	The target processor is offline.
IE.RRR -	A request to reconfigure the switched bus was 'refused' by the offline system.
IE.OFL -	Specified memory box is not present in the target system.

Set Configuration, Controllers:

Resource Name: See Table 5-2.

Controller i/d: 7 bit ASCII character, as described in Chapter 2. Wildcards are not allowed.

Controller flag: Set to 1

Unit number: 0

State/function: 0 = online, 1 = mark for offline

Bus mask: 0

Configuration words 0-3: 0

Declaring a controller online allows it to be used for servicing I/O requests. Marking the controller online does not permit it to be accessed until the reconfiguration task verifies that the controller CSR can be addressed from the online system.

[illegible] it can be accessed, the driver is called at the powerfail entry int.

To place a controller offline, the following activities must be performed by the reconfiguration task:

- o If any devices are mounted, the task must ensure that there is an alternate path to the device from the online system.
- o 'Quiesce' the controller by blocking the queuing of further requests-in-progress to complete.
- o Set controller interrupt vectors to the nonsense interrupt.

Status Returns:

The following values are returned to the low byte of the I/O status block word 0. Unless otherwise specified, word 1 is zero.

IS.SUC	Request succeeded.
IE.NSR	Specified controller does not exist.
IE.BAD	Invalid controller identification (non-alphabetic) or function code was specified.
IE.ONL	Specified controller was already online when an 'online' request was issued.
IE.NLN	Specified controller was already offline when an 'offline' request was issued.

IE.MOU - One or more devices are mounted but no alternate path to the device exists from the online system.

IE.SPC - Entry was out of the address space of the issuing task.

IE.OFL - Specified controller cannot be accessed by the online system. The controller is marked 'online' but declared not-present.

Set Configuration, Devices:

Resource Name: See Table 5-2.

Controller I/D: 7 bit ASCII character or wildcard (*)

Controller flag: 0

Unit number: Device physical unit number in the range 0-99

Slave unit number: Device slave unit number in the range 0-99

State/function: 0 = online, 1 = mark for offline

Bus Mask: 0

Configuration words 0-3: Cleared (0)

The function and controller I/D are interpreted as follows:

Offline Request:

A~cess to the device is denied to all controllers to which it is tached. The controller I/D is ignored.

Online Request:

The device is 'bound' to the controller specified in the I/D field. All I/O requests to the device must be s
erviced only by the specified
controller. If a wildcard is specified, the device can be serviced by all controllers to which it is attached.

Before a device can be placed offline, the following conditions must be met:

- o The device may not be mounted.
- o All further requests to the device are blocked. All requests in progress must be allowed to complete.

Status Returns:

The following values are returned to the low byte of I/O status block word 0. Unless specified, word 1 is zero.

IS.SUC - Request succeeded.

IE.NSR - Specified device does not exist.

IE.BAD - Invalid function code, controller I/D, unit or slave unit number.

IE.ONL - Specified device was already online when an 'online' request was issued.

IE.NLN -	Specified device was already offline when an 'offline' request was issued.
IE.MOU -	Device is mounted.
IE.SPC -	Entry is out of the address space of the issuing task.
IE.BYT -	Entry is not aligned on a word boundary.
IE.PRI -	Issuing task was not privileged.

5.1.2.5 Error Logging Interface

The reconfiguration task will initiate an error log entry whenever system configuration is changed.

(Interface is to be specified)

5.2 DT03 FP DRIVER

The DT03 FP is a multi-unit, multi-controller device. There is one controller/unit combination for each switched bus in the system. The unit number is the same as the switched bus run number.

The DT03 driver performs the following operations:

- o Switch the bus to a target processor under manual or automatic control.
- o Place the bus offline.

Requests are only serviced when issued by a privileged task.

5.2.1 GET LUN INFORMATION Directives

The GET LUN INFORMATON directive returns the following in words 3 and 4 of the buffer. All other characteristics words are zero.

Word 3 contains a mask with a bit set to one for each processor bus to which the switched bus can be connected. Word 4 contains a single bit set defining the current state of the switched bus (0 = bus offline).

5.2.2 QIO Functions

The table below lists the QIO functions serviced by the driver.

Function	Description
QIO$C IO.FLN ...	Place bus offline.
QIO$C IO.CON ... <mask,man>	Connect bus to specified bus run.

where:

mask is a 16 bit run mask specifying the bus run to which the connection is to be made.

man is a flag that, when non-zero, specifies than a manual connection is requested.

5.2.2.1 Place Bus Offline

When an offline request is received, the driver transfers control to the processor from which the bus is currently accessed and issues a reset, for the switched bus only, by setting bit 9 in the DT03 CSR. All interrupts from the DT03 are disabled to all processors having access to the bus. The bus mask in device characteristics word 3 is cleared.

The power recovery routine will not re-enable DT03 interrupts while the bus is offline.

It is the responsibility of the issuing task to gracefully terminate I/O for all devices on the bus and block references to the CSRs of all connected devices before issuing the offline request.

Status Returns:

The following values are returned in the low byte of word 0 in the I/O status block. Unless specified otherwise, Word 1 is zero.

IS.SUC -	Request succesfully completed; the bus is offline and all interrupts from it are disabled.
IE.PRI -	Issuing task is not privileged.
IE.NLN -	Bus was already offline when the 'offline' request was issued.
IE.OFL -	DT03 FP is not present on the system.

5.2.2.2 Connect Bus -- Automatic Mode

The bus to which the connection is to be made is specified in the first parameter word of the QIO DPB. The processor owning the bus run must be online.

A request for automatic connection is serviced as follows:

The driver transfers control to the processor currently owning the bus (if the CPU is online), issues RESET (CSR bit 9) and clears the REQ and interrupt enable bits in the CSR. (If no online processor owns the bus, this step is omitted.)

The driver then tranfers control to the target processor, enables DT03 interrupts on this CPU and asserts REQ to request control of the bus. A connection must occur within 1 second.

On occurrences of an interrupt, the ISR will:

- o Verify that the CONN (connect bit) is set on the CPU. If so, terminate the request successfully.
- o If the interrupt was due to on external request (EXT INIT), the bit will be cleared (connection refused).
- o If the interrupt was due to request refused, the QIO request will be terminated with a status of IE.RRR (reconfiguration request refused).
- o If the interrupt was due to the watchdog timer, the online system will be 'CRASHED'.

In the steady-state, online condition, the controlling CPU will be the only processor with DT03 interrupts enabled. In this case, it must service and clear the external interrupt (EXT INIT) flag to prevent loss of the bus to a contending offline system.

NOTE

All requests to connect the bus are rejected when AC LO is set in the CSR (bus is in power-fail state).

5.2.2.3 Connect Bus -- Manual Mode

A manual connection is requested by setting the second parameter word the DPB to a non-zero value.

The chain of events is identical to the automatic mode except that the one second timeout is not initiated. The request remains pending until:

- o A 'connect' interrupt is received at the target procesor.

- o An IO.KIL request is received by the driver.

The 'kill' request is serviced immediately. A subsequent 'connect' interrupt is ignored. The bus is considered to be offline and all DT03 interrupts are disabled.

NOTE

> All requests to connect the bus are rejected when AC LO is set in the CSR (bus in power fail state).

5.2.2.4 Connect Bus -- Status Returns

The following status is returned in the low byte of word 0 in the I/O status block. Unless otherwise specified, word 1 is cleared.

IS.SUC -	Request successfully completed.
IE.RRR -	Reconfiguration request refused.
IE.DNR -	'Connect' interrupt was not received at the target processor within 1 second after a request was posted.
IE.TPF -	Target processor is offline.
IE.NLN -	The switched bus was already offline when the offline request was issued.
IE.BAD -	Bus mask is invalid (bus not interfaced to specified DT03 or nonexistent bus specified).
IE.PRI -	Issuing task was not privileged.
IE.PWF -	Requested connection could not be made because AC LO was set (the bus was in a powerfail state).
IE.OFL -	DT03 FP is not present in the system.

5.2.3 <u>Power Recovery</u>

The DT03 FP driver will re-establish the state of each bus as follows:

- o If the CONN bit is set in the CSR, the bus is marked online and under control of the current processor.

- o If AC LO is set, the bus is declared offline.
- o If the bus and processor to which it was connected are online, the driver will:
  - . Set the bus offline
  - . Issue a 'connect' on the target processor

The bus is not marked online unless a 'connect' interrupt is received within 1 second.

Note that the Executive must call the DT03 driver before any other drivers are invoked at their power recovery entry points. The 1-second delay is necessary to insure that the bus has reconnected before any drivers are called.

5.2.4 DT03 FP Error Logging Interface

: DT03 driver will initiate an error log whenever:

- o A connect interrupt was not received at the target processor within 1-second (for an automatic request).
- o A 'request refused' interrupt is received.

5.3 MK11 MEMORY CONTROLLER DRIVER (MB:)

The purpose of this driver is two-fold:

- o It services memory reconfiguration requests that are in a hardware-independent format.
- o It translates hardware-reported errors into a standard format for error logging entries.

Requests are only serviced when issued by a privileged task.

5.3.1 GET LUN INFORMATION Directive

The GET LUN INFORMATION directive returns the following information in :rds 3 and 4 of the buffer:

Word 3: Number of memory boxes (1-8)

Word 4: Total memory capacity (in units of 1024 word blocks)

The remaining words are cleared.

5.3.2 QIO Functions

The table below lists the QIO functions serviced by the driver.

Function		Description
QIO$C	IO.ONL...<Int, box-list>	Configure memory boxes online.
QIO$C	IO.RCF...<buf>	Read memory box configuration.
QIO$C	IO.KIL...	Terminate I/O for a task.

where:

buf	is a 32-word buffer to receive memory configuration status.
box-list	is a list of one-byte box numbers in parameter words 2-3.
Int	is a two-byte parameter. The low byte specifies the number of boxes in the box list. The high byte specifies interleaving. A value of 0 indicates no interleaving, 1 = interleaving.

Configuration requests (IO.RCF) are only valid when issued by a privileged task.

5.3.2.1 Configure Memory Boxes Online

When an 'online' request is received, the driver performs the following operations for each online processor.

The memory controller for each box is accessed in the order in which the box was specified to setup the base address and interleaving. Internal interleaving is always enabled.

The base address is setup as follows:

o When no interleaving is requested, the address is set to the current 'top' of the physical address space. The top is advanced by the amount of memory in the box. This operation is performed sequentially for each box in the list.

- When interleaving is requested, the address is set to the current top of physical memory. All boxes in the list are interleaved in the order specified, namely:

  first box: A00
  second box: A02
  third box: A04
  fourth box: A06

The top of memory is advanced by the total capacity of all boxes.

The number of boxes specified for an interleave request must be a multiple of the interleave factor (2 or 4).

Status Returns:

The low byte of I/O status block word 0 is set as follows. Unless otherwise noted, word 1 is always 0.

IS.SUC -	Success. Word 1 contains the amount of physical memory online (in units of 1024 word blocks).
IE.NSR -	Nonexistent memory box specified.
IE.BAD -	Number of boxes specified not in the range 1-4 or memory box number exceeds 7.
IE.ONL -	A memory box was already online when the online request was received.
IE.INT -	Memory boxes could not be interleaved because the boxes do not have the same capacity or the number of boxes is incorrect (must be 2 or 4).
IE.MAN -	Memory boxes could not be placed online because one or more port controllers were in manual mode. Word 1 contains the box number (low byte) and processor number (high byte).
IE.ADR -	Requested configuration could not be performed without creating a gap in the physical address space.
IE.PRI -	Issuing task is not privileged.
IE.OFL -	Specified memory box is not present in the system.

5.3.2.2 Return Configuration Status

.is function fills a buffer with 'n', 5-word entries, where n is the number of boxes in the system (as recorded in GET LUN device characteristics word 2). Each entry is in the following format:

Word 0: Status, 0 = online, 1 = offline

Word 1: Bus mask representing each bus to which the ports are connected, where:

Bit 0 - Bus run 0
Bit 1 - Bus run 1
Bit 2 - Bus run 2
Bit 3 - Bus run 3

Word 2: Physical base address of the box (always 0 if offline).

Word 3: Box size (in units of 1024 word blocks).

Word 4: The interleave factor and address, where:

The low byte represents the interleave address as follows:

Value	Address
0	A00
2	A02
4	A04
6	A06

The high byte represents the interleave factor as follows:

Value	Interleave Factor
0	not interleaved
2	2-way interleaving
4	4-way interleaving

Status Returns:

IE.SPC - Part or all of all of buffer is out of the issuing task's address space.

IE.BYT - Buffer is byte aligned.

5.3.3 Power Recovery

Power recovery events occur as the aftermath of an AC LO condition or whenever the system is bootstrapped. In the latter case, the MK11 driver must insure that a valid memory configuration exists before system operation is allowed to continue.

During power recovery, the MK11 driver will access the port control registers to re-establish the box configuration as follows:

- o The integrity of the internal configuration table will be checked. A discrepancy will force a system crash.
- o All boxes under manual control will be checked against the internal memory configuration tables. A difference in configuration for any port will force a system crash.
- o All boxes that are online but not under manual control will have configuration restored. Inability to restore configuration at a port, due to port in manual mode, will force a system crash.

NOTE

> On power recovery, following a system bootstrap, it is the responsibility of 'SAV' to establish initial memory state, as described in Chapter 6.

5.3.4 Exception Handling

The MK11 driver will field all memory exception traps. It will be the driver's responsibility to report all faults to the error logger in a hardware-independent format.

(The format for the information furnished to the error logger is to be specified.)

5.4 ERROR LOGGER ENHANCEMENTS

(To be specified)

5.5 DIAGNOSTIC LOADER

The function of the diagnostic loader, LOD, is to load a stand-alone diagnostic into a memory box from a Files-11 device. The independent port controls on the box allow it to serve as a window between the online system and an offline maintenance configuration. The ability to load diagnostics in this fashion eliminates the requirement for dedicating an operable mass storage device for stand-alone diagnostic operation.

e loader operates as a privileged task under RSX-11mP. On command, it will access a specified device to load a core image of a diagnostic into a 'maintenance' partition. The partition, which is established by the SET /MAIN command described in Chapter 6, has the following attributes:

- o The partition is exactly aligned on the range of physical addresses spanned by one or more memory boxes.
- o No tasks can be installed in the partition.

5.5.1 File Format

The diagnostic file must be in the image format (without a header or stack) produced by the RSX-11M Task Builder. The file need not be contiguous.

The loader will read the file until:

- o 'End of file' is encountered, or
- o No more space is available to store file contents in the maintenance partition.

The latter condition is an error.

On completion of a successful load, the message:

```
LOD - DIAGNOSTIC LOADED
```

will be printed.

5.5.2 Diagnostic Load Command Syntax

The command to load a diagnostic is:

```
LOD     filename/PAR=name
```

where:

filename	is a standard RSX-11 file specification.
name	is the name of a maintenance partition.

5.5.3 Error Conditions

The following message is preceded by the loader. Each message is preceded by the string: 'LOD - '

```
SYNTAX ERROR
```

Cause:

The command line contained unrecognizable syntax.

Remedial Action:

Verify that the file specification is in the correct format, then reissue the command.

INCORRECT OR INVALID PARTITION SPECIFIED

Cause:

The partition is non-existent or is not a maintenance partition.

Remedial Action:

Allocate the partition (or correct the name), then reissue the command.

FILE NOT FOUND

Cause:

e specified file could not be located.

Remedial Action:

Correct any erroneous portions of the file specification, if applicable, and reissue the command.

FILE TOO BIG

Cause:

The diagnostic load terminated because the file contents exceeded the size of the specified maintenance partition.

Remedial Action:

Re-allocate a larger partition and reissue the request.

CHAPTER 6

OPERATIONAL AND BOOTSTRAP PROCEDURES

6.0 INTRODUCTION

This chapter describes:

- o The procedures for bootstrapping the multiprocessor syste
- o The procedures and MCR commands used during normal system operation.
- o Special VMR commands required to setup the multiprocessor environment.
- o Modifications to XDT (the executive debugging tool).

6.1 SYSTEM BOOTSTRAPPING

The system bootstrap procedure is comprised of the following elements:

- o Code that is resident in the Executive to cold-start a new multiprocessor system.
- o Transient code to start a previously running system.
- o Permanently resident code to cycle-up a processor into a running multiprocessor system.
- o A multiprocessor hardware bootstrap that contains provisions for warm-starting a processor as described in the previous item.

Succeeding sections will describe each of these elements in detail.

6.1.1 Memory Configuration at Startup

In the multiprocessing environment, the user may not want all available physical memory to be allocated to the operating system because:

- o One or more boxes may be in use by another thread.
- o One or more boxes may be offline for repair.

Therefore, on startup, only those boxes electrically online and under manual control, as seen from the CPU on which the boostrap is running, will be included in the resources available to the operating system.

If this amount of memory is less than the size recorded in the system image, the startup procedure will issue the warning message:

```
INSUFFICIENT MEMORY AVAILABLE
```

It is the operator's responsibility to ensure that:

- o Memory is correctly configured at all ports.
- o Enough memory is online to contain the operating system and all allocated partitions.

The startup procedure will attempt to ensure that memory is configured identically at all ports accessible to the online system. If an error is found, system operation will terminate with a 'crash' notification.

On startup, SAV will extend or contract the highest partition in the system if the partition is system controlled. (SAV currently performs this function.) Expansion will only take place throughout memory boxes under MANUAL control.

After the system is running, more memory can be added through the program-assisted reconfiguration commands.

6.1.1.1 Recommended Memory Configuration Procedure

The amount of memory allocated for use by the online system, as recorded in the system image, should require a subset of all available memory boxes. The user should defer the allocation of additional memory until startup. This can be done by an indirect command file that issues the necessary reconfiguration commands.

Before SAVing a system to which additional memory has been added rough reconfiguration, the user should remove any partitions located in the space. SAVE will then truncate the required system size accordingly before writing it out to the load device.

If the space is not deallocated, the system size remains at its current value.

NOTE

> The above procedure is the only way to safely contract the physical address space of a system. There is no reconfiguration command that will place memory boxes offline.

6.1.2 Bootstrap Device Identification

All RSX-11M bootstraps will be modified to return the CSR address of the device from which the system was booted to allow the startup code to identify the controller and device.

This will allow a system to be booted from any random-access device-controller combination.

6.1.3 Coldstarting a New System

To minimize the size and complexity of resident executive bootstrap code, a new system will only run in single-processor mode. The user must execute the MCR SAVE command to initiate multiprocessing.

Coldstarting a new system is performed by Executive module INITL. This module will initially poll all MK11 CSRs to size memory and determine the boxes that are present. If a CSR does not exist, the message:

```
MEMORY BOX nn NOT PRESENT
```

will be printed. The appropriate status is recorded in the system data base.

Next, all DT03 FP CSRs that were specified at system generation as being controllable from the current CPU will be polled.

If a CSR does not exist, the message:

```
SWITCHED BUS nn NOT PRESENT
```

ll be printed.

If a CSR exists, but the bus is not connected, the message:

```
SWITCHED BUS nn OFFLINE
```

is displayed.

The data base is updated as necessary to note those busses that are accessible from the current processor.

After the bus configuration has been established, INITL will initiate a poll of all devices, as it does now, reporting any that are offline.

Finally, the remaining startup code is executed to cycle up the processor.

On cold-start of a new system, the MK11 and DT03 drivers will not be called at the powerfail entry points.

6.1.4 <u>Multiprocessor Cold Start</u>

Multiprocessor cold-start is initiated by the MCR command SAV. SAV is subdivided into three parts:

- o Code that runs in conjunction with the operating system to prepare the system for recording on the bootstrapping device.
- o Stand-alone code that writes out and reads the system image from the load device.
- o Code that initiates the operating system.

The stand-alone code to read in and start the multiprocessing system is executed each time a previously SAVed system is bootstrapped.

6.1.4.1 System Shutdown

Preparing the system for writing on the bootstrap medium requires the following activities:

- o All devices are checked to verify that none are mounted.
- o All active tasks (except SAV itself) are blocked.
- o The interprocessor interrupt for each online CPU (except the one on which SAV is running) is vectored to shutdown code which:
  - . saves CPU context in a fixed physical location reserved for each processor
  - . disables the sanity timer and interprocessor interrupt
  - . disables memory management

. executes a halt instruction

- SAV preserves the contents of physical locations 0-1200(8), writes a device bootstrap in these locations, then records the system image and enters the boostrap code to restart the system.

6.1.4.2 System Bootstrap

The system can be rebooted on any processor. The system boostrap reads in the system image from the load device and transfers control to the startup code with sufficient information to allow the boot device to be identified by the system.

The boostrap must re-establish the mapping context as follows:

Kernel, User APR 0-4	-> Physical locations from 0-20K
Kernel, User APR's 5-6	-> Set to map task SAV
Kernel, User APR 7	-> I/O page

Control is then returned to task SAV, with the processor in 22-bit mode, to perform system initiation.

6.1.4.3 System Initiation

Upon regaining control, SAV restores the contents of physical locations 0-1200(8) but re-directs all interrupt vectors to the 'nonsense' interrupt.

The IIST vectors are setup to the normal entry point.

Next, the MK11 control registers on each CPU are accessed to establish the initial memory configuration. Only those boxes in manual mode are considered part of the online system. All remaining boxes are removed from the physical address space and marked offline. If, at this point, a CSR that was present no longer exists, the message:

```
MEMORY BOX nn NOT PRESENT
```

is issued.

SAV will then verify that a valid memory configuration exists, namely:

- Physical memory must be contiguous.

- No more than one box can respond to a given physical address.

If one of these conditions is violated, SAV will print the message:

```
     ILLEGAL MEMORY CONFIGURATION
```

and 'halt' the system.

Memory size is checked to ensure that it is equal to or greater than the value recorded in the system. If a reduction in system size occurs, the warning:

```
     INSUFFICIENT MEMORY AVALABLE
```

will be printed.

SAV then verifies that the executive 'low core' partition is allocated and can be addressed. If not, the message:

```
     NO STORAGE FOR MULTIPROCESSOR OPERATION
```

is printed. The system will then only run in single-processor mode.

NOTE

Memory for this region must be allocated by the VMR command:

```
SET /EXEPAR=base
```

SAV fills in the table of physical low-core addresses allocated to each CPU as follows (xxx=base of executive partition):

Physical Memory	CPU
0-8KB	Processor from which the system was booted.
xxx-xxx+8KB	CPU n
xxx+8KB-xxx+16KB	CPU n+1
	etc.

The table, when indexed by CPU number *2, yields the contents of APR0 for the specified processor. To initiate multiprocessor operation, the 8KB low-core context for all other processors is established by copying the contents of physical 0-8KB to the space allocated as described above. SAV must set up the initial context for the other processors and establish itself as the 'current' task on the processor from which the system was booted.

Interprocessor boot commands are transmitted to the CPUs that were online when the system was written to disk. Each processor responding to the boot command enters the warmstart code, that must reside in the first 8KB of physical memory, to establish its kernel mapping and initial state. Processors that do not respond to the boot signal are reported to the user with the message:

```
     PROCESSOR nn OFFLINE
```

Processor status is changed to 'OFFLINE'.

Next, the status of all DT03 FP controllers is established by testing for the existence of CSRs on all 'online' processors.

If a CSR does not exist, the message:

```
     SWITCHED BUS nn NOT PRESENT
```

is printed. As in the current procedure for displaying device status, this message only appears if the condition represents a change in the bus status.

The state of each bus switch is set by calling the driver at its power-fail entry point. A change in bus-state to 'offline', because of inability to connect it to an online CPU, is reported with the message:

```
     SWITCHED BUS nn IS OFFLINE
```

The final step is to poll all controllers. A controller is declared 'not present' if:

- o It's CSR does not respond.
- o The switched bus on which it is installed cannot be accessed.
- o The CPU that controls the device cannot be accessed.

If a transition from 'online' to 'not present' is detected, the message:

```
     CONTROLLER <name> IS NOT PRESENT
```

is printed. Upon completion, a similar message is printed for all devices that cannot be accessed because the processor, bus, or controller or device itself are not present, i.e.,:

```
     DEVICE <name> IS NOT PRESENT
```

For each device that is on the system, the appropriate interrupt vector is setup overwriting the 'nonsense' vector that is permanently recorded in the disk image.

At this point, the system is cycled through power recovery to reset system time (from the TOD), initialize all resident online devices, and start up the system clocks. The error logger is invoked with a 'boot' message so it can record a snapshot of the system configuration in the log.

6.1.5 Hardware Bootstrap Facility

Before a processor can be safely booted and brought under control of an operational multiprocessor system, the provisions described below must be incorporated in the hardware bootstrap.

- o A set of preliminary, read-only, processor sanity checks.
- o A procedure for recognizing a boot request transmitted using the Interprocessor Interrupt protocol defined below.

The Interprocessor Interrupt boot protocol is as follows:

Online System	Offline CPU
Transmit boot	Enter bootstrap
Stall for 1 second	Issue 'RESET'. Perform 'read only' sanity checks.
Issue Interprocessor Interrupt	Wait for interrupt flag for 1 second.
Stall for 1 second If no response from CPU then report boot failure.	If flag set, enable IIST interrupts, else enter normal boot sequence.

The occurrence of an interrupt will vector the offline processor to the warmstart code described in paragraph 6.1.6.

6.1.6 Processor 'Warm Start' Code

Processor 'warm start' code allows a previously halted and reset processor to re-establish its memory management and register context before rejoining the online system. This code resides within the first 8KB of physical memory. A CPU will enter this code whenever it services an interprocessor interrupt and a test of the memory management status register indicates that the processor is running unmapped.

6.2. ONLINE OPERATOR COMMANDS AND STATUS DISPLAYS

The following operator commands and status displays are available, in addition to the reconfiguration facilities discussed in Chapter 5.

Multiprocessor Displays:

RMDEMO	Displays the current task on each processor that is online.
DEV	Displays the physical device name and status of each device/unit.
TASKLIST/TAL	Displays task-CPU affinity, Task-Bus affinity for installed tasks.
ATL/ACT	Displays task-CPU affinity, Task-Bus affinity for active tasks.
OPEN	Open a location in processor-specific kernel space. Open a location in a bus-specific I/O page.
R	Display diagnostic partitions, cache-bypass partitions, partition-bus masks for device common blocks.
TIM	Display Time-of-Day clock setting, set TOD clock.
TKTN	Print processor number on Task Terminations.

Commands:

EXTPAR	Extend partition size.
RUN	Modified to accept processor affinity specification.
SET/MAIN (SET/SUB)	Allocate diagnostic partition, set cache-bypass attribute, allocate device partition on one bus.

6.2.1 <u>RMDEMO Modificatons</u>

RMDEMO will be modified to expand the 'current task' display so that the active task on each processor will be displayed.

6.2.2 <u>DEV</u>

The device list will be modified to display:

- o The physical device name for all reconfigurable devices. (Refer to Section 2.2, "NAMING CONVENTIONS".)
- o The status display will be modified to distinguish between devices that are 'offline' or 'not present'.

6.2.3 <u>TASKLIST/TAL</u>

Tasklist will be modified to display the following:

AFF	Task-CPU affinity attribute
mask	Octal mask indicating task-bus affinity

6.2.4 <u>ATL/ACT</u>

ATL and ACT will be modified to display the following:

AFF	Task-CPU affinity attribute
mask	Octal mask indicating task-bus affinity
cpu	Current processor (or last processor if task is not active)

6.2.5 PAR

PAR will be modified to display:

DIAG	Diagnostic partition
mask	Partition-bus mask
BYP	'cache bypass' attribute for a partition

6.2.6 TIM

The syntax of the TIM command will be modified to allow the time-of-day clock setting to be displayed alongside the time recorded by the operating system.

In addition, the command will be modified to update the TOD hardware registers whenever the time is reset. A change in base year recorded in the task image, if required, will be announced by an error message.

Syntax:

```
TIM /TOD
```

Error Messages:

```
TIM - TIME-OF-DAY CLOCK NOT PRESENT
```

Cause:

The Time-of-Day clock is not installed on the system.

Remedial Action:

Reissue the command without the /TOD switch.

```
TIME-OF-DAY CLOCK NOT UPDATED
```

Cause:

This message, recorded both on the operators terminal and the terminal of the issuing task, was issued because a TIM command to reset the system clock was unable to access the write-protected TOD registers.

Remedial Action:

To eliminate the discrepancy between the TOD and operating system time, write-enable the registers and re-issue the command.

```
BASE YEAR MUST BE RESET
```

Cause:

A command to reset the TIM resulted in a discrepancy between the 'base-year' recorded in the system image and the current base-year. If a subsequent power recovery or reboot is done, there will be a discrepancy between actual time and TOD time as read from the device registers.

Remedial Action:

Update the base year in the system image, using the VMR TIM command.

6.2.7 Task Termination Notification (TKTN)

Task TKTN is responsible for notifying the operator of abnormal task termination. This service will be modified for the multiprocesig environment to display processor number.

6.2.8 RUN

ɜ following syntax will be added to the 'RUN' command to start a task having task-CPU affinity.

Syntax:

```
     RUN taskname/switches/CPU=n
```

where:

taskname	is a task specification in the currently accepted format.
switches	are one or more switch mnemonics from among those now recognized.
n	is the number of a processor for which task-CPU affinity is requested.

Error Messages:

```
     NON-EXISTENT PROCESSOR SPECIFIED
```

Cause:

The processor is not known to the operating system.

6.2.9 SET /MAIN (SET /SUB)

These functions will be modified to allow the allocation of maintenance partitions, bus-local device partitions, and the enabling/disabling of the cache-bypass attribute for a partition. Device partitions can have physical memory allocations that overlap, provided each partition is on a different bus.

Syntax:

Device partitions:

```
     SET /MAIN = name:base:length:DEV:nn
     SET /SUB = main:sub:base:length:nn
```

where:

nn	is a bus run number. This parameter is mandatory in a multiprocessor system whenever a device partition is allocated.

Cache Bypass Attribute

```
     SET /MAIN = name:base:length:type:attr
     SET /SUB = main:sub:base:length:attr
```

where:

attr	is the string 'BYP' specifying 'cache bypass'. The default is cache on.

The remaining parameters are as currently defined.

When the cache bypass attribute is specified for a common block, all task references to the portion of the virtual address used to map the shared region are diverted to backing store.

Maintenance Partitions:

```
     SET /MAIN = name:box:MNT
```

where:

name	is the partition name.
box	is the memory box on which the base address of the partition will be aligned.
MNT	specifies the maintenance attribute.
	The partition is sized and aligned on box boundaries based on the interleaving in effect when the command was issued, as shown below.

<u>Interleave factor</u>	<u>Boxes allocated</u>
none	1
2-way	2
4-way	4

The partition is sized according to the amount of memory in the allocated boxes (as read from the MK11 CSR registers).

NOTE

This command cannot be invoked by VMR.

Error Messages:

NON-EXISTENT BOX SPECIFIED

Cause:

The box number for the device partition specified is not known to the operating system.

Remedial Action:

Reissue the command with a valid box number.

BOX IS OFFLINE

Cause:

The specified box is not allocated for use by the online system.

Remedial Action:

Use the reconfiguration commands to place the box online.

6.2.10 <u>OPEN</u>

MCR command OPEN must be modified to recognize one of the following qualifiers:

- o An optional CPU number whenever the /KNL switch is invoked.
- o An optional bus number whenever a physical address on the I/O page is specified.

Syntax

```
     OPE  vloc/KNL[:cpu]
     OPE  ploc/[BUS:bus]
```

where:

vloc is a virtual address in the kernel space of the specified CPU.

cpu is a CPU number. If unspecified, a value of 0 is assumed.

ploc is a physical memory address.

bus is a bus run specification in the form:

Sn	for switched bus n or
Pn	for processor bus n

If unspecified, the string 'P0' (processor bus run 0) is assumed.

Whenever the switch:

/PAR=name

is invoked, the OPE command will assume the appropriate bus run if the named partition resides on the I/O page.

Bus need not be specified whenever:

- o ploc is not in the I/O page or
- o processor bus 0 is referenced.

'cpu' need not be specified whenever:

- o A kernel reference to processor 0 address space is intended.
- o A kernel reference to a virtual address higher than 4K but less than 160000 is specified.

Kernel references to the I/O page (addresses above 157777) should be qualified by a processor number.

Error Messages:

NON-EXISTENT PROCESSOR OR BUS

Cause:

OPE command referenced a bus or processor that was not known to the operating system.

Remedial Action:

Correct the CPU or bus specification and reissue the command.

PROCESSOR OR BUS IS OFFLINE

Cause:

An OPEN command directed to an I/O page address could not be performed because the switched bus or controlling processor was offline.

Remedial Action:

Use the reconfiguration commands discussed in Chapter 5 to display the status of the functional unit and configure the appropriate system component online, then reissue the OPE command.

6.2.11 EXTPAR

This function is provided to allow the size of a partition to be extended into unallocated memory. Extension is only permitted for a system controlled partition. When issued, the command causes the size be increased until:

- o The end of memory is reached, or
- o Another partition is encountered.

Syntax

```
EXTPAR = parname
```

Where:

parname	is the name of a system controlled partition.

Error messages

NONEXISTENT PARTITION

Cause:

Named partition does not exist.

Remedial Action:

None

NOT A SYSTEM CONTROLLED PARTITION

Cause:

The named partition was not system controlled.

Remedial Action:

Use the PAR command to display partitions and attributes, then reissue the command if appropriate.

6.3 OFFLINE OPERATOR COMMANDS AND SYSTEM DISPLAYS

The following VMR commands will be modified to accept the syntax and functional extensions described for MCR:

```
DEVICES
TASKLIST
RUN
SET /MAIN
SET /SUB
PAR
EXTPAR
```

NOTE

Maintenance partitions cannot be setup by VMR.

6.3.1 VMR Commands to Setup the Multiprocessor Environment

A single VMR command is provided to perform the allocation of private memory to each processor, as described in Chapter 3.

Syntax:

```
SET /(NO) EXEPAR= base
```

where:

base	is a physical address that defines the location of an area size: (n-1)*8KB 'n' is the number of processors in the system.

The command references a pre-allocated PCB in SYSCM that is setup with e base and top addresses. The partition is threaded into the PCB ıst and marked busy.

The partition is removed by issuing the command:

```
     SET /NOEXEPAR
```

Error Messages:

```
     NOT A MULTIPROCESSOR SYSTEM
```

Cause:

The command was issued for a non-multiprocessor operating system.

Remedial Action:

None.

```
     SPACE USED
```

Cause:

An attempt has been made to create an Executive partition in an area already occupied.

Remedial Action:

Use the PAR command to display the status of memory, then reissue the command

```
     NO PARTITION ALLOCATED
```

Cause:

An attempt was made to delete a nonexistent Executive partition

Remedial Action:

None

6.4 INTERNALLY EXTENDED OR MODIFIED OPERATOR COMMANDS

The following VMR and MCR commands have been modified internally because of multiprocessing or I/O data structure changes. No added functionality is visible to the user.

LOAD/UNLOAD	- Recognize new data structures for I/O.
INSTALL	- Set cache - bypass attribute for global common blocks. Detect conflicts in device common block references.

INSTALL will detect and report a conflict whenever a task references two device common blocks that are not visible from the same processor.

Error Message

INS - MULTIPLE DEVICE COMMONS NOT ON ONE CPU

Cause:

As described above, an attempt was made to install a task referencing device common blocks that are on seperate processors.

Remedial Action:

If possible, remove the conlict by issuing a reconfiguration command to appropriately reposition a switched bus.

6.5 XDT/BRK MODIFICATIONS

The Executive Debugging tool must be modified to recognize the 'low core' region associated with each processor in the multiprocessor environment. This utility will also suspend all processors whenever an Executive breakpoint or other processor trap occurs.

The changes required are summarized as follows:

- o Provide a command to switch CPU context.
- o Correctly process breakpoints associated with the low core region belonging to each CPU.
- o Suspend operation of all processors whenever a synchronous trap occurs in the Executive.
- o Modify the XDT terminal driver to direct I/O to a single console terminal (regardless of the processor context).

When multiprocessing is in effect, the Executive debugger should precede all displays of memory context with processor number.

When the 'proceed' command is issued, the debugger must correctly re-dispatch all processors.

All breakpoints apply to every processor. There will be no way to restrict a breakpoint to a single CPU. The trap is sprung by the first CPU that executes the BRK instruction.

5 CHANGES IN POST-MORTEM DUMP FORMAT

Task PMD dumps the task image to a hard-copy device in case of abnormal termination. This task will be modified to print the CPU on which the task was running whenever the system incorporates multiprocessor support.

CHAPTER 7

SYSTEM GENERATION DIALOGUE

(To be supplied)

INDEX to Appendix A

ADDENDUM A

NEW DATA STRUCTURES

In the descriptions that follow, "same" implies that the description of a certain field is the same at that found in the current Guide to Writing an I/O Driver manual.

A number of fields are conditional within both the SCB and the KRB. These are flagged by a "note number" in the NOTES column on the diagram.

NOTE

These figures are included for illustrative purposes only. They are not to be used for coding.

S C B

STATUS CONTROL BLOCK

```
                 N
                 O
                 T
                 E
                 S
                 +===============================================+
                 !                                               !  :0
S.LHD            +------------  INPUT/OUTPUT QUEUE  -------------+
                 !                                               !  :2
                 +-----------------------------------------------+
S.URM            ! 1        FORK UNIBUS RUN MASK                 !  :4
                 +-----------------------------------------------+
S.FRK            !              FORK LINK                        !  :6
                 +-----------------------------------------------+
                 !               FORK PC                         !  :10
                 +-----------------------------------------------+
                 !               FORK R5                         !  :12
                 +-----------------------------------------------+
                 !               FORK R4                         !  :14
                 +-----------------------------------------------+
S.KS5            ! 2          DRIVER/FORK KISAR5                 !  :16
                 +-----------------------------------------------+
S.PKT            !            I/O PACKET ADDRESS                 !  :20
S.CTM            +-----------------------------------------------+
S.ITM            !  INITIAL TIMEOUT CNT  !  CURRENT TIMEOUT CNT  !  :22
S.STS            +-----------------------------------------------+
S.ST2            !   STATUS EXTENSION    !         STATUS        !  :24
                 +-----------------------------------------------+
S.KRB            !              KRB ADDRESS                      !  :26
S.ROFR           +-----------------------------------------------+
S.RCNT           ! 3 OFFSET TO DEV REG.  !  NUMBER TO COPY       !  :30
                 +-----------------------------------------------+
S.KTB            ! 4           KRB ADDRESS 0                     !  :32
                 +-----------------------------------------------+
                 ! 4           KRB ADDRESS 1                     !  :34
                 +-----------------------------------------------+
                                    .
                                    .
                                    .
                 +-----------------------------------------------+
                 ! 5           KRB ADDRESS N                     !
                 +-----------------------------------------------+
                 ! 5                0                            !
                 +-----------------------------------------------+
```

NOTES:

THE NOTED WORDS APPEAR IF THE SYMBOLS BELOW ARE DEFINED BY SYSGEN.

	IF DF ...
1	M$$PRO (MULTIPROCESSOR SUPPORT)
2	L$$DRV (LOADABLE DRIVER SUPPORT)
3	E$$DVC (ERROR LOGGING)
4	M$$ACD AND S2.MAD (MULTIACCESS DEVICE)
5	M$$ACD AND (S2.MAD AND (NOT S2.DAD))

The SCBs in the system are generally the same length, except for the length of the KRB table at the end.

LHD (first word equal zero; second word points to first).

Driver access:

Initialized, not referenced.

Description:

Two words which form the I/O queue listhead. The first word points to the first I/O packet in the queue, and the second word points to the last I/O packet in the queue. If the queue is empty, the first word is zero, and the second word points to the first word.

S.URM (controller unibus run mask)

Driver access:

Initialized, not referenced.

Description:

Contains a unibus run mask that defines the unibus segment to which the currently assigned controller is attached. When controller assignment is made, this cell is set from K.URM. ADJACENCY WITH THE FORK BLOCK IS ASSUMED!

S.FRK (reserve four words of storage).

Driver access:

Not initialized or accessed.

Description:

The four words starting at S.FRK are used for fork block storage if and when the driver deems it necessary to establish itself as a fork process. Fork block storage preserves the state of the driver, which is restored when the driver regains control at fork level. This area is automatically used if the driver calls $FORK. The FORK processor also depends on the adjecency of S.URM and S.KS5 if the required support is generated into the system.

S.KS5 (initialized to zero).

Driver access:

Initialized, not referenced.

Description:

This word contains the contents of KISAR5 necessary to correctly alter the execute mapping to reach the driver for this unit. It has no meaning for a driver that is not loadable. It is set by LOA, and whenever a fork block is dequeued and executed, this word is unconditionally jammed into KISAR5. ADJACENCY WITH THE FORK BLOCK IS ASSUMED!

S.PKT (reserve one word of storage).

Same

S.CTM/ITM

Same

S.ST2 (controller status extension).

Driver access:

Initialized, not referenced.

Description:

This status byte defines certain relatively static status conditions for the controller - unit combination. The currently defined bits are:

```
S2.MAD=1 ; Multiaccess Device (1=Yes)
S2.LDS=2 ; Load Sharing Enabled (1=Yes)
S2.DAD=4 ; Dual Access Device (1=Yes)
S2.SOP=10 ; Seek Optimization Supported (1=Yes)
S2.ERL=20 ; Error Logging Supported (1=Yes)
```

S.STS (initialize to zero)

Driver access:

Initialized, not referenced.

Description:

Establishes the controller as busy/not busy. This byte is the interlock mechanism for marking a driver as busy for a specific controller. Tested and set by $GTPKT and tested by $IODON.

S.KRB (pointer to currently assigned KRB)

Driver access:

Initialized, referenced by driver to get to KRB.

Description:

This word points to the currently assigned KRB. For non-multiaccess devices, it is set at SYSGEN time and never altered. For multiaccess devices with load-sharing enabled, it may take on the value of one of the KRB pointers in the KRB table, S.CTB.

S.KTB (table of KRB address)

Driver access:

Initialized, not referenced.

Description:

Every controller to which the unit (UCB-SCB combination) can communicate is represented in this table by a KRB address. If S2.MAD this table will exist, and if S2.DAD it contains only two entries, with no terminating zero word.

Otherwise, it may contain any number of entries, with the list terminated by a zero word. Bit zero of each word is an on/offline flag which, when set, indicates that KRB is offline with respect to this SCB and should not be considered for controller assignment.

```
                                   K R B

                          CONTROLLER REQUEST BLOCK


                     N
                     O
                     T
                     E
                     S

   SCB    KRB
  S.VCT  K.VCT  +------------------------------------------------+
  S.CON  K.CON  !        VECTOR/4        !       PRIORITY        !  :-6
         K.IOC  +------------------------------------------------+
  S.PRI  K.PRI  !  CONTROLLER I/O COUNT !  CONTROLLER INDEX      !  :-4
                +------------------------------------------------+
         K.STS  !               CONTROLLER STATUS                !  :-2
                +================================================+
  S.CSR  K.CSR  !                  CSR ADDRESS                   !  :0
         K.HPU  +------------------------------------------------+
         K.OFF  ! HIGHEST PHYSICAL UNIT ! OFFSET TO UCB TABLE    !  :2
                +------------------------------------------------+
         K.OWN  !                     OWNER                      !  :4
                +------------------------------------------------+
         K.CRQ  ! 2                                              !  :6
                +------      CONTROLLER REQUEST QUEUE      ------+
                ! 2                                              !  :10
                +------------------------------------------------+
         K.URM  ! 3        CONTROLLER UNIBUS RUN MASK            !  :12
                +------------------------------------------------+




                +------------------------------------------------+
                ! 4       11/70 UMR/RHBAE OFFSET                 !
                +------------------------------------------------+
                ! 4                     -                        !
                +------------------------------------------------+
                ! 4                     -                        !
                +------------------------------------------------+
                ! 4                     -                        !
                +------------------------------------------------+
                ! 4                     -                        !
                +------------------------------------------------+
                ! 4       11/70 UNIBUS MAPPING REGISTER          !
                +------------------------------------------------+
                ! 5        UCB ADDRESS PHYSICAL UNIT 0           !
```

```
+-----------------------------------------------+
                      .
                      .
                      .
+-----------------------------------------------+
! 5       UCB ADDRESS PHYSICAL UNIT N           !
+-----------------------------------------------+
! 5                     0                       !
+-----------------------------------------------+
```

NOTES:

IF DF ...

1. THIS FIELD WILL OVERLAP THE I/O REQUEST QUEUE WHEN CONTIGUOUS ALLOCATION OF KRB AND SCB IS USED.
2. M$$PRO (MULTIPROCESSOR SUPPORT)
3. M$$EXT (11/70 EXTENDED MEMORY SUPPORT)
4. M$$ACD (MULTIACCESS DEVICE SUPPORT)
5. E$$DVC OR R$$CON (ERROR-LOGGING SUPPORT)

VCT (interrupt vector divided by four) same

K.PRI (device priority) same

K.CON (controller number times 2) same

K.IOC (Initially zero).

Driver access:

Initialized, not used.

Description:

This is an I/O count used by the system to keep track of how busy the controller is. The value is directly proportional to the number of outstanding requests queued for this controller. This is a weighted number to be used to judge the relative activity of a controller with respect to another controller, and the actual numeric value should not be used, as it may change when different weighting schemes are used.

K.STS (initialize with appropriate status bits)

Driver access:

Initialized, not referenced.

Description:

This word is used as a status word that concerns the controller. Currently defined bits are:

```
KS.OFL=1        ;controller offline
KS.SDX=2        ;seeks allowed during data transfers
KS.UOP=4        ;unit operations in parallel supported
KS.MBC=10       ;massbus controller
KS.MOF=20       ;marked for offline
```

K.CSR (Controller Status Register address) same

K.CSR=0 is guaranteed.

K.OFF (offset in bytes to start of UCB table).

Driver access:

Initialized, referenced by interrupt dispatch code.

Description:

This byte contains the byte offset to the beginning of the UCB table. When added to the starting address of the SCB it will yield the UCB table address. The unibus mapping registers (UMRs) extend in a negative direction from the start of the UCB table.

K.HPU (highest physical unit number).

Driver access:

Initialized.

Description:

This byte contains the value of the highest physical unit number used on this controller.

K.OWN (initially zero).

Driver access:

Initialized, referenced for actual unit.

Description:

This a busy/non-busy interlock for the controller. If the controller is busy, this word contains the UCB address of the currently active unit. This word in each KRB can replace the old CNTBL concept for multi-controller drivers. Drivers may still use an internal CNTBL if they so desire.

K.CRQ (first word equal zero; second word points to first).

Driver access:

Initialized, not referenced.

Description:

Two words which form the controller wait queue. Fork blocks are queued here for driver processes that have requested controller access. Driver processes that request access for control functions are queued on the front of the list, and those that request access for data transfer are queued on the end of the list.

K.URM (controller unibus run mask)

Driver access:

Initialized, not referenced.

Description:

Contains a unibus run mask that defines the unibus segment to which the controller is attached. When controller assignment is made, the cell is moved into S.URM for the fork block there.

Table of UCB addresses -- pointed to by K.OFF

Driver access:

Initialized, referenced by interrupt dispatch code.

Description:

Table contains the UCB addresses for the units on this controller. Physical unit zero is in the first word, unit one in the second and so on. The table is terminated by a zero word. Currently, a controller may have a maximum of eight units (hardware limitation).

Unibus Mapping Registers:

The UMR work area extends down from the start of the UCB table (if any). The first word is used as the RHBAE offset for the RH70. This byte value is the offset that, when added to the CSR address contained in K.CSR, will yield the address of the RMBAE register on the RH70.

```
                          K R B 1

                  SUB-CONTROLLER REQUEST BLOCK

K.IOC  +-------------------------------------------------+
K.PRI  ! CONTROLLER I/O COUNT ! CONTROLLER INDEX          ! :-4
       +-------------------------------------------------+
K.STS  !              CONTROLLER STATUS                   ! :-2
       +=================================================+
K.CSR  !                 CSR ADDRESS                      ! :0
K.HPU  +-------------------------------------------------+
K.OFF  ! CONTROLLER I/O COUNT  !  OFFSET TO UCB TABLE     ! :2
       +-------------------------------------------------+
K.OWN  !                    OWNER                         ! :4
       +-------------------------------------------------+
K.CRQ  !                                                  ! :6
       +------    CONTROLLER REQUEST QUEUE          ------+
       !                                                  ! :10
       +-------------------------------------------------+



       +-------------------------------------------------+
       !          UCB ADDRESS PHYSICAL UNIT 0             !
       +-------------------------------------------------+
                               .
                               .
                               .
       +-------------------------------------------------+
       !          UCB ADDRESS OF PHYSICAL UNIT N          !
       +-------------------------------------------------+
       !                       0                          !
       +-------------------------------------------------+
```

This entire control block appears only if S$$UBD is defined at SYSGEN time.

NOTE

This control block looks very similar to the beginning of the KRB. This is to allow the same routines to be used to request and release sub-controller access as are used for controller access. Furthermore, the table of UCB addresses must only be accessed by using the offset (to allow for expansion of the KRB1 in the future).

K.PRI (device priority) same

K.IOC (initially zero).

Driver access:

Initialized, not used.

Description:

This is an I/O count used by the system to keep track of how busy the controller is.

K.STS (initialize with appropriate status bits)

Driver access:

Initialized, not referenced.

Description:

This word is used as a status word that concerns the controller. Currently defined bits are:

```
KS.OFL=1         ;controller offline
KS.SDX=2         ;seeks allowed during data transfers
KS.UOP=4         ;unit operations in parallel supported
KS.MOF=20        ;controller marked for offline
```

K.CSR (Controller Status Register address) same

K.CSR=0 is guaranteed.

K.OFF (offset in bytes to start of UCB table).

Driver access:

Initialized, referenced by interrupt dispatch code.

Description:

This byte contains the byte offset to the beginning of the UCB table. When added to the starting address of the SCB it will yield the UCB table address. The unibus mapping registers (UMRs) extend in a negative direction from the start of the UCB table.

K.HPU (highest physical unit number).

Driver access:

Initialized.

Description:

This byte contains the value of the highest physical unit number used on this controller.

K.OWN (initially zero).

Driver access:

Initialized, not referenced.

Description:

This a busy/non-busy interlock for the controller. If the controller is busy, this word contains the UCB address of the current controller.

K.CRQ (first word equal zero; second word points to first).

Driver access:

Initialized, not referenced.

Description:

This byte is currently unused.

Beginning of table of UCB addresses for slave units.

Driver access:

Referenced at interrupt.

Description:

Used by driver to determine the UCB of an interrupting slave unit from the unit number returned by the sub-controller. This table need only be as long as the number of slave units. It is terminated by a zero word.

DEVICE DISPATCH TABLE

```
                +=================================================+
D.VINI          !             INITIATOR ENTRY POINT               !  :0
                +-------------------------------------------------+
D.VCAN          !             CANCEL ENTRY POINT                  !  :2
                +-------------------------------------------------+
D.VTIM          !             TIMEOUT ENTRY POINT                 !  :4
                +-------------------------------------------------+
D.VPWF          !             POWERFAIL ENTRY POINT               !  :6
                +-------------------------------------------------+
D.VINT          !             GENERIC CONTROLLER NAME (ASCII)     !  :10
                +-------------------------------------------------+
                !             INTERRUPT ENTRY POINT 0             !
                +-------------------------------------------------+
                                        .
                                        .
                                        .
                +-------------------------------------------------+
                !             INTERRUPT ENTRY POINT N             !
                +-------------------------------------------------+
                !                       0                         !
                +-------------------------------------------------+
                !             GENERIC CONTROLLER NAME (ASCII      !
                +-------------------------------------------------+
                !             INTERRUPT ENTRY POINT 0             !
                +-------------------------------------------------+
                                        .
                                        .
                                        .
                +-------------------------------------------------+
                !             INTERRUPT ENTRY POINT N             !
                +-------------------------------------------------+
                !                       0                         !
                +-------------------------------------------------+
                                        .
                                        .
                                        .
```

D.VINI-
D.VPWF same

VINT (beginning of interrupt address blocks)

Driver access:

Initialized, not used.

Description:

The interrrupt address block is a structure that defines the address(es) to include in the vector for this driver. It is general enough to support multicontroller drivers, such as the terminal driver. Furthermore, it sets no restriction on the number of vectors each controller may have. The number of vectors is implicit in the number of addresses in the interrupt address block.

NOTE

No end-of-table marker exists. Every entrance to this table must be to search for a specific generic controller name. It is not possible to scan this table to the end.

CONTROLLER TABLE

(CTB)

```
               +================================================+
   L.LNK       !                LINK TO NEXT CTB                ! :0
               +------------------------------------------------+
   L.NAM       !         GENERIC CONTROLLER NAME (ASCII)        ! :2
               +------------------------------------------------+
   L.DCB       !                  DCB ADDRESS                   ! :4
               +------------------------------------------------+
   L.MAX       !       UNUSED          ! MAXIMUM KRB INDEX      ! :6
               +------------------------------------------------+
   L.KRB       !                 KRB ADDRESS 0                  ! :10
               +------------------------------------------------+
                                        .
                                        .
                                        .
               +------------------------------------------------+
               !                 KRB ADDRESS N                  ! :10+N
               +------------------------------------------------+
```

L.LNK (link to next CTB in list. Master listhead is $CTLST)

Driver access:

Not initialized or referenced.

Description

All of the CTBs in the system are linked together so they can be found, and they are threaded through this first word. A zero link terminates this list.

A CTB must exist for every controller type in the system. Drivers may continue to use their internal CTB if they wish.

L.NAM (two character ASCII device name)

Driver access:

Initialized once for the controller.

Description:

This two-character ASCII string is the controller mnemonic used to find this CTB from among all the others in the system. For the RH11 controller, it is RH instead of DB or DS or MM.

L.DCB (DCB address of the device for which this is the controller)

Driver access:

None.

Description:

The DCB pointer is used to reach the DCB, and thereby UCB and driver dispatch table for a driver. If bit 0 is a 1 (i.e., the address is odd) it is an interrupt address for a multidriver controller (i.e., the RH11/RH70 dispatch routine.)

MAX (maximum controller index)

Driver access:

None.

Description:

Used by programs which scan the CTBs to determine the number of KRB addresses.

L.KRB (KRB address of controllers)

Driver access:

Initialized once for the controller.

Description:

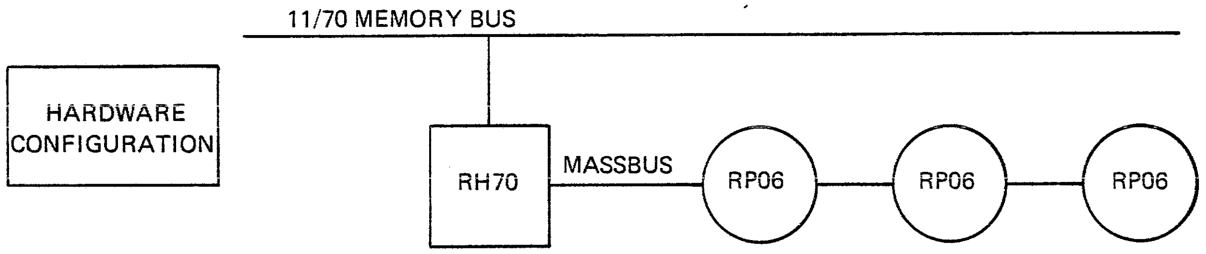
A list of the KRB addresses ordered by their respective system wide controller numbers. This table is indexed by the controller index retrieved from the PS word immediately after an interrupt. The table is terminated by a zero word: while the interrupt routines will not have to scan it in a linear fashion, the only way to find all the KRBs in the system includes a linear scan of all the CTBs. The CTB is static and created at SYSGEN time.

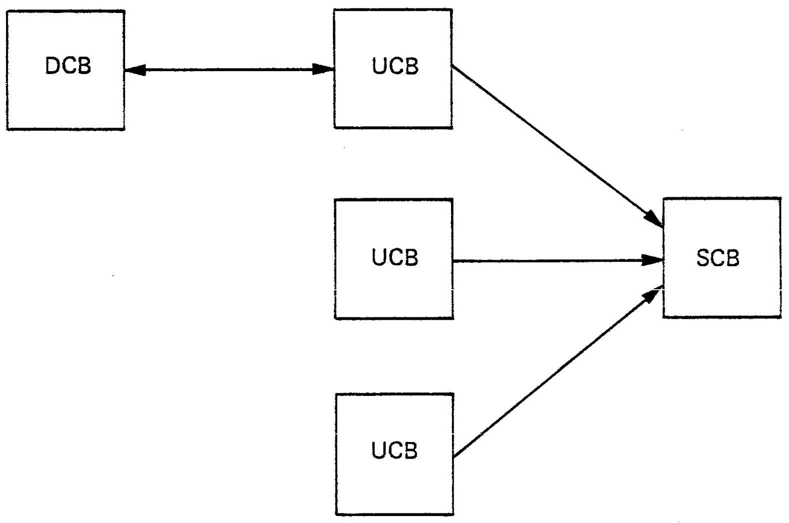
The address of the start of the KRB address in the CTB is $yzCTB where y,z are the controller mnemonic.

APPENDIX B

RELATIONSHIPS BETWEEN HARDWARE CONFIGURATIONS AND THE I/O DATA BASE

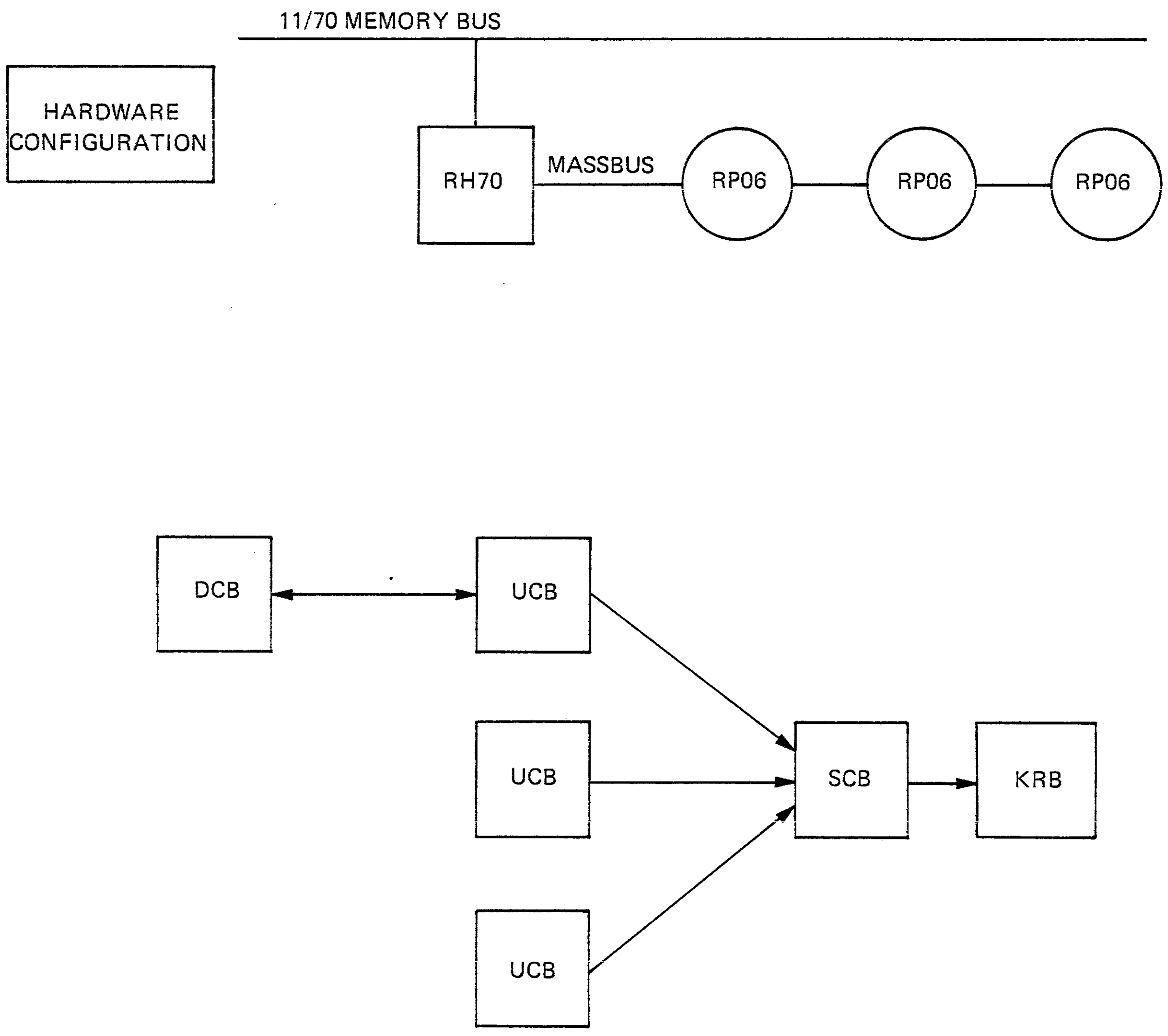
This Appendix B consists of a packet of illustrations, numbered 1 through 9.





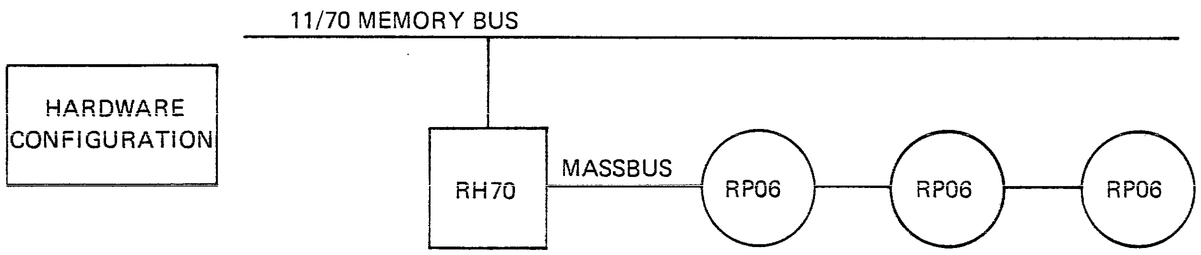
- o 3 units
- o only one unit may be active at one time

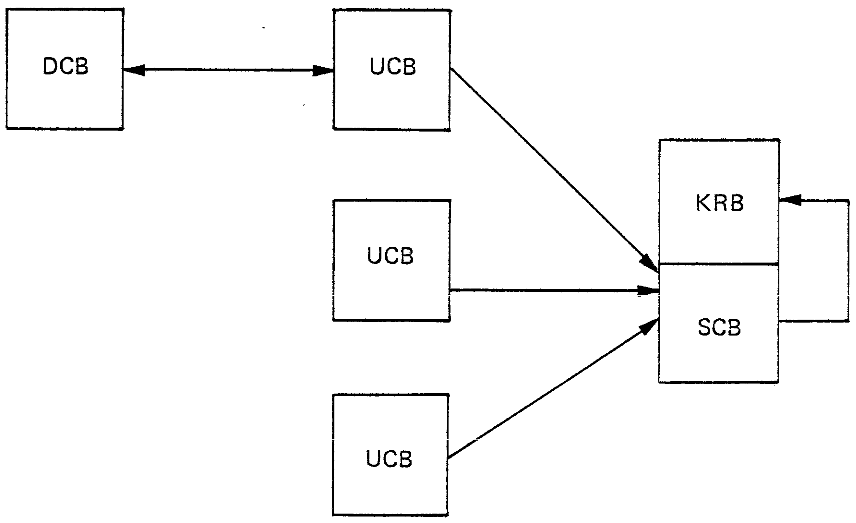
Figure 1 Logical UCB/SCB Relationship, RSX-11M V3



- o 3 units
- o only one unit may be active at one time

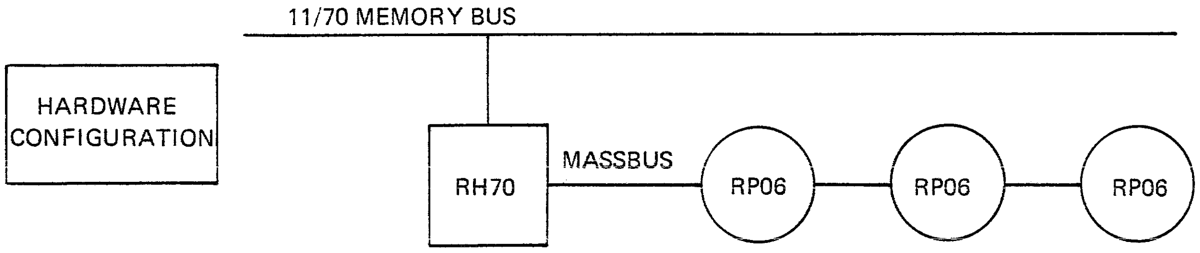
Figure 2 Logical UCB/SCB/KRB Relationship, RSX-11M V4

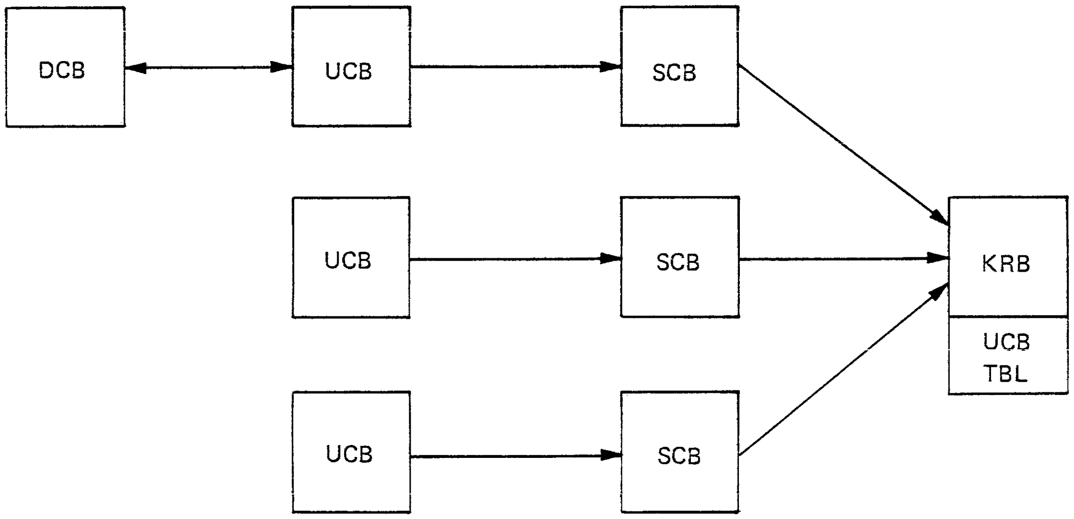




- o 3 units
- o contigious KRB allocation

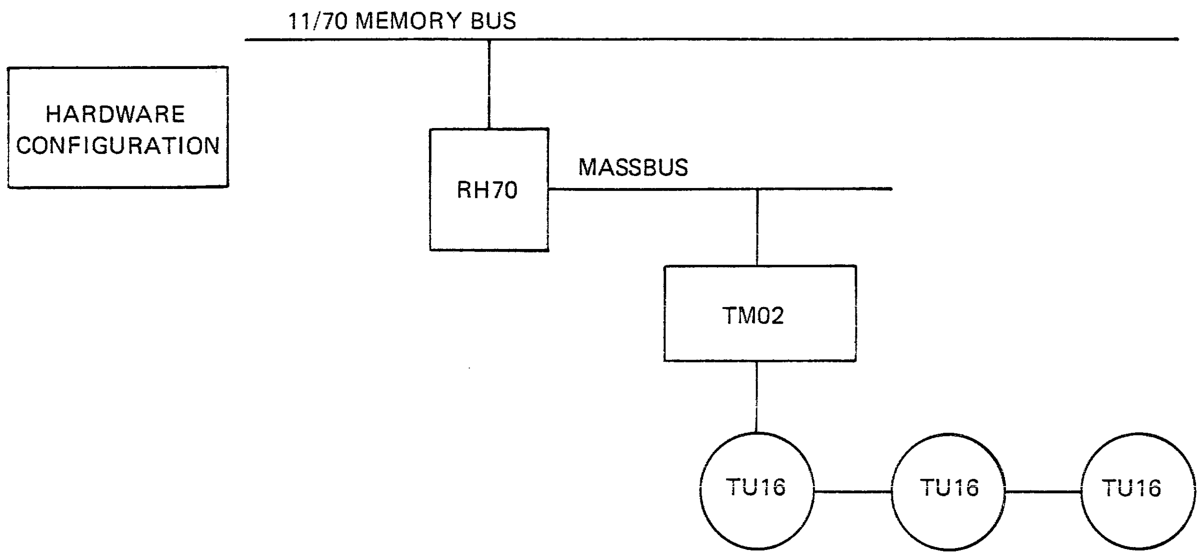
Figure 3 Physical UCB/SCB/KRB Relationship for RSX-11M V4; No Unit Operation in Parallel

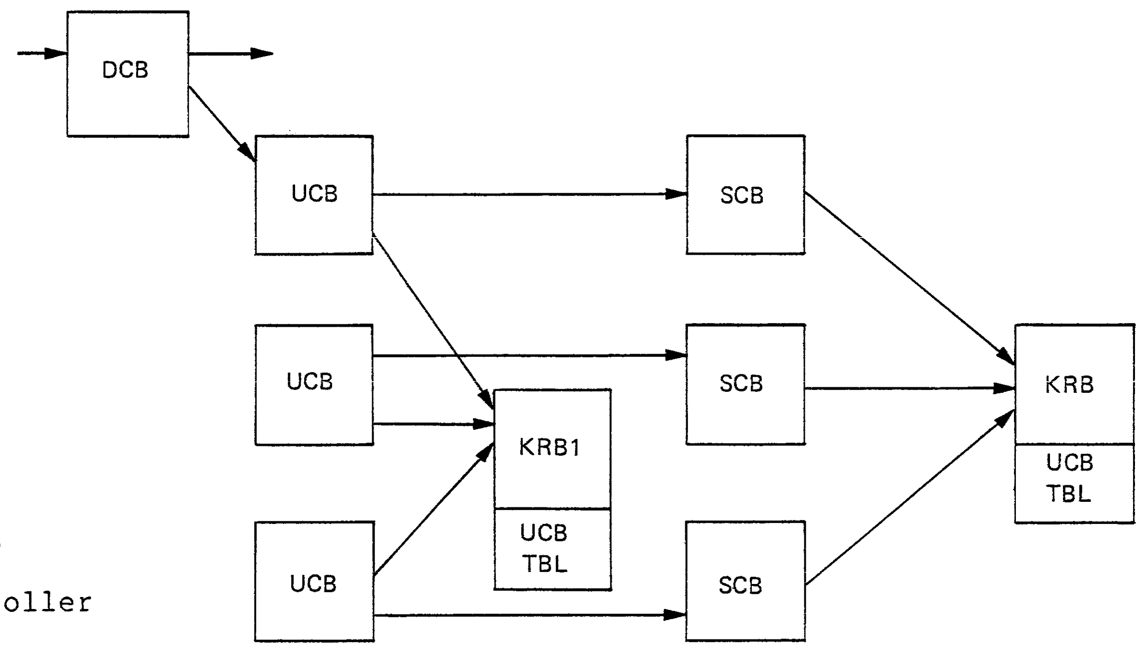




- 3 units
- unit operation in parallel

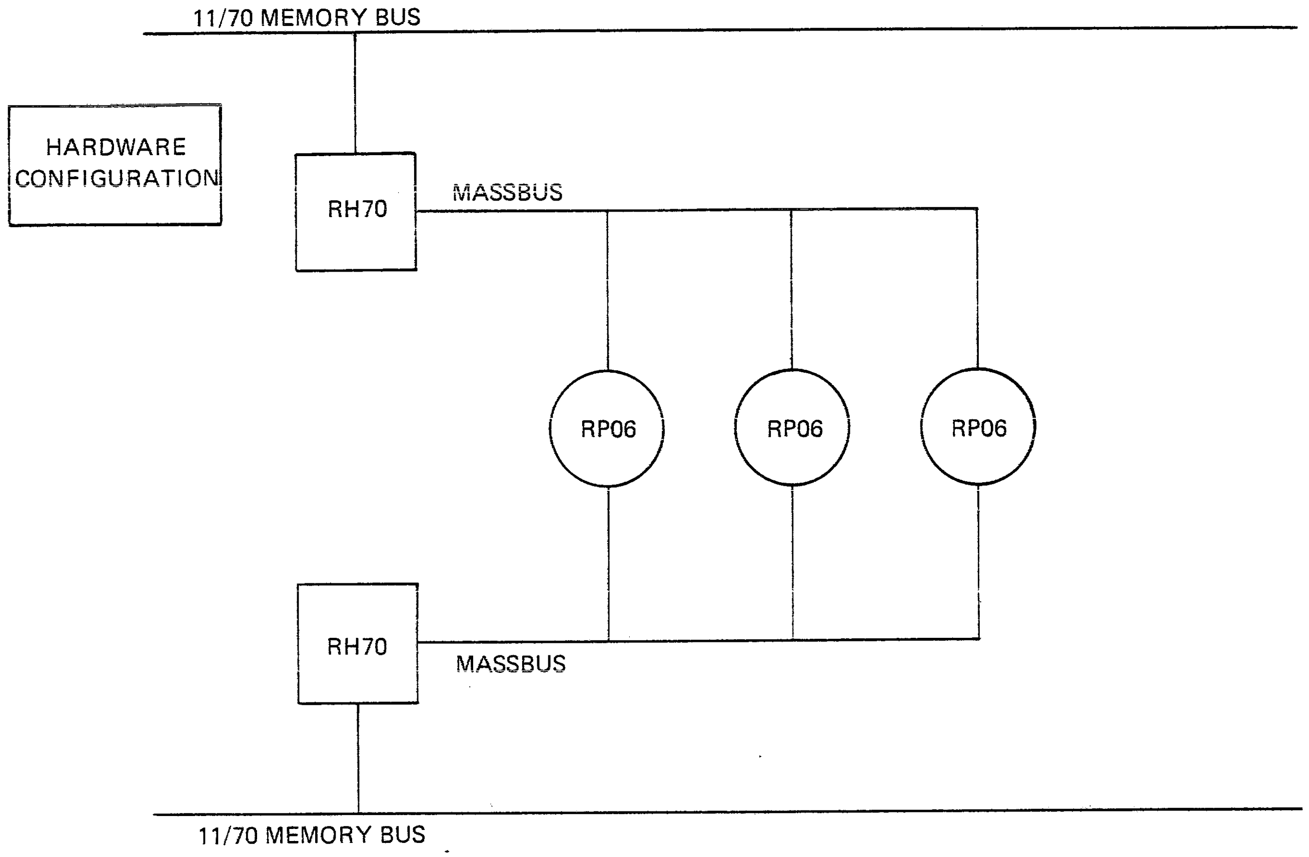
Figure 4 Physical UCB/SCB/KRB Relationship for RSX-11M V4; Unit Operation in Parallel

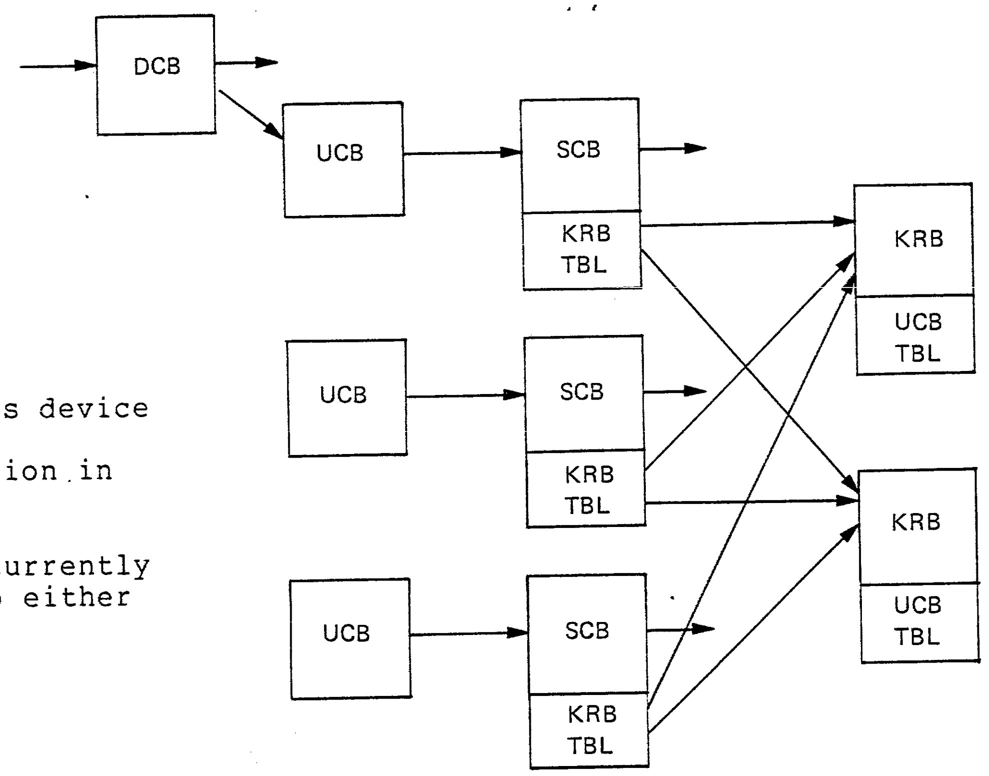




- o 3 units
- o 1 controller
- o 1 sub-controller
- o unit operation in parallel

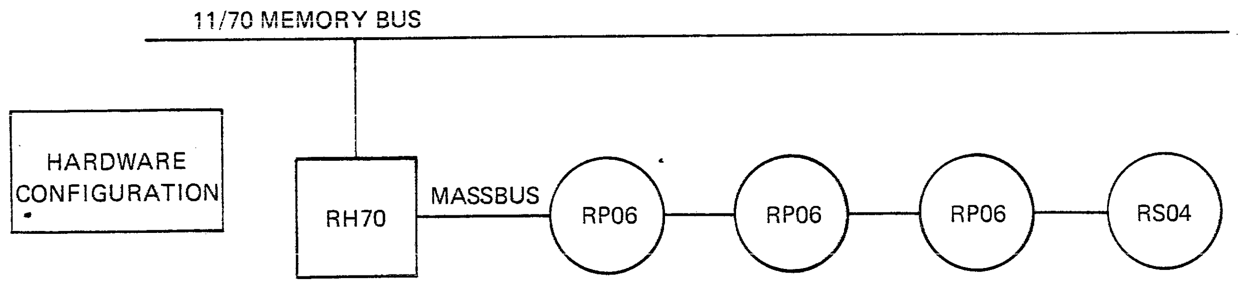
igure 5 Physical UCB/SCB/KRB/KRB1 Relationship for RSX-11M V4; Subcontroller Device

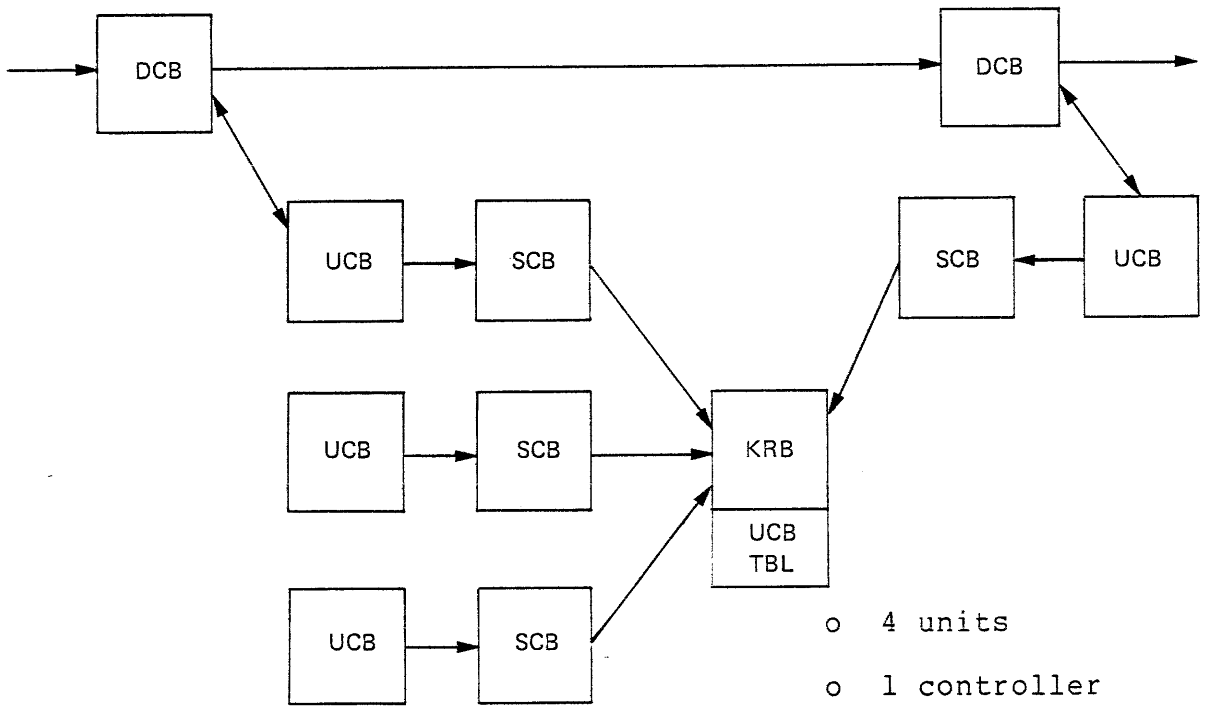




- o 3 units
- o multi-access device
- o unit operation in parallel
- o units not currently assigned to either controller

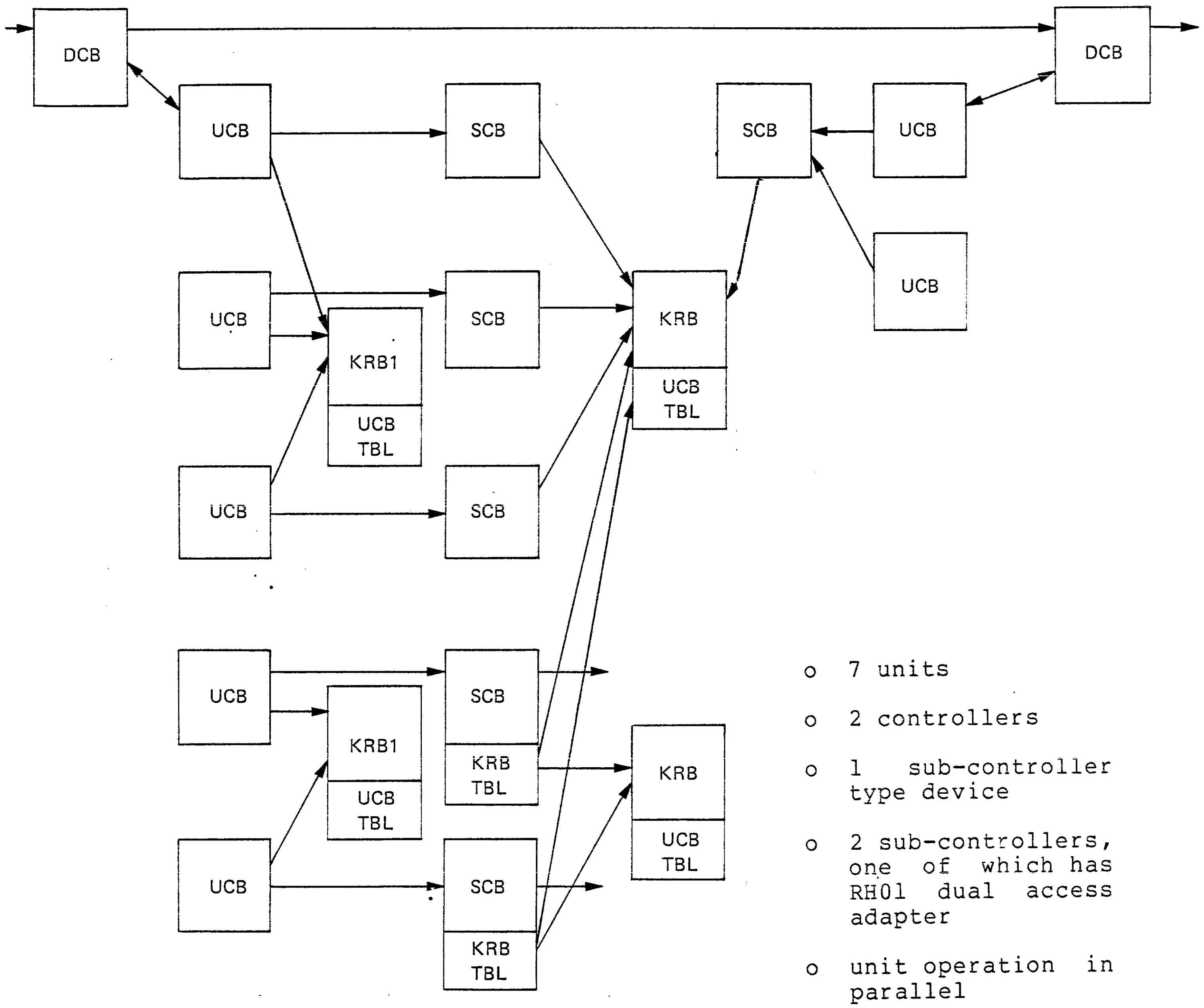
Figure 6 Physical UCB/SCB/KRB Relationship for RSX-11M V4; Multi-access Device





- 4 units
- 1 controller
- mixed massbus operation
- unit operation in parallel

Figure 7 Physical UCB/SCB/KRB Relationship for RSX-11M V4; Mixed Massbus Operation



- o 7 units
- o 2 controllers
- o 1 sub-controller type device
- o 2 sub-controllers, one of which has RH01 dual access adapter
- o unit operation in parallel

Figure 8a System Logical Relationship; UCB/SCB/KRB/KRB1

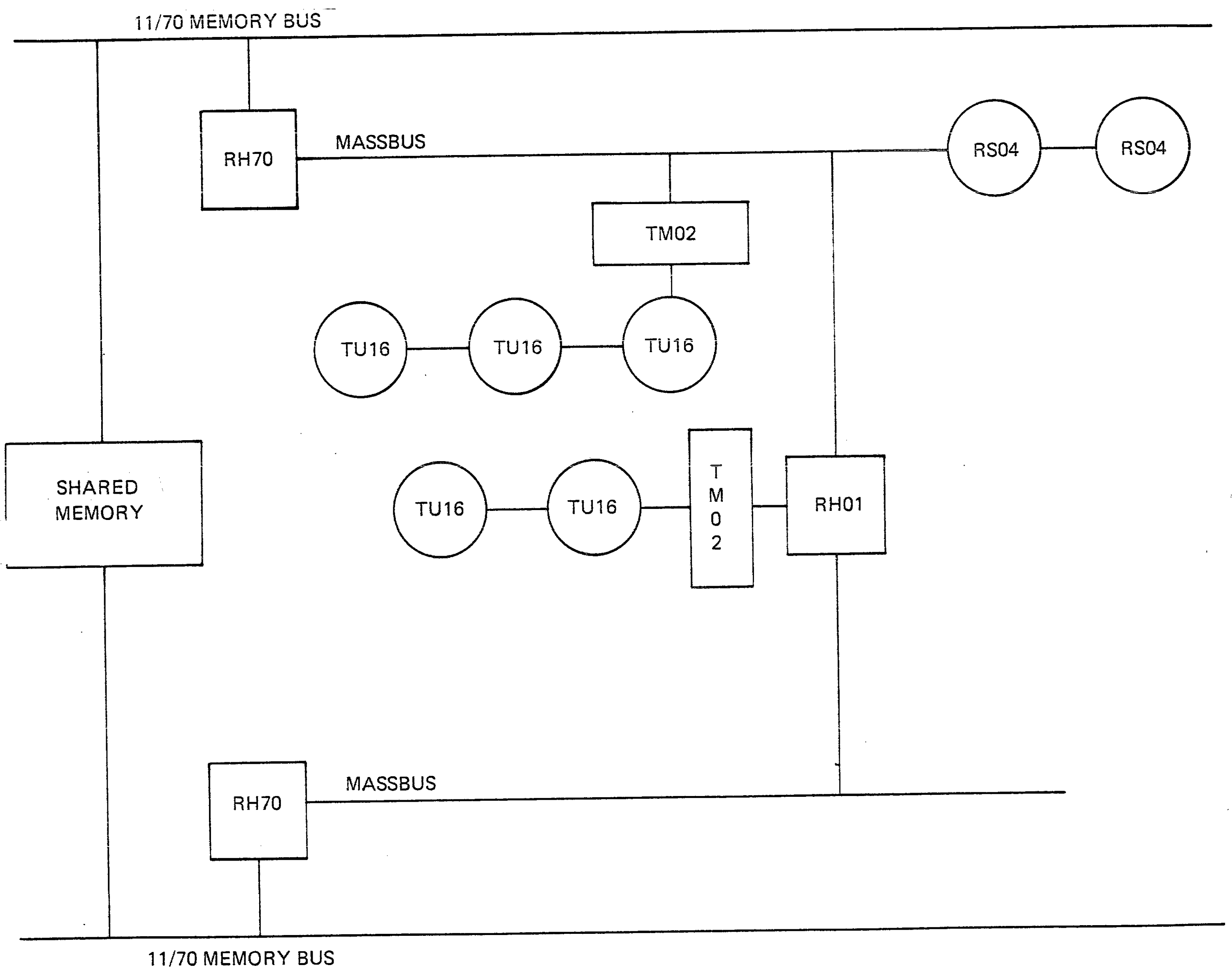


Figure 8b System Hardware Relationships

(Hardware configuration for Figure 8a)

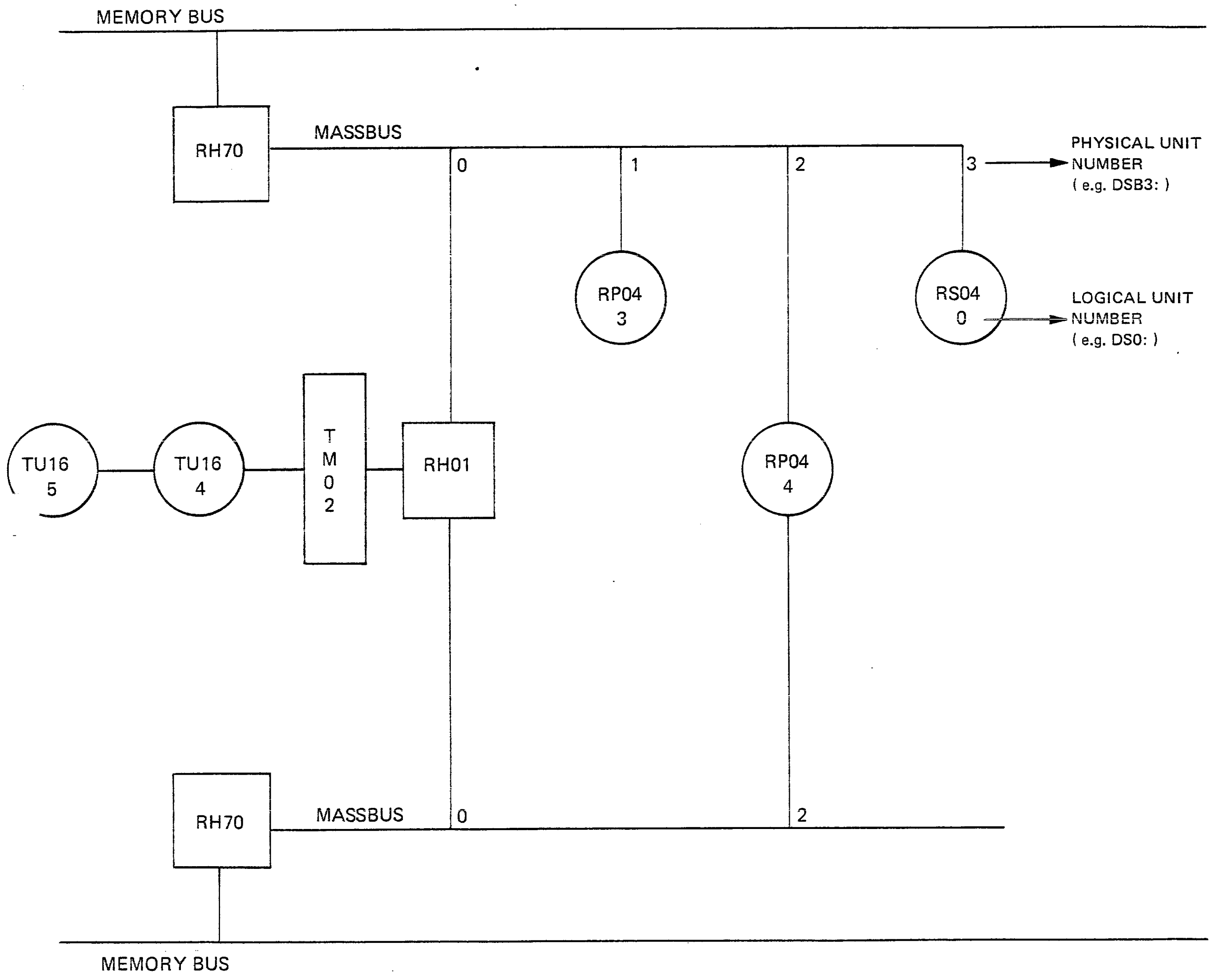


Figure 9 Example Configuration