EK-KA881-TD-PRE

VAX 8800 System Technical Description

Volume 1

FOR INTERNAL USE ONLY

digital™

EK-KA881-TD-PRE

VAX 8800 System Technical Description

Volume 1

FOR INTERNAL USE ONLY

Prepared by Educational Services
of
Digital Equipment Corporation

Preliminary Edition, July 1986

Copyright Digital Equipment Corporation 1986
All Rights Reserved

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

Printed in U.S.A.

Class A Computing Devices

Notice: This equipment generates, uses, and may emit radio frequency energy. The equipment has been type tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC rules, which are designed to provide reasonable protection against such radio frequency interference when operated in a commercial environment. Operation of this equipment in a residential area may cause interference in which case the user at his own expense may be required to take measures to correct the interference.

The following are trademarks of Digital Equipment Corporation:

logo	DECwriter	RSX
logo	DIBOL	Scholar
DEC	MASSBUS	ULTRIX
DECmate	PDP	UNIBUS
DECset	P/OS	VAX
DECsystem-10	Professional	VMS
DECSYSTEM-20	Rainbow	VT
DECUS	RSTS	Work Processor

CONTENTS

SECTION 1 INTRODUCTION AND SYSTEM OVERVIEW

CHAPTER 1 INTRODUCTION

1.1 MANUAL SCOPE 1-1
1.2 MANUAL ORGANIZATION. 1-1
1.3 RELATED DOCUMENTATION. 1-2
1.4 SYSTEM DESCRIPTION 1-3
1.5 PHYSICAL DESCRIPTION 1-3
1.6 FUNCTIONAL DESCRIPTION 1-7
1.6.1 Console Subsystem. 1-8
1.6.2 Central Processing Unit. 1-10
1.6.2.1 Instruction Box. 1-13
1.6.2.2 Execution Box. 1-16
1.6.2.3 Cache Box. 1-20
1.6.3 Clock Module 1-21
1.6.4 Memory (MBox). 1-23
1.6.4.1 Memory Control Logic 1-24
1.6.4.2 MAR4 Memory Array. 1-25
1.6.5 System Buses 1-25
1.6.5.1 VAX 8800 Memory Interconnect (NMI) 1-25
1.6.5.2 VAX Bus Interconnect (VAXBI) 1-27
1.6.5.3 Visibility Bus (VBus). 1-29
1.6.6 VAX Bus Interconnect and I/O Adapters. 1-30
1.6.7 Power System Complex 1-33
1.6.7.1 876 Power Controller 1-34
1.6.7.2 NBox Port Conditioner. 1-34
1.6.7.3 Module Power Supplies. 1-35
1.6.7.4 Environmental Monitoring Module. 1-35
1.6.7.5 Battery Backup Unit. 1-35

CHAPTER 2 SYSTEM CONTROL

2.1 GENERAL. 2-1
2.2 SYSTEM CONTROL 2-1
2.2.1 Software 2-1
2.2.2 Hardware 2-1
2.3 CONSOLE SOFTWARE COMPONENTS. 2-4
2.3.1 Control Program. 2-4
2.3.1.1 Special Control Program Features 2-4
2.3.1.2 Multiple Command Streams 2-5
2.3.2 File Transfer Program. 2-6
2.3.3 Logical Block Server Program 2-6
2.3.4 Real-Time Interface Driver 2-6
2.4 CONSOLE SUPPORT MICROCODE (CSM). 2-6
2.4.1 Console Support Microcode Structure. 2-6
2.4.2 CSM Data Transfers/Protocol. 2-7
2.4.3 Console Support Microcode Entry Points 2-7

2.5 OPERATIONAL MODES 2-8
2.5.1 Console Mode 2-8
2.5.2 Program Mode 2-9
2.6 OPERATOR/CONSOLE INTERACTION 2-9
2.6.1 Console State Bits 2-9
2.6.1.1 Command Validity 2-10
2.6.1.2 Saving Console State 2-10
2.6.2 Console Commands 2-10
2.6.2.1 Executing Console Commands 2-12
2.6.3 Console/Operator Display 2-13
2.6.3.1 Local Display During Remote Operation. . . . 2-13
2.6.3.2 Console Command Language Display Prompts . . 2-15
2.6.4 System Logfile 2-15
2.6.4.1 Displaying the Logfile 2-15
2.6.4.2 Logical Terminals/Logfile Integrity. 2-15
2.6.4.3 Saving the Logfile 2-16
2.7 POWER-UP/DOWN SEQUENCING 2-16
2.7.1 Power On 2-16
2.7.2 Power Fail 2-17
2.7.3 Powerdown. 2-17
2.7.4 Warm Restart 2-17

CHAPTER 3 SYSTEM OPERATION

3.1 INTRODUCTION 3-1
3.2 MICROCODE. 3-1
3.2.1 Microcode Characteristics. 3-1
3.2.1.1 Functionality. 3-1
3.2.1.2 Operation. 3-1
3.2.1.3 Structure. 3-1
3.2.2 Microcode Control. 3-2
3.2.2.1 Interrupt and Processor Register (INPR). . . . 3-2
3.2.2.2 Condition Code and Branch (CCBR) 3-3
3.2.2.3 Gateway Control (GWYC) 3-3
3.2.2.4 Microtrap (UTRP) 3-3
3.2.2.5 Microbranch Slices (UBRS). 3-3
3.2.3 Pipelining 3-3
3.2.4 Microtraps 3-4
3.2.5 Micromatch 3-5
3.2.5.1 Stop on Match. 3-5
3.2.5.2 Trap on Match. 3-6
3.3 MEMORY ADDRESSING AND READ/WRITE OPERATIONS. . . . 3-6
3.3.1 Virtual Addresses. 3-6
3.3.1.1 Layout 3-6
3.3.1.2 Format 3-6
3.3.2 Physical Addresses 3-9
3.3.3 Address Translation. 3-10
3.3.3.1 Page Table Entry 3-10
3.3.4 Cache Operation. 3-15
3.3.4.1 Translation Buffer 3-15
3.3.4.2 Cache. 3-16
3.3.4.3 NMI Interface. 3-17

3.3.5 Read/Write Operations. 3-17
3.3.6 Device Address Selection 3-19
3.3.6.1 Transaction Significant Address Bits 3-19
3.4 INTERRUPTS AND EXCEPTIONS. 3-20
3.4.1 Servicing. 3-20
3.4.1.1 NMI Interrupt Enable Register. 3-20
3.4.1.2 Types of Interrupts. 3-20
3.4.1.3 Servicing the Interrupt. 3-20
3.4.2 Priority Levels. 3-20
3.4.3 System Control Block (SCB) Format. 3-21
3.4.3.1 SCB Pagination 3-23
3.4.3.2 Offsetable Devices 3-23
3.4.3.3 VAXBI Node Direct Connected Devices. 3-23
3.4.3.4 SCB Format 3-24
3.4.4 Machine Check Exception. 3-25
3.4.4.1 Types of Exceptions. 3-25

CHAPTER 4 DIAGNOSTIC AND MAINTENANCE AIDS

4.1 INTRODUCTION 4-1
4.2 GENERAL. 4-1
4.3 DIAGNOSTICS. 4-3
4.3.1 Console Selftest 4-3
4.3.2 Microdiagnostics 4-3
4.3.2.1 CSM Commands 4-4
4.3.2.2 Status and Error Information 4-6
4.3.2.3 Micromonitor Error Messages. 4-8
4.3.3 Macrodiagnostics 4-8
4.3.4 Customer Runnable Diagnostics. 4-9
4.3.4.1 Auto-Test Mode 4-10
4.3.4.2 Menu Mode. 4-10
4.3.5 Remote Diagnostics 4-10
4.4 POWER/ENVIRONMENTAL SYSTEM 4-10
4.4.1 Module Placement Verification. 4-10
4.4.1.1 Module Key Test. 4-10
4.4.1.2 Module Placement 4-12
4.4.2 Power Monitoring/Error Reporting 4-12
4.4.2.1 Default Mode Error Reporting 4-13
4.4.2.2 Operational Error Reporting. 4-13
4.4.3 Voltage Margining. 4-14
4.5 MAINTENANCE AIDS 4-16
4.5.1 Machine Check Logout Stack 4-16

FIGURES

No.	Title	Page
1-1	VAX 8800 System Major Component Locations (Rear View)	1-4
1-2	Cardcage Module Layout (Front View)	1-6
1-3	Simplified Block Diagram of the VAX 8800 System	1-8
1-4	Simplified Block Diagram of the Console Subsystem	1-10
1-5	Simplified Block Diagram of Single CPU	1-12
1-6	Simplified Block Diagram of the CPU IBox	1-13
1-7	Simplified Block Diagram of the CPU Execution Box	1-17
1-8	Simplified Block Diagram of the CBox	1-21
1-9	Simplified Block Diagram of the Clock Module	1-22
1-10	Simplified Block Diagram of the MBox	1-24
1-11	Simplified Block Diagram of the VAX 8800 Memory Interconnect	1-26
1-12	Simplified Diagram of VAX Bus Interconnect (Maximum Configuration)	1-28
1-13	I/O Interconnect and Adapters	1-31
1-14	NMI-to-VAXBI Adapter	1-32
1-15	Simplified Block Diagram of the Power System Complex	1-34
2-1	VAX 8800 System Hardware and Software Control Components	2-2
2-2	Console Operational Modes	2-8
2-3	Local/Remote Display Character Flow	2-14
3-1	Simplified Pipelining	3-4
3-2	Virtual Address Space Layout	3-7
3-3	Virtual Address Format	3-8
3-4	Physical Address Space Layout	3-9
3-5	MAP Enable Register Bit Configuration	3-10
3-6	Page Table Entry Bit Configuration	3-10
3-7	System Space Virtual-to-Physical Address Translation	3-12
3-8	Process Space (P0 Region) Virtual-to-Physical Address Translation	3-13
3-9	Process Space (P1 Region) Virtual-to-Physical Address Translation	3-14
3-10	CBox Functional Components	3-15
3-11	NMI Address Selection	3-18
3-12	NMI Address Bit Significance	3-19
3-13	NBI I/O Adapter SCB Vector Offset Format	3-24

4-1	Bottom-Up Testing.	4-2
4-2	Module Keying Test Simplified Block Diagram.	4-11
4-3	Module Key Test Connections.	4-12
4-4	Margin Enable and Margin Hi Lo Registers	4-15
4-5	Machine Check Logout Stack	4-17
4-6	CBox Error Register.	4-18
4-7	IBox Error Register.	4-18
4-8	EBox Error Register.	4-19
4-9	NMI Interrupt Control Register	4-19
4-10	NMI Fault Summary.	4-20
4-11	NMI Silo Data.	4-21
4-12	NMI Error Address Register	4-21
4-13	Cache ON Register.	4-22
4-14	Machine Check Status	4-22
4-15	Revision 1/2 Registers	4-22

TABLES

No.	Title	Page
1-1	Technical Description Manual Organization.	1-2
1-2	Related Documentation.	1-3
1-3	VAX 8800 System Physical Characteristics	1-5
1-4	Cabinet Module Identification.	1-7
1-5	Power Supply Identification.	1-7
1-6	VAX 8800 Processor Functional Units/Data Bus Descriptions	1-11
1-7	System Clocks.	1-23
1-8	MCL Command Operations	1-24
1-9	NMI Function Descriptions.	1-27
1-10	VAXBI Function Descriptions.	1-29
1-11	Optional VAX Bus Interconnect Adapters	1-30
1-12	NBIA Registers	1-33
1-13	NBIB Registers	1-33
1-14	NBox Modules	1-35
2-1	Hardware and Software Component Description.	2-3
2-2	Major Sections of CSM Code	2-7
2-3	Console Support Microcode Entry Points	2-8
2-4	Bit Examples	2-9
2-5	Console Command Overview	2-11
2-6	Console Command Language Prompts	2-15
2-7	Module Power Supply Turn-On Sequence	2-17
3-1	VAX 8800 System Microtraps	3-5
3-2	Page Table Entry Bit Description	3-11
3-4	Translation Buffer Field Description	3-15
3-4	Hardware Interrupt Priority Level Assignments.	3-21
3-5	System Control Block Page 0 (000--1FF)	3-22
3-6	Machine Check Exception Examples	2-25

4-1	TEMP Register Addresses for Use with Diagnostic CSM	4-5
4-2	Microcode Error Register Addresses	4-6
4-3	Macrodiagnostic Tests	4-9

EXAMPLES

No.	Title	Page
4-1	Sample Microdiagnostic Display Output	4-7
4-2	Sample Microdiagnostic Error Display	4-7
4-3	EMM Warning Message	4-14

SECTION 2 SYSTEM BUS SUMMARY

CHAPTER 1 MEMORY INTERCONNECT (NMI)

1.1	INTRODUCTION	1-1
1.2	BASIC FUNCTIONS	1-4
1.3	NMI SIGNALS AND TIMING	1-5
1.4	NMI ADDRESS SPACE	1-18
1.5	READ/WRITE TRANSACTIONS	1-25
1.6	INTERLOCKED OPERATIONS	1-32
1.7	BUS ARBITRATION	1-33
1.8	INTERRUPTS	1-39
1.8.1	NMI Interrupt Priority Levels	1-39
1.8.2	Device Interrupts	1-40
1.8.3	NMI Faults	1-41
1.9	NMI ERRORS	1-42

CHAPTER 2 VAX BUS INTERCONNECT (VAXBI)

2.1	INTRODUCTION	2-1
2.2	BASIC FUNCTIONS	2-4
2.3	VAXBI SIGNALS AND TIMING	2-5
2.4	VAXBI ADDRESS SPACE	2-11
2.4.1	Memory Address Space	2-11
2.4.2	I/O Address Space	2-11
2.4.3	Address Selection	2-12
2.5	BASIC VAXBI TRANSACTION FORMAT	2-18
2.5.1	Command/Address Cycle	2-18
2.5.2	Embedded Arbitration Cycle	2-18
2.5.3	Data Cycles	2-19
2.5.4	Bus Parity	2-19
2.6	READ/WRITE TRANSACTIONS	2-20
2.6.1	Write Data Cycles	2-20
2.6.2	Read Data Cycles	2-20
2.6.3	Nonexistent Addresses	2-23

2.6.4	Stalls	2-23
2.6.5	Retries	2-24
2.7	BROADCAST TRANSACTIONS	2-24
2.8	INVALIDATE TRANSACTIONS	2-26
2.9	INTERRUPT OPERATION (INTR, IDENT, AND IPINTR TRANSACTIONS	2-28
2.9.1	Interrupt (INTR) Transactions	2-29
2.9.2	Identify (IDENT) Transactions	2-31
2.9.3	Interprocessor Interrupt (IPINTR) Transaction	2-34
2.10	STOP TRANSACTIONS	2-36
2.11	BUS ARBITRATION AND CONTROL	2-38
2.11.1	Bus Requests	2-38
2.11.2	Arbitration Modes	2-39
2.11.3	Arbitration Control	2-39
2.11.4	Extending a Transaction	2-41
2.11.5	Special Mode Functions	2-41
2.12	VAXBI ERRORS	2-42
2.12.1	Parity Checking	2-42
2.12.2	Transmit Check Error Detection	2-43
2.12.3	Protocol Checking	2-44

CHAPTER 3 VISIBILITY BUS (VBUS)

3.1	INTRODUCTION	3-1
3.2	BASIC FUNCTIONS	3-1
3.3	VBUS SIGNALS	3-1
3.4	VBUS REGISTERS	3-4
3.5	MODULE VBUS CHANNEL CIRCUITRY	3-7
3.5.1	Minimum Configuration	3-7
3.5.2	Expanded Configuration	3-7
3.6	VBUS ADDRESS/DATA SUMMARY	3-10
3.7	VBUS CONSOLE COMMANDS	3-10

FIGURES

No.	Title	Page
1-1	Memory Interconnect (NMI)	1-2
1-2	Basic NMI Timing	1-5
1-3	NMI Signals	1-6
1-4	NMI Address Space	1-20
1-5	NMI Address Bits	1-23
1-6	NMI Address Selection	1-24
1-7	NMI Write Transaction	1-26
1-8	NMI Read Transaction	1-27
1-9	NMI Write Transaction Types	1-29

1-10	NMI Read Transaction Types	1-30
1-11	Basic NMI Arbitration Line Timing	1-33
1-12	NMI Arbitrator Operation	1-36
1-13	Detailed NMI Arbitration Line Timing (Typical)	1-37
1-14	MEMORY BUSY Timing	1-38
1-15	Fault Signal Timing	1-42
2-1	VAX Bus Interconnect (VAXBI)	2-2
2-2	VAXBI Signals	2-6
2-3	Basic VAXBI Timing	2-10
2-4	VAXBI Address Space	2-13
2-5	VAXBI Node Register Space	2-14
2-6	VAXBI Required Registers	2-15
2-7	BIIC-Specific Device Registers	2-16
2-8	VAXBI Read/Write Address Bits	2-17
2-9	Basic VAXBI Transaction Format	2-18
2-10	VAXBI Write Transaction (Octaword Length)	2-21
2-11	VAXBI Read Transaction (Octaword Length)	2-22
2-12	VAXBI Broadcast (BDCST) Transaction (Octaword Length)	2-25
2-13	VAXBI Invalidate (INVAL) Transaction	2-27
2-14	VAXBI Interrupt (INTR) Transaction	2-30
2-15	VAXBI Identify (IDENT) Transaction	2-33
2-16	VAXBI Interprocessor Interrupt (IPINTR) Transaction	2-35
2-17	VAXBI STOP Transaction	2-37
2-18	Bus Arbitration Request Lines	2-38
2-19	Arbitration State Diagram	2-40
2-20	VAXBI Arbitration (Example)	2-42
3-1	Visibility Bus (VBus) and VBus Control (on CLK Module)	3-2
3-2	VBus Control Register	3-5
3-3	VBus Access Register	3-6
3-4	VBus Channel in CPU Module (Minimum Configuration)	3-8
3-5	VBus Channel in CPU Module (Expanded Configuration)	3-9

TABLES

No.	Title	Page
1-1	Glossary of NMI Terms	1-3
1-2	NMI Signal Descriptions	1-10
1-3	I/O Registers in NBI and Memory Controller	1-21
1-4	NMI Interrupt Priority Levels (IPLs)	1-39
1-5	NMI Errors	1-43

2-1 Glossary of VAXBI Terms. 2-3
2-2 VAXBI Signal Descriptions. 2-7

3-1 VBUS Signal Descriptions 3-3
3-2 VBus Control Register Bit Descriptions 3-5
3-3 VBus Directory (Excerpt) 3-11

SECTION 3 CONSOLE SUBSYSTEM

CHAPTER 1 INTRODUCTION

1.1 GENERAL. 1-1
1.2 RELATED DOCUMENTATION AND REFERENCES 1-1
1.3 FUNCTION AND PURPOSE 1-2
1.4 SUBSYSTEM COMPONENTS 1-2
1.5 CONSOLE/VAX 8800 INTERACTION 1-4
1.5.1 Power-Up Mode. 1-4
1.5.2 Console I/O. 1-4
1.5.3 VAX 8800 State Description 1-6
1.5.3.1 Power Off. 1-6
1.5.3.2 Clock Stopped/WCS Invalid. 1-6
1.5.3.3 Clock Running/WCS Invalid. 1-6
1.5.3.4 Clock Running/WCS Valid. 1-6
1.5.4 Console State Description. 1-7
1.6 CONSOLE SOFTWARE COMPONENTS. 1-9
1.6.1 Control Program. 1-9
1.6.2 Logical Block Server Program 1-9
1.6.3 Real-Time Interface Driver 1-9
1.7 CONSOLE/VAX 8800 POWER SEQUENCE. 1-10
1.7.1 Powerup (Refer to Figure 1-3). 1-10
1.7.2 EMM/Console Initialize (Refer to Figure 1-4). 1-12
1.7.3 Restart/Boot/Halt (Refer to Figure 1-5). . . . 1-17
1.7.4 Power Fail (Refer to Figure 1-6) 1-19
1.7.5 Powerdown (Refer to Figure 1-7). 1-21

CHAPTER 2 FUNCTION DESCRIPTION

2.1 GENERAL. 2-1
2.2 REAL-TIME INTERFACE (RTI). 2-1
2.2.1 Programmable Peripheral Interface (PPI). 2-3
2.2.1.1 Port A 2-3
2.2.1.2 Port B 2-3
2.2.1.3 Port C 2-6
2.2.1.4 PPI Control. 2-8
2.2.2 Serial Line Port 2-10
2.2.2.1 ECPI Registers 2-10
2.2.2.2 Data Transfer and Status Registers 2-14

2.3 CONSOLE INTERFACE. 2-16
2.3.1 Buffer Translate and Synchronize 2-16
2.3.2 Console Address Decode 2-16
2.3.3 Console Sequencer (CSEQ MCA) 2-17
2.3.4 Terminal Register/Interval Clock (TRIC) MCA. . 2-17
2.3.4.1 Program Mode 2-19
2.3.4.2 Console Mode 2-20
2.3.5 Data Output Mux. 2-20
2.3.6 Control Registers. 2-21
2.3.7 Visibility Bus Control 2-22
2.3.8 Console Interrupt Generation 2-23
2.3.9 Power Status 2-23
2.4 CONSOLE/VAX 8800 INTERACTION 2-24
2.4.1 Initialization 2-24
2.4.1.1 Turn ON and Monitor System Power/Reset EMM. 2-24
2.4.1.2 Console Poweron. 2-27
2.4.1.3 Load and Run Console Power-Up Software . . . 2-28
2.4.1.4 Sequenced Power Application. 2-32
2.4.1.5 Initialize Hardware. 2-36
2.4.1.6 Test and Checkout. 2-44
2.4.1.7 Load RAMs and DRAMs. 2-46
2.4.2 VAX 8800 CPU Control 2-63
2.4.2.1 Console Sequencer. 2-64
2.4.2.2 Control Registers. 2-68
2.4.3 Data Transfers 2-70
2.5 CONSOLE/VAX 8800 CLOCKS AND TIMING 2-73
2.5.1 One-MHz Clock. 2-73
2.5.2 Interval Clock 2-75
2.5.3 CPU Timeouts 2-78
2.5.4 Visibility Bus 2-79

CHAPTER 3 DETAILED DESCRIPTION

3.1 GENERAL. 3-1
3.2 TERMINAL REGISTER INTERVAL CLOCK (TRIC). 3-1
3.3 CONSOLE SEQUENCER MCA (CSEQ) 3-9
3.3.1 Console Strobe Sequencer 3-9
3.3.2 Read Acknowledge 3-9
3.3.3 Console Write Sequencer. 3-9
3.3.4 Control Store Load Sequencer 3-9
3.4 CONSOLE/VAX 8800 REGISTER SUMMARY. 3-17
3.4.1 Console Registers (Refer to Figure 3-7). 3-17
3.4.2 VAX 8800 CPU Registers (Refer to Figure 3-8) . . 3-24
3.5 CONSOLE CABLING. 3-28

FIGURES

No.	Title	Page
1-1	Simplified Block Diagram of the Console Subsystem.	1-3
1-2	Modes of Operation	1-5
1-3	Power-Up Sequence.	1-11
1-4	EMM/Console Initialize	1-13
1-5	Restart/Boot/Halt.	1-18
1-6	Power-Fail Sequence.	1-20
1-7	Powerdown Procedure.	1-22
2-1	Console Subsystem Functional Block Diagram	2-2
2-2	PPI Port A Format.	2-3
2-3	PPI Port B Format.	2-3
2-4	PPI Port C Format.	2-6
2-5	PPI Control Register Format.	2-9
2-6	ECPI Mode 1 Register	2-10
2-7	ECPI Mode 2 Register	2-12
2-8	ECPI Command Register.	2-13
2-9	Serial Line Port Data and Status Registers	2-14
2-10	RXDB, TXDB, and DBCS	2-18
2-11	Control Registers.	2-21
2-12	VBus Control and Access Registers.	2-22
2-13	System Power-On Sequence	2-25
2-14	Environmental Monitoring Module Reset Sequence	2-26
2-15	Console Power-On Events.	2-27
2-16	Serial Line Port Data Transfer Registers	2-28
2-17	Load/Run Console Power-Up Software Events.	2-30
2-18	Sequenced Power Application Events	2-32
2-19	Console Interconnect Loopback Testing Through Ports A, B, and C.	2-37
2-20	Console Interconnect Loopback Testing Through Ports B and C.	2-37
2-21	Interface Data Path Loopback Test of Unbuffered Data	2-38
2-22	Interface Data Path Loopback Test of Buffered Data Through RXDB and TXDB	2-39
2-23	Console Sequencer Enable Logic	2-40
2-24	Control Register Initialization.	2-41
2-25	Hardware Initialization Events	2-42
2-26	Test and Checkout Events	2-44
2-27	VBus Parity Bits	2-49
2-28	DRAM Address	2-51
2-29	MNI Control Store Address.	2-53
2-30	RAM Loading Simplified Block Diagram	2-56
2-31	RAM/DRAM Loading Events.	2-57
2-32	Console/Interface Timing Signals	2-64
2-33	Write Sequence	2-65
2-34	Read Sequence (Setup).	2-66
2-35	Read Sequence (Data Out)	2-67

2-36	Simplified Diagram of Control Register 0	2-68
2-37	Simplified Diagram of Control Register 1	2-69
2-38	Simplified Diagram of Control Register 2	2-69
2-39	Data Transfer Control Interface-to-VAX 8800 CPU.	2-71
2-40	Data Transfer VAX 8800 CPU-to-Console Interface.	2-72
2-41	1 MHz Clock Generation	2-74
2-42	Interval Clock Registers Bit Configuration	2-75
2-43	Interval Clock Simplified Block Diagram.	2-77
2-44	Simplified CPU Timeout	2-78
2-45	Clock Status and Timeout Register (CST).	2-78
2-46	VBus Control/Data Signals.	2-79
2-47	VBus Logic Simplified Block Diagram.	2-80
3-1	TRIC MCA Block Diagram	3-2
3-2	TRIC MCA Pin Layout.	3-3
3-3	TRIC MCA Body Drawing.	3-4
3-4	CSEQ MCA Block Diagram	3-10
3-5	CSEQ MCA Pin Layout.	3-11
3-6	CSEQ MCA Body Drawing.	3-12
3-7	Console Registers.	3-17
3-8	VAX 8800 CPU Registers	3-24
3-9	Console Subsystem Cabling Diagram.	3-29

TABLES

No.	Title	Page
2-1	PPI Port B Bit Description	2-4
2-2	PPI Port C bit Description	2-6
2-3	PPI Control Register Bit Description	2-9
2-4	ECPI Mode 1 Register Bit Description	2-11
2-5	ECPI Mode 2 Register Bit Description	2-12
2-6	ECPI Command Register Bit Description.	2-13
2-7	Serial Line Port Data and Status Registers Bit Description.	2-15
2-8	Serial Line Port Data Transfer Registers Bit Description.	2-29
2-9	Key Initialization Signal/Functions.	2-62
2-10	ICCS Bit Configuration	2-76
3-1	TRIC MCA Pin Assignments	3-5
3-2	TRIC MCA Signal Descriptions	3-6
3-3	CSEQ MCA Pin Assignments	3-13
3-4	CSEQ MCA Signal Descriptions	3-14
3-5	Console Cable List	3-28

SECTION 4 POWER SYSTEM COMPLEX

CHAPTER 1 GENERAL DESCRIPTION

1.1 INTRODUCTION 1-1
1.2 SYSTEM COMPONENTS. 1-4
1.2.1 876A Power Controller. 1-5
1.2.2 NBox Power Converter 1-5
1.2.3 Modular Power System (MPS) 1-6
1.2.4 Environmental Monitoring Module. 1-6
1.2.5 Cooling System 1-7
1.2.6 Battery Backup Unit H7231-M. 1-7
1.3 MECHANICAL CONFIGURATION 1-8
1.3.1 876A Power Controller. 1-11
1.3.2 NBox Port Conditioner. 1-11
1.3.3 MPS Modules (Regulators) and Cage. 1-11
1.3.4 Battery Backup Unit. 1-12
1.3.5 Air Flow System. 1-12
1.4 POWER DISTRIBUTION 1-13
1.4.1 AC Power 1-13
1.4.2 DC Power 1-15
1.4.3 Controls and Breakers. 1-17
1.4.3.1 Controls 1-17
1.4.3.2 Circuit Breakers 1-17
1.5 AC POWER SPECIFICATIONS. 1-19
1.5.1 Electrical Requirements. 1-19
1.5.1.1 AC Power Sources 1-19
1.6 FAULT AND STATUS INDICATORS. 1-21
1.6.1 876A Power Controller. 1-21
1.6.2 NBox 1-21
1.6.2.1 H7170 Built-In Test Equipment. 1-23
1.6.2.2 ILM Built-In Test Equipment. 1-23
1.6.3 Modular Power Supply Regulators. 1-24
1.6.4 Environmental Monitoring Module. 1-25
1.6.5 System Console Device. 1-26

CHAPTER 2 FUNCTIONAL DESCRIPTION

2.1 INTRODUCTION 2-1
2.2 POWER SYSTEM BLOCK DIAGRAM 2-1
2.3 SIMPLIFIED OPERATION 2-3
2.3.1 876A Power Controller. 2-3
2.3.2 NBox 2-4
2.3.2.1 H7170 Power Converter. 2-4
2.3.2.2 Control Start-up Power Module (CSP). 2-5
2.3.2.3 Interface Logic Module (ILM) 2-5
2.3.2.4 New Box Translator Module (NBT). 2-5
2.3.3 Modular Power System (MPS) 2-6
2.3.4 Battery Backup Unit (BBU Model H7231-M). 2-8
2.3.4.1 BBU Control. 2-9
2.3.5 Environmental Monitoring Module (EMM). 2-11

2.3.6 Power System Monitoring. 2-13
2.3.6.1 Key Monitoring 2-14
2.3.6.2 Regulator Control. 2-14
2.3.6.3 BBU Control. 2-14
2.3.6.4 Air Flow Status. 2-14
2.3.6.5 AC/DC LO Signals 2-15
2.3.6.6 Regulator Overtemp. and CPU Cabinet Temperature (Thermistor Volts) 2-15
2.4 POWER SEQUENCES. 2-16
2.4.1 Circuit Breakers 2-16
2.4.2 Summary of Power-Up Sequence 2-18
2.5 SYSTEM POWER-UP FLOWCHART (FIGURE 2-5) 2-24
2.6 CONSOLE POWER-DOWN FLOWCHART (FIGURE 2-6). . . . 2-29
2.7 POWER-DOWN/POWER INTERRUPT WITH BBU FLOWCHART (FIGURE 2-7) 2-31
2.8 POWERUP FROM BBU FLOWCHART (FIGURE 2-8). 2-35

CHAPTER 3 DETAILED DESCRIPTION

3.1 INTRODUCTION 3-1
3.2 BLOCK DIAGRAM OF THE VAX 8800 POWER SYSTEM 3-1
3.3 876A POWER CONTROLLER. 3-3
3.4 NBOX POWER CONVERTER ASSEMBLY. 3-6
3.4.1 NBox Modules 3-8
3.4.1.1 H7170. 3-8
3.4.2 CSP. 3-9
3.4.2.1 ILM. 3-9
3.4.2.2 NBT Module 3-14
3.4.3 Modular Power System -- MPS. 3-16
3.4.3.1 H7186 +5.0-Volt Regulator. 3-18
3.4.3.2 Side Panel 3-20
3.4.3.3 H7186 Main Board 3-21
3.4.4 H7187 -2.0-Volt Regulator. 3-22
3.4.5 H7180 -5.2-Volt Regulator. 3-22
3.4.5.1 H7180 Side Panel 3-24
3.4.5.2 H7180 Main PC Board. 3-24
3.4.6 H7189 BIP Regulator. 3-26
3.4.6.1 H7189 Functional Description 3-29
3.4.7 MPS Regulator BITE Indicators. 3-33
3.4.8 Buses and Backplanes 3-35
3.4.9 MPS Backplane. 3-35
3.4.9.1 300-Vdc Buses. 3-40
3.4.10 Environmental Monitoring Module. 3-41
3.4.10.1 8085A Microprocessor System. 3-43
3.4.10.2 Electric Key Monitor 3-44
3.4.10.3 Regulator Control Circuits 3-45
3.4.10.4 Regulator On/Off Control Circuits. 3-46
3.4.10.5 Regulator Margin Control Circuits. 3-46
3.4.11 Status Registers 3-48
3.4.11.1 AC/DC LO Circuits. 3-49
3.4.11.2 Total-Off Control and Indicator Circuits . . 3-52

3.4.11.3 Temperature Sensing Circuits 3-53
3.4.11.4 Voltage Measuring Circuit. 3-55
3.4.11.5 EMM/Console Voltage Tests. 3-58
3.4.11.6 Battery Backup Unit (BBU) Control. 3-58
3.4.12 Battery Backup Unit (H7231-M). 3-59
3.4.13 Air Flow Sensing Circuit 3-63
3.5 COOLING SUBSYSTEM. 3-65

FIGURES

No.	Title	Page
1-1	VAX 8800 System Physical Layout (Front View)	1-2
1-2	VAX 8800 Power System Block Diagram (60 Hz)	1-3
1-3	VAX 8800 Power System Block Diagram (50 Hz)	1-4
1-4	VAX 8800 CPU Cabinet - Front View	1-9
1-5	VAX 8800 CPU Cabinet - Rear View	1-10
1-6	876A Rear View Showing Receptacles	1-14
1-7	DC Power Section Block Diagram	1-16
1-8	VAX 8800 Power System Circuit Location Diagram	1-18
1-9	876A Power Controller Front Panel	1-21
1-10	NBox Front-Panel Indicators	1-22
1-11	Indicators for the Modular Power Regulators	1-24
1-12	EMM Front-Panel Indicators	1-25
2-1	VAX 8800 Power System Block Diagram	2-2
2-2	MPS Backplane Configuration (Rear View)	2-7
2-3	Battery Backup Subsystem Functional Block Diagram	2-10
2-4	EMM Functional Block Diagram	2-12
2-5	System Power-Up Flowchart	2-20
2-6	System Power-Down Flowchart	2-28
2-7	Powerdown/Power Interrupt with BBU Flowchart	2-30
2-8	AC Powerup from BBU Operation Flowchart	2-34
3-1	Power System Block Diagram	3-2
3-2	876A Front and Rear Panels	3-4
3-3	876A Power Controller Block Diagram	3-5
3-4	NBox System Interconnect	3-7
3-5	ILM PC Board Signals	3-10
3-6	NBT PC Board Signals	3-15
3-7	MPS Regulator Configuration	3-17
3-8	H7186 Block Diagram	3-19
3-9	H7180 Block Diagram	3-23
3-10	H7189 Block Diagram	3-27
3-11	BITE Indicators	3-34
3-12	Organization of the Power System	3-37
3-13	MPS I Backplane	3-38
3-14	MPS II Backplane	3-39
3-15	EMM Block Diagram	3-42
3-16	Voltage Margining Circuit	3-47

3-17	AC/DC LO Timing Diagram.	3-50
3-18	AC/DC LO Circuit	3-51
3-19	Temperature Sensing Circuit.	3-54
3-20	Voltage Measuring Circuit.	3-56
3-21	Voltage Measuring Technique.	3-57
3-22	BBU Block Diagram.	3-62
3-23	Air Flow Sensing Circuit	3-64
3-24	Air Flow Path.	3-66

TABLES

No.	Title	Page
1-1	Power System Components.	1-4
1-2	NBox Modules	1-5
1-3	876A Power Distribution.	1-15
1-4	VAX 8800 Circuit Breakers.	1-17
1-5	H7170 Status Indicators.	1-23
1-6	MPS Regulator Indicators	1-24
1-7	EMM Magnetic Status Indicator Codes.	1-26
2-1	876A Power Distribution.	2-4
2-2	Modules Using CSP Bias Voltages.	2-5
2-3	Voltage Regulators	2-7
2-4	System Circuit Breakers.	2-17
3-1	876A AC Power Distribution	3-3
3-2	NBox Modules	3-8
3-3	CSP Voltages	3-9
3-4	VAX 8800 MPS Regulators.	3-18
3-5	H7186 Side-Panel Components and Interconnects. .	3-20
3-6	H7186 Main Board Circuits and Interconnects. . .	3-21
3-7	H7180 Side Panel Components and Interconnects. .	3-24
3-8	H7180 Main PCB Circuits and Interconnects. . . .	3-24
3-9	H7189 Module Functions	3-28
3-10	H7189 Outputs.	3-31
3-11	H7189 Module I Circuits and Interconnects. . . .	3-32
3-12	H7189 Module II Circuits and Interconnects . . .	3-32
3-13	MPS Regulator Connectors	3-36
3-14	300-V Buses, Power Sources, and Loads.	3-40
3-15	Battery Backup Interface Signals	3-61

SECTION 5 CLOCK MODULE

CHAPTER 1 INTRODUCTION

1.1	BASIC OPERATION.	1-1
1.2	BASIC COMPONENTS AND TIMING.	1-3
1.3	CLOCK CONTROL (BY CONSOLE)	1-5
1.4	CLOCK STALLS	1-7
1.5	CLOCK STATUS	1-7

CHAPTER 2 FUNCTIONAL DESCRIPTION

2.1 DETAILED BLOCK DIAGRAM 2-1
2.1.1 Oscillator 2-1
2.1.2 Phase Generator. 2-2
2.1.3 Clock Control Logic. 2-2
2.1.4 Clock Distribution Circuits. 2-2
2.2 CLOCK GENERATOR INITIALIZATION 2-7
2.3 SYSTEM CLOCK PERIOD CONTROL. 2-8
2.3.1 Phase-Locked Loop Operation. 2-8
2.3.2 Changing Clock Period. 2-11
2.4 SYSTEM CLOCK START/STOP/BURST CONTROL. 2-12
2.4.1 Starting the Clocks. 2-13
2.4.2 Stopping the Clocks Unconditionally. 2-13
2.4.3 Stopping the Clocks on Micromatch/Scope Sync Generation 2-16
2.4.4 Bursting the Clocks. 2-17
2.4.5 Single-Stepping the Clocks 2-17
2.4.6 Single-Stepping the B CLK. 2-18
2.5 SLOW CLOCK GENERATION AND CONTROL. 2-19
2.6 CLOCK CONSOLE COMMANDS 2-20

FIGURES

No.	Title	Page
1-1	Clock Generator (and Console Interface) on Clock Module.	1-2
1-2	System Clock Timing Diagram.	1-4
1-3	Clock Control Register	1-8
1-4	Burst Count Register	1-9
1-5	Clock Period Register.	1-9
1-6	Clock/Timeout Status Register.	1-10
2-1	Clock Generator (Detailed Block Diagram)	2-3
2-2	Simplified Clock Frequency Control Circuitry	2-9
2-3	Simplified Clock Start/Stop/Burst Control Logic.	2-14
2-4	Start/Stop/Burst Control Timing Diagram.	2-15
2-5	Micromatch and Scope Sync Timing Diagram	2-16
2-6	Single-Stepping B CLK Timing Diagram	2-18
2-7	Slow Clock Timing Diagram.	2-19

TABLES

No.	Title	Page
1-1	System Clocks.	1-1
1-2	Clock Control Register Bit Descriptions.	1-8
1-3	Clock/Timeout Status Register Bit Descriptions	1-10
2-1	Clock Generator Inputs	2-4
2-2	Clock Generator Outputs.	2-5
2-3	Clock Console Commands	2-20

SECTION 6 INSTRUCTION BOX (IBOX)

CHAPTER 1 INTRODUCTION

1.1 OVERVIEW 1-1
1.1.1 Dual-Processor Configuration 1-1
1.2 LOGIC ELEMENTS 1-2
1.2.1 Physical Implementation. 1-2
1.2.2 Instruction Buffer (IB). 1-5
1.2.2.1 Writing the IB 1-5
1.2.2.2 Reading the IB 1-5
1.2.3 IB Manager 1-6
1.2.3.1 IB Read/Write Control. 1-6
1.2.3.2 Computing Amount of IB Data Consumed 1-6
1.2.4 Decoder Logic. 1-7
1.2.4.1 Decoder RAMs 1-7
1.2.4.2 Special Address Encoder. 1-8
1.2.5 Microsequencer Logic 1-8
1.2.6 Control Store. 1-9
1.2.7 Condition Code and Macrobranch Logic 1-9
1.2.7.1 PSL CC Bits. 1-9
1.2.7.2 CPU State Flags. 1-10
1.2.8 Interrupt and Processor Register Logic 1-10
1.2.8.1 Interrupt Logic. 1-10
1.2.8.2 Processor Register Logic 1-10
1.2.9 File Address Generator 1-10
1.2.10 Gateway Control Logic. 1-11
1.2.10.1 Primary Functions. 1-11
1.3 IBOX BUSES 1-12
1.3.1 Cache Data Bus 1-12
1.3.2 IB Data Bus. 1-12
1.3.3 Cons Bidi Data Bus 1-12
1.4 IBOX RESIDENT INTERNAL PRIVILEGED REGISTERS (IPRs) 1-13
1.4.1 VAX Architecture IPRs. 1-13
1.4.2 VAX 8800-Specific IPRs 1-14
1.4.2.1 NMI Interrupt Control Register (NICTRL). . . 1-14
1.4.2.2 Interrupt Other Processor Register (INOP). . 1-15
1.5 IBOX MICROCODE VISIBLE ONLY REGISTERS. 1-16
1.5.1 Clear Interrupt Other Processor (CIOP) 1-16
1.5.2 IBox Error Register (IBER) 1-16
1.5.2.1 IBER Usage 1-16
1.5.2.2 IBER Bits <7:0>. 1-17
1.5.2.3 IBER Bits <11:08>. 1-17
1.5.3 Clear Error Register (CER) 1-19

CHAPTER 2 MICROCODE OVERVIEW AND PIPELINE CONCEPTS

2.1 CHAPTER SCOPE. 2-1
2.2 VAX 8800 MAIN CONTROL STORE OVERVIEW 2-1
2.2.1 Microcode Size and Allocation. 2-1
2.2.2 Microcode File Structure 2-1
2.2.3 Microcode Assembly 2-2
2.2.3.1 Other Loadable Binary Files. 2-3
2.2.4 Microword Format 2-3
2.2.4.1 Field Naming Convention. 2-3
2.2.4.2 Field Functionality. 2-3
2.2.5 Microcode Definition Files 2-11
2.2.5.1 Field Definition File - DEFIN.MIC. 2-11
2.2.5.2 Macrodefinition File - MACRO.MIC 2-13
2.2.6 Microcode Related Documentation. 2-15
2.3 MICROCODE PIPELINING CONCEPTS. 2-20
2.3.1 Pipelining Rationale 2-20
2.3.2 Pipelined Versus Nonpipelined Machines 2-20
2.3.2.1 Performance Factors. 2-22
2.4 VAX 8800 PIPELINE CHARACTERISTICS. 2-23
2.4.1 CPU Clock Cycle. 2-23
2.4.2 CPU Hardware Design. 2-23
2.4.3 Relationship Between Microcycles and CPU Functions. 2-25
2.4.4 IB Decode Cycle. 2-25
2.4.5 Canonical Time States. 2-27
2.4.5.1 Definition of a Canonical Time State 2-27
2.4.5.2 Overlapping Time States. 2-27
2.4.6 Time State Events. 2-29

CHAPTER 3 IBOX FUNCTIONAL DESCRIPTION

3.1 CHAPTER SCOPE. 3-1
3.2 CONTROL STORE LOGIC. 3-2
3.2.1 Control Store RAM Segments 3-2
3.2.2 Control Store RAM Addressing 3-5
3.2.3 Control Store RAM Data Latches 3-5
3.2.4 Loading the Control Store RAMs 3-6
3.2.4.1 Load Control Store Microaddress. 3-6
3.2.4.2 Write Data to Selected Address 3-7
3.3 MICROSEQUENCING. 3-9
3.3.1 Microsequencer Hardware. 3-9
3.3.1.1 Microbranch Slice (UBRS) MCAs. 3-9
3.3.1.2 Microtrap (UTRP) MCA 3-9
3.3.1.3 Microstack 3-9
3.3.2 Normal Microcode Flow. 3-11
3.3.3 IB Decoder Supplied Microaddress 3-11
3.3.4 Microbranching 3-11
3.3.4.1 Microbranch Conditions 3-13
3.3.4.2 Microbranch Latency 3-19
3.3.5 Microsubroutine Calls and Returns. 3-21
3.3.5.1 Normal Microstack Operation. 3-21
3.3.5.2 Microsubroutine Calls. 3-21

3.3.5.3 Microsubroutine Returns. 3-23
3.3.6 Microtraps 3-23
3.3.6.1 Microtrap Servicing. 3-25
3.3.6.2 Disabling Microtraps 3-30
3.3.7 Console Supplied Microaddresses. 3-30
3.4 MACROINSTRUCTION DECODING. 3-32
3.4.1 Initializing the IB (IB Flush) 3-32
3.4.1.1 Full IB Flush. 3-32
3.4.1.2 IB Flush Logic 3-34
3.4.1.3 Partial IB Flush 3-38
3.4.2 I-Stream Prefetching 3-39
3.4.2.1 General Description. 3-39
3.4.2.2 Refilling Cache. 3-39
3.4.3 Loading the IB 3-40
3.4.3.1 IB Write Control 3-40
3.4.3.2 Cache Monitor Logic. 3-42
3.4.3.3 IB Full Logic. 3-43
3.4.3.4 IB Load Example. 3-43
3.4.4 Reading The IB 3-45
3.4.4.1 Pipeline Timing. 3-45
3.4.4.2 IB Read Ports. 3-46
3.4.4.3 IB Data Aligner. 3-51
3.4.4.4 IB Data Formatter and Data Scrambler 3-60
3.4.4.5 IB Read Example. 3-66
3.4.5 IB Manager Operations. 3-73
3.4.5.1 IB Read Address Logic. 3-73
3.4.5.2 Opcode Watcher Logic 3-76
3.4.5.3 Specifier Size Logic 3-78
3.4.5.4 Checking TEMPINC <2:0> Validity. 3-81
3.4.5.5 Decoder Stall. 3-82
3.4.5.6 Modifying the IB Pointer 3-82
3.4.5.7 IB Read Address Logic Control Signals. . . . 3-85
3.4.5.8 Computing Number of IB Longwords Consumed. . 3-91
3.4.6 Instruction Decoder Operation. 3-93
3.4.6.1 Pipeline Timing Considerations 3-93
3.4.6.2 Operand Specifier Entry Point Addresses. . . 3-96
3.4.6.3 Opcode Entry Point Microaddresses. 3-104
3.4.6.4 Special Microaddresses 3-111
3.4.6.5 IBST MCA Signals Related to Instruction Decoding 3-118
3.4.6.6 Decoder RAM (DRAM) 3-121
3.4.7 Optimized Instructions 3-128
3.4.7.1 Simple Move Instructions 3-128
3.4.7.2 Simple Branch Instructions 3-129
3.5 MACROBRANCH INSTRUCTIONS 3-129
3.5.1 Branch Instruction Basics. 3-129
3.5.2 Branch Instruction Classes 3-130
3.5.3 Unconditional Branches 3-130
3.5.4 Short Conditional Branches 3-135
3.5.5 Long Conditional Branches. 3-141
3.5.6 Condition Code and Macro Branch Logic. 3-147

3.6	SPECIAL REGISTER ADDRESSING.	3-150
3.6.1	RNUM1 and RNUM2 Registers.	3-150
3.6.2	RLOG Register.	3-150
3.6.3	MDNUM Register	3-152
3.7	INTERRUPTS	3-153
3.7.1	Interrupt Requests	3-153
3.7.2	Interrupt Servicing.	3-153
3.8	CONSOLE GATEWAY CONTROL.	3-158
3.8.1	Loading Control Store and Decoder RAMs	3-158
3.8.2	Starting the Micromachine.	3-160
3.8.3	Data Transfer with Console Resident IPRs . . .	3-160
3.8.4	Breakpoint Microtrap	3-160
3.8.5	Console Data Parity Check.	3-161

FIGURES

No.	Title	Page
1-1	IBox Block Diagram	1-3
1-2	NMI Interrupt Control (NICTRL) Register Bit Map.	1-14
1-3	Interrupt Other Processor (INOP) Register Bit Map.	1-15
1-4	IBox Error Register (IBER) Bit Map	1-18
2-1	Microword Bit Format	2-4
2-2	Sample Microword Field Definiton - I_APORT Field.	2-11
2-3	Basic Time State Diagram - Nonpipelined CPU. . .	2-21
2-4	Basic Time State Diagram - Pipelined CPU	2-21
2-5	Basic CPU Timing	2-24
2-6	Microcycles/CPU Functions.	2-26
2-7	IB Decode Cycle.	2-28
2-8	Canonical Time States.	2-28
2-9	VAX 8800 Pipeline Time State Diagram	2-30
3-1	Control Store Logic Simplified Block Diagram . . .	3-3
3-2	Microaddress Bit Slices for Micromatch Register Loading.	3-7
3-3	Control Store RAM Load Path.	3-8
3-4	Microsequencer Logic	3-10
3-5	Normal INEXT Field Addressing.	3-12
3-6	Microbranch Condition Selection.	3-14
3-7	Microbranch Latency.	3-20
3-8	Microstack Operation	3-22
3-9	Microtrap Servicing.	3-26
3-10	Microtrap Latency	3-27
3-11	Console Supplied Microaddress.	3-31
3-12	Instruction Buffer Logic	3-33
3-13	IB Flush Logic	3-35
3-14	IB Load Logic.	3-41

3-15	I-Stream Data Entering the IB.	3-44
3-16	IB Memory Unit Contents for MOVL Example	3-47
3-17	IB Read Port Example - Part 1.	3-48
3-18	IB Read Port Example - Part 2.	3-49
3-19	IB Data Aligner Muxes.	3-51
3-20	IB Data Aligner Output Example - Part 1.	3-54
3-21	IB Data Aligner Output Example - Part 2.	3-55
3-22	Opcode Mux Sources	3-57
3-23	IB Data Formatter and Data Scrambler Logic . . .	3-61
3-24	IB Read Example.	3-69
3-25	PCNC MCA Block Diagram	3-74
3-26	Simplified IB Read Address Logic	3-83
3-27	Instruction Decode Logic	3-94
3-28	Operand Specifier Entry Address Format	3-96
3-29	Opcode Entry Address Format.	3-104
3-30	Special Microaddress Format.	3-111
3-31	Special Address Encoder Logic.	3-112
3-32	Decoder RAM Read Address Format.	3-121
3-33	Decoder RAM Output Signals	3-123
3-34	Pipeline State for a BRB Instruction	3-131
3-35	Pipeline State for a Successful BEQL Instruction.	3-140
3-36	Pipeline State for a Successful AOBLEQ Instruction.	3-145
3-37	Condition Code and Macro Branch Logic.	3-148
3-38	File Address Slice MCAs.	3-151
3-39	Interrupt Logic Simplified Block Diagram	3-155
3-40	Gateway Control Logic.	3-159

TABLES

No.	Title	Page
1-1	Microcode Features	1-9
1-2	IBox Resident IPRs	1-13
1-3	NICTRL Register Bit Descriptions	1-14
1-4	INOP Register Bit Description.	1-15
1-5	IBER Bit Descriptions.	1-10
2-1	VAX 8800 Microcode Files	2-2
2-2	Microword Field Definitions.	2-6
2-3	Macroexpression Classes.	2-15
2-4	Sample Register Transfer Macros.	2-16
2-5	Sample Cache Command Macros.	2-17
2-6	Sample CREG/IREG Macros.	2-17
2-7	Sample Microbranch Macros.	2-18
2-8	Sample Miscellaneous Macros.	2-19
2-9	Pipelined/Nonpipelined CPU Comparisons	2-22
2-10	Pipeline Time States/CPU Events.	2-31

3-1	Control Store RAM Segment Functionality.	3-2
3-2	Next Microaddress Sources.	3-5
3-3	IBRTYPE/IBRMASK Microword Field Relationship . .	3-15
3-4	Microbranch Conditions	3-16
3-5	Special Microbranch Condition Bit Usage.	3-18
3-6	Microtrap Conditions and Vectors	3-24
3-7	Machine Check Microtrap Conditions	3-25
3-8	IBST and PCNC MCA Outputs After an IB Flush. . .	3-37
3-9	IB Flush Relative State Changes - IBST and PCNC MCAs	3-38
3-10	IB Read Address/IB Read Port Source.	3-46
3-11	Data Aligner Control Signals/Data Selection. . .	3-52
3-12	IB Data Format Control Signals/Functions	3-62
3-13	IB Format Control Signals/Specifier Data Type. .	3-63
3-14	IB Data Formatter and Data Scrambler Output. . .	3-67
3-15	Floating Point Short Literal Formats	3-68
3-16	Specifier Size Logic Control Signals	3-79
3-17	Slow Spec Size <2:0> Values.	3-80
3-18	IB Pointer Source.	3-84
3-19	Operand Specifier Entry Address Bit Descriptions	3-97
3-20	Operand Data Size/Access Type Correlation. . . .	3-98
3-21	Operand Specifier Entry Address Symbolic Labels	3-100
3-22	Opcode Entry Address Bit Descriptions.	3-105
3-23	Special Microaddress Conditions.	3-114
3-24	Special Conditions Serviced During IB Decode Cycles	3-116
3-25	Decoder RAM Output Signal Descriptions	3-124
3-26	Execute Code for a BRB Instruction	3-132
3-27	Microword CTL.BRB.MEM Event Timing	3-134
3-28	IMISC Field Settings for Macrobranch Recipes . .	3-136
3-29	IMISC Field Settings for PSL Condition Code Recipes	3-143
3-30	Execute Code for a AOBLEQ Instruction.	3-146
3-31	IMISC Field Settings for State Flag Control. . .	3-149
3-32	Hardware Interrupt Priority Levels	3-154
3-33	Interrupt ID Codes/IPLs.	3-157

EXAMPLE

No.	Title	Page
2-1	Sample Field Value Assignments - I_APORT	2-12

SECTION 7 EXECUTION BOX LOGIC (EBOX)

CHAPTER 1 INTRODUCTION

1.1 GENERAL. 1-1
1.1.1 EBox Organization. 1-2
1.1.2 EBox Operators 1-4
1.1.2.1 Main ALU 1-4
1.1.2.2 Cache Data Path (CDP) and Bus Watcher/Decoder (BWD). 1-4
1.2 SLICE MODULE (SLC1/SLC0) FUNCTIONS 1-5
1.2.1 Parity Generator/Checker (PAR) 1-5
1.2.2 Register File (RGF). 1-6
1.2.3 Slow Data File (SDF) 1-6
1.2.4 Program Counter (PC) Subsystem 1-7
1.2.5 Cache Data Path (CDP). 1-7
1.2.6 Main Arithmetic Logic Unit (Main ALU). 1-8
1.2.7 Bus Watcher/Decoder (BWD). 1-8
1.3 SHIFTER MODULE (SHR) FUNCTIONS 1-9
1.3.1 Shifter (SHF). 1-9
1.3.1.1 Integer Data 1-9
1.3.1.2 Floating-Point Data. 1-9
1.3.1.3 Decimal String Data. 1-10
1.3.2 Floating-Point (FP) Support. 1-10
1.3.2.1 Priority Encoder (PEN) 1-10
1.3.2.2 Shift ALU (SALU) 1-11
1.3.2.3 Exponent ALU (XALU). 1-11
1.3.3 Multiplier/Divider (MULT). 1-11
1.4 EBOX REGISTERS 1-12
1.4.1 P0LR, P1LR, and SLR Internal Bit Formats . . . 1-13
1.4.2 VAX 8800-Specific Registers. 1-13
1.4.2.1 Machine Check Status Register (MCSTS). . . . 1-13
1.4.2.2 System Identification (SID) Register 1-14
1.4.2.3 Revision Registers (REVR1 and REVR2) 1-15
1.4.2.4 EBox Parity Error Register (EBER). 1-17

CHAPTER 2 FUNCTIONAL DESCRIPTION

2.1 GENERAL. 2-1
2.2 SLICE MODULE (SLC1/SLC0) DESCRIPTION 2-1
2.2.1 Parity Generator/Checker (PAR) 2-1
2.2.1.1 Parity Generator 2-8
2.2.1.2 Parity Checker 2-8
2.2.1.3 EBox Parity Error Register (EBER). 2-11
2.2.1.4 Carry Save Logic 2-12
2.2.2 Register File (RGF). 2-14
2.2.2.1 Floating-Point Shuffle (FPS) 2-14
2.2.2.2 Memory Data (MD) Registers 2-16
2.2.2.3 Traps and Stalls 2-16
2.2.3 Slow Data File (SDF) 2-18
2.2.3.1 Writes 2-19
2.2.3.2 Reads. 2-19
2.2.3.3 Stalls and Traps 2-19

2.2.4	Program Counter (PC) Subsystem	2-22
2.2.4.1	PC VA FA Multiplexer	2-22
2.2.4.2	Virtual Address (VA) File	2-22
2.2.4.3	Trap Shadow Logic	2-23
2.2.4.4	Program Counter (PC)	2-24
2.2.5	Cache Data Path (CDP)	2-31
2.2.5.1	Cache Data Buffer (CDBF)	2-31
2.2.5.2	Cache Data Store (CDS)	2-33
2.2.6	Main Arithmetic Logic Unit (Main ALU)	2-41
2.2.6.1	ALU First Half (ALF)	2-41
2.2.6.2	Main ALU Functions	2-44
2.3	SHIFTER MODULE (SHR) DESCRIPTION	2-51
2.3.1	Shifter (SHF)	2-51
2.3.1.1	Shift Count Bus	2-55
2.3.1.2	General Function Selection	2-55
2.3.1.3	Logical Shift or Rotate	2-58
2.3.1.4	Arithmetic Shift	2-58
2.3.1.5	Decimal String Conversion	2-58
2.3.2	Floating-Point (FP) Support	2-60
2.3.2.1	Priority Encoder (PEN) MCA	2-62
2.3.2.2	Shift ALU (SALU)	2-70
2.3.2.3	Exponent ALU (XALU)	2-78
2.3.3	Multiplier/Divider (MULT)	2-86
2.3.3.1	Data Interface Signals	2-86
2.3.3.2	Carry and Control Signals	2-86
2.3.3.3	Logical and Arithmetic Functions	2-86
2.3.3.4	Multiplier Operation	2-95
2.3.3.5	Divider Operation	2-96

FIGURES

No.	Title	Page
1-1	VAX 8800 CPU Kernel Block Diagram	1-1
1-2	Execution Unit (EBox) Block Diagram	1-3
1-3	Machine Check Status Register (MCSTS)	1-13
1-4	System Identification (SID) Register	1-14
1-5	Revision Register 1 (REVR1)	1-15
1-6	Revision Register 2 (REVR2)	1-16
2-1	Slice Module (SLC1/SLC0) Block Diagram	2-2
2-2	Parity Generator/Checker (PAR) Block Diagram	2-3
2-3	EBox Parity Error Register (EBER)	2-11
2-4	Register File (RGF) Block Diagram	2-15
2-5	Slow Data File (SDF) Block Diagram	2-19
2-6	Program Counter (PC) Subsystem Block Diagram	2-23
2-7	Cache Data Path (CDP) Block Diagram	2-32
2-8	Main Arithmetic Logic Unit (Main ALU) Block Diagram	2-43
2-9	Shifter Module (SHR) Block Diagram	2-52
2-10	Shift MCA (SHFT) Logic and Gating Block Diagram	2-53

2-11	Shift Control MCA (SHC) Block Diagram.	2-54
2-12	Shift Count Bus Signal and Gating Block Diagram.	2-55
2-13	VAX-11 Floating-Point Formats.	2-61
2-14	Priority Encoder (PEN) Block Diagram	2-64
2-15	INMUX Mapping of the BPORT Input Data.	2-65
2-16	Shift ALU (SALU) Block Diagram	2-73
2-17	Exponent ALU (XALU) Block Diagram.	2-79
2-18	Multiplier/Divider (MULT) Block Diagram.	2-88

TABLES

No.	Title	Page
1-1	Privileged IPRs Maintained by the EBox	1-12
1-2	P0LR, P1LR, and SLR Internal Formats	1-13
1-3	Machine Check Status Register (MCSTS) Bit Descriptions	1-14
1-4	System Identification (SID) Register Bit Field Descriptions	1-15
1-5	Revision Register 1 (REVR1) Bit Field Descriptions	1-16
1-6	Revision Register 2 (REVR2) Bit Field Descriptions	1-17
2-1	Parity Generator/Checker (PAR) Signal Descriptions	2-3
2-2	A-Side Port Control.	2-10
2-3	B-Side Port Control.	2-10
2-4	Keepgoing Conditions for A CD PAR<3,1>	2-10
2-5	E_SHFT<4:0> Control of the EBox Parity Error Register (EBER).	2-12
2-6	E_ALUCI<1:0> Control of the ALU Carry-In Source	2-13
2-7	Register File (RGF) Address Allocation	2-16
2-8	Register File (RGF) Signal Descriptions.	2-17
2-9	Slow Data File (SDF) Signal Descriptions	2-20
2-10	Program Counter (PC) Subsystem Signal Descriptions	2-27
2-11	E_VAWRT and I_APORT<7:6> Control of PC VA FA Multiplexer Input Selection.	2-30
2-12	PC Multiplexer Input Selection PC Multiplexer Select Signals	2-30
2-13	Cache Data Buffer (CDBF) Signal Description. . .	2-34
2-14	Cache Data Store (CDS) Signal Description. . . .	2-38
2-15	CDS Output Multiplexer Control for Each Slice. .	2-40
2-16	ALU First Half (ALF) Signal Descriptions	2-45
2-17	ALU Second Half (ALS) Signal Descriptions. . . .	2-48
2-18	A-Side Select (ASEL) Input Control Signals . . .	2-49
2-19	B-Side Select (BSEL) Input Control Signals . . .	2-49
2-20	Keepgoing/Stall Conditions	2-49
2-21	EALU<5:0> Field Control of the Main ALU.	2-50

2-22 Shift Count Bus Signals and Source 2-55
2-23 ESHFT<4:0> Field Selection of Shifter (SHF) MCA Logic Functions. 2-56
2-24 ESHFTSEL Selection of a Result Output to the BP Bus. 2-57
2-25 EFPFORMAT<1:0> Field Control of Decimal String Data Conversions. 2-59
2-26 EPEFUNC Field Selection of PEN Functions 2-65
2-27 Priority Encoder (PE) Results Passed to the Shift Count Bus 2-67
2-28 Increment Multiplexer Data (INCD) Selection to the Incrementer (INCR). 2-68
2-29 Sticky Bit Logic Input and Test Selection. . . . 2-68
2-30 G<1:0> Guard Bit Input Selection 2-69
2-31 Round Bit R<1:0> Input Selection 2-69
2-32 SALU and XALU Control Signals from the Microcode 2-71
2-33 ESXALUFN Field Control of the SALU Functions . . 2-72
2-34 Resulting Sign of the Fraction 2-74
2-35 SALU Selection of the APORT and BPORT Inputs . . 2-74
2-36 A-Latched Condition Code Inputs to the Branch Multiplexer. 2-76
2-37 Microbranch Condition Code Description 2-77
2-38 ESXALUFN<5:3) Control of the General XALU Functions. 2-80
2-39 XALU Functions with E_SXALUFN<5:3> Equal to 000 2-81
2-40 XALU Functions with ESXALUFN<5:3> Not Equal to 000 2-81
2-41 XALU Condition Code (XALUCC) Tests 2-82
2-42 M1 Inputs Passed to M3 2-83
2-43 M3 Inputs Passed to the Adder B-side 2-84
2-44 M2 Outputs to M6 or the XREG 2-85
2-45 M2 Data Passed to the BP Bus by M6 2-85
2-46 Multiplier/Divider (MULT) Control Signals from the Microcode 2-90
2-47 E_MULDIV Field Control of the MULT Functions . . 2-91
2-48 MULT Logic Signal Port Function Description. . . 2-93

SECTION 8 CACHE BOX LOGIC (CBOX)

CHAPTER 1 INTRODUCTION

1.1 CACHE BOX SYSTEM DESCRIPTION 1-1
1.2 CBOX OPERATION 1-5
1.2.1 CBox Cycles. 1-6
1.2.1.1 Quiescent State. 1-7
1.2.1.2 Read Cycle 1-8
1.2.1.3 Write Cycles 1-8
1.2.1.4 The PIBA 1-9

1.2.1.5 TB Cycles 1-9
1.2.1.6 Refill Operation 1-10
1.2.1.7 Invalidate Cycle 1-10
1.2.2 CBox Stalls 1-11

CHAPTER 2 FUNCTIONAL DESCRIPTION

2.1 CBOX SUBSYSTEMS DESCRIPTION 2-1
2.1.1 Translation Buffer 2-1
2.1.1.1 VA Latch 2-4
2.1.1.2 TB RAM 2-5
2.1.1.3 TB Match MCA 2-8
2.1.1.4 TB RAM Bypass 2-10
2.1.1.5 PA Latch 2-14
2.2 CBOX SUBSYSTEMS DESCRIPTION 2-19
2.2.1 Cache 2-19
2.2.1.1 Cache Data Path Logic 2-19
2.2.1.2 Cache Tag MCA 2-25
2.2.1.3 Cache Match MCA 2-25
2.2.1.4 Cache Control MCA 2-27
2.2.1.5 MD Number MCA 2-28
2.3 CBOX SUBSYSTEMS DESCRIPTION 2-29
2.3.1 NMI Interface 2-29
2.3.1.1 NMI Address/Data Slices 2-31
2.3.1.2 NMI Out Control 2-36
2.3.1.3 NMI In Control 2-42
2.3.1.4 NMI Arbitration/Acknowledgment 2-47
2.3.1.5 CBox NMI Registers 2-52

FIGURES

No. Title Page

1-1 CBox -- Block Diagram 1-3
1-2 CBox Cycle Timing 1-5

2-1 Translation Buffer -- Block Diagram 2-2
2-2 Virtual Address Fields 2-4
2-3 TB -- Write Sequence Diagram 2-6
2-4 TB Match MCA -- Simplified Block Diagram 2-9
2-5 PABH MCA -- Simplified Block Diagram 2-12
2-6 PABL MCA -- Simplified Block Diagram 2-13
2-7 PA Latch -- Logic Diagram 2-15
2-8 PA Latch Bit Routing -- Refill Cycle 2-17
2-9 PA Latch Bit Routing -- VA Reference 2-18
2-10 Cache -- Block Diagram 2-20
2-11 NMI Interface -- Block Diagram 2-30
2-12 NMI Address/Data Slices MCA -- Simplified Block Diagram 2-32
2-13 NMI Out Control -- Simplified Block Diagram . . . 2-37
2-14 Control Store Microword Format Diagram 2-41
2-15 NMI In Control -- Block Diagram 2-42

2-16 NMI Arbitration/Acknowledgment Control -- Simplified Block Diagram 2-49
2-17 Timeout -- Flow Diagram. 2-51
2-18 CBox NMI Registers -- Location Diagram 2-52
2-19 Cache Register -- Bit Format 2-53
2-20 Cache Error Register Byte 2 -- Bit Format. . . . 2-55
2-21 Cache Error Register Byte 1 -- Bit Format. . . . 2-56
2-22 Cache Error Register Byte 0 -- Bit Format. . . . 2-57
2-23 NMI Fault/Status Register Byte 1 -- Bit Format 2-58
2-24 NMI Fault/Status Register Byte 0 -- Bit Format 2-60
2-25 NMI Error Address Register -- Bit Format 2-61
2-26 NMI Silo Byte 2 -- Bit Format. 2-63
2-27 NMI Silo Byte 1 -- Bit Format. 2-65
2-28 NMI Silo Byte 0 -- Bit Format. 2-66
2-29 Cache TAG Initialization Register -- Bit Format 2-67
2-30 Diagnostic ID Register -- Bit Format 2-68
2-33 Diagnostic Control Register -- Bit Format. . . . 2-69

TABLES

No. Title Page

1-1 CBox Cycles. 1-7

2-1 PROTection Field <03:00> Coding and Access Allowed. 2-7
2-2 TB Match MCA Operation Coding. 2-8
2-3 Cache Register - Bit Descriptions. 2-53
2-4 Cache Error Register - Byte 2 Bit Descriptions 2-55
2-5 Cache Error Register - Byte 1 Bit Descriptions 2-56
2-6 Cache Error Register - Byte 0 Bit Descriptions 2-57
2-7 NMI Fault/Status Register Byte 1 - Bit Descriptions 2-58
2-8 NMI Fault/Status Register Byte 0 - Bit Descriptions 2-60
2-9 NMI Error Address Register -- Bit Descriptions 2-61
2-10 NMI Silo Byte 2 -- Bit Descriptions. 2-64
2-11 NMI Silo Byte 1 -- Bit Descriptions. 2-65
2-12 NMI Silo Byte 0 -- Bit Descriptions. 2-66
2-13 Cache TAG Initialization Register -- Bit Descriptions 2-67
2-14 Diagnostic ID Register -- Bit Descriptions . . . 2-68
2-15 Diagnostic Control Register -- Bit Descriptions 2-69

SECTION 9 MEMORY SYSTEM (MBOX)

CHAPTER 1 INTRODUCTION

1.1 MANUAL SCOPE 1-1
1.2 WRITING PHILOSOPHY 1-1
1.3 MBOX FUNCTIONS 1-2
1.4 MBOX OVERVIEW. 1-4
1.4.1 NMI Signals Used by the MBox 1-6
1.4.2 MBox Operations. 1-6
1.4.3 Octaword Sort-of-Write 1-6
1.4.4 Command Bus Cycles 1-6
1.4.4.1 Write Longword Bus Cycles (Table 1-3). 1-9
1.4.4.2 Write Quadword Bus Cycles (Table 1-5). . . . 1-10
1.4.4.3 Write Octaword Bus Cycles (Table 1-6). . . . 1-11
1.4.4.4 Read Longword Bus Cycles (Table 1-7) 1-11
1.4.4.5 Read Octaword Bus Cycles (Table 1-8) 1-13
1.4.4.6 Read Hexword Bus Cycles (Table 1-9). 1-14
1.4.5 Memory Controller (MCL). 1-14
1.4.5.1 Command/Address Sequence 1-16
1.4.5.2 Normal Write 1-20
1.4.5.3 Read 1-22
1.4.5.4 Masked Write 1-25
1.4.6 Four-Megabyte Memory Array Board (MAR4). . . . 1-29
1.4.6.1 MAR4 Select and Command/Address. 1-30
1.4.6.2 Write Operation. 1-32
1.4.6.3 Read Operation 1-33

CHAPTER 2 MEMORY CONTROLLER (MCL)

2.1 DFA OVERVIEW (Figure 2-1). 2-1
2.1.1 Command/Address Cycle. 2-1
2.1.2 Write Data Cycle(s). 2-5
2.1.3 First Read Data Cycle. 2-7
2.1.4 "Next" Read Data Cycles. 2-8
2.1.5 NMI Interrupt. 2-9
2.1.6 NMI Memory Busy. 2-9
2.1.7 Single-Bit Error Correction During a Masked Write 2-10
2.1.8 CSR Reads. 2-10
2.2 DATA/ADDRESS (DAD) MCAS. 2-10
2.2.1 DAD Ports. 2-13
2.2.2 NMI Writes to Memory 2-13
2.2.3 NMI Reads from Memory. 2-13
2.2.4 Masked Writes Requiring Single-Bit Error Correction 2-14
2.2.5 Error-Free Masked Writes 2-15
2.2.6 Decode RAM Addressing. 2-15
2.2.6.1 Initially Loading the Decode RAM 2-15
2.2.6.2 Reading CSR1 2-15
2.2.7 CSR Reads to the NMI 2-16
2.3 FUNCTION (FUNK) MCA. 2-17
2.3.1 Function Field Parity. 2-20

2.3.2 Function Decoder 2-20
2.3.3 NEW CMD EARLY/NEW CMD LATE 2-21
2.3.4 Read Lock Function 2-22
2.3.5 Write Unlock Function. 2-22
2.3.6 Lock-Timeout Counter 2-23
2.3.7 Block Command. 2-23
2.3.8 Write Sequence Fault 2-24
2.3.8.1 Write Longword to Memory 2-24
2.3.8.2 Write Longword to CSR. 2-26
2.3.8.3 Write Quadword 2-26
2.3.8.4 Write Octaword 2-27
2.3.9 NMI Faults 2-28
2.3.10 NMI Confirmation 2-30
2.3.11 NMI DEAD 2-31
2.3.12 CSRs (Figure 2-4). 2-32
2.3.13 Read/Return and Read/Continue (Figure 2-5) . . 2-32
2.3.13.1 MCL Immediately Gets the NMI 2-33
2.3.13.2 MCL Waits for the NMI. 2-34
2.3.14 MRM Hold Command (Figure 2-6). 2-35
2.3.15 Force One Cycle (Figure 2-6) 2-35
2.4 ARBITRATION/ID (ARID) MCA. 2-35
2.4.1 NMI Data Parity (Figure 2-7) 2-35
2.4.1.1 Parity In. 2-34
2.4.1.2 Parity Out 2-37
2.4.2 NMI Function/ID Parity (Figure 2-7). 2-38
2.4.2.1 Parity In. 2-38
2.4.2.2 Parity Out 2-38
2.4.3 Fault Detect (Figure 2-8). 2-38
2.4.4 NMI ID/Mask (Figure 2-9) 2-40
2.4.4.1 ID/Mask In 2-40
2.4.4.2 ID Out 2-42
2.4.5 Arbitration/Hold Logic 2-43
2.4.5.1 Memory Gets the Bus Right Away - Longword Read. 2-43
2.4.5.2 Memory Gets the Bus Right Away - Octaword Read (Figure 2-11). 2-45
2.4.5.3 Memory Gets the Bus Right Away - Longword Read Back-to-Back with Another Read Function. 2-45
2.4.5.4 Memory Gets the Bus Right Away - Hexword Read 2-46
2.4.5.5 Memory Does Not Get the Bus Right Away - Longword Read. 2-46
2.4.5.6 Memory Does not Get the Bus Right Away - Two Longword Reads Back-to-Back or an Octaword Read. 2-46
2.4.6 Interrupts (Figure 2-12) 2-47
2.4.7 CSRs (Figure 2-13) 2-49
2.4.7.1 CSR0 2-49
2.4.7.2 CSR3 2-49
2.4.8 Memory Busy (Figure 2-14). 2-49
2.4.9 Clocks and Clock Control (Figure 2-15) 2-50
2.5 MDP OVERVIEW (Figure 2-16) 2-54
2.5.1 MDB Address In 2-54

2.5.2 MDB Address Out. 2-54
2.5.3 MDB Data In. 2-54
2.5.4 MDB Data Out 2-56
2.5.5 Write-Enable and Bad-Data Bits 2-56
2.5.6 Data Parity. 2-57
2.5.7 Data Read Operation. 2-57
2.5.8 Masked Write Operation 2-58
2.5.9 CSR Reads. 2-59
2.6 MEMORY DATA BUFFER (MDB) (Figure 2-17) 2-59
2.6.1 Address Read Port. 2-61
2.6.2 Data Read Port 2-62
2.6.3 W Write Port 2-62
2.6.4 C Write Port 2-63
2.6.5 CSR Logic. 2-64
2.7 MEMORY DATA BUFFER CONTROL (MDBC) MCA. 2-66
2.7.1 Loading Data into the MDB. 2-67
2.7.1.1 Normal Octaword Write With X and Y Buffers Empty. 2-67
2.7.1.2 Normal Octaword Write With Data Already in the X or Y Buffer 2-74
2.7.1.3 Masked Write With No Errors. 2-76
2.7.1.4 Masked Write With a Correctable Error. . . . 2-78
2.7.1.5 Masked Write With an Uncorrectable Error . . 2-79
2.7.2 Unloading Data From the MDB. 2-80
2.7.2.1 General. 2-80
2.7.2.2 Detailed 2-82
2.7.3 Y Out Select Logic 2-83
2.7.3.1 General. 2-83
2.7.3.2 Detailed 2-84
2.7.4 Internal Error, Write Decode RAM, and Clocks 2-87
2.8 DATA CHECK (DCHK) MCA. 2-87
2.8.1 Syndrome Generate. 2-89
2.8.2 Error Check. 2-89
2.8.3 Error Status 2-92
2.8.4 Serializer and CSR2. 2-95
2.8.5 Diagnostic Mode (Figure 2-28). 2-95
2.8.6 Reset and Clocks (Figure 2-28) 2-98
2.9 MRM OVERVIEW (Figure 2-33) 2-99
2.9.1 NAB Board Select 2-104
2.9.2 NAB Command Field and Parity 2-105
2.9.3 MDB Address Selection. 2-105
2.9.4 Internal Error 2-106
2.9.5 MDB Write-Data Load. 2-106
2.9.6 Octaword Writes. 2-106
2.9.7 Read-Data Cycle(s) 2-106
2.9.8 Masked Writes. 2-107
2.9.9 CSR READS. 2-108
2.10 MEMORY SEQUENCE CONTROL (MSC) MCA) 2-108
2.10.1 MSC Buffer Control (Figure 2-35) 2-109
2.10.1.1 Buffer Control Operation 2-109
2.10.1.2 AMRM BUSY REQ. 2-112
2.10.1.3 A CMD PROC START 2-113

2.10.2 BNUM Probe Buffer and Error Logic (Figure 2-36). 2-113
2.10.2.1 Probe Logic. 2-113
2.10.2.2 Error Logic. 2-116
2.10.3 Command/Address/Size Buffer (Figure 2-37). . . 2-116
2.10.3.1 Command Channel. 2-116
2.10.3.2 Address/Size Channel 2-118
2.10.4 Size Logic (Figure 2-38) 2-118
2.10.5 Starting Address Logic (Figure 2-40) 2-120
2.10.5.1 Initial Starting Address 2-120
2.10.5.2 Address Incrementation 2-124
2.10.6 Mask Address/Size Buffer (Figure 2-41) 2-124
2.10.7 Write Command Logic (Figure 2-42). 2-126
2.10.7.1 Write Machine. 2-126
2.10.7.2 Write Command Bits 2-128
2.10.8 Masked-Write Logic (Figure 2-43) 2-129
2.10.9 Command Done Logic (Figure 2-44) 2-131
2.10.9.1 BMSC PRE CMD DONE. 2-131
2.10.9.2 BMSC PRE MASK DONE 2-133
2.10.10 Command Parity 2-133
2.11 MEMORY SEQUENCE CONTROL 1 (MSC1) MCA 2-133
2.11.1 Mask Store (Figure 2-45) 2-134
2.11.2 Select Out Buffer Control (Figure 2-46). . . . 2-137
2.11.3 Read Buffer Control (Figure 2-47). 2-139
2.11.4 MDB Address I/O Select Logic (Figure 2-48) . . 2-139
2.11.4.1 MDB Address In Select Bits 2-139
2.11.4.2 MDB Address Out Select Bits. 2-142
2.11.5 Error Address Pointer. 2-143
2.11.6 BMRM INVERT ADDR4. 2-143
2.12 MEMORY ARRAY SEQUENCE CONTROL (MASC) MCA (Figure 2-49). 2-143
2.12.1 Command/Address Parity 2-143
2.12.2 Force Parity Error 2-145
2.12.3 Board Number (BMAS BNUM<2:0>). 2-145
2.12.4 Send No Command. 2-145
2.12.5 MASC Empty 2-146
2.12.6 Board Select (BMAS BD SEL<2:0>). 2-146
2.12.7 Command Accept (BMAS CMD ACPT) and Board Valid (BMAS BD VALID). 2-146
2.13 READ CONTROL SEQUENCER (RCS) MCA 2-147
2.13.1 Power Control (Figure 2-50). 2-147
2.13.2 CSR Control (Figure 2-51). 2-149
2.13.2.1 BMRM EN SERIAL RD. 2-149
2.13.2.2 BMRM SERIAL RD<2:0>. 2-149
2.13.2.3 AMRM CSR WRITE 2-149
2.13.2.4 ARCS FORCE CMD ACPT. 2-151
2.13.2.5 AMRM MPR DATA SEL. 2-151
2.13.2.6 BMRM FAKE CMD ACPT 2-151
2.13.3 Read Command Bits (Figure 2-52). 2-151
2.13.3.1 AMRM READ CMD<0> 2-151
2.13.3.2 BRCS READ CMD<1> 2-151
2.13.4 Board Select/Enable (Figure 2-52). 2-151
2.13.4.1 Board Select 2-151
2.13.4.2 Board Select Enable. 2-153

2.13.5 Read Data In Signals (Figure 2-52) 2-153
2.13.5.1 AMRM DRIVE NEW DATA. 2-153
2.13.5.2 BMRM NAB GATE. 2-153
2.13.6 RCS Full/Empty Status (Figure 2-52). 2-154
2.13.6.1 ARCS FULL. 2-154
2.13.6.2 BRCS EMPTY 2-154
2.14 BATTERY BACKUP UNIT (BBU). 2-154
2.14.1 Loss of Power. 2-156
2.14.2 Return of Power. 2-156

CHAPTER 3 FOUR MEGABYTE MEMORY ARRAY BOARD (MAR4)

3.1 VAX 8800 ARRAY BUS (NAB) 3-1
3.1.1 Signal Clocks. 3-1
3.1.2 Longword Write Timing. 3-1
3.1.3 Longword Read Timing 3-8
3.1.4 Octaword Read Timing 3-8
3.2 MAR4 OVERVIEW. 3-10
3.2.1 Write Operation (Figures 3-4 and 3-5). 3-12
3.2.2 Read Operation (Figures 3-4 and 3-6) 3-19
3.3 MAR4 DETAILED DESCRIPTIONS 3-22
3.3.1 Clock Logic. 3-23
3.3.2 4MBARRAY Banks 3-26
3.3.2.1 Array Bank Components. 3-26
3.3.2.2 Data Flow. 3-29
3.3.2.3 Array Command Sequencing 3-29
3.3.2.4 Array Refresh Sequencing 3-34
3.3.2.5 Battery Mode 3-36
3.3.2.6 Cold Start 3-38
3.3.2.7 Array Bank Differences 3-38
3.3.3 Input Parser 3-38
3.3.3.1 Command/Address Parity Check 3-39
3.3.3.2 Write Inhibit. 3-39
3.3.3.3 Data Ready Done. 3-39
3.3.3.4 MAR4 Board Selection 3-41
3.3.3.5 Generation of Array Bank Signals 3-41
3.3.3.6 Array Not Busy 3-42
3.3.3.7 Battery Mode 3-42
3.3.4 ECC/DPARITY. 3-42
3.3.4.1 ECC Check Bits 3-42
3.3.4.2 Write Inhibit. 3-42
3.3.4.3 Write Data Parity Check. 3-44
3.3.4.4 INT BAD DATA 3-44
3.3.5 Data Output Control. 3-44
3.3.5.1 MAR4 Read Enable 3-44
3.3.5.2 Bank Select. 3-46
3.3.5.3 MAR4 Data Transfer Enable. 3-47
3.3.5.4 Control of Read Data Transfer. 3-47
3.3.5.5 DRDY SNC CLK 3-48
3.3.5.6 Battery Mode 3-48
3.3.6 Refresh. 3-48
3.3.6.1 Normal Mode. 3-49
3.3.6.2 Battery Mode 3-53

FIGURES

No.	Title	Page
1-1	MBox Simplified Block Diagram.	1-3
1-2	MBox Read/Write Simplified Block Diagram	1-5
1-3	MBox Block Diagram	1-17
1-4	Command/Address Flow Diagram	1-18
1-5	Write Data Cycly Flow Diagram.	1-21
1-6	Read Data Cycle Flow Diagram	1-23
1-7	Masked Write Data Cycle Flow Diagram	1-27
1-8	MAR4 Read/Write Flow Diagram	1-31
1-9	MAR4 Command Fields.	1-32
2-1	DFA Block Diagram.	2-3
2-2	DAD Block Diagram.	2-11
2-3	FUNK Function and Control Logic.	2-18
2-4	FUNK CSRs.	2-25
2-5	Read/Return and Read/Continue Logic.	2-29
2-6	Clock and Command Control Logic.	2-36
2-7	Parity Generation and Checking	2-39
2-8	Fault Detect Logic	2-40
2-9	ID/Mask Logic.	2-41
2-10	Arbitration/Hold Logic	2-44
2-11	NMI Arbitration/Hold Timing.	2-45
2-12	Interrupt Logic.	2-48
2-13	CSR Logic.	2-51
2-14	Memory Busy Logic.	2-52
2-15	Clock, Reset, and Unjam Logic.	2-53
2-16	Memory Data Path (MDP) Block Diagram	2-55
2-17	Memory Data Buffer (MDB) Block Diagram	2-60
2-18	CSR Logic.	2-65
2-19	MDBC -- MDB Data-In Selection.	2-69
2-20	Input Load Command Detect Logic.	2-70
2-21	Full Logic	2-71
2-22	X and Y Bit Storage.	2-72
2-23	MDBC -- MDB Feedback Selection	2-77
2-24	Double-Bit Error Logic	2-80
2-25	MDBC -- MDB Data-Out Selection	2-81
2-26	Y Out Select Flow Diagram.	2-85
2-27	MDBC -- Internal Error, Write Decode RAM, and Clocks	2-88
2-28	DCHK Block Diagram	2-90
2-29	Error Check Block Diagram.	2-91
2-30	Error Status Block Diagram	2-93
2-31	Serializer Block Diagram	2-96
2-32	CSR2 Bit Map	2-97
2-33	MRM Block Diagram.	2-100
2-34	MSC Block Diagram.	2-110
2-35	Buffer Control	2-111
2-36	BNUM Probe Buffer and Error Logic.	2-114
2-37	Command/Address/Size Buffer.	2-117
2-38	Size Logic	2-119
2-39	Hex State Machine Flow Diagram	2-121

2-40	Starting Address Logic	2-122
2-41	Mask Address/Size Buffer	2-125
2-42	Write Command Logic	2-127
2-43	Mask Write Logic	2-130
2-44	Command Done Logic	2-132
2-45	Mask Store Block Diagram	2-135
2-46	Select-Out Buffer Control Block Diagram	2-138
2-47	Read Buffer Control Block Diagram	2-140
2-48	MDB Address I/O Select Block Diagram	2-141
2-49	MASC Block Diagram	2-144
2-50	Power Control	2-148
2-51	CSR Control	2-150
2-52	Array Read Control	2-152
2-53	MCL BBU Block Diagram	2-155
2-54	Power Down Flow Diagram	2-157
2-55	Power Up Flow Diagram	2-158
3-1	Longword Write Timing Diagram	3-5
3-2	Longword Read Timing Diagram	3-6
3-3	Octaword Read Timing Diagram	3-7
3-4	MAR4 Block Diagram	3-11
3-5	Write Flow Diagram	3-13
3-6	Read Flow Diagram	3-15
3-7	Clock Logic Block Diagram	3-24
3-8	Clock Timing Diagram	3-25
3-9	4MBARRAY Bank Block Diagram (Bank 0 Shown)	3-27
3-10	Array Command Flow Diagram	3-30
3-11	Array Refresh Flow Diagram	3-35
3-12	Input Parser Block Diagram	3-40
3-13	ECC/DPARITY Block Diagram	3-43
3-14	Data Output Control Block Diagram	3-45
3-15	Refresh Time Periods	3-50
3-16	Refresh Block Diagram	3-51
3-17	Refresh Flow Diagram -- Normal Mode	3-52
3-18	Refresh Flow Diagram -- Battery Mode	3-55
3-19	Initiation of Battery Mode Refreshes	3-57
3-20	Termination of Battery Mode Refreshes	3-58
A-1	Flow-Diagram Symbols	A-1

TABLES

No.	Title	Page
1-1	NMI Signals Used by the MBox	1-7
1-2	MBox Command Functions	1-8
1-3	Write Longword Bus Cycles	1-9
1-4	NMI Confirmation Codes	1-10
1-5	Write Quadword* Bus Cycles	1-10
1-6	Write Octaword Bus Cycles	1-11
1-7	Read Longword Bus Cycles	1-12
1-8	Read Octaword Bus Cycles	1-13
1-9	Read Hexword Bus Cycles	1-15

2-1 Command Code 2-2
2-2 Size Code. 2-2
2-3 Function Codes 2-20
2-4 NMI Confirmation Codes 2-30
2-5 Read Function Codes. 2-34
2-6 Read Command Code. 2-59
2-7 Write Commands 2-82
2-8 ENABLE ECC Truth Table 2-99
2-9 Size Code. 2-118
2-10 Initial Address Truth Table. 2-123
2-11 Write State Code 2-128
2-12 Write Command Code 2-128

3-1 NAB Signals. 3-2
3-2 Clock Distribution 3-26
3-3 Read Bank Shift Register Modes 2-46

SECTION 10 NBI (NMI TO VAXBI ADAPTER)

CHAPTER 1 INTRODUCTION

1.1 GENERAL INFORMATION. 1-1
1.2 PHYSICAL DESCRIPTION AND CIRCUIT TECHNOLOGY. . . . 1-3
1.3 BASIC BLOCK DIAGRAM. 1-3
1.4 BASIC OPERATION. 1-7
1.4.1 CPU Read/Write Data Transfers. 1-7
1.4.2 DMA Read/Write Data Transfers. 1-10
1.4.3 Interrupt Operation. 1-13
1.5 NBI REGISTERS. 1-17
1.5.1 NBIA Registers 1-17
1.5.1.1 Control/Status Registers (CSR0 and CSR1) . . 1-18
1.5.1.2 Vector Registers (BR4VR through BR7VR) . . . 1-24
1.5.2 NBIB (BIIC) Registers. 1-25
1.5.2.1 Device Register (DTYPE). 1-26
1.5.2.2 VAXBI Control/Status Register (BICSR). . . . 1-27
1.5.2.3 Bus Error Register (BER) 1-30
1.5.2.4 Error Interrupt Control Register (EINTRCSR) 1-34
1.5.2.5 INTR Destination Register (INTRDES). 1-36
1.5.2.6 IPINTR Mask Register (IPINTRMSK) 1-37
1.5.2.7 IPINTR/STOP Destination Register (FIPSDES). 1-38
1.5.2.8 IPINTR Source Register (IPINTRSRC) 1-39
1.5.2.9 Starting and Ending Address Registers (SADR and EADR). 1-40
1.5.2.10 BCI Control and Status Register (BCICSR) . . 1-42
1.5.2.11 Write Status Register (WSTAT). 1-45
1.5.2.12 Force IPINTR/STOP Command Register (FIPSCMD). 1-46
1.5.2.13 User Interrupt Control Register (UINTRCSR) 1-47
1.5.2.14 General Purpose Registers (GPR <3:0>). . . . 1-49

CHAPTER 2 INTERFACE DESCRIPTIONS

2.1 NMI 2-1
2.1.1 NMI Signals. 2-4
2.1.2 Basic Timing 2-4
2.1.3 NMI Address Space. 2-11
2.1.4 NMI Read/Write Transactions. 2-11
2.1.5 NMI Arbitration/Memory Busy. 2-16
2.1.6 NMI Interrupts 2-20
2.1.7 NMI Errors 2-23
2.2 DATA BUS (BETWEEN NBIA AND NBIB) 2-24
2.3 VAXBI. 2-29
2.3.1 VAXBI Signals. 2-29
2.3.2 Basic Timing 2-34
2.3.3 VAXBI Address Space. 2-35
2.3.4 VAXBI Read/Write Transactions. 2-40
2.3.5 Interrupt Operation (INTR, IDENT, and IPINTR Transactions). 2-44
2.3.5.1 Interrupt (INTR) Transactions. 2-45
2.3.5.2 Identify (IDENT) Transactions. 2-47
2.3.5.3 Interprocessor Interrupt (IPINTR) Transactions 2-50
2.3.6 STOP Transactions. 2-52
2.3.7 Invalidate (INVAL) Transactions. 2-54
2.3.8 Bus Arbitration. 2-56
2.3.8.1 Bus Requests 2-56
2.3.8.2 Arbitration Modes. 2-57
2.3.8.3 Arbitration Control. 2-57
2.3.8.4 Extending a Transaction. 2-59
2.3.8.5 Special Mode Functions 2-59
2.3.9 VAXBI Errors 2-60
2.3.9.1 Parity Checking. 2-61
2.3.9.2 Transmit Check Error Detection 2-61
2.3.9.3 Protocol Checking. 2-62

CHAPTER 3 FUNCTIONAL DESCRIPTION

3.1 INTRODUCTION 3-1
3.1.1 NBIA Block Diagram 3-1
3.1.1.1 NMI Data Buffer. 3-2
3.1.1.2 NPAR MCA 3-4
3.1.1.3 NBIM MCA 3-4
3.1.1.4 NBFD MCA 3-4
3.1.1.5 NBAP MCA 3-4
3.1.1.6 NBCT MCA 3-5
3.1.1.7 DSEQ MCA 3-5
3.1.1.8 DC022 Transaction Buffer 3-5
3.1.1.9 Data (Bus) Buffers 3-5
3.1.1.10 Data Bus (and Transaction Buffer) Controls . . 3-5
3.1.2 NBIB Block Diagram 3-6
3.1.2.1 Data Bus Data Buffer 3-6
3.1.2.2 BCI Data Buffer. 3-6
3.1.2.3 Parity and Translation Logic 3-6

3.1.2.4 Data Buffer Read/Write Control 3-8
3.1.2.5 Length and Interrupt Control Logic 3-8
3.1.2.6 Master and Slave Port Sequencers 3-8
3.1.2.7 BIIC 3-8
3.1.2.8 VAXBI Clock Driver/Receiver. 3-9
3.2 INITIALIZATION/SELFTEST. 3-12
3.2.1 Basic NBI Initialization 3-12
3.2.2 BIIC Initialization/Selftest 3-14
3.2.3 Powerup. 3-14
3.2.4 NBI INIT/UNJAM 3-16
3.2.5 RESET (By Connected VAXBI Device). 3-18
3.3 CPU READ/WRITE OPERATIONS. 3-20
3.3.1 NMI Address Decoding and Translation 3-22
3.3.2 Local Read/Write Operations. 3-24
3.3.2.1 Command/Address Cycle. 3-25
3.3.2.2 Write Data Cycle 3-28
3.3.2.3 Return Data Cycle. 3-30
3.3.2.4 Parity Generation and Checking 3-32
3.3.3 VAXBI Read/Write (and IDENT) Operations. . . . 3-33
3.3.3.1 Command/Address Transfer 3-36
3.3.3.2 Write Data Transfer. 3-43
3.3.3.3 Return Read Data Transfer. 3-49
3.3.3.4 Parity Generation and Checking 3-55
3.3.4 Write Sequence Faults. 3-58
3.3.5 NMI BUS ACCESS TIMEOUTS. 3-58
3.3.6 VAXBI Errors 3-58
3.4 DMA READ/WRITE OPERATIONS. 3-60
3.4.1 Command/Address Transfer 3-66
3.4.1.1 Command/Address to BCI Data Buffer and Data Bus Buffer. 3-66
3.4.1.2 Command/Address to NBIA's Transaction Buffer 3-70
3.4.1.3 Command/Address to NMI (NMI Command/Address Cycle) 3-70
3.4.2 Write Data Transfer. 3-72
3.4.2.1 Write Data to BCI Data Buffer and Data Bus Buffer. 3-73
3.4.2.2 End of VAXBI Transaction (and Retries) . . . 3-76
3.4.2.3 Write Data to NBIA's Transaction Buffer. . . 3-76
3.4.2.4 Write Data to NMI (NMI Write Data Cycle or Cycles) 3-77
3.4.2.5 NMI Write Transaction Retries (NO ACCESS/MEMORY BUSY/NOACK). 3-78
3.4.2.6 DMA Errors 3-78
3.4.3 Return Read Data Transfer. 3-79
3.4.3.1 Return Read Data to Transaction Buffer . . . 3-79
3.4.3.2 NMI Read Transaction Retries (MEMORY BUSY). 3-83
3.4.3.3 Return Read Data to NBIB 3-83
3.4.3.4 Return Read Data to BCI Data Buffer and BIIC (VAXBI READ DATA CYCLE). 3-83
3.4.3.5 End of VAXBI Transaction 3-84
3.4.3.6 DMA Errors (VAXBI Transaction Retries) . . . 3-85
3.4.4 Parity Generation and Checking 3-86

3.4.4.1 Command/Address and Write Data/Mask Parity 3-86
3.4.4.2 Return Read Data/Status Parity 3-88
3.4.5 Timeouts. 3-90
3.4.6 Read Sequence Faults 3-90
3.5 INTERRUPT (INTR AND IPINTR) OPERATIONS 3-91
3.5.1 Decoding Interrupt Requests. 3-92
3.6 MISCELLANEOUS OPERATIONS 3-96
3.6.1 BIIC Register Read/Write Operations (by Other VAXBI Nodes) 3-96
3.6.2 VAXBI Stop Transactions. 3-96
3.6.3 VAXBI INVAL and BROADCAST Transactions 3-96
3.7 DIAGNOSTIC DATA TRANSFERS. 3-97
3.7.1 BIIC Loopback Requests 3-97
3.7.2 NBIA Wraparound. 3-98
3.7.3 CPU Read/Write to Memory (Flip Address Bits <29> and <22>). 3-99

FIGURES

No.	Title	Page
1-1	NBI Configuration.	1-2
1-2	NBI Basic Block Diagram	1-4
1-3	DC022 Transaction Buffer Organization	1-6
1-4	CPU Read/Write Data Transfer	1-8
1-5	DMA Read/Write Data Transfers.	1-11
1-6	INTR/IPINTR Operation	1-14
1-7	Interrupt Vector Format	1-15
1-8	SCB Format (Example)	1-16
1-9	Control/Status Register 0 (CSR0)	1-18
1-10	Control/Status Register 0 (CSR1)	1-22
1-11	Vector Registers (BR4VR through BR7VR)	1-24
1-12	Device Register (DTYPE).	1-26
1-13	VAXBI Control/Status Register (BICSR).	1-27
1-14	Bus Error Register (BER)	1-33
1-15	Error Interrupt Control Register (EINTRCSR).	1-34
1-16	INTR Destination Register (INTRDES).	1-36
1-17	IPINTR Mask Register (IPINTRMSK)	1-37
1-18	IPINTR/STOP Destination Register (FIPSDES)	1-38
1-19	IPINTR Source Register (IPINTRSRC)	1-39
1-20	Starting Address Register (SADR)	1-40
1-21	Ending Address Register (SADR)	1-41
1-22	VAXBI Control/Status Register (BICSR).	1-42
1-23	Write Status Register (WSTAT).	1-45
1-24	Force IPINTR/STOP Command Register (FIPSCMD)	1-46
1-25	User Interrupt Control Register (UINTRCSR)	1-47
1-26	General-Purpose Registers (GPR <3:0>).	1-49
2-1	NBIA and NBIB Input/Output Signals	2-2
2-2	Basic NMI Timing	2-4
2-3	NMI Address Space	2-12

2-4 NMI Write Transaction 2-13
2-5 NMI Read Transaction 2-14
2-6 Basic NMI Arbitration Line Timing 2-16
2-7 NMI Arbitration Line Timing (Typical) 2-18
2-8 MEMORY BUSY Timing 2-19
2-9 Fault Signal Timing 2-22
2-10 Basic VAXBI Timing 2-34
2-11 VAXBI Address Space. 2-37
2-12 VAXBI Node Register Space. 2-38
2-13 VAXBI-Required Registers 2-38
2-14 BIIC-Specific Device Registers 2-39
2-15 VAXBI Write Transaction (Octaword Length). . . . 2-42
2-16 VAXBI Read Transaction (Octaword Length) 2-43
2-17 VAXBI Interrupt (INTR) Transaction 2-46
2-18 VAXBI Identify (IDENT) Transaction 2-49
2-19 VAXBI Interprocessor Interrupt (IPINTR) Transaction. 2-51
2-20 VAXBI STOP Transaction 2-53
2-21 VAXBI Invalidate (INVAL) Transaction 2-55
2-22 Bus Arbitration Request Lines. 2-56
2-23 Arbitration State Diagram. 2-58
2-24 VAXBI Arbitration (Example). 2-60

3-1 NBIA Detailed Block Diagram. 3-3
3-2 NBIB Detailed Block Diagram 3-7
3-3 NBI Powerup 3-15
3-4 Reset by VAXBI Node 3-17
3-5 UNJAM/Programmed NBI INIT 3-19
3-6 NMI Address Decoding and Translation 3-23
3-7 Local Read/Write Command/Address Cycle 3-27
3-8 Local Write Data Cycle 3-29
3-9 Local Read Data Cycle 3-31
3-10 Basic Information Flow Between NMI and VAXBI . . 3-34
3-11 NMI to VAXBI Command/Address Transfer. 3-37
3-12 NMI to VAXBI Write Data Transfer 3-44
3-13 VAXBI to NMI Return Read Data Transfer 3-50
3-14 Aligned and Unaligned Quadword Read Data Ordering 3-62
3-15 Basic Information Flow Between VAXBI and NMI During DMA Read/Write Operations 3-64
3-16 VAXBI to NMI Command/Address Transfer. 3-67
3-17 VAXBI to NMI Write Data Transfer 3-74
3-18 NMI to VAXBI Return Read Data Transfer 3-80
3-19 INTR/IPINTR Operations 3-93
3-20 FLIP 29/22 Diagnostic Data Transfers 3-101

TABLES

No.	Title	Page
1-1	NBIA Registers	1-17
1-2	Control/Status Register 0 (CSR0) Bit Descriptions	1-19
1-3	Control/Status Register 1 (CSR1) Bit Descriptions	1-22
1-4	NBIB (BIIC) Registers.	1-25
1-5	Device Register (DTYPE) Bit Descriptions	1-26
1-6	VAXBI Control/Status Register (BICSR) Bit Descriptions	1-28
1-7	Bus Error Register Bit Descriptions.	1-30
1-8	Error Interrupt Control Register (EINTRCSR) Bit Descriptions	1-35
1-9	INTR Destination Register (INTRDES) Bit Descriptions	1-36
1-10	IPINTR Mask Register (IPINTRMSK) Bit Descriptions	1-37
1-11	IPINTR/STOP Destination Register (FIPSDES)	1-38
1-12	IPINTR Source Register (IPINTRSRC) Bit Descriptions	1-39
1-13	Starting Address Register (SADR) Bit Descriptions	1-40
1-14	Ending Address Register (EADR) Bit Descriptions	1-41
1-15	BCI Control/Status Register (BCICSR) Bit Descriptions	1-42
1-16	Write Status Register (WSTAT) Bit Descriptions	1-45
1-17	Force IPINTR/STOP Command Register (FIPSCMD)	1-46
1-18	User Interrupt Control Register (UINTRCSR) Bit Descriptions	1-48
2-1	NMI Signals Connecting to NBIA	2-5
2-2	Data Bus Signals	2-25
2-3	VAXBI Signals.	2-30
3-1	BCI Signals.	3-9
3-2	NBI Initialization	3-13
3-3	CPU Read/Write Summary	3-21
3-4	DMA Read/Write Summary	3-61

SECTION 1
INTRODUCTION AND SYSTEM OVERVIEW

CHAPTER 1
INTRODUCTION

1.1 MANUAL SCOPE

This manual contains a comprehensive technical description of the functional and operational characteristics of the VAX 8800 system. This manual is written at two levels of detail:

1. Introduction and overview
2. Functional and detailed logic level

The introduction and overview level provides a description of the major components and the function and relationship of each component.

The functional and detailed logic level is provided in separate sections of the manual and contains a description of each of the major components and a detailed explanation of the operational characteristics of each component.

1.2 MANUAL ORGANIZATION

This section provides an overview of the VAX 8800 System Technical Description manual and an introduction to the VAX 8800 system and contains four chapters with the following information:

Chapter 1 Introduction to the manual and the overall VAX 8800 system. Contains a physical and functional description of the system and provides a list of reference documentation.

Chapter 2 System Control. Includes a description of the console software, system operating modes, and console/operator interaction. Provides an overview of the console commands, operator displays, and the console logfile.

Chapter 3 System Operation. Description of the CPU microcode including sequencing, format, parallel operations, and stalls and traps. Memory I/O transfers, and interrupts and exceptions.

Chapter 4 Diagnostic and Maintenance Aids. Discussion of available diagnostics, error logging, voltage margining, and remote diagnostic procedures.

The entire manual is divided into 10 sections as shown in Table 1-1.

Table 1-1 Technical Description Manual Organization

Section	Description
1	Introduction and overview of the VAX 8800 system and its major components.
2	System Bus Summary. Description of the VAX 8800 bus architecture and how the system components are interconnected.
3	Console Subsystem. Introduction and functional description of the console subsystem. Includes an explanation of how the console interacts with the VAX 8800 system.
4	Power System Complex. Description of the VAX 8800 power system including major components, controls and indicators, power modules, environmental monitoring, and distribution.
5	Clock Module. Functional description of the system clock generation logic.
6	IBox. Functional description of the instruction box logic, including microsequencer, writable control store, instruction buffer, and instruction stream decoder hardware.
7	EBox. Functional description of the execution box logic including slow data file, register file, main ALU, BCD ALU, and array processor.
8	CBox. Description of the cache box logic including translation buffer, microsequencer, and NMI write buffer.
9	MBox. Description of the VAX 8800 memory system logic including 4-megabyte memory array board, VAX 8800 array bus, and NMI signal functions.
10	NBI (NMI to VAXBI Adapter). Description of the VAX 8800 memory interconnect (NMI) to VAX bus interconnect (VAXBI) adapter including NBIA and NBIB modules, and VAXBI interface.

1.3 RELATED DOCUMENTATION

Table 1-2 provides a list of related documents containing additional information pertaining to the VAX 8800 system. The references listed in the table refer to documentation available at the system level only. Each section of this manual provides a list of related information for the specific area of the system if applicable.

Table 1-2 Related Documentation

Title	Document Number
VAX 8800 System Hardware User's Guide	EK-8800H-UG-001
VAX 8800 System Diagnostic User's Guide	EK-KA88D-UG-001
VAX 8800 System Maintenance Guide	EK-88XV1-MG-001
VAX 8800 System Installation Guide	EK-8800I-IN-001
VAX 8800 Site Preparation and Planning Guide	EK-8800P-SP-001
VAX 8800 Field Maintenance Print Set	

1.4 SYSTEM DESCRIPTION

The following paragraphs provide a general, physical, and functional description of the VAX 8800 system. The functional description is intended to provide an overview of the system and an explanation of how the major components described in the subsequent chapters interact with each other.

The VAX 8800 system is an LSI-based, high performance system adaptable to all VAX technical and commercial applications. The system supports the VAX/VMS operating system and is configured as either a single or dual-CPU system.

The dual-CPU system allows asymmetrical multiprocessing with the primary CPU performing all I/O transfers. The secondary CPU is assigned to perform compute-ready processes.

1.5 PHYSICAL DESCRIPTION

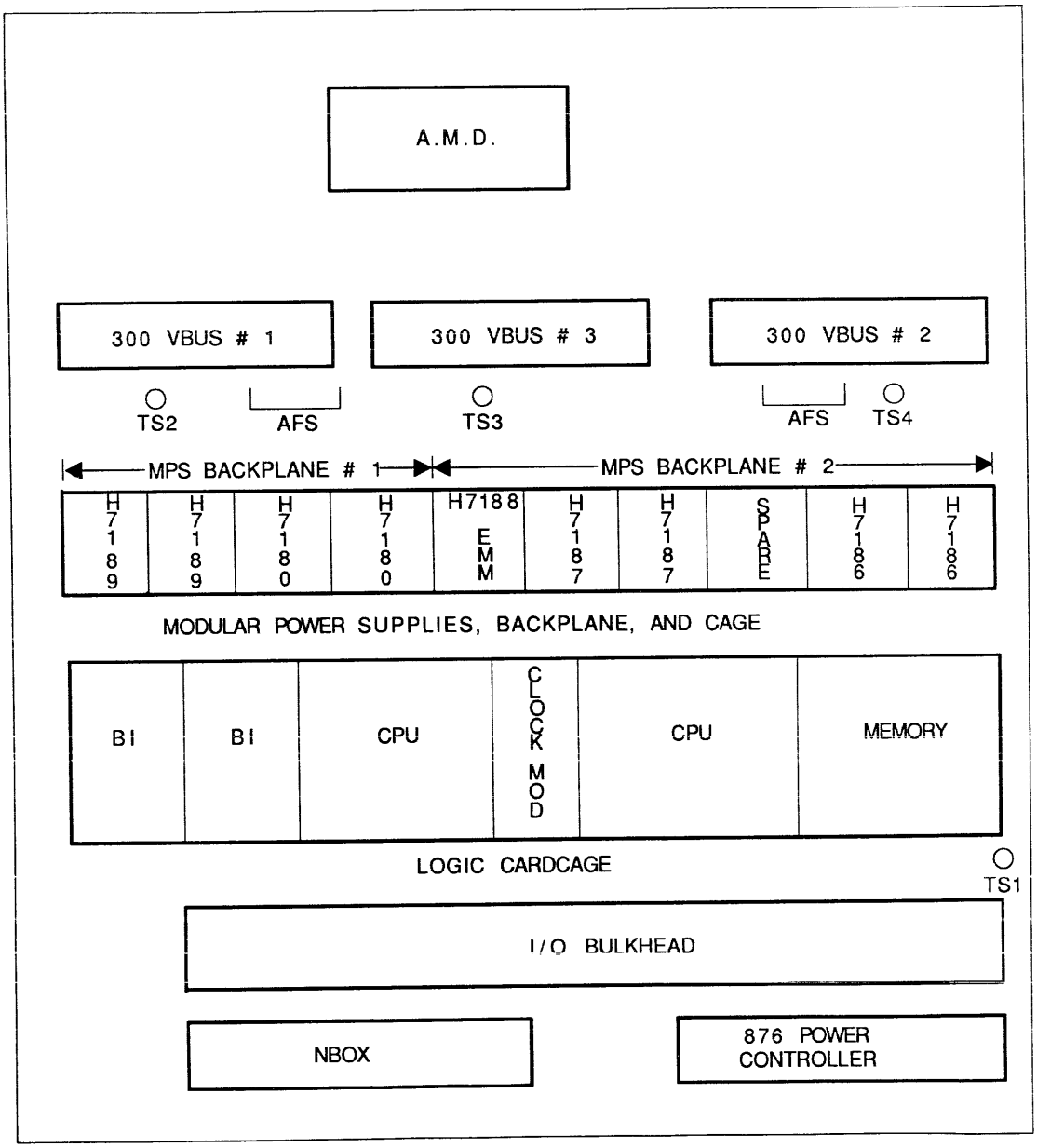
Figure 1-1 shows the cabinet layout of a dual-CPU VAX 8800 system and the location of the major components. Table 1-3 provides a list of the physical characteristics and applicable parameters for each characteristic.

The system is housed in an H9650 cabinet and contains the following:

- Module power system
- Cooling system
- Module cardcage (backplane)
- Input/output bulkhead

An H9652 expansion cabinet is used to house the power system ac input transformer and the system battery backup unit.

Figure 1-2 shows a layout of the VAX 8800 cardcage and Table 1-4 lists and identifies the modules contained in the cardcage. Table 1-5 identifies the modules used in the module power system (MPS).



NOTE: 1. TS = TEMPERATURE SENSE
2. AFS = AIR FLOW SENSE
3. A.M.D = AIR MOVING DEVICE

SCLD-74

Figure 1-1 VAX 8800 System Major Component Locations (Rear View)

Table 1-3 VAX 8800 System Physical Characteristics

Characteristics	Parameters
Cabinet Type	H9650
Cabinet Dimensions	
Width	46.5 inches
Height	61.5 inches
Depth	30.0 inches
Environmental	
Estimated Maximum Heat	6500 Watts
Maximum Temperature Rise	17 Degrees Celsius
Temperature Range	
Operational	10 to 40 Degrees Celsius
Nonoperational	-40 to 66 Degrees Celsius
Humidity	
Operational	10 to 90 percent relative humidity
Nonoperational	10 to 95 percent relative humidity
Cooling System	
Type	Air moving device
Drive	Three phase 208 Vac, 60 Hz/440 Vac, 50 Hz induction motor
Air Mover	Quad inlet, dual outlet centrifugal blower
Air Source	Filtered ambient air
Electrical	
Input Requirements	Three phase 208 Vac, 60 Hz/440 Vac, 50 Hz
Internally Generated	300 Vdc +/-5 Vdc +/-12 Vdc +10 Vdc -5.2 Vdc -2.0 Vdc

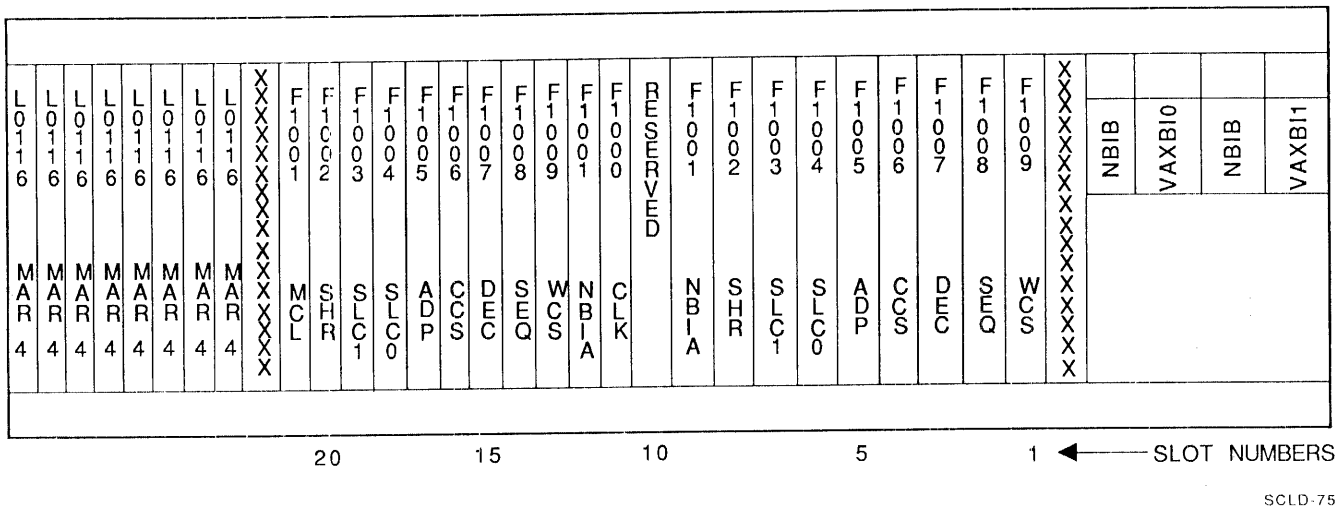


Figure 1-2 Cardcage Module Layout (Front View)

Table 1-4 Cabinet Module Identification

Slot	Module	Description	CPU
1	WCS	Writable Control Store	Right
2	SEQ	Sequencer	Right
3	DEC	Instruction Decoder	Right
4	CCS	Cache Control Sequencer	Right
5	ADP	Address Data Path	Right
6	SLC0	Data Slice 0	Right
7	SLC1	Data Slice 1	Right
8	SHR	Shifter	Right
9	NBIA I/O 1		Both
10		Reserved	
11	CLK	Clock and Console Interface	Both
12	NBIA I/O 2		Both
13	WCS	Writable Control Store	Left
14	SEQ	Sequencer	Left
15	DEC	Instruction Decoder	Left
16	CCS	Cache Control Sequencer	Left
17	ADP	Address Data Path	Left
18	SLC0	Data Slice 0	Left
19	SLC1	Data Slice 1	Left
20	SHR	Shifter	Left
21	MCL	Memory Controller	Both
	MAR4	4-Mbyte Memory Array	Both
	VAXBI0/ VAXBI1	VAX Bus Interconnect Options	Both

Table 1-5 Power Supply Identification

Power Supply	Description
MOD C H7186	+5.0 Vdc Supply
MOD B H7186	+5.0 Vdc Supply (Battery Backup)
MOD D H7187	-2.0 Vdc Supply
EMM	Environmental Monitoring Module
MOD E H7180	-5.2 Vdc Supply
MOD F H7189	+5.0, -2.0, +/-12, -5.2, +15 Vdc Supply
MOD H H7189	+5.0, -2.0, +/-12, -5.2, +15 Vdc Supply

1.6 FUNCTIONAL DESCRIPTION

The VAX 8800 system configuration as shown in Figure 1-3 consists of the console subsystem, a primary CPU, secondary CPU, memory subsystem with one to eight 4-Mbyte arrays, power system, and one or two VAX bus interconnect (VAXBI) adapters. The system is interconnected through a synchronous backplane bus called the VAX 8800 memory interconnect (NMI). The NMI provides the system with a communications path between the CPUs, memory, and the adapters that connect to the VAX bus interconnects.

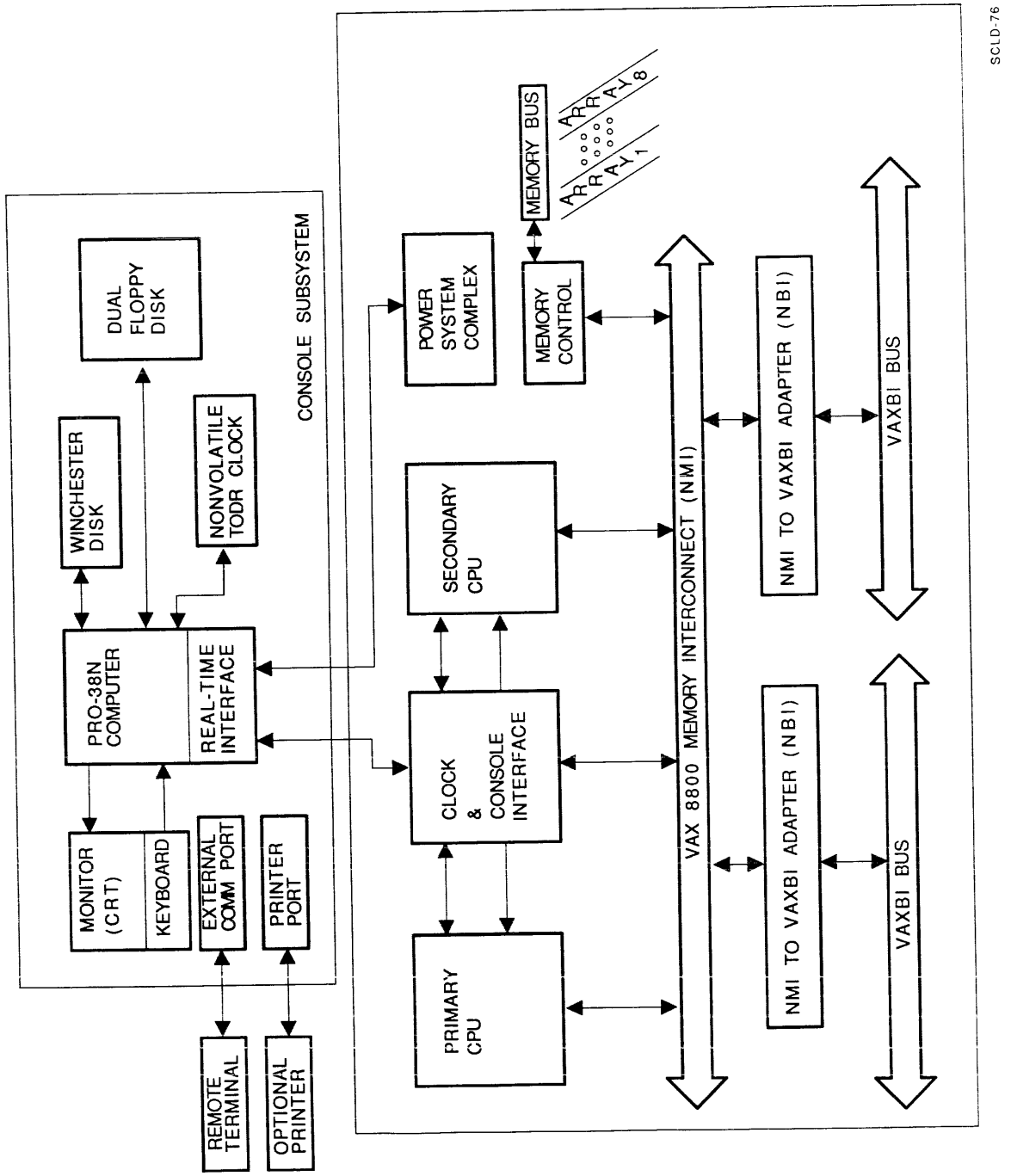


Figure 1-3 Simplified Block Diagram of the VAX 8800 System

1.6.1 Console Subsystem

The console subsystem shown in the simplified block diagram of Figure 1-4, controls the power sequencing, loading of control stores, and general operation of the VAX 8800 system. A Professional Series 380 Computer (PRO-38N) with a Winchester disk and dual floppy disks is used as a console device for the VAX 8800 system.

Communication between the console and either of the VAX 8800 CPUs is accomplished by transferring data to and from a console interface on the VAX 8800 clock module.

The console computer contains an I/O option called the real-time interface (RTI) that is installed in slot six of the six-slot PRO-38N VAX bus interconnect. The RTI provides a communications path between the PRO-38N and the console interface located on the clock module.

The RTI provides the console processor with several I/O ports including a programmable peripheral interface (PPI) and two serial line units (SLU).

The PPI contains three 8-bit ports for transferring data, address, and control signals between the console and the console interface. One of the SLUs is connected to the environmental monitoring module (EMM) in the power system complex, and provides the console with the ability to control and monitor the power and environmental parameters of the VAX 8800 system.

The other SLU has the RECEIVE DATA input and TRANSMIT DATA output connected to a spare connector located on the VAX 8800 bulkhead. This unused spare SLU connector could be used for simple data transfers to the PRO-38N, but it is not currently supported by the VAX 8800 console software.

The rear of the PRO-38N contains a serial printer port for connecting an optional printer to the console.

A remote diagnostic link can be established through an existing port on the rear of the PRO-38N that is configured for modem control.

Refer to Section 3 of this manual for detailed information concerning the console subsystem.

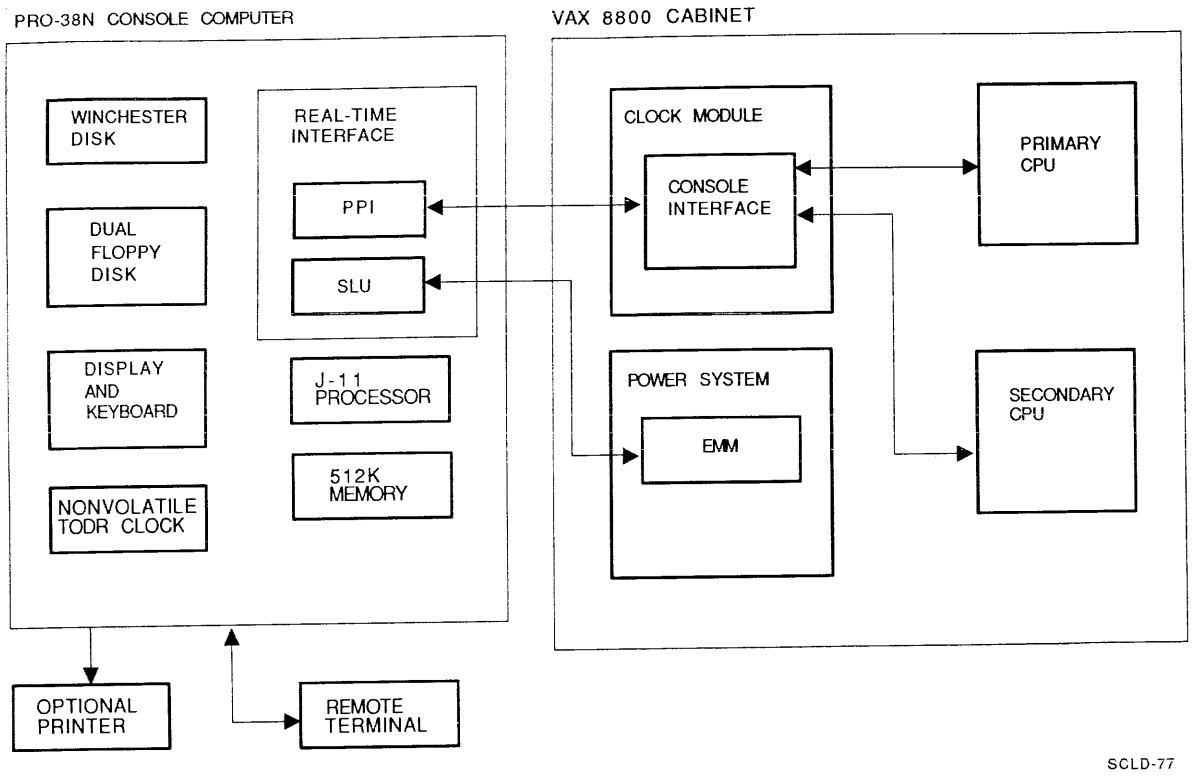


Figure 1-4 Simplified Block Diagram of the Console Subsystem

1.6.2 Central Processing Unit

The VAX 8800 processor shown in Figure 1-5 consists of three functional units and associated data transfer buses. The functional units and data buses are listed and described in Table 1-6. Refer to the following individual sections of this manual for additional information concerning the functional units of the CPU:

Functional Unit	Section
Instruction Box	5
Execution Box	6
Cache Box	7

Table 1-6 VAX 8800 Processor Functional Units/Data Bus Descriptions

Functional Unit/Bus	Description
Instruction Box	CPU microcode store and control. Consists of the writable control store (WCS), and the decoder and sequencer modules (DEC) (SEQ).
Execution Box	Processes data received from the cache and instruction boxes. Performs arithmetic, logical, and bit shift operations. The EBox consists of the data slice modules (SLC0 and SLC1), and the shifter (SHR) module.
Cache Box	Contains the cache, translation buffer, and the interface to memory and the I/O. The cache is a 64-KByte physical index, direct mapped and buffered write-through cache. The translation buffer is a 1K direct mapped cache of virtual-to-physical address translations. The CBox consists of the address data path (ADP) and cache control sequencer (CCS) modules.
Virtual Address Bus	Data path for transferring virtual address from the execution box to the cache.
Cache Data Bus	Data path from the cache to the execution box and instruction parser.
Write Data Bus	Data path for write data from the execution box to the cache.
Cache/ALU Bypass Bus	Bypass register data that is scheduled to be written into, but does not yet have valid data.
Visibility Bus	Slow speed data bus that allows the console to access internally latched data in the CPU modules.
Instruction Buffer Data Bus	Data path for transfer to the execution box. The data consists of byte, word, and longword address displacements, absolute addresses, and immediate data. Branch displacements and literals are also transferred over the instruction buffer data bus.

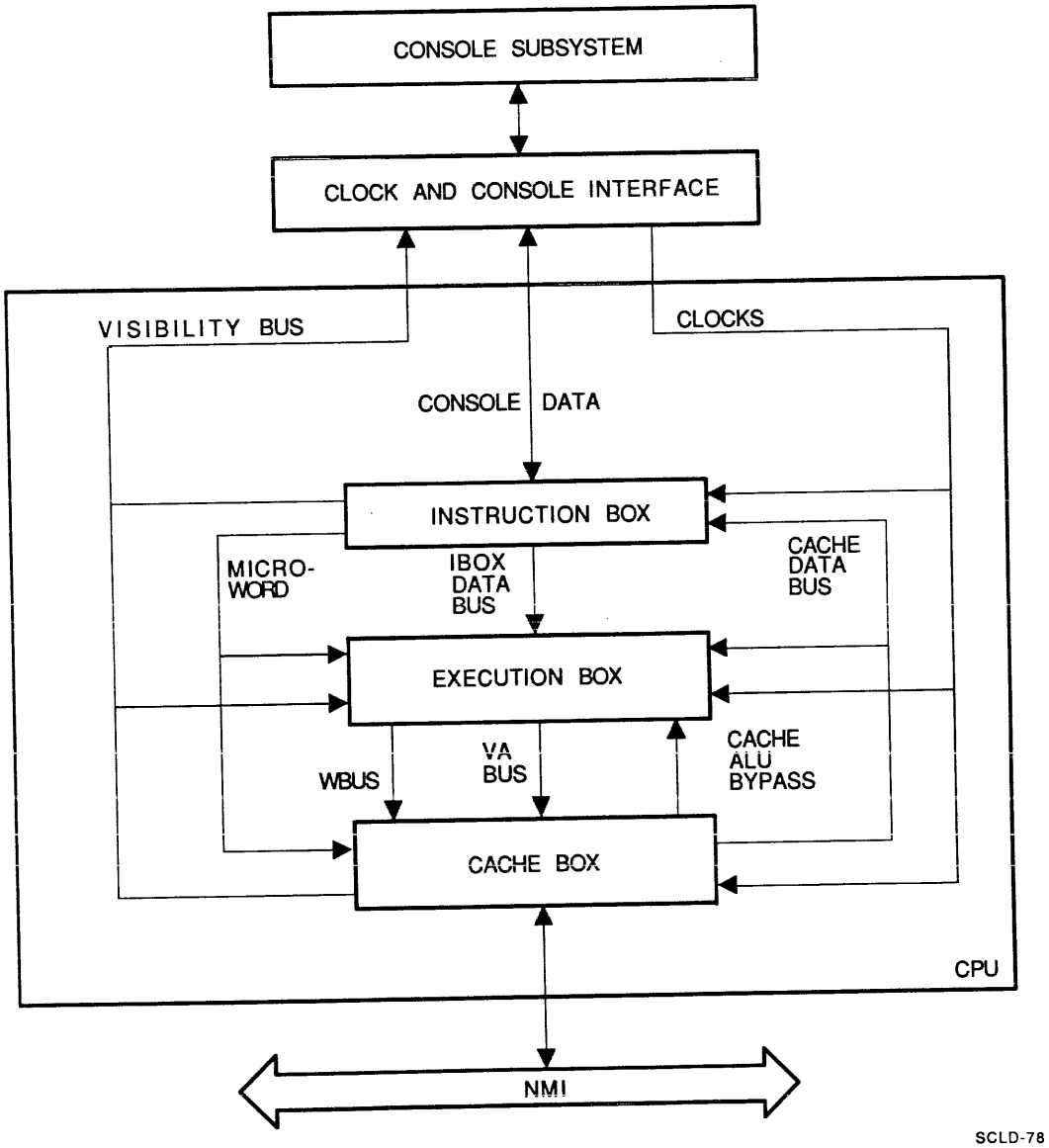


Figure 1-5 Simplified Block Diagram of Single CPU

1.6.2.1 Instruction Box -- The instruction box (IBox) shown in Figure 1-6 is the CPU's microcode store and control center. Major functions performed by the IBox include:

- Buffering the prefetched VAX instruction stream data received from the cache box.
- Decoding and controlling the execution of microinstructions.
- Monitoring and servicing microtraps, interrupts, and exceptions.
- Supplying instruction stream embedded data (literals, immediate data) to the execution box.
- Providing an interface between the clock module and the CPU.

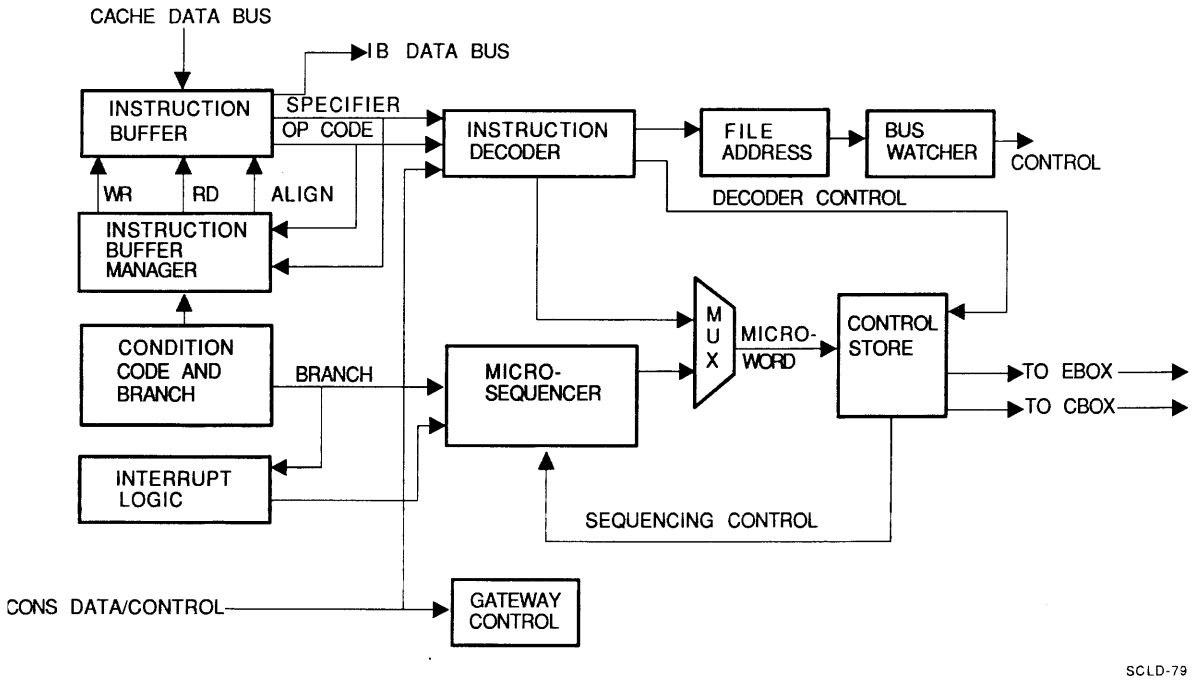


Figure 1-6 Simplified Block Diagram of the CPU IBox

The IBox resides on the DEC, SEQ, and WCS modules and consists of the following elements:

- Instruction buffer (IB)
- Decoder
- Microsequencer
- Control store
- Condition code and microbranch logic
- Interrupt and processor register logic
- File address generator
- Console gateway control

Instruction Buffer

The instruction buffer is a 4-longword, 16-byte memory that receives prefetched VAX I-stream data from the CBox. The instruction buffer outputs the following macroinstruction data:

- Op code byte
- Current operand specifier
- GPR number of the current specifier
- Specifier extension bytes

Instruction stream data enters the instruction buffer one longword at a time from the cache data bus. The data is loaded into a longword location that is indicated by the write address. An instruction buffer manager controls the read/write operations of the instruction buffer.

Decoder

The decoder consists of a 4K x 17-bit writable RAM (DRAM), and a special address encoder composed of discrete priority encoders and multiplexers.

The DRAMs are addressed by the current specifier number, the op code byte, and a 2-byte indicator signal. The specifier number and the 2-byte indicator signal are received from the instruction buffer manager, and the op code from the instruction buffer. The DRAMs perform the following functions:

- Supply the microsequencer with part of the entry point address for op code and specifier microroutines.

- Assist the instruction buffer manager in controlling the instruction buffer.

- Indicate which EBox memory data register is to receive the data from memory for those specifiers requesting data.

Microsequencer Logic

The microsequencer logic determines which one of several possible sources is to supply the control store RAMs with the address of the next microword to be executed. The next possible microwords to be executed include:

- Current microword
- Decoder entry-point microaddress
- EBox or CBox microtrap vector
- Machine check microtrap vector
- Trapped microPC from a microPC silo
- Microsubroutine return address from a microstack break
- Console supplied address

The 14 bit wide selected address is stored in microPC latches and presented to the control store.

Control Store

The microcode control store of 16K x 143 bits, resides in a set of 16K x 1-bit writable RAMs. The RAMs are loaded from the console by means of the gateway controller during system initialization. Approximate RAM usage consists of:

Amount	Use
15K	CPU Control
1K	User-Written Code

Condition Code and Macrobranch Logic

The condition code and macrobranch logic are responsible for maintaining the condition code bits of the processor status longword, and seven CPU state bits. Raw condition codes from various EBox operations are used in the generation of microbranch conditions based on the size of the data being processed and the raw condition codes. The raw condition codes can be compared with the current longword condition code bits to create a macrobranch instruction, or stored as the new longword condition bits.

The CPU state bits are microprogramming aids that provide firmware writers with a method of controlling microcode flow. The bits can be set or cleared in a microroutine and then tested as conditions in later routines.

Interrupt and Processor Logic

The interrupt and processor logic contains the priority interrupt hardware and the four internal processor registers (IPRs). The interrupt portion of the logic monitors all hardware interrupts, encodes the level of the highest pending request, and compares it to the current priority level. If the encoded level is higher than the current level, the interrupt logic will request an interrupt by asserting an interrupt pending line.

The internal processor registers control and supply data to the interrupt logic, microsequencer, and the memory management logic in the CBox.

File Address Generator

The file address generator performs the following functions:

- Supplies addressing for the EBox register file and slow data file
- Stores general-purpose register numbers referenced by operand specifiers
- Records changes made to general-purpose registers in auto-increment/decrement operations

Gateway Control Logic

The gateway control (GWC) controls the data paths between the CPU and the console interface. The GWC controls the loading of the control store RAMs and micromatch register, and the loading of the decoder and cache control RAMs.

1.6.2.2 Execution Box -- The execution box shown in Figure 1-7 processes data received from the CBox and the IBox, and returns the processed data along with the virtual address to the CBox. Functions performed by the EBox include:

- All CPU required arithmetic, logical, and bit shift operations.
- Maintaining the program counter and general-purpose registers.
- Maintaining the internal processor registers.
- Controlling data transfers between the CBox, IBox, and clock module registers.
- Providing condition code information to the IBox microsequencer.

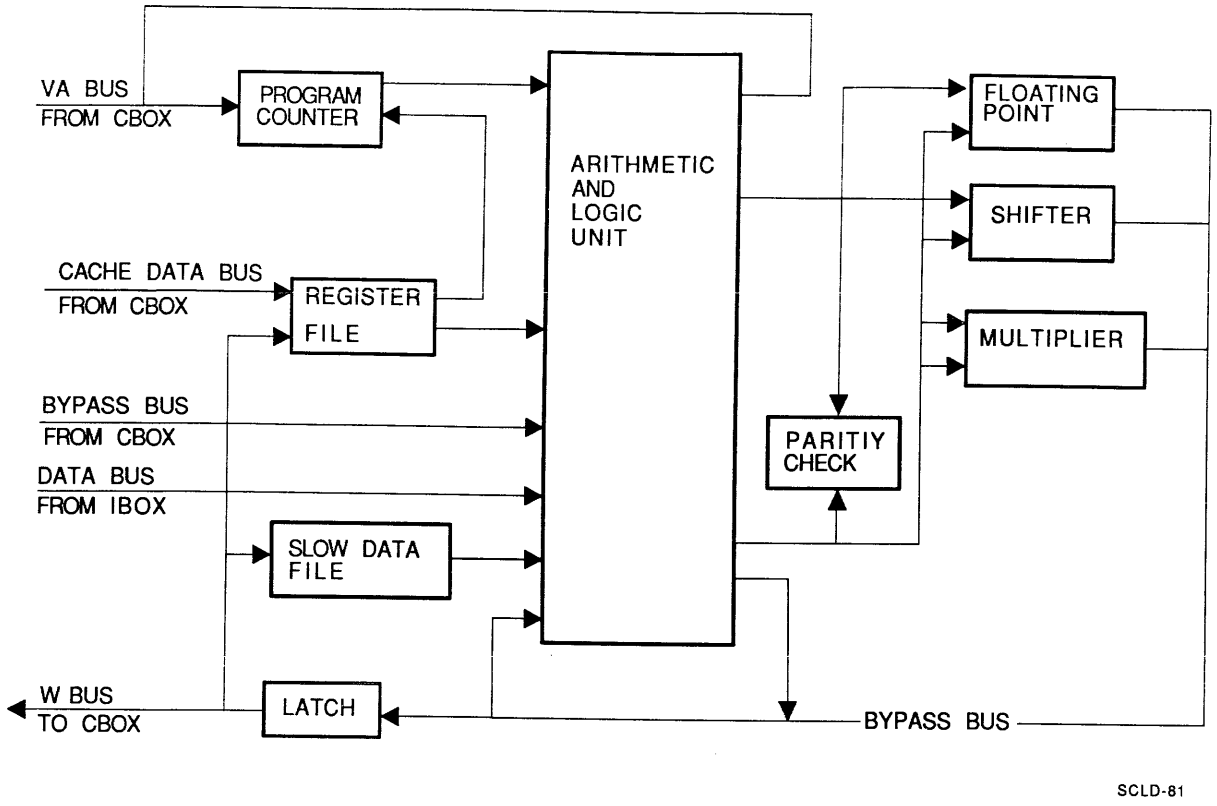


Figure 1-7 Simplified Block Diagram of the CPU Execution Box

The EBox consists of the following major elements:

- Register file
- Slow data file
- Program counter logic
- Main arithmetic and logic unit
- Shifter
- Floating-point support logic
- Multiplier

The major elements that make up the EBox are located on the data slice modules (SLC0 and SLC1), the shifter (SHR), and part of the CBox decoder (DEC) module.

Register File

The register file consists of 32 high-speed 36-bit registers (32 bits of data and 4 parity bits). The 32 registers in the register file include:

- 15 general-purpose registers (GPR)
- 9 temporary registers (TEMPS) microcode scratchpad registers
- 8 memory data registers (MDR) store data received from the cache

Slow Data File

The slow data file has a timing restraint that inhibits access to the file for three cycles following a write operation. The file consists of 256 low data rate 36-bit registers (32 bits of data and 4 parity bits). Slow data file registers consist of:

- VAX internal processor registers (IPR)
- Data path constants
- Diagnostic test patterns

Program Counter

The program counter (PC) maintains the VAX PC, PC incrementer, backup and trap PCs, and the virtual address file (VAF) register.

VAX PC

The VAX PC supplies the CBox translation buffer with the virtual address for each op code, operand, and operand specifier of the instruction stream.

PC Incrementer

The PC incrementer updates the PC by adding an increment value equal to the size of I-stream data being processed. The increment value (0-6) is supplied by the PC increment generator in the IBox.

Backup PC

The backup PC saves macroinstruction op codes and restores the PC if the instruction results in a macroexception. Saving the op code allows a service routine to examine the op code of a failing instruction and service the fault.

Trap PC

The trap PC maintains a history of recent PC activity, and provides microtrap service routines with the active PC at the time a microtrap occurs.

Virtual Address File

A copy of each virtual address sent to the CBox is saved in the virtual address file. This copy is used as a backup if the address causes a microtrap.

Main Arithmetic and Logic Unit

The arithmetic and logic unit is a 32-bit adder that processes integer, floating-point, and binary coded decimal data to perform the following functions:

- Addition and subtraction (carries propagated)
- Logical AND, OR, and exclusive OR

Auxiliary functions performed include:

- Multiplexing data received by the data slice modules
- Supplying memory and register data to the CBox
- Supplying virtual address to the CBox translation buffer
- Routing data to and from the shift register module
- Providing carry and condition codes to the IBox

Shifter

The shifter is a 64-bit input, 32-bit output shift matrix that handles data in all formats in one of the following three modes of operation:

- Integer
- Floating-point
- Binary coded decimal

Floating-Point

This support system processes the sign and exponent fields of floating-point data. Included in the floating-point support logic are: a shift count ALU, priority encoder, and an exponent ALU.

Multiplier

The multiplier is a 64-bit multiplier that enhances the speed of integer and floating-point multiplication. Characteristics of the multiplier are:

- Incorporates an "eight bit at a time" multiply algorithm that generates eight result bits per cycle.
- Produces or generates the correct two's complement results for integer data.
- Incorporates a "one bit at a time" division algorithm that generates 1-quotient bits per cycle.

1.6.2.3 Cache Box -- The cache box (CBox) shown in Figure 1-8, is a 64K byte physical indexed, direct mapped, and buffered write-through cache that speeds address translations and provides a communication path for the CPU to the NMI.

The CBox consists of a translation buffer, NMI interface, and the 64K byte data store cache.

Translation Buffer

The translation buffer (TB) is a 1K direct mapped cache of virtual to physical address translations. The TB consists of a tag store and a data store. The TB is organized into 512 per process translations and 512 system region translations.

The tag store uses a portion of the virtual address (VA) to access a RAM array and compares the contents of the RAM with the remaining VA bits. When the comparison results are equal and the TB valid bit is set, the address has "HIT" and the contents of the Data Store will be valid for that address.

The data store uses the page frame number (PFN) of the page table entry (PTE) for the virtual address. If the tag store comparison results in a TB HIT, the PFN concatenated with VA<8:0> is used as the physical address.

Cache

The cache is a hardware mechanism used to provide fast access to frequently used data, and is addressed by a physical address. If required read data is available in the cache, the data is extracted and a memory request is not required. If the data is not available in the cache, a memory request is initiated to provide the data. The data from memory is passed on to the requester and is also placed in the cache for subsequent use.

NMI Interface

The NMI interface provides the CPU with a control and data path for communication with the VAX 8800 memory interconnect (NMI). When a cache read MISSES (no comparison), the interface uses the missed address to build a command/address transaction to send to memory. This allows the translation buffer and the cache to be free to process additional CPU requests until the requested data arrives from memory. When the requested data arrives from memory, the interface assumes control of the cache and loads the new data into the cache data store.

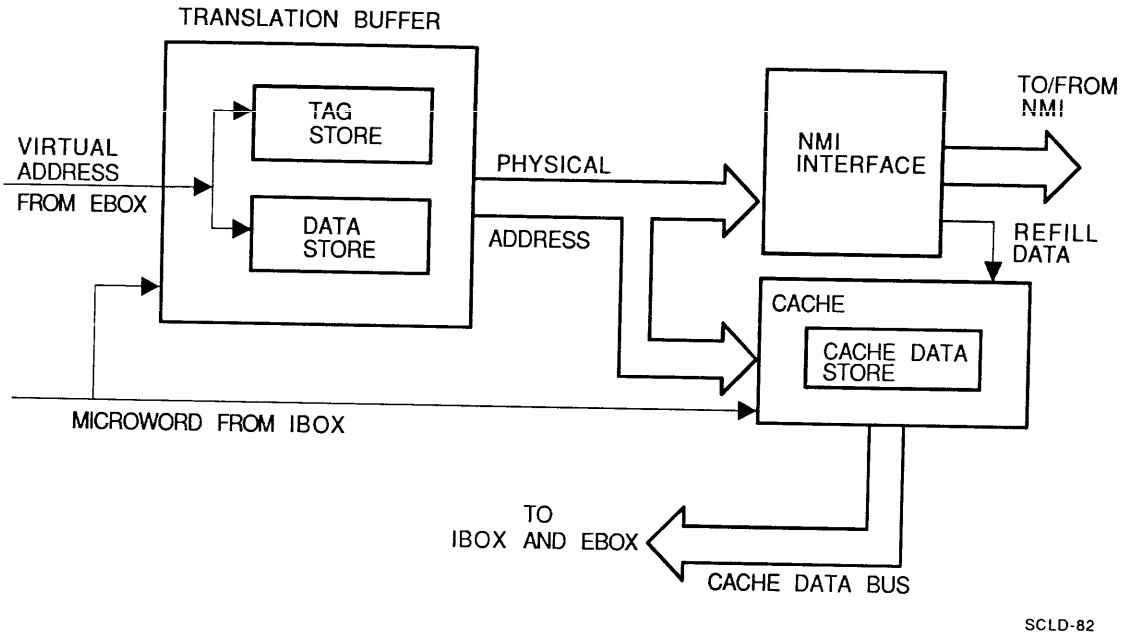


Figure 1-8 Simplified Block Diagram of the CBox

1.6.3 Clock Module

The clock module contains all of the hardware necessary to generate, control, and distribute the VAX 8800 system timing to all system modules. Included on the clock module is the interface between the console and the rest of the VAX 8800 system. Refer to Section 4 of this manual for detailed information pertaining to the clock logic.

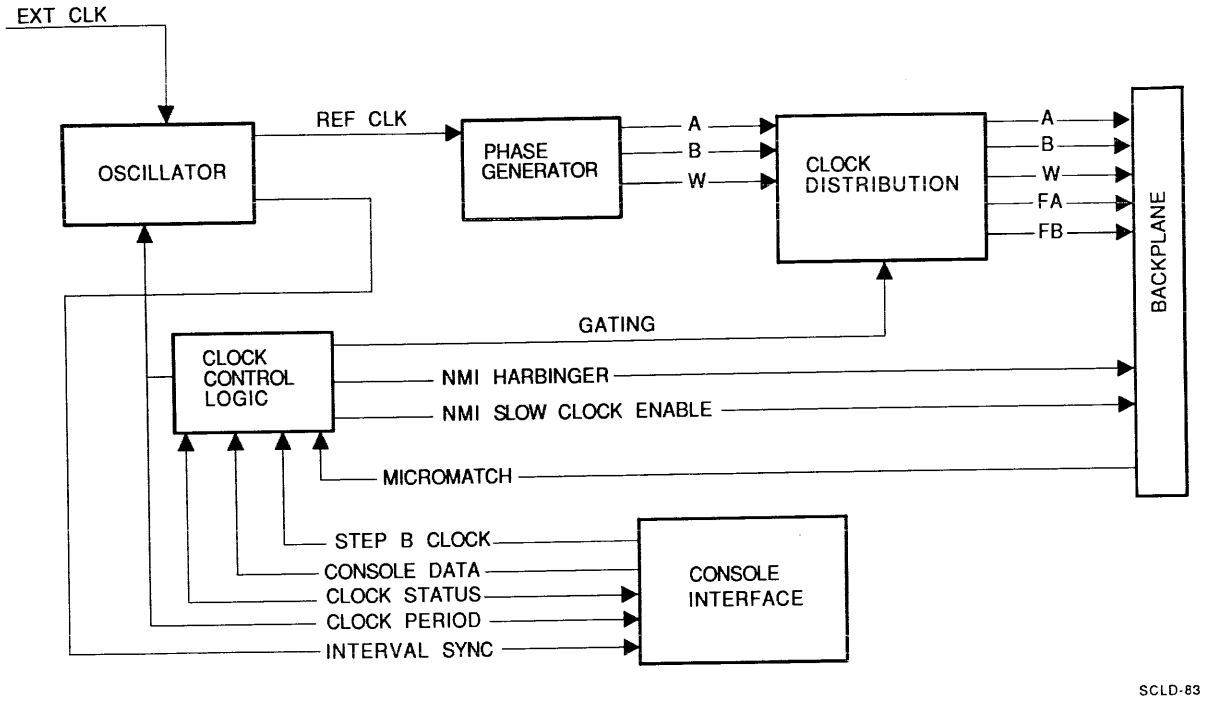


Figure 1-9 Simplified Block Diagram of the Clock Module

The basic timing source of the clock module is a 250 kHz oscillator that generates a reference signal used by the phase generator to produce two nonoverlapping clock phases.

The phase generator output consists of Phase A, B, and W. The W phase signal is similar to the A phase, but longer in duration. The A and B clock phases are gated in the distribution logic by signals from the clock control logic to produce gated system clocks that can be started, stopped, or burst by the clock control logic.

The distribution logic also produces free-running (ungated) clocks that are used in the VAX 8800 system. Table 1-7 lists and identifies the clock signals generated and distributed by the clock module.

Table 1-7 System Clocks

Clock Signal	Function
A and B CLK	Main system clocks used by the CPU modules and the I/O adapters to sequence and synchronize operations.
W CLK	Free-running system clock used in RAM write operations of CPU modules SLC0 and SLC1.
FA and FB CLK	Free-running system clocks used by the memory controller and I/O adapters to sequence and synchronize operations. Also used by the console interface to control and monitor the CPU when clocks are stopped.

The console subsystem controls the clock generation logic with operator initiated commands by means of the console interface. Three registers within the clock logic are used for control of the system clocks by the console. A fourth register provides status information for the console operator. The registers used are:

- Clock control
- Clock period
- Burst count
- Timeout and status

Console control of the clocks allows the system operator to:

- Start and stop the clocks
- Burst the clocks
- Single step the clocks
- Enable a clock stop on a micromatch
- Change the clock period
- Disable clock stalls
- Control the NMI timeout clock (NMI slow clock enable)

1.6.4 Memory (MBox)

The VAX 8800 memory system (MBox) consists of a memory control logic (MCL) module located in slot 21 of the cardcage, and one to eight MAR4 4-Mbyte memory array boards. The array boards are located in slots 22 through 29 of the cardcage. Figure 1-10 shows a simplified block diagram of the MBox with one MAR4 array board. Refer to Section 8 of this manual for additional information pertaining to the MBox.

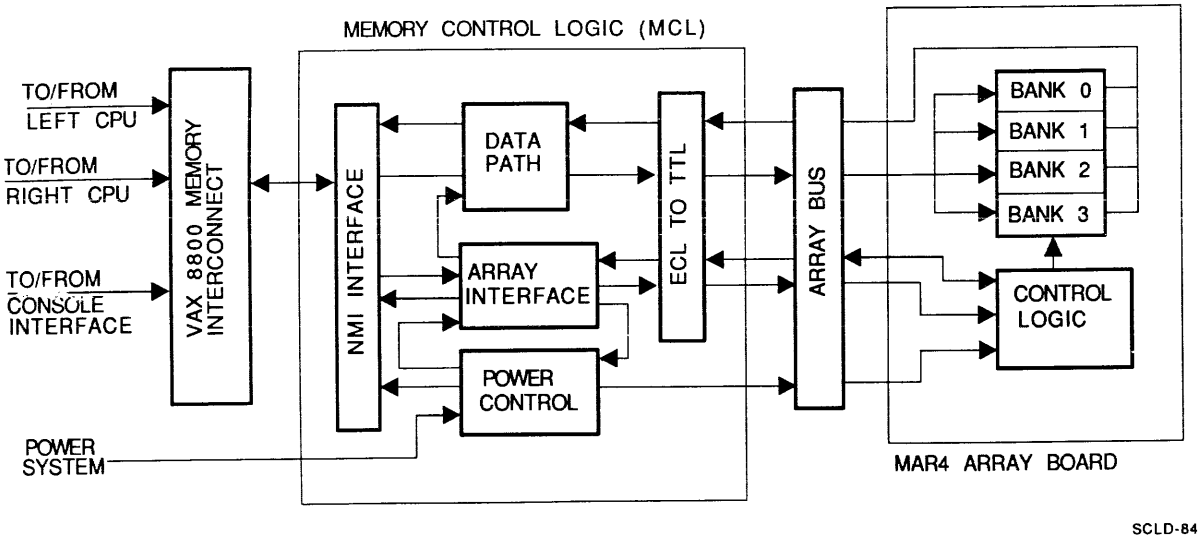


Figure 1-10 Simplified Block Diagram of the MBox

1.6.4.1 Memory Control Logic -- The MCL provides control and a communications interface between the NMI and the memory array boards. The single MCL has the capability to control up to eight array boards, and can monitor operations on three arrays simultaneously. Table 1-8 describes the command operations that the MCL performs on the MAR4 array.

Table 1-8 MCL Command Operations

Operation	Description
Longword Write	Writes a data longword and seven ECC check bits into one of the MAR4 array banks.
Longword Read	Reads a data longword and seven ECC check bits from one of the MAR4 array banks.
Octaword Read	Reads four data longwords and associated check bits from a MAR4 array.

MAR4 selection is accomplished by asserting a board select line and enabling the control logic on the appropriate array board. The MCL performs periodic memory refresh and distributes battery backup power from the power system during power interruptions.

1.6.4.2 MAR4 Memory Array -- The MAR4 array board consists of four 1-Mbyte array banks for a total of 4 Mbytes of memory for each MAR4. The array banks contain a 39 bit wide common I/O longword and seven error correction code (ECC) check bits. Data transfer and interfacing between the memory control logic and the array boards is accomplished by means of the VAX 8800 array bus (NAB).

1.6.5 System Buses

The VAX 8800 has three system level buses that provide a path for data transfers and status information.

- VAX 8800 memory interconnect
- VAX bus interconnect
- Visibility bus

The VAX 8800 memory interconnect (NMI) is a backplane bus that interconnects the major system components and allows transfer of data between the connected units.

The VAX bus interconnect (VAXBI) is the system I/O bus that provides a connection point between the VAX 8800 system and external device adapters.

The visibility bus provides the console operator with diagnostic access to internal latch contents. The VBus is also used to perform error checks during initialization.

1.6.5.1 VAX 8800 Memory Interconnect (NMI) -- The NMI shown in Figure 1-11 is a synchronous backplane bus that interconnects the following major components of the VAX 8800 system:

- Primary CPU
- Secondary CPU
- Memory system controller
- NMI to VAXBI adapters (NBI)

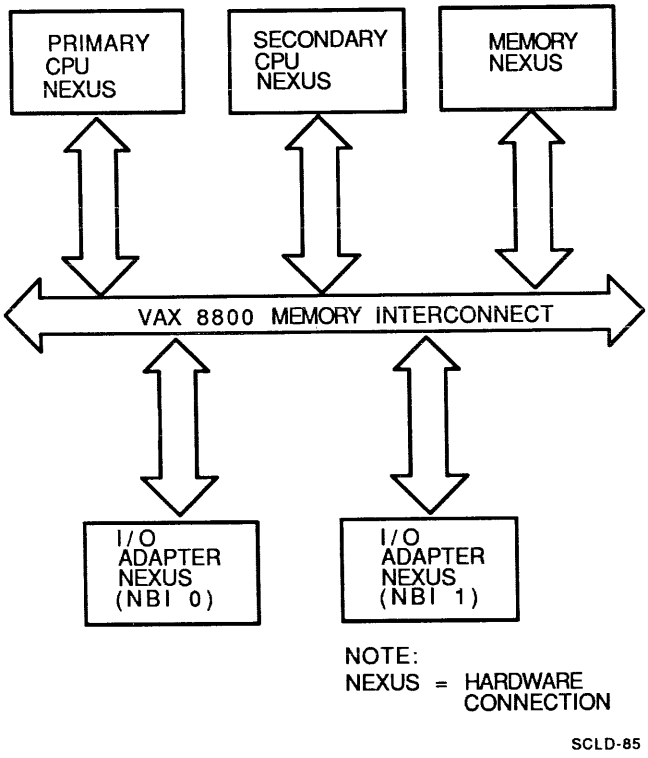


Figure 1-11 Simplified Block Diagram of the VAX 8800 Memory Interconnect

Table 1-9 lists the primary functions performed by the NMI, and provides a brief description of each of them. Refer to Section 2 of this manual for detailed information concerning the NMI.

Table 1-9 NMI Function Descriptions

Function	Description
Write Transactions	Supports longword, quadword, and octaword write.
Read Transactions	Supports longword, octaword, and hexword read.
Memory Read/Write Operations	Allows the CPUs and I/O adapters to access memory through bus read/write transactions.
I/O Register Read/ Write Operations	Allows the primary CPU to access I/O registers in the memory, I/O adapters, and I/O devices through bus read/write transactions.
Interrupt Handling	Transmits interrupt requests generated by the memory and I/O adapters to the primary CPU.
System Synchronization	Provides system clocks to all nexus (hardware blocks physically connected to the NMI).
System Initialization	Allows the console to initialize all nexus.
Power-Fail Warning	Provides ACLO and DCLO signals to all nexus.

1.6.5.2 VAX Bus Interconnect (VAXBI) -- The VAX bus interface (VAXBI) is the I/O bus for VAX 8800 and is connected to the system through the NMI to VAXBI (NBI) adapter as shown in Figure 1-12. Each NBI adapter can interface up to two VAX bus interconnects and provides for a maximum of four VAXBIs to be connected to the system when two NBI adapters are installed.

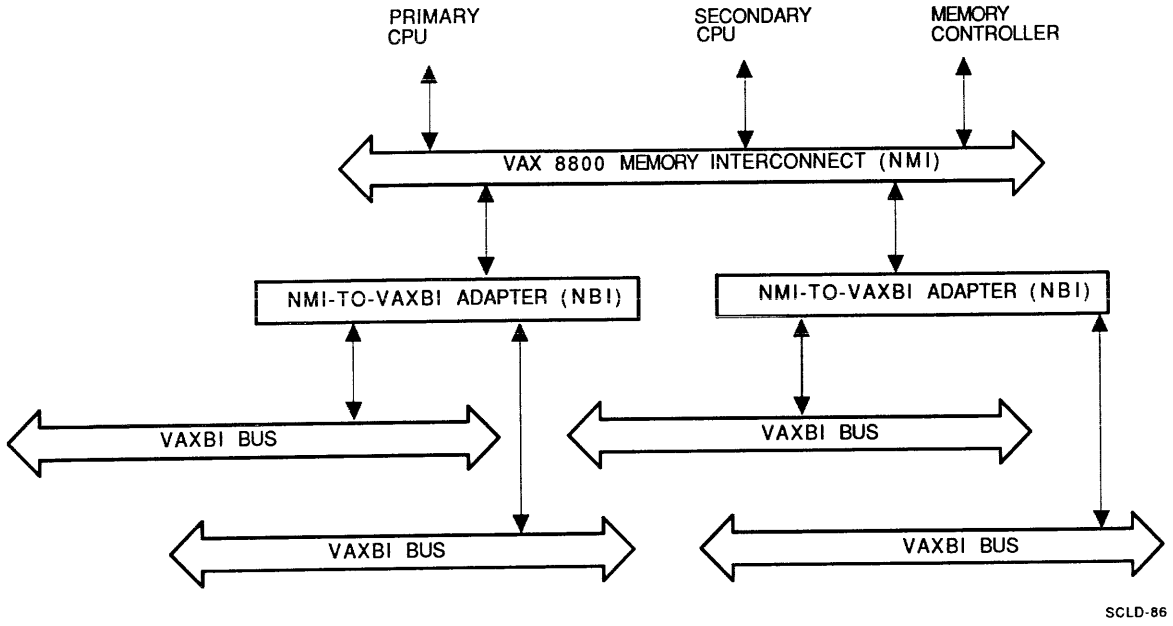


Figure 1-12 Simplified Diagram of VAX Bus Interconnect (Maximum Configuration)

The VAXBI is a 32 bit wide synchronous bus that interconnects up to 16 VAXBI interfaces (VAXBI nodes) with logical addresses of 0 through 15. The address of a node is determined by an ID plug inserted on the backplane. Address 0 is reserved for the NBI node.

The remaining nodes consist of I/O device controllers, and bus adapters interfacing the system's I/O devices to the VAXBI. A description of some of the different types of bus adapters can be found in Section 1.6.6.

Table 1-10 identifies and describes the basic functions performed by the VAX bus interconnect.

Table 1-10 VAXBI Function Descriptions

Function	Description
Memory Read/Write Operations	Allows DMA transfers between an I/O device on the VAXBI and the VAX 8800 main memory through bus read/write transactions.
I/O Register Read/ Write Operations	Allows the primary CPU to access I/O registers in the I/O devices on the VAXBI through bus read/write transactions originated on the NBI.
Interrupt Handling	Enables I/O devices on the VAXBI to interrupt the primary CPU through bus INTR transactions directed to the NBI node.
System Synchronization	Provides NBI generated clocks to all nodes connected to the VAXBI.
System	Allows nodes to assert a reset line, and initialize a simulated VAXBI power-fail sequence generated by the NBI.
Power-Fail Warning	Provides ACLO and DCLO signals to all nodes.

1.6.5.3 Visibility Bus (VBus) -- The visibility bus is a slow speed bus consisting of sixteen data lines and two control lines. The VBus allows the console operator to read internally latched data in the VAX 8800 CPU modules during the execution of microdiagnostics and system initialization. The VBus is used when the system clocks are stopped. Major functions performed by the VBus include:

- Monitoring the state of the CPUs during the execution of microdiagnostics or in response to commands entered at the console during system debug.
- Verifying CPU module installation and revision during system initialization.
- Ensuring that control store parity errors do not occur when loading microcode during system initialization.

The console controls and reads the VBus by means of two registers located on the clock module's console interface. The VBus control and access registers perform the following functions:

- Select the VBus input channel
- Step the clocks that operate the VBus
- Send serial VBus addresses to the CPU modules
- Halt the operation of the VBus address shift register

1.6.6 VAX Bus Interconnect and I/O Adapters

The VAX 8800 memory interconnect-to-VAXBI interconnect (NBI) adapters provide a connection point for the VAX 8800 CPUs to the backplane. Additional optional adapters can be connected to the backplane as shown in Figure 1-13 to allow for connection of the CPUs to other I/O devices. Table 1-11 lists and identifies some of the optional adapters that can be connected to the VAXBI.

Table 1-11 Optional VAX Bus Interconnect Adapters

Adapter	Function
DWBUA-CA	VAXBI Bus-to-UNIBUS controller. Requires an expansion cabinet and uses existing UNIBUS adapters.
KDB50-BA	VAXBI Bus-to-Storage Interconnect disk controller.
BSA	VAXBI Bus-to-Storage Interconnect disk and magnetic tape controller.
DEBNT-AA	VAXBI BUS-to-Network Interconnect controller. Uses Ethernet.
CIBCA-AA	VAXBI Bus-to-Computer Interconnect controller.
CIBCI-CA	VAXBI Bus-to-Computer Interconnect controller.
KLESI-AA	VAXBI Bus-to-Native tape controller.
DMB32-LF	VAXBI Bus-to-Native asynchronous comm. controller.
DRB32-AA	VAXBI Bus-to-Multifunction controller.

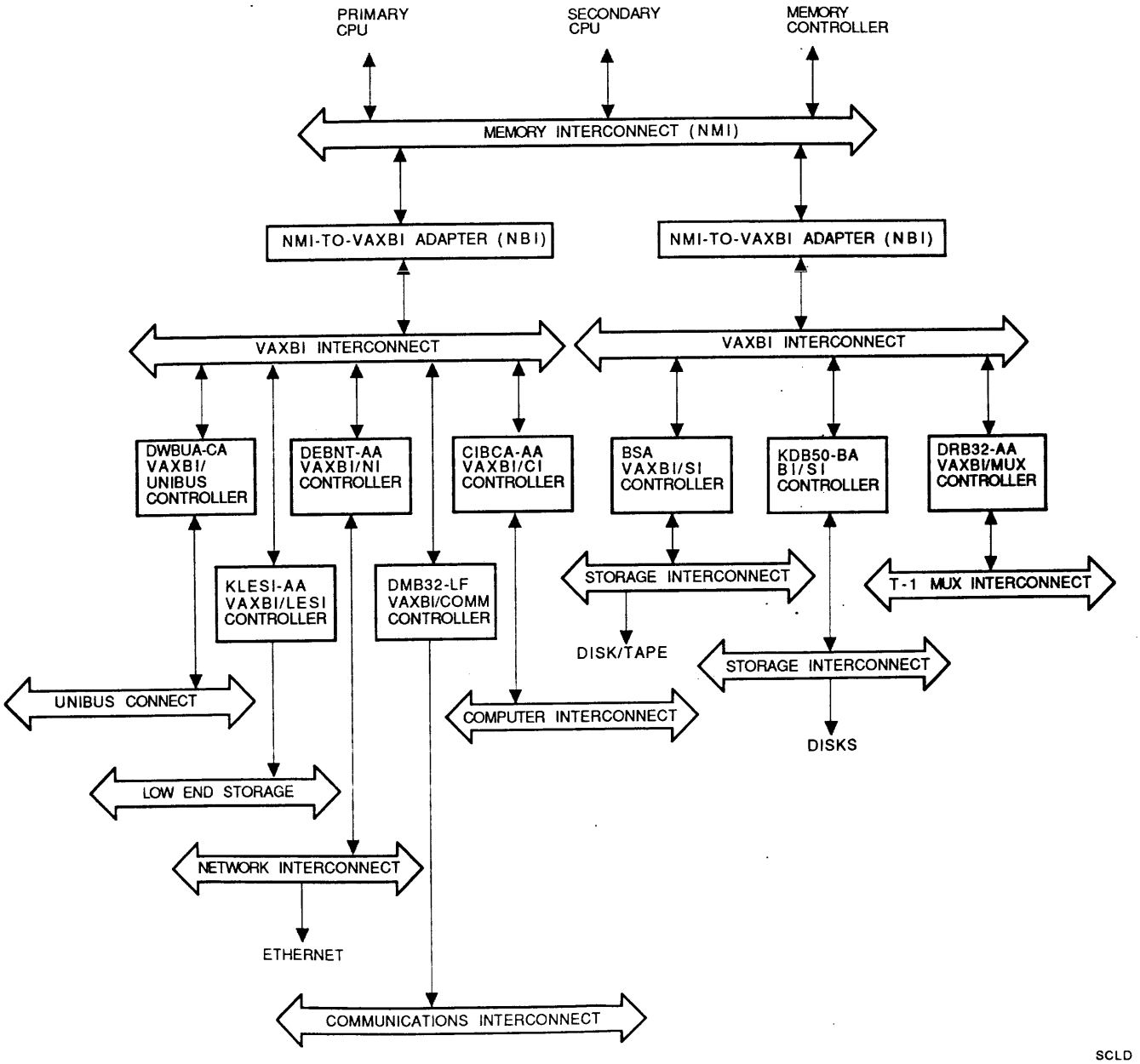


Figure 1-13 I/O Interconnect and Adapters

The NMI-to-VAXBI (NBI) adapter shown in Figure 1-14 consists of an NBIA module that interfaces to the NMI, and one or two NBIB modules. The NBIB modules interface to the VAXBI through the use of a VAXBI interface chip (BIIC).

The NBIA module contains the NMI nexus registers, and the NBIB module contains the VAXBI connecting registers. Tables 1-12 and 1-13 list and identify the registers on each module.

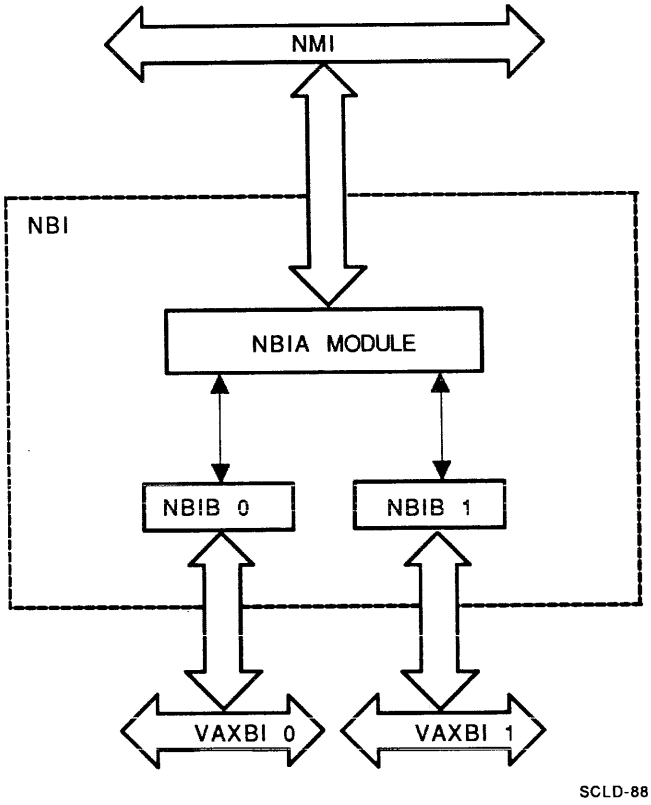


Figure 1-14 NMI-to-VAXBI Adapter

Table 1-12 NBIA Registers

Address	Register
2X080000	Control/Status 0 (CSR0)
2X080004	Control/Status 1 (CSR1)
2X080008	VAXBI 0 Stop Register (BI0I)
2X08000C	VAXBI 1 Stop Register (BI1I)
2X080010	BR4 Vector Offset (BR4VR)
2X080014	BR5 Vector Offset (BR5VR)
2X080018	BR6 Vector Offset (BR6VR)
2X08001C	BR7 Vector Offset (BR7VR)

X = 0 = NBIA 0
4 = NBIA 1

Table 1-13 NBIB Registers

Address	Register
2X000000	Device Type
2X000004	VAXBI Control/Status
2X000008	Bus Error
2X00000C	Error Interrupt Control
2X000010	Interrupt Destination
2X000014	IP Interrupt Mask
2X000018	IP Interrupt Destination
2X00001C	IP Interrupt Source
2X000020	Starting Address
2X000024	Ending Address
2X000028	BCI Control
2X00002C	Write Status
2X000040	User Interrupt Control

X = 0 = NBIA 0/BI 0
2 = NBIA 0/BI 1
4 = NBIA 1/BI 0
6 = NBIA 1/BI 1

1.6.7 Power System Complex

The power system complex provides the voltages necessary to operate the CPUs, memory, and the VAX bus interconnect of the VAX 8800 system. Three phase ac utility power is used as the primary source for the system. The required ac and dc voltages are developed using the power modules and voltage regulators located within the VAX 8800 cabinet. Figure 1-15 shows a simplified block diagram of the power system complex. Refer to Section 10 of this manual for detailed information concerning the power system complex.

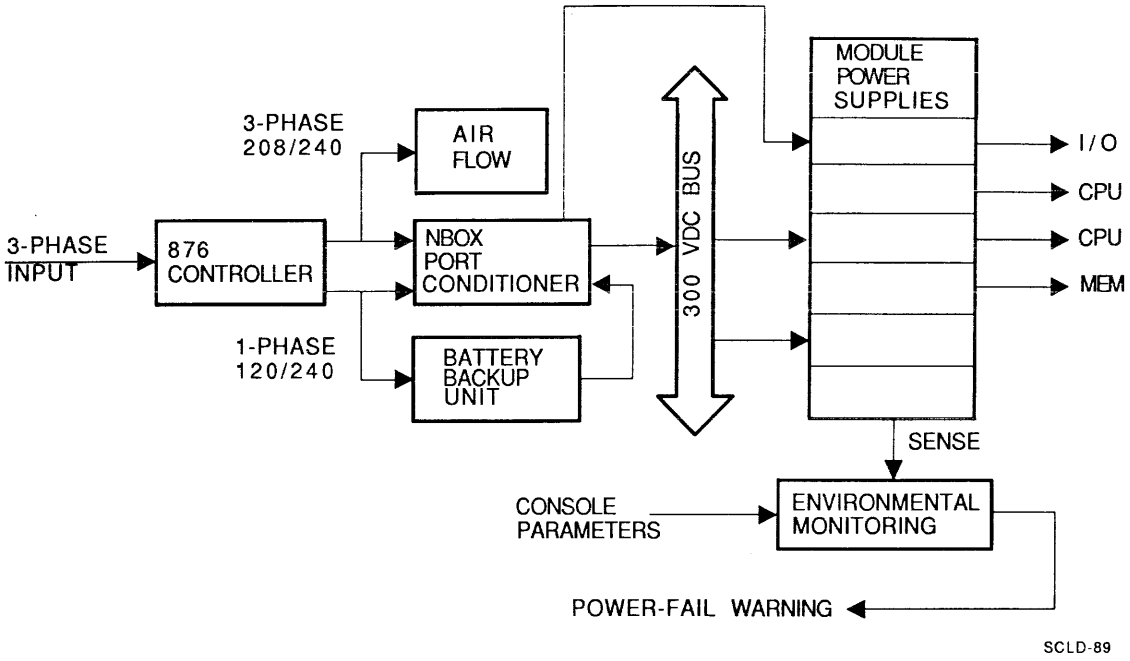


Figure 1-15 Simplified Block Diagram of the Power System Complex

1.6.7.1 876 Power Controller -- The 876 controller is the main ac input module for the power system. Power is received from the main circuit breaker and distributed to the other system components by the 876 controller. The controller distributes the following power:

Unit	Power
NBox	Single and three phase
BBU	Unswitched single phase
Console	Unswitched single phase
Air Mover	Unswitched three phase

1.6.7.2 NBox Port Conditioner -- The NBox is a multifunction power assembly containing five modules. Table 1-14 identifies the modules and describes the function of each module.

Table 1-14 NBox Modules

Module	Function
H7170 X	Convert three-phase ac power into 300 Vdc output.
H7170 Y	Same as H7170 X.
Interface Logic Module (ILM)	Provide logic signal interface between EMM and other power system components. Controls BBU operation.
Control and Startup Power (CSP)	Convert single-phase ac power to logic level voltages. Supplies: +5, +12, -12, and +10.5 for the EMM and ILM.
New Box Translator (NBT)	Converts logic signals for startup and initialization.

1.6.7.3 Module Power Supplies -- The module power supplies (MPS) are a group of dc power modules and backplane located above the CPU cardcage in the main CPU cabinet. The MPS contains the regulated dc power supplies that provide the operating power for the CPU, memory, and extender modules.

1.6.7.4 Environmental Monitoring Module -- The environmental monitoring module (EMM) is a microprocessor-based unit that monitors the power and environmental conditions within the VAX 8800 system. The EMM responds to console control commands during power-up and power-down sequencing, initialization, and battery backup operations. The console controls the power system through the EMM.

1.6.7.5 Battery Backup Unit -- The battery backup unit (BBU) gives the power system a method of providing protection voltage to the main CPU memory during an ac power failure. The BBU contains a 48-Vdc rechargeable battery pack, charging circuit, and a dc-to-dc converter. The converter provides 300 Vdc during the backup mode.

CHAPTER 2
SYSTEM CONTROL

2.1 GENERAL

This chapter describes the interaction between the VAX 8800 system and the console subsystem, and how the VAX 8800 system is controlled by the console software. Included in this chapter is an overview of the console commands, operator displays, and the system logfile.

2.2 SYSTEM CONTROL

The console subsystem is a complex mixture of software and hardware that controls the VAX 8800 system.

2.2.1 Software

The primary function of the console software is to control the console and VAX hardware, and provide a communications media between the VAX microcode and the console.

2.2.2 Hardware

Control of the hardware is accomplished using encoded command, status, and data streams by means of several data paths and registers located within the VAX system and console. A body of VAX resident code, called the console support microcode (CSM), allows the console to access portions of the VAX 8800 CPU that the console cannot access directly by means of a hardware link. The CSM allows the console to access the internal CPU registers and locations in main memory.

Figure 2-1 shows a simplified block diagram of some of the key hardware and software involved in the control of the VAX 8800 system by the console. Table 2-1 describes the components identified in the block diagram.

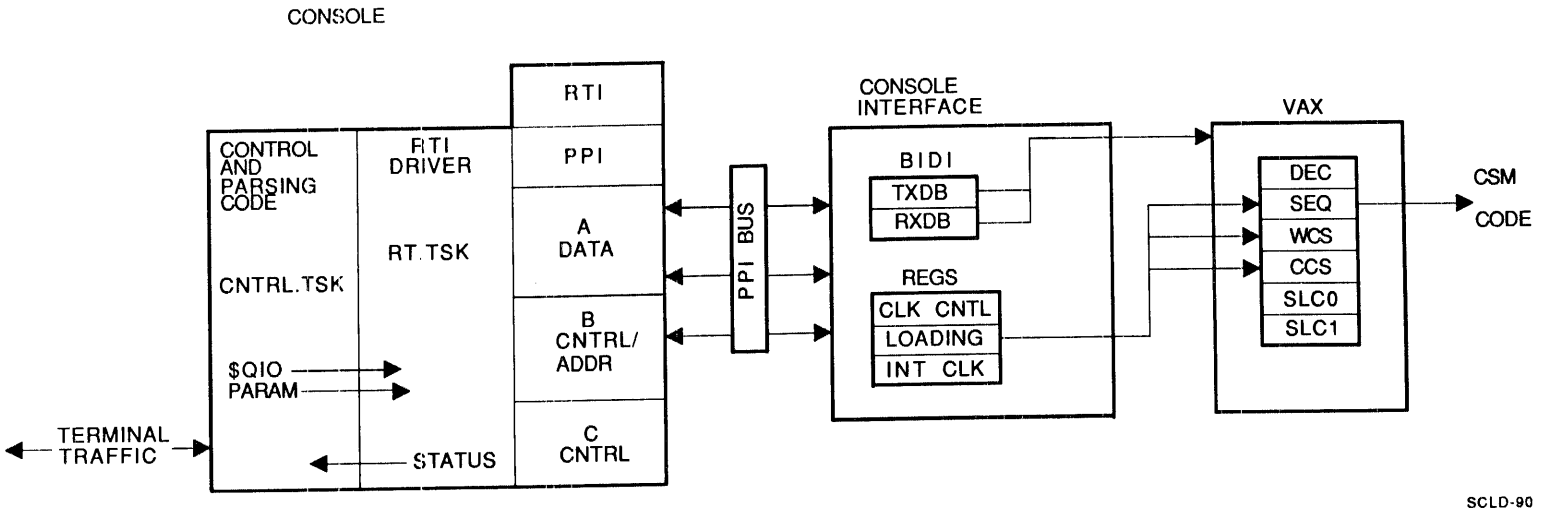


Figure 2-1 VAX 8800 System Hardware and Software Control Components

Table 2-1 Hardware and Software Component Description

Component	Function
Console Software	Handles input from the VAX system and EMM.
	Responds to VAX system and EMM with appropriate control actions.
	Commands and issues RSX-based $QIOs to the RTI driver.
RTI Driver	Services all $QIO service requests for data transfers between the console, and VAX system and EMM.
	Generates encoded commands for the VAX system and EMM.
	Handles asynchronous input from the VAX system and EMM.
Console Support Microcode (CSM)	VAX/WCS resident microcode that executes the functions requested by the console software.
	The CSM receives encoded commands and parameters from the RTI driver under interrupt control as a result of console generated commands.
RTI Hardware	The RTI module in the PRO-38N console computer, contains a programmable peripheral interface (PPI) that communicates with the VAX 8800 system through three addressable ports.
	Data is transferred to and from the ports across a bus connecting the PPI to the console interface on the clock module.
	The RTI contains a serial line port for communication with the EMM.
Console Interface	The console interface resides on the clock module and contains the hardware necessary to receive, route, and control the data stream between the console and the VAX system.
	The console interface communicates by means of the PPI bus on the console side and the bidirectional data bus on the VAX side.
	Transmit and receive data buffers provide the data transfer link between the PPI bus and the VAX 8800 system internal data buses.

2.3 CONSOLE SOFTWARE COMPONENTS

The console software consists of the following four major components:

1. Control program
2. File transfer program
3. Logical block server program
4. Real-time interface driver

2.3.1 Control Program

The control program is the main console program and implements both program and console mode, logfiles, EMM support, micromonitor, and remote access.

The control program is composed of interruptable code and asynchronous system traps (ASTs), which are software interrupt routines. The main body of code executes, and is interrupted periodically, when ASTs execute. AST routines are executed when some type of I/O or other system event occurs.

The control program normally has a QIO outstanding to the EMM, remote port (when enabled), and local terminal. When operating in the console mode, the local and remote QIOs represent entire lines. During program mode operation, the QIOs represent single characters. The local and remote ports can be in different modes or the same mode simultaneously. The console terminal driver echoes and implements special characters like ^U and RUBOUT during the console mode, and the VAX terminal driver performs the identical function during program mode.

The control program makes full use of the following RSX system directives:

- Set
- Clear
- Wait for event flag
- Timed requests
- Q/IO with wait
- QIO without wait
- QIO with AST completion routines
- QIO without AST completion routines

2.3.1.1 Special Control Program Features

CPU Designation

Console software maintains mappings from the LOGICAL "primary" and "secondary" CPUs, to the physical "left" and "right" CPUs. Start-up procedures refer to primary or secondary CPU rather than left or right.

Automatic Remapping

If the CPU designated as "primary" is not available, the control program can remap the "primary" to the other physical CPU and the start-up procedures will continue to operate properly.

Disable Secondary CPU

A secondary disabled state bit in the console is set by the control program when the secondary CPU is not operational. If one CPU is not operational, the other CPU is automatically designated as the primary CPU.

The secondary disabled bit can also be set by either a console command or a miscellaneous VMS command from the primary, to force single CPU operation.

When secondary disabled is set, the control program will assert CPU INIT and disable the NMI microsequencer. BOOT SECONDARY miscellaneous commands from the primary CPU will be ignored if the secondary CPU is disabled. The primary CPU will timeout waiting for the secondary to come up.

Current Primary and Next Primary

The SET NEXTPRIMARY command allows the operator to designate either the left or right CPU as the next primary. The keywords CURRENTPRIMARY and NEXTPRIMARY apply to the SET CPU command, and allow the VAX 8800 system operator to specify the logical CPU.

2.3.1.2 Multiple Command Streams -- The control program listens to and accepts commands from both the local and remote ports at all times. This requirement results in simultaneous 'local' and 'remote' command streams. An additional requirement of the dual-CPU system is that the secondary CPU can be rebooting independent of activity in the primary. The additional requirement creates another stream of commands that can be executing independently from commands entered at either the local or remote console.

The control program implements the three independent command streams by having a separate database for each stream. A global variable STREAM selects which copy of the database to use.

A fourth copy of the database is used to prevent database corruption by incorrect console commands. The control program creates a temporary copy of the database prior to processing each command. If the command is not interpreted successfully, the temporary database is copied back.

2.3.2 File Transfer Program

The file transfer program transfers data files between the VAX and the console Winchester disk drive. The VAX accesses the console disk by transferring files to/from the file transfer program. This transfer process allows the console to control all disk I/O functions without interference from the VAX.

The record management system (RMS) is used to read, write, and delete files.

2.3.3 Logical Block Server Program

The logical block server program (LBS) is the mechanism by which the operating system (VMS/UNIX™) reads or writes console media.

The VAX 8800 console provides arbitrarily large virtual floppies to VMB. A virtual floppy is a file on the Winchester disk. When VMB is reading or writing a logical block on a floppy, the LBS converts the logical block request into a virtual block operation.

The logical block server supports the read and write logical block functions and allows the VAX system to read or write to two logical devices (CSA1: and CSA2:).

LBS communicates with an operating system device driver through RXDB/TXDB and the RTI program.

2.3.4 Real-Time Interface Driver

The RTI driver program provides the link between the RTI hardware and the control, logical block server, and file transfer programs.

Separate sub-drivers are used for communications with the PPI port or serial line unit. The PPI sub-driver translates console commands, such as EXAMINE or DEPOSIT, into the appropriate sequence of READ, WRITE, and ACKNOWLEDGE commands for the PPI.

2.4 CONSOLE SUPPORT MICROCODE (CSM)

The CSM is a special group of microcoded routines that control console command processing inside the VAX system where the console cannot directly access the hardware. The CSM is used to access the internal CPU registers and main memory.

2.4.1 Console Support Microcode Structure

The CSM consists of four major sections of code. Table 2-2 lists and describes each of the sections.

UNIX is a trademark of AT&T.

Table 2-2 Major Sections of CSM Code

Section	Description
Executive Wait Loop	Polls RXCS and RXDB for console-to-VAX data.
Entry Points into specific CSM routines	Many entry-point microaddresses.
Command Dispatcher	Tests ID bits for CSM commands, and dispatches microcode to correct routine.
Exit Routines	Cleans up and transcends to other VAX/CONSOLE states.

In order to utilize WCS space in an efficient manner, the console support microcode resides in two locations (resident and nonresident).

The resident CSM is in instruction set processing (ISP) code space and is always loaded to allow MTPR, MFPR, PROBER, and PROBEW instructions to utilize CSM routines to access registers.

Nonresident CSM is overlaid on top of USER WCS code space. This code is used for initialization and troubleshooting.

2.4.2 CSM Data Transfers/Protocol

Data is transmitted in one or more bytes of information. Most routines require numerous byte transfers to execute successfully. The executive routines are responsible for building "packets" to send to the console.

The following guidelines describe the protocol during data transfers:

- Commands are 1 to 9 bytes long.
- Responses are 1 to 5 bytes long.
- During console-to-VAX transfers, the most significant byte is transmitted first.
- During VAX-to-console transfers, the least significant byte is transmitted first.

2.4.3 Console Support Microcode Entry Points

The CSM can be entered by both hardware and ISP (instruction) code execution. Table 2-3 lists the entry points.

Table 2-3 Console Support Microcode Entry Points

Type of Entry	Entry Point
Hardware	Microtraps Special Address Startup Constrained Addresses
Software (ISP Shared)	MTPR/MFPR Instructions VAX Communications PROBER/PROBEW Instructions

2.5 OPERATIONAL MODES

The console operates in one of two modes as shown in Figure 2-2. Refer to Section 3 (Console) of the manual for additional information concerning modes of operation and machine states.

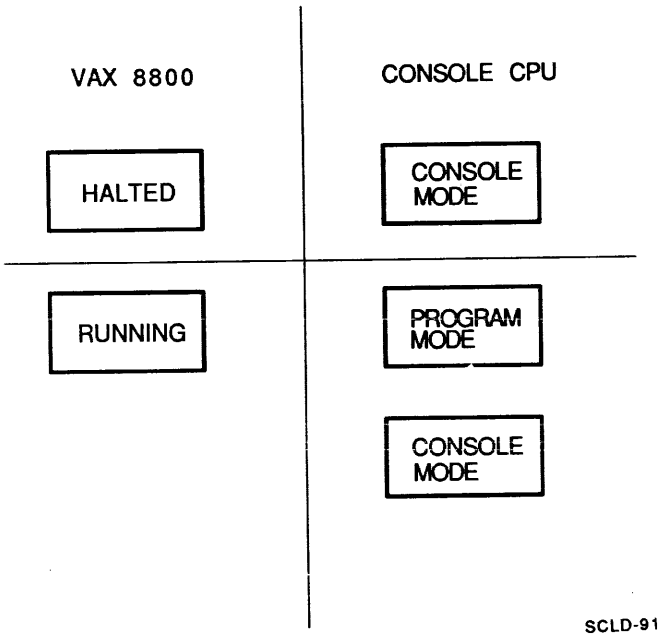


Figure 2-2 Console Operational Modes

2.5.1 Console Mode

The console mode can be in effect with the VAX 8800 system halted or running. While the VAX system is halted, the entire range of console commands is available to the operator.

Typing a CTRL/P while the VAX system is running causes the console to enter the console mode (leaving the VAX system running). Since the VAX system is running, the available commands during the console mode are limited.

2.5.2 Program Mode

Program mode allows the console to act as a VAX terminal. During this mode, communications with the VAX system are interrupt-driven.

2.6 OPERATOR/CONSOLE INTERACTION

The VAX 8800 system and console CPUs constitute a complex system that must be initialized, operated under normal and abnormal conditions, and turned off. Which functions are valid at any particular time are dependent on the "state" of the VAX and console.

The console and VAX system are two separate CPUs that can be operating in any combination of states. The state of one CPU is independent of the other CPU.

2.6.1 Console State Bits

The states of the VAX system and console are maintained by console software, and are used to implement restrictions on state transitions and console commands. The following state information is available:

- Left CPU
- Right CPU
- Common hardware
- Command streams

Table 2-4 provides examples of the state bits.

Table 2-4 Bit Examples

State	Type
VAX power on ?	Common
Remote port enabled ?	Common
Battery Backup unit working ?	Common
AUTOBOOT enabled ?	Common
EMM OK ?	Common
Clock running ?	Common
CPU halted ?	Left/Right
Terminal in Console or Program mode ?	Per command stream

2.6.1.1 Command Validity -- Each console command calls a utility routine that specifies which state bits must be TRUE and which must be FALSE for a specific command. Unspecified bits are not checked. If the specified bits are not in the correct states, the command will not be executed and an error message will be printed.

Some bits require access to the hardware each time it is referenced. (Example: The clock may stop if the microPC matches and stop-on-micromatch was set. It must be checked each time the clock bit is referenced.)

Each console command can specify whether the CPU-specific bits to be checked are in the CPU for which the command was issued, the other CPU, or both CPUs. This causes some commands to be dependent on the states of both CPUs.

2.6.1.2 Saving Console State -- Important console state variables are preserved during console power failures. These include:

- Default settings -- anything that can be set with SET DEFAULT.
- Logfile control block.
- Console "state bits", which include the AUTO BOOT/RESTART/POWERON switches.
- Clock rate.
- Reboot-Primary ID.
- Shadow copies of clock board registers.
- TODR, date, and time TODR was written.

2.6.2 Console Commands

The VAX 8800 Console Command Language (CCL) is the interface between the operator and the control and monitoring capabilities of the VAX 8800 console and micromonitor subsystems. Table 2-5 provides an overview of VAX 8800 commands. Refer to the Console User's Guide for a complete description of each command.

Table 2-5 Console Command Overview

Command	Description
@	Opens specified file and processes the records within the file
BOOT	Executes Device_nameBOO.CMD; if no device is specified, executes DEFBOO.CMD
CLEAR ACCUMULATOR SOMM TOMM	Resets the settings made by the SET and PROBE commands
CONTINUE	Starts execution at current PC
DEPOSIT/EXAMINE	Deposit/Examine address/data
DISABLE/ENABLE	Establishes console parameters: Auto Restart/Boot/Poweron Odometer, Printer, Console User monitoring
EXIT	Terminates console program
FIND	Enables memory search: 64Kb of good memory Restart Parameter Block
HALT	CPU halts at the next macroinstruction boundary
HELP	Prints help for the specified topic
IF	Enables conditional execution of commands
INITIALIZE	Sets CPUs to defined state
LINK	Creates temporary indirect command file
LOAD	Loads memory/control store
MICROSTEP	Bursts the clock (n) cycles
NEXT	Steps the VAX through (n) macroinstructions
PERFORM	Executes the temporary command file created with LINK

Table 2-5 Console Command Overview (Cont)

Command	Description
POWER	Turns power ON/OFF/STANDBY
PROBE	ORs VBus bit into Accumulator
REPEAT	Causes command line to be executed continuously
SENSE REVISION	Senses revisions of specified component
SET	Sets Clocks, CPU Primary, Defaults EMM Power Margins, Relocation Values for EXAM/DEP, SOMM, TOMM, Verify/Nonverify
SHOW	Shows Value/Status: Accumulator, CPU, Defaults, EMM, Logfile, Revision History
START	Starts execution of macro/microcode
TEST	Starts customer diagnostics or micromonitor
UNJAM	Asserts/Deasserts UNJAM, reloads NBI with hi and lo memory limits
VERIFY	Verifies EMM, module placement, revision history

2.6.2.1 Executing Console Commands -- All commands that read or write the VAX 8800 system hardware are implemented by means of QIOs to the RTI driver program. Commands that require the power, clocks, or CPU to be in a specific state, generate a state check prior to executing the command.

Commands from the console are passed to the VAX system as 8 bits of data by the RTI driver. The command data is transferred by means of the PPI bus connecting the console computer and the console interface on the clock module.

The command data being transferred contains encoded identification and data fields describing the type, purpose, and destination of the command. The console interface places the command data into the receive data buffer (RXDB) so that it can be transferred to the VAX CPU decoder module by means of the bidirectional data bus connecting the console interface and the VAX 8800 CPU.

2.6.3 Console/Operator Display

The display for the VAX 8800 system is the PRO-38N console display terminal. Only one of the VAX 8800 system's dual CPUs will be shown on the display screen during normal operation. During system testing in the micromonitor mode, the output from both CPUs may be observed simultaneously.

Status is displayed at the bottom of the screen to allow for longer view time during scrolling when the screen is responding to other console output.

2.6.3.1 Local Display During Remote Operation -- During remote operations, both the local and remote terminals can be operational at the same time. The local operator can monitor the input/output over the remote port by enabling the remote monitoring function.

When remote monitoring is enabled, all remote activity appears in the logfile as well as on the local screen. Figure 2-3 shows an example of character flow during simultaneous local and remote operations.

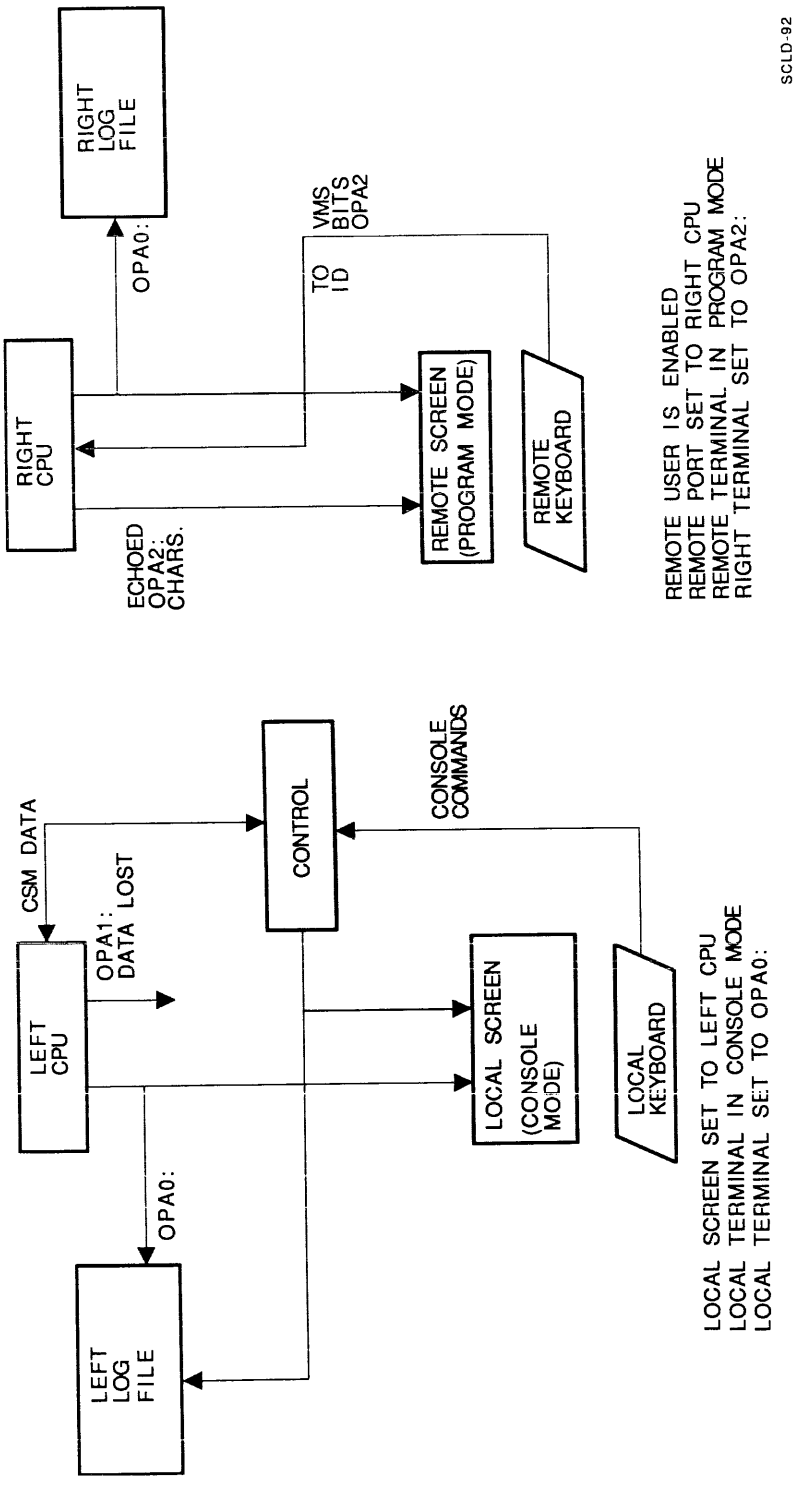


Figure 2-3 Local/Remote Display Character Flow

2.6.3.2 Console Command Language Display Prompts -- Table 2-6 describes the console command language prompts that appear on the local and remote display screens.

Table 2-6 Console Command Language Prompts

Prompt	Description
>>>	Console Command Level Input
MIC>	Micromonitor Input
<<<	Link Mode Input (Reference LINK and PERFORM Commands)

2.6.4 System Logfile

The console maintains two logfiles for the purpose of saving and reviewing console output pertaining to each CPU. Each logfile consists of a circular buffer capable of storing 720 lines (30 screens) of 80 characters per line.

All of the data that appears on the display screen during console mode operations, as well as the output from the VAX, is saved in the logfile. Console entries made during the program mode are echoed back from the VAX, but are not saved in the logfile.

2.6.4.1 Displaying the Logfile -- The SHOW LOGFILE command allows the operator to look at the contents of the logfile with the provision for scrolling up and down, and changing pages of the display. Data received during the display of the logfile will be appended to the end of the file and not displayed with the present viewing.

2.6.4.2 Logical Terminals/Logfile Integrity -- Logical terminal OPA0: is the standard logical terminal for normal system operations, and all OPA0: data is placed in the logfile. Entering the SET TERMINAL OPA1: command prior to program mode operations will eliminate filling the logfile with unwanted data.

2.6.4.3 Saving the Logfile -- Logfile data is preserved if either the control program stops, or the console power fails. Nonvolatile data is stored in the first block of the logfile:

- Logfile control block
- Micromatch address
- Verify switch
- Examine/Deposit defaults
- Default radix
- Internal trace bits
- State bits
- Clock rate
- Regulator margin mask
- Reboot primary identity
- Memory limits
- Shadow copies of clock board registers
- TODR written by the VAX system
- P/OS date and time when TODR written
- Micromonitor state variables

2.7 POWER-UP/DOWN SEQUENCING

The following paragraphs provide a brief overview of the power-up/down and restart functions. Refer to the Console and Power Sections of the manual for detailed information concerning power-on sequencing, recovery, and initialization and boot processes.

2.7.1 Power On

The power-on sequence can be initiated by either a VAX 8800 system power-fail recovery, or by entering the POWER ON command at the console terminal.

The console software determines whether the recovery was a console-only failure by examining the VAX power status by means of the environmental monitoring module in the power system.

The power-up procedure requires that the console send commands to the EMM to apply system power in a particular sequence. Table 2-7 shows the sequence in which the power supply modules are turned on.

Table 2-7 Module Power Supply Turn-On Sequence

Order	Module	Power Application
1	J	+300
2	B	+5 Memory/Battery
3	C	+5
4	F	+/- 12, -5.2, +5, -2 to VAXBI
5	H	+/- 12, -5.2, +5, -2 to VAXBI
6	E	-5.2
7	D	-2

2.7.2 Power Fail

Power failure recognition by the environmental monitoring module in the power system will result in an interrupt to the VAX 8800 system by means of the console interface. The EMM is directed to assert the ACLO and DCLO signals, and initiate the power-fail sequence.

2.7.3 Powerdown

An intentional powerdown is initiated by operator command at the console. The console software checks the status of the battery backup unit and informs the operator if the BBU is not present or enabled.

If backup power is not available, the software asks the operator if it should continue or abort the powerdown process. If the BBU is enabled or the operator wishes to continue without backup power, the EMM is directed to assert the ACLO and DCLO signals and initiate power shutdown.

2.7.4 Warm Restart

Warm restart uses the same procedures as the power-on sequence for turning on the system power. Memory must be backed up and a restart parameter block must be available for a warm restart.

3.1 INTRODUCTION

This chapter contains a brief overview of the CPU microcode, memory addressing, read/write operations, and interrupts and exceptions. Refer to the individual sections of this manual for detailed information.

3.2 MICROCODE

The control store microcode resides in the IBox and consists of 16K words x 143 bits. The microcode is loaded into the control store RAMs from the console during system initialization. The control store usage is:

- CPU control -- 15K
- User-written code -- 1K

3.2.1 Microcode Characteristics

3.2.1.1 Functionality

Horizontal -- Microword bits are grouped into fields with each field directly feeding and controlling a specific CPU element. (Some fields have vertical functionality in that they control more than one element.)

3.2.1.2 Operation

Pipelined -- More than one microword is active at any given time. Pipelining allows the CPU to perform multiple operations simultaneously.

3.2.1.3 Structure

Segmented -- Each microword is divided into three segments, CS0, CS1, and CS2. Dividing the microword into three segments enhances the pipelining process.

CS0 RAM Segment

Resides on the SEQ module and consists of microword bits <47:0>. Provides early control - IBox and EBox operations.

CS1 RAM Segment

Resides on the WCS module and consists of microword bits <95:48>. Provides mid/late control - EBox operations.

CS2 RAM Segment

Resides on the WCS module and consists of microword bits <142:96>. Provides late control - EBox and CBox operations.

3.2.2 Microcode Control

The sequencer module (SEQ) controls the operation of the microcode. The primary duties performed by the SEQ module are:

- Select the appropriate microPC input for the control store address latches.
- Monitor hardware and software interrupt requests.
- Provide CS0 segment microcode data to the kernel.
- Monitor microbranch and microtrap conditions.
- Provide a data path between the CPU and the console.

The functional areas of the sequencer are:

- INPR
- CCBR
- GWYC
- UTRP
- UBRS

3.2.2.1 Interrupt and Processor Register (INPR)

- Receives and prioritizes hardware interrupts (software interrupts are handled by microcode).
- Provides interrupt microtrap vectors and microbranch conditions to the microsequencer logic for interrupt processing.

3.2.2.2 Condition Code and Branch (CCBR)

- Monitors EBox and CBox condition bits.
- Provides macrobranch and trap vectors during instruction execution.
- Contains the PSL <N, C, V, Z> bits.
- Controls instruction buffer flushing during macrobranch and macrotrap.
- Contains STATE flags for microcode branching.

3.2.2.3 Gateway Control (GWYC)

- Controls console access to the micromachine during RAM loading and microcode requests.
- Monitors console parity errors during data transfers.

3.2.2.4 Microtrap (UTRP)

- Receives and prioritizes all macro- and microtrap conditions.
- Provides microvectors to the microbranch slice logic.
- Controls microstack addressing and uPC mux.
- Provides SILO control to save machine state during traps.

3.2.2.5 Microbranch Slices (UBRS)

- Monitors microbranch conditions from the CPU kernel.
- Monitors microvectors from the microtrap logic.
- Supplies the next microPC to the control store address latches.
- Provides microstack data and receives RETURN data.

3.2.3 Pipelining

The process of pipelining (processing multiple microwords simultaneously) allows the VAX 8800 system to utilize the hardware in a more efficient manner, and increase the performance of the CPU. Figure 3-1 shows a simplified drawing of VAX 8800 pipelining.

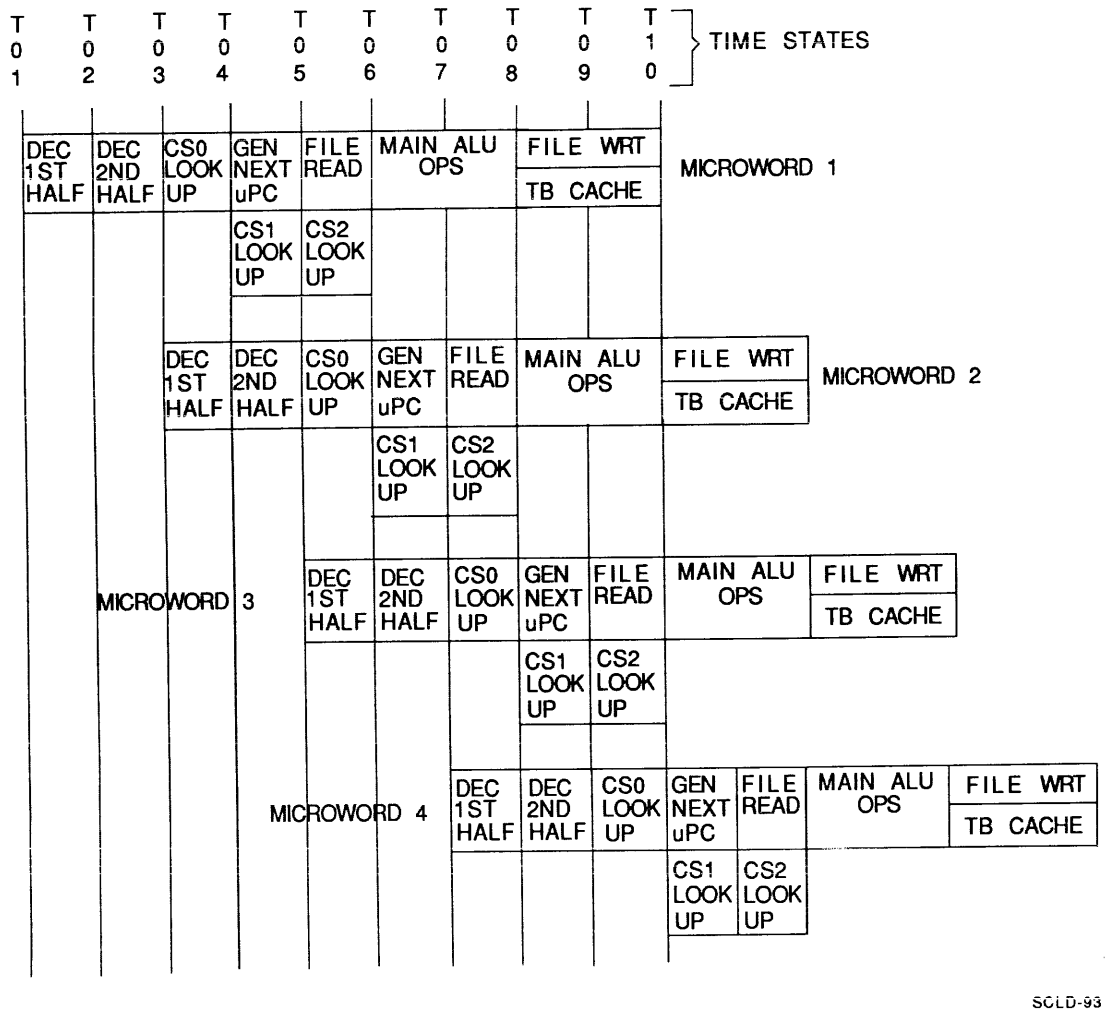


Figure 3-1 Simplified Pipelining

3.2.4 Microtraps

Microtraps are hardware conditions that prevent the current microword from executing properly. When a microtrap occurs, the hardware forces the control store address to a fixed location (the location is dependent on the type of trap) overriding the address that would have been selected. The forced address is the starting location of the trap handler microcode routine.

Microtraps are used extensively by the memory management system, and for resolving system faults such as parity and bus errors. Table 3-1 lists the various types of microtraps for the VAX 8800 system.

Table 3-1 VAX 8800 System Microtraps

Microtrap Condition	Priority
Microbreak	Highest
Machine Check *	
VA Parity Error *	
TB Tag Parity Error *	
Not Used	
Reserved Floating-Point Operand	
Floating-Point Round Error, ADD	
Floating-Point Round Error, Multiply	
Integer Overflow	
TB Miss	
Access Violation	
Modify Bit	
Page Cross	
Unaligned Page Cross	
Unaligned Conditional Branch	
Integer Overflow	Lowest

* = Machine Check

When a microtrap occurs, any register or cache writes that would otherwise be performed by the microwords in the pipeline are blocked until the cause of the trap has been cleared. The hardware performs most of the state-saving and restoration for return to the blocked microinstructions.

Microaddresses of the blocked microinstructions are saved in a queued microPC silo to allow for the possibility of a branch in one of the blocked microinstructions.

3.2.5 Micromatch

One method of halting the micromachine is through the use of a micromatch. [The hardware matches every microPC (program counter) with the specified breakpoint microaddress.]

The micromatch function provides for one of two actions to be specified when a micromatch occurs:

1. Stop on Match
2. Trap on Match

3.2.5.1 Stop on Match -- When the clocks are stopped during a micromatch, the state of the machine is frozen until a command from the console restarts the system clocks.

3.2.5.2 Trap on Match -- Following a microPC match, the micromachine traps to a specified location to perform the intended operation. Original microcode flow can be continued at the completion of the trap routine.

3.3 MEMORY ADDRESSING AND READ/WRITE OPERATIONS

3.3.1 Virtual Addresses

The processor generates a 32-bit virtual address for each instruction and operand in memory. As the process executes, the system translates each virtual address to a physical address.

Virtual address space is a 32-bit unsigned integer specifying a byte location in the physical address space and is arranged in 512-byte units, called pages.

A memory management system provides the mechanism for mapping the active part of the virtual address space to the available physical address space and provides page protection between separate processes. The operating system controls the virtual-to-physical address mapping tables and places the inactive parts of the virtual address space on the external storage media.

When memory mapping is disabled, virtual addresses are translated to physical addresses by ignoring bits 30 and 31.

The translation buffer in the CBox is used to speed virtual address translations by holding calculated address translations for future use. If the translation for a virtual address exists in the translation buffer, the data is used for the physical address. If the desired address translation does not exist, a microtrap occurs and a trap routine performs the address translation. When the trap routine completes the translation process, the translated address is written into the translation buffer.

3.3.1.1 Layout -- Virtual address space is divided into two equal size spaces:

1. Per-process address space
2. System address space

Per-process space is distinct for each process running on the system and system space is shared by all processes. Figure 3-2 shows the layout for virtual address space.

3.3.1.2 Format -- The format for the 32-bit virtual address is shown in Figure 3-3.

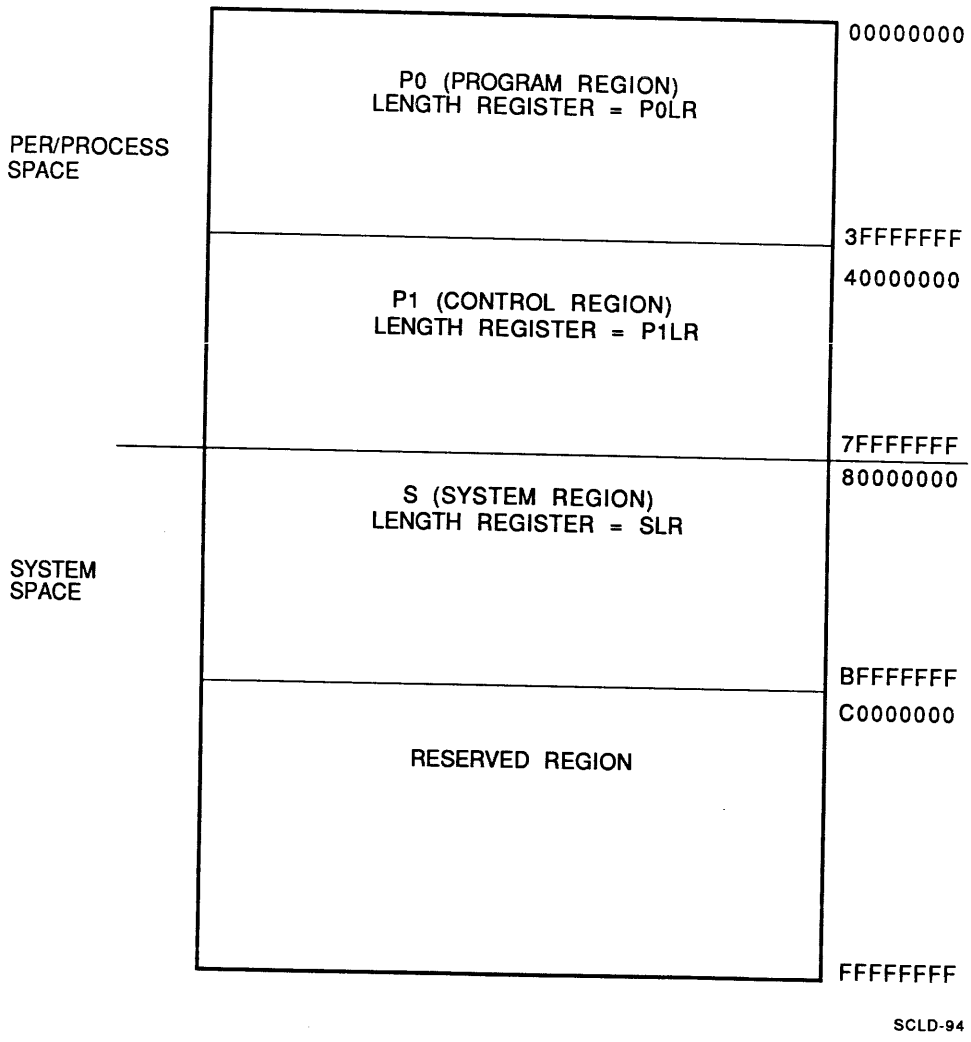
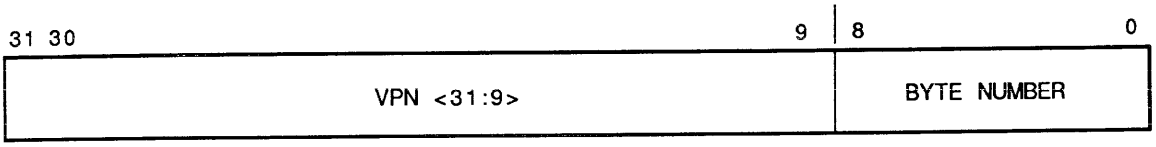


Figure 3-2 Virtual Address Space Layout



VPN (VIRTUAL PAGE NUMBER) -- SPECIFIES THE VIRTUAL PAGE TO BE REFERENCED

BIT 31 1 = SYSTEM SPACE ADDRESS
0 = PER-PROCESS SPACE ADDRESS

BIT 30 PER-PROCESS SPACE
1 = CONTROL REGION
0 = PROGRAM REGION

BYTE NUMBER -- SPECIFIES THE BYTE ADDRESS WITHIN THE PAGE

SCLD-95

Figure 3-3 Virtual Address Format

3.3.2 Physical Addresses

Physical address space consists of two parts, memory space and I/O space. Memory space starts at address zero and continues to 1FFFFFFF. I/O space begins at 20000000 and continues to the end of the physical address space 3FFFFFFF. Figure 3-4 shows the layout of physical address space.

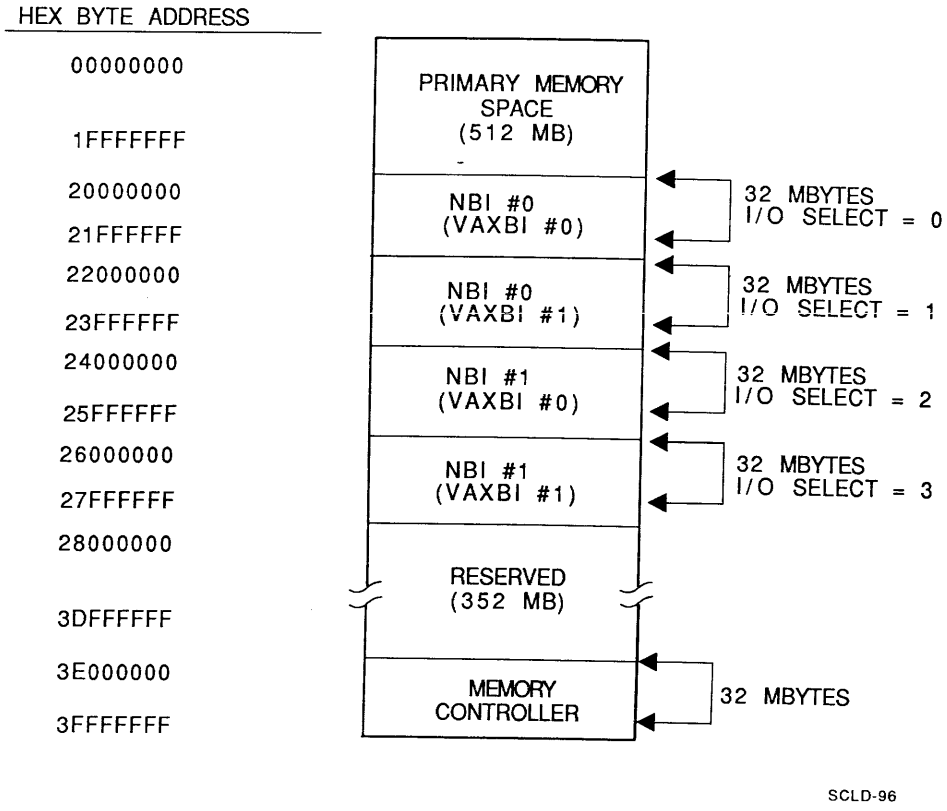


Figure 3-4 Physical Address Space Layout

3.3.3 Address Translation

The process of translating a virtual address to a physical address is controlled by setting the memory mapping enable (MME) bit in the MAPEN (MAP ENABLE) internal processor register. Figure 3-5 shows the bit configuration of the MAP enable register.

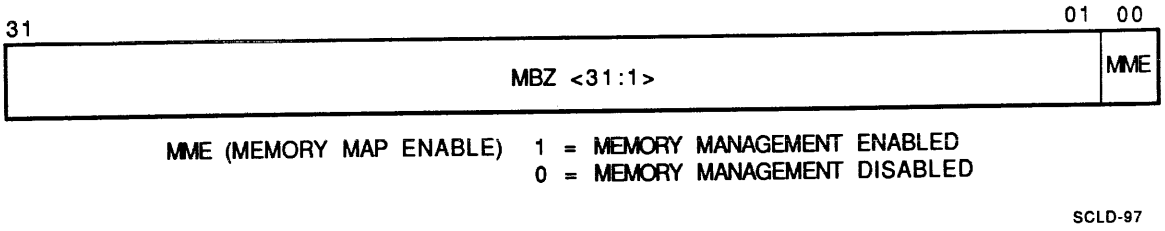


Figure 3-5 MAP Enable Register Bit Configuration

3.3.3.1 Page Table Entry -- The CPU uses a page table entry (PTE) to translate virtual addresses to physical addresses. Figure 3-6 shows the format and bit configuration of the PTE. Table 3-2 describes the function and purpose of the PTE bits.

Figures 3-7 through 3-9 show simplified diagrams of the translation process for system space and process space regions 0 and 1.

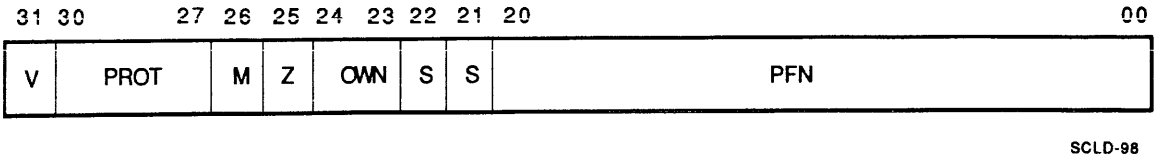


Figure 3-6 Page Table Entry Bit Configuration

Table 3-2 Page Table Entry Bit Description

Bit	Symbol	Function	Comments
31	V	Valid Bit	Determines validity of the M bit and PFN field. 1 = Valid, 0 = not valid. When V = 0, the M and PFN fields are reserved for DIGITAL software.
30-27	PROT	Protection Field	Always valid. Used by the CPU hardware.
26	M	Modify Bit	When bit 31 is set, the M bit indicates whether the page has been modified. If the M bit is set, the page may have been modified.
25	Z	Zero Bit	Reserved for DIGITAL software. Must be zero.
24-23	OWN	Owner Bits	Reserved for DIGITAL software. Use as the access mode for the page owner.
22-21	S	Software Bits	Reserved for DIGITAL software.
20-0	PFN	Page Frame Number	Upper 21 bits of the physical address of the base of the page.

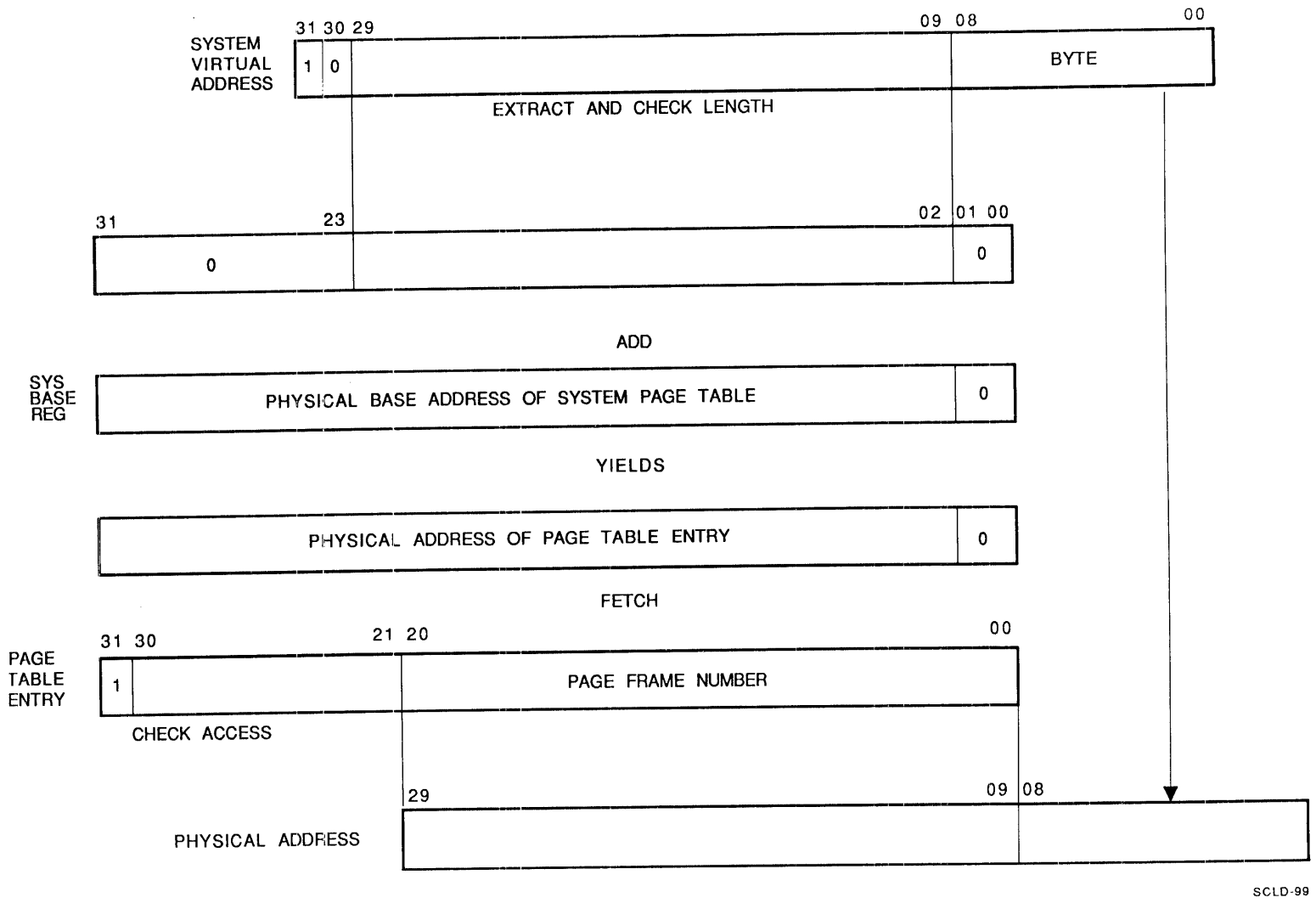


Figure 3-7 System Space Virtual-to-Physical Address Translation

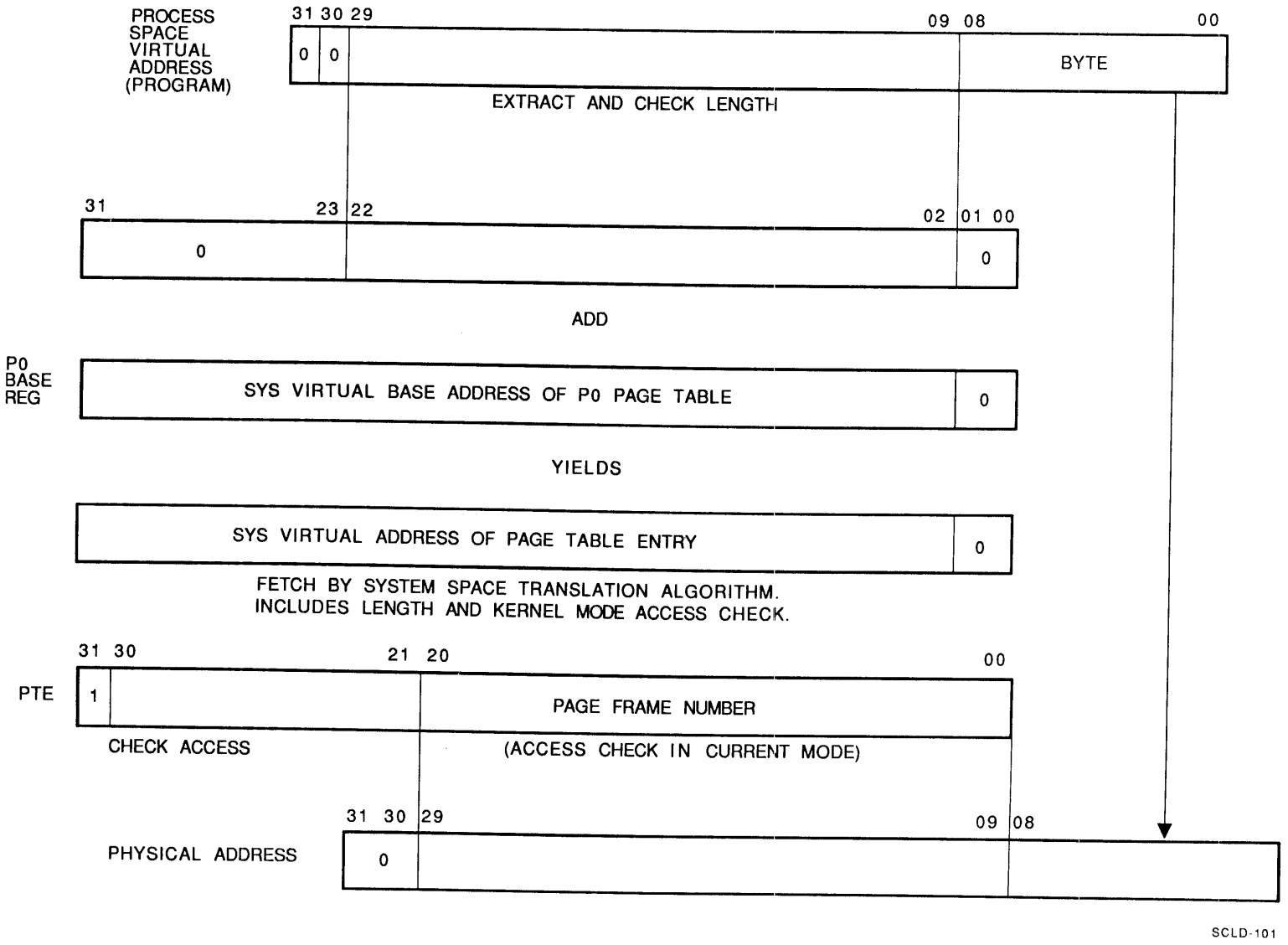


Figure 3-8 Process Space (P0 Region) Virtual-to-Physical Address Translation

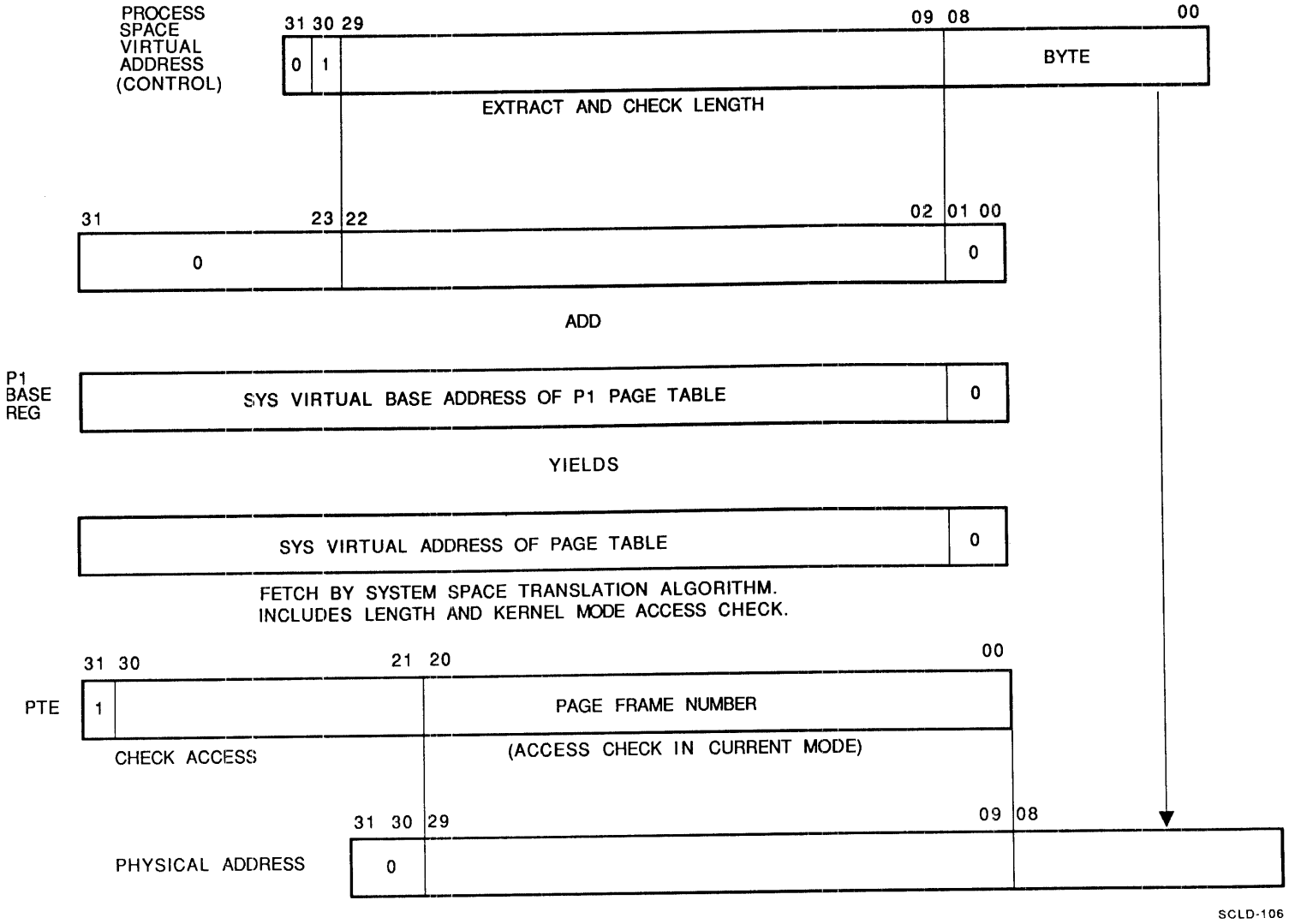


Figure 3-9 Process Space (P1 Region) Virtual-to-Physical Address Translation

3.3.4 Cache Operation

The CBox shown in Figure 3-10 consists of three major functional areas:

1. Translation Buffer (TB)
2. Cache
3. NMI Interface

3.3.4.1 Translation Buffer -- The translation buffer is a hardware buffer used to hold calculated address translations for future use. The TB also contains control information concerning the physical page. Table 3-3 lists and describes the fields of the translation buffer.

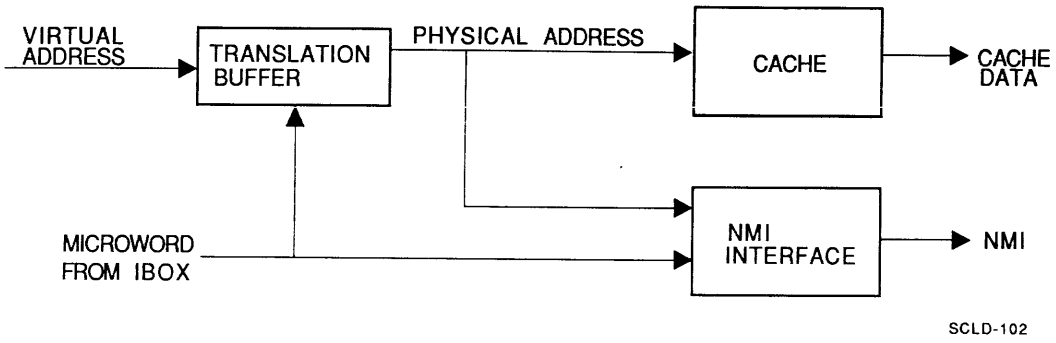


Figure 3-10 CBox Functional Components

Table 3-3 Translation Buffer Field Description

Field	Field Description
TB Tag	Used in the matching process to determine if the TB contains a translation for a virtual address.
TB Data	Consists of the page frame number (PFN) of the page table entry (PTE) for the virtual address.
Protection	Protection information from the page table entry of the page. Compared with the current processor mode to determine access capabilities to the page.
M Bit	Modify bit from the page table entry.
TB Valid Bit	Indicates whether a TB entry is valid. The TB valid bit is not the same as the valid bit in the page table entry.

TB Hit

A translation buffer hit indicates that the currently accessed TB entry is the correct translation for the virtual address. A TB hit is determined by accessing the TB with virtual address bits <31, 17:9> and matching the TB tag field with virtual address bits <30:18>. If the addresses match and the entry is valid, the translation is for the virtual address, and protection and modify bit checks are initiated.

If there are no protection or Mbit problems, then it is a TB hit and the PFN concatenated with VA<8:0> is used as the physical address.

TB Miss

A TB miss occurs when there is no translation of the virtual address to a physical address.

3.3.4.2 Cache -- The cache is a read allocate only device, and cache locations are updated only if a previous cache hit existed. Write misses do not affect the cache. In order for the cache to process either reads or writes in one cycle and in any order, the cache uses a delay write algorithm that delays the update until the next write cycle.

The cache is divided into three sections:

1. Cache Tag Store
2. Cache Tag Valids
3. Cache Data Store

Cache Tag Store

The cache tag store holds the cache tags that are compared with the incoming PA<28:16> as part of the effort to determine a cache hit.

Cache Tags Valid

Each tag entry has four associated valid bits that are used to indicate an octaword valid. The valid bits are used in the matching process to determine a cache hit.

Cache Data Store

The cache data store contains the data corresponding to the cache tag. If the access results in a cache hit, the data being read from the cache data store is valid data and is passed to either the IBox or EBox.

3.3.4.3 NMI Interface -- The NMI interface provides the control and data path that allows the CPU to communicate with the VAX 8800 memory interconnect. Following a cache read miss, the NMI interface uses the missed read address to build an NMI command/address transaction and sends it to memory by means of the NMI. The translation buffer and cache are free to process additional CPU requests while the NMI interface handles the memory transaction with the NMI.

When the memory data arrives, the NMI interface assumes control of the cache, loads the data into the cache store, and validates the cache tag valids with the new tag address.

Read/write command/address transactions generated by the NMI interface in the CBox are transferred to memory and I/O devices by means of the VAX 8800 memory interconnect, which is a synchronous backplane bus that interconnects the CPUs, memory, and I/O adapters.

3.3.5 Read/Write Operations

Before any device can perform a transfer of data on the NMI, the device must request and be granted use of the bus by asserting an NMI arbitration request line. The bus is granted to the requesting device by the assertion of a bus enable signal for one cycle.

A write transaction requires only one transfer. The device initiating the transaction arbitrates for the bus and then transfers command/address information followed by write data.

Read transactions require multiple transfers to complete the transaction. First, the device requesting the read arbitrates for the bus and transfers the command/address. The device containing the data to be read arbitrates for the bus and transfers the read data during a second transfer cycle. A third transfer of read data may be required for some read hexword transactions.

At the beginning of a read/write transaction, the 30-bit NMI address is asserted during the command/address cycle to identify the device involved in the transaction. NMI address selection is shown in Figure 3-11.

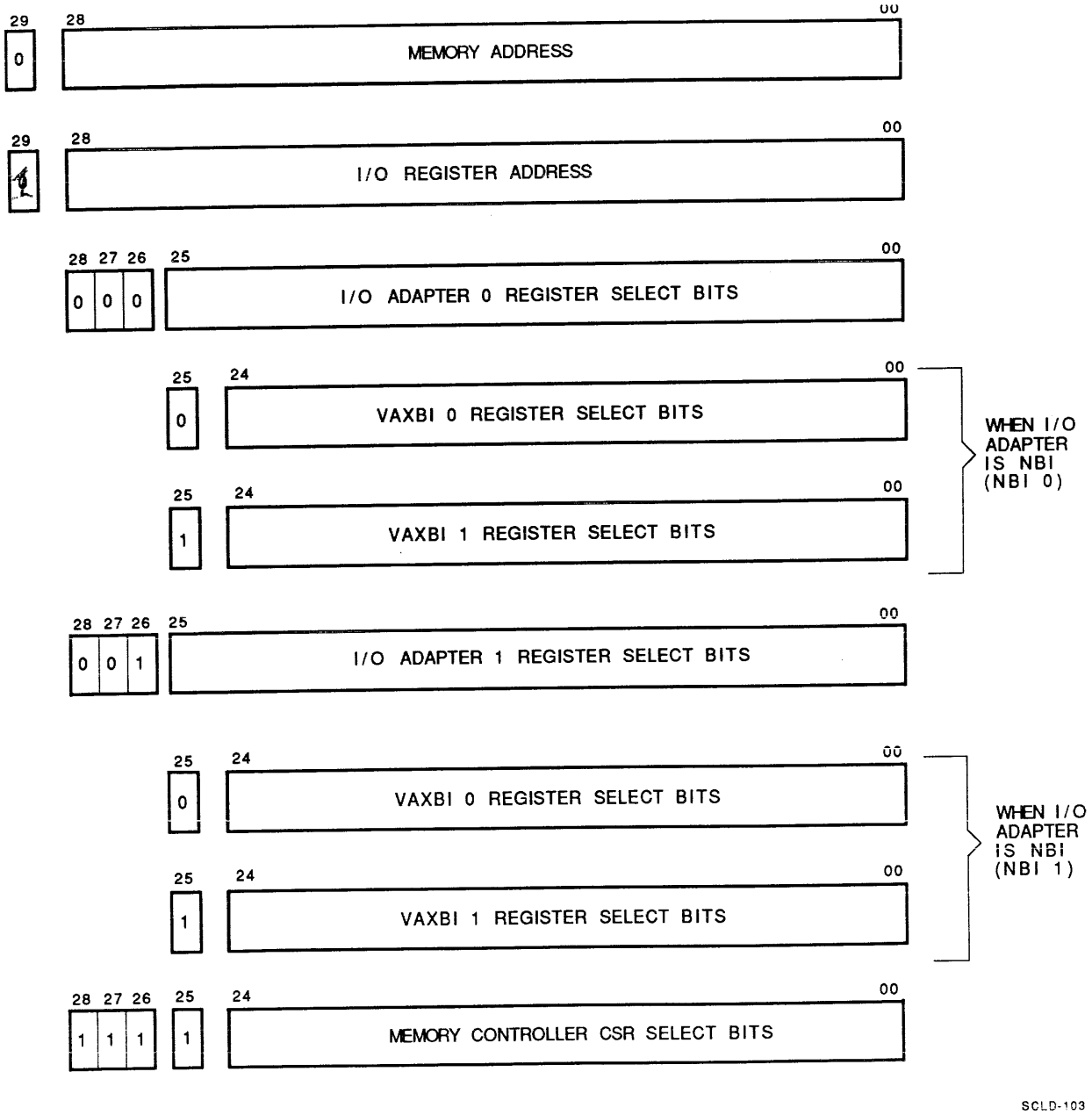
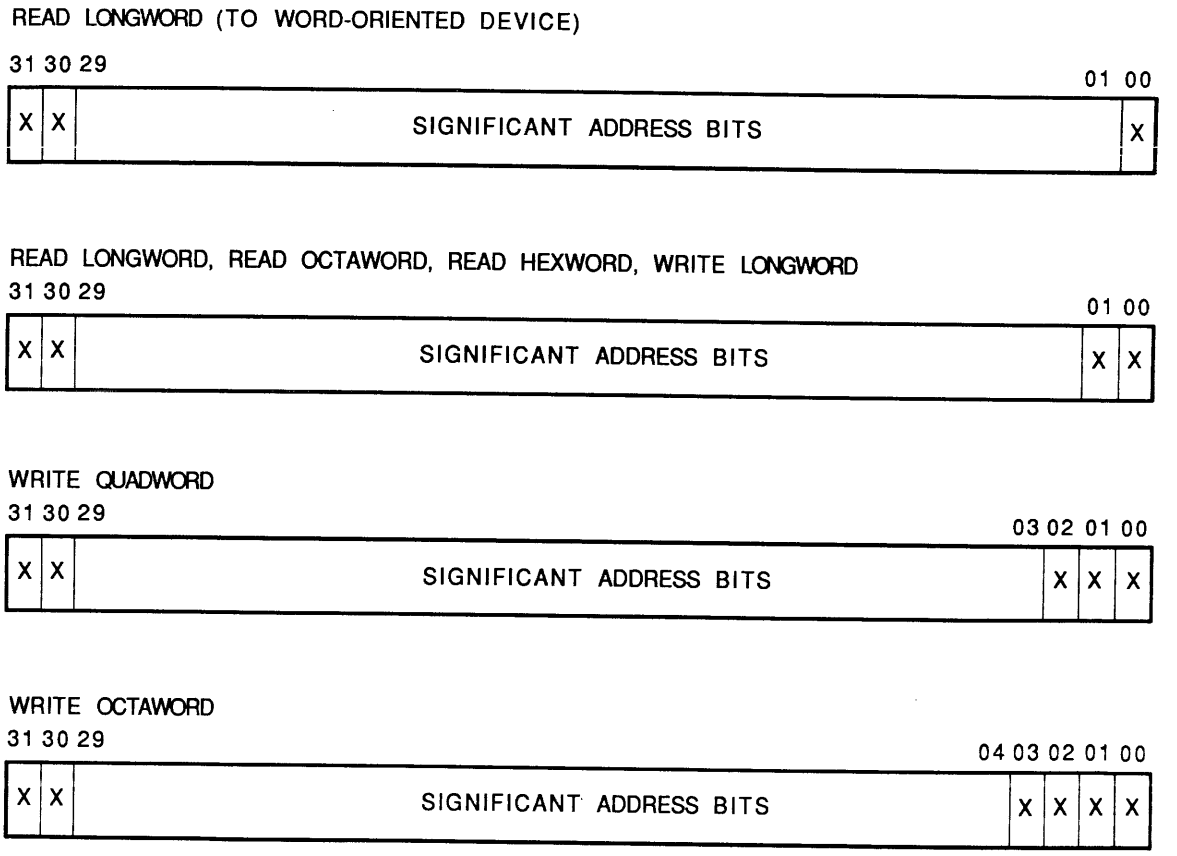


Figure 3-11 NMI Address Selection

3.3.6 Device Address Selection

Address bit <29> determines if the specified address is a memory or I/O address. If it is an I/O address, then bits <28:26> further define which I/O adapter (NBI 0 or NBI 1) is being addressed. If an NBI is addressed, then bit <25> selects the appropriate VAXBI (0 or 1). Bits <25:29> must be set to one in order to select a memory controller CSR.

3.3.6.1 Transaction Significant Address Bits -- Which bits of an address are significant, depends on the type of transaction being performed. Figure 3-12 shows the significant and nonsignificant bits for different transactions.



NOTE: X INDICATES THAT THE BIT IS NOT SIGNIFICANT AND IS IGNORED BY THE RESPONDING DEVICE.

SCLD-104

Figure 3-12 NMI Address Bit Significance

3.4 INTERRUPTS AND EXCEPTIONS

3.4.1 Servicing

The two VAX 8800 CPUs handle interrupts that originate from within the processor or from the other processor. Interrupts from the NMI nexus (I/O devices and memory controller) are handled exclusively by the CPU that is designated as primary.

3.4.1.1 NMI Interrupt Enable Register -- Each processor has an interrupt control register (NICTRL) that enables or disables interrupt handling for that processor. The bits of the NICTRL are used as follows:

Bit	Function
7	Enable Interrupts from NMI Device 0
6	Enable Interrupts from NMI Device 1
5	Enable Interrupts from Main Memory

3.4.1.2 Types of Interrupts -- There are two types of interrupt requests generated by the NBIs. An interrupt request can be generated locally by the NBI, or it can be generated by any one of the I/O devices connected to the NBI.

Locally generated interrupts are at the BR4 level and I/O device interrupt requests can be at any BR level. (Refer to Table 3-4.)

3.4.1.3 Servicing the Interrupt -- To service an NBI interrupt, the CPU performs a firmware initiated read of a system control block vector offset value from one of the four vector offset registers in the NBI. The BR level of the interrupt determines which one of the four registers is used.

Locally generated interrupts will use page zero (0) of the SCB for the vector offset, and I/O devices use pages 1 through 63 for the vector offset.

3.4.2 Priority Levels

In order for the CPU to respond to an interrupt, the incoming interrupt must have an interrupt priority level (IPL) greater than the IPL in the processor status longword (PSL). Table 3-4 lists the hardware IPL assignments.

Table 3-4 Hardware Interrupt Priority Level Assignments

Device or Condition	Interrupt Priority Level (HEX)
Unassigned	1F
Power Fail	1E
CBox Error	1D
NMI Fault	1C
Unassigned	<1B:18>
I/O Adapter #0 (NBI #0), NMI BR7	17
I/O Adapter #1 (NBI #1), NMI BR7	17
Interval Timer	16
I/O Adapter #0 (NBI #0), NMI BR6	16
I/O Adapter #1 (NBI #1), NMI BR6	16
I/O Adapter #0 (NBI #0), NMI BR5	15
I/O Adapter #1 (NBI #1), NMI BR5	15
Memory, NMI BR5	15
I/O Adapter #0 (NBI #0), NMI BR4	14
I/O Adapter #1 (NBI #1), NMI BR4	14
Console Terminal Receive	14
Console Terminal Transmit	14
Other Processor	14
Unassigned	<13:10>

3.4.3 System Control Block (SCB) Format

The system control block for the VAX 8800 system can have up to 64 pages of vectors that are allocated to three levels of interrupt vectors. The first page (page 0) is reserved for VAX architectural and NMI nexus vectors, and subsequent pages are allocated to I/O devices connected through the NBI adapters (maximum of 60 devices). Table 3-5 shows the configuration of SCB page 0.

Table 3-5 System Control Block Page 0 (000--1FF)

Vector (HEX)	Condition
00	NOP
04	Machine Check
08	Kernel Stack not Valid Abort
0C	Power Fail
10	Reserved/Privileged Instruction
14	Customer Reserved Instruction
18	Reserved Operand
1C	Reserved Operating Mode
20	Access Control Violation Fault
24	Translation Not Valid
28	Trace Pending
2C	Breakpoint Instruction
30	Not Used
34	Arithmetic
38	Not Used
3C	Not Used
40	CHMK
44	CHME
48	CHMS
4C	CHMU
50	Not Used
54	Not Used
58	Not Used
5C	NMI Fault
60 -- 6C	Not Used
70 -- 7C	Not Used
80	Interrupt Other Processor
84	Software Level 1
88	Software Level 2
8C	Software Level 3
90	Software Level 4
94	Software Level 5
98	Software Level 6
9C	Software Level 7
A0	Software Level 8
A4	Software Level 9
A8	Software Level A
AC	Software Level B

Table 3-5 System Control Block Page 0 (000--1FF) (Cont)

Vector (HEX)	Condition
B0	Software Level C
B4	Software Level D
B8	Software Level E
BC	Software Level F
C0	Interval Timer
C4 -- CC	Not Used
D0 -- DC	Not Used
E0 -- EC	Not Used
F0, FC	Not Used
F8	Console Terminal Receive
FC	Console Terminal Transmit
100 -- 11C	Not Used
120	I/O Adapter #0 (NMI)
124 -- 12C	Not Used
130	I/O Adapter #1 (NMI)
134 -- 13C	Not Used
140, 144	Not Used
148	Memory (NMI)
14C -- 1FF	Not Used

3.4.3.1 SCB Pagination -- The pages of the SCB following page 0 are allocated for devices connected through the NMI adapters. Interrupt devices are classified as either offsetable or directly connected.

3.4.3.2 Offsetable Devices -- Offsetable devices that pass interrupts from devices on another bus will use the first page following page 0.

3.4.3.3 VAXBI Node Direct Connected Devices -- Vectors for direct connected devices will be located beginning on the first even numbered page following the pages for offsetable devices. Depending on the amount of pages required for offsetable devices, a blank page may exist as a result of starting on an even numbered page with direct connected devices. One page will be allocated for each VAXBI.

3.4.3.4 SCB Format -- The format for the SCB vector offset values supplied by an NBI I/O adapter are shown in Figure 3-13.

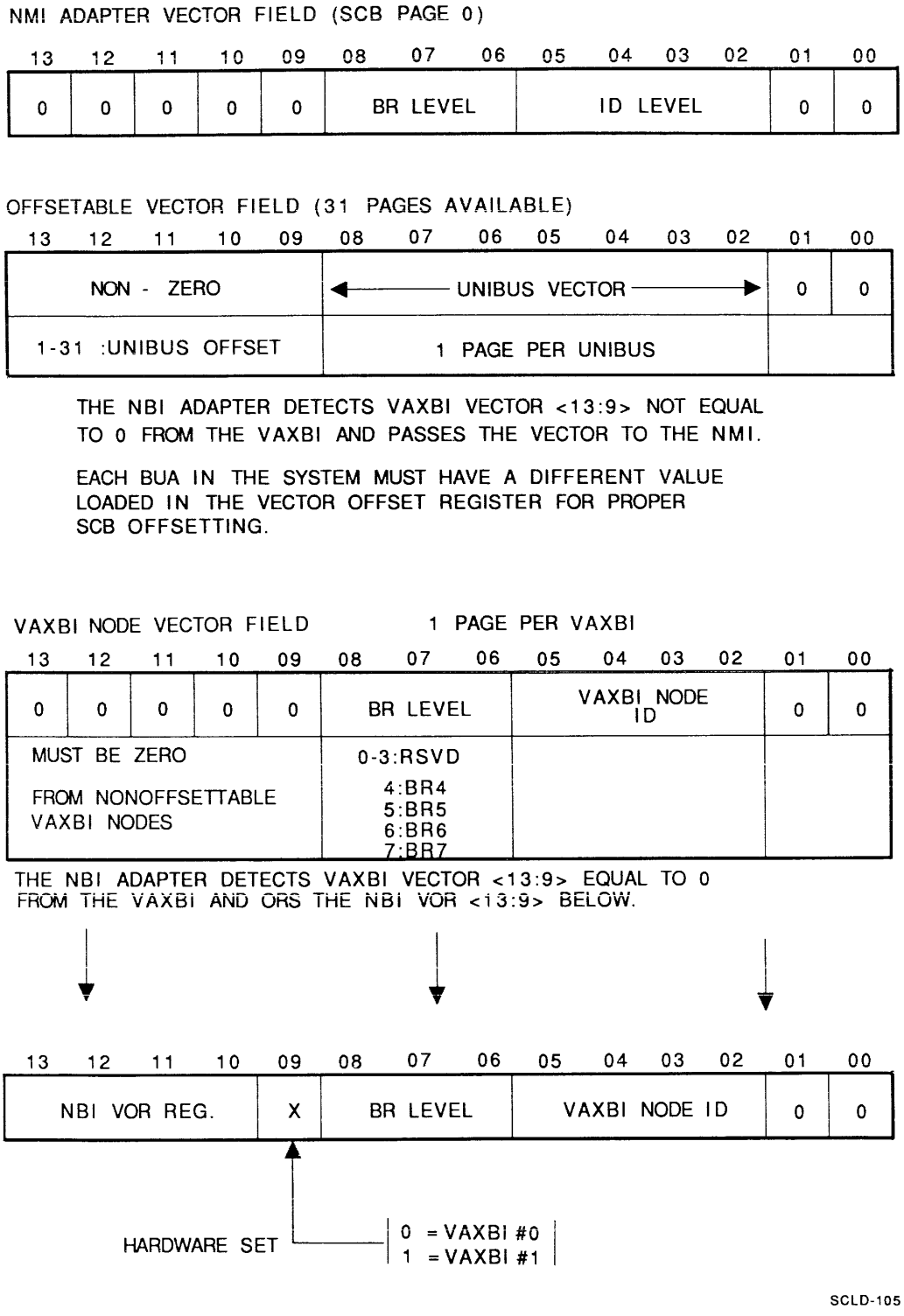


Figure 3-13 NBI I/O Adapter SCB Vector Offset Format

3.4.4 Machine Check Exception

A machine check exception occurs any time the processor detects an internal error and reports the error to the VMS operating system. The exception is taken into the system control block at virtual address SCBB +04. Communication between the VMS system and the CPU microcode during machine check handling is accomplished by means of the machine check status register.

3.4.4.1 Types of Exceptions -- Machine check exceptions consist of either a fault or an abort. Faults allow the instruction that created the exception to be retried because it assumes the state of the machine is stable. Aborts do not allow for a retry. Table 3-6 lists examples of faults and aborts that may occur in the VAX 8800 system.

Table 3-6 Machine Check Exception Examples

Type	Description	Location
FAULTS	Sequencer-to-Console Processor Register Data Parity Error	IBox
	Console Processor Register-to-Sequencer Data Parity Error	IBox
	Decoder RAM-to-Sequencer Data Parity Error	IBox
	Console Processor Register Input Data Parity Error	IBox
	Memory Failure	CBox
	Cache Tag Parity Error	CBox
	Memory Data Parity Error	CBox
	Virtual Address Parity Error	CBox
	Translation Buffer Tag Parity Error	CBox
	Translation Buffer Data Parity Error	CBox
	Data Read Error	CBox

Table 3-6 Machine Check Exception Examples (Cont)

Type	Description	Location
ABORTS	CS0 Parity Error	IBox
	CS1 Parity Error	IBox
	CS2 Parity Error	IBox
	NMI Sequencer Control Store Parity Error	CBox
	Decoder RAM Output Data Parity Error	IBox
FAULTS OR ABORTS DEPENDING ON SOURCE		
	B - Side Byte 3 Parity Error	EBox
	A - Side Byte 3 Parity Error	EBox
	B - Side Byte 2 Parity Error	EBox
	A - Side Byte 2 Parity Error	EBox
	B - Side Byte 1 Parity Error	EBox
	A - Side Byte 1 Parity Error	EBox
	B - Side Byte 0 Parity Error	EBox
	A - Side Byte 0 Parity Error	EBox

CHAPTER 4
DIAGNOSTIC AND MAINTENANCE AIDS

4.1 INTRODUCTION

This chapter contains a brief overview of the diagnostics and maintenance aids used in the VAX 8800 system. Included in this chapter is a description of the module keying verification and the power and environmental system. Refer to the System Diagnostic User's Guide for detailed information concerning diagnostic use.

4.2 GENERAL

The VAX 8800 diagnostic software package provides a bottom-up sequence for verifying the CPU cluster and the system. Tests and test programs should be run in the sequence they were designed for, in order to achieve maximum coverage and isolation.

A top-down test sequence is also provided for installation or repair verification if the run time of the bottom-up sequence exceeds field service limitations.

In a dual processor configuration, some microdiagnostic tests can be run concurrently in both processors, and others must be run sequentially in one processor at a time. The microdiagnostic monitor controls dispatching to microdiagnostics making this transparent to the user.

The default diagnostic mode in the dual processor configuration is to test both left and right CPUs. The user must explicitly select a single processor if both are not to be tested.

CAUTION

Diagnostics cannot be run in one processor while an operating system is running in the other.

VAX 8800 specific diagnostics (for example, NMI exerciser, cluster exerciser) require both processors to be running cooperative macroprograms.

The flowchart in Figure 4-1 shows a complete bottom-up test sequence for the VAX 8800 system. Normally, the sequence can be abbreviated, but when a failure is detected, more information will be gained by running every test.

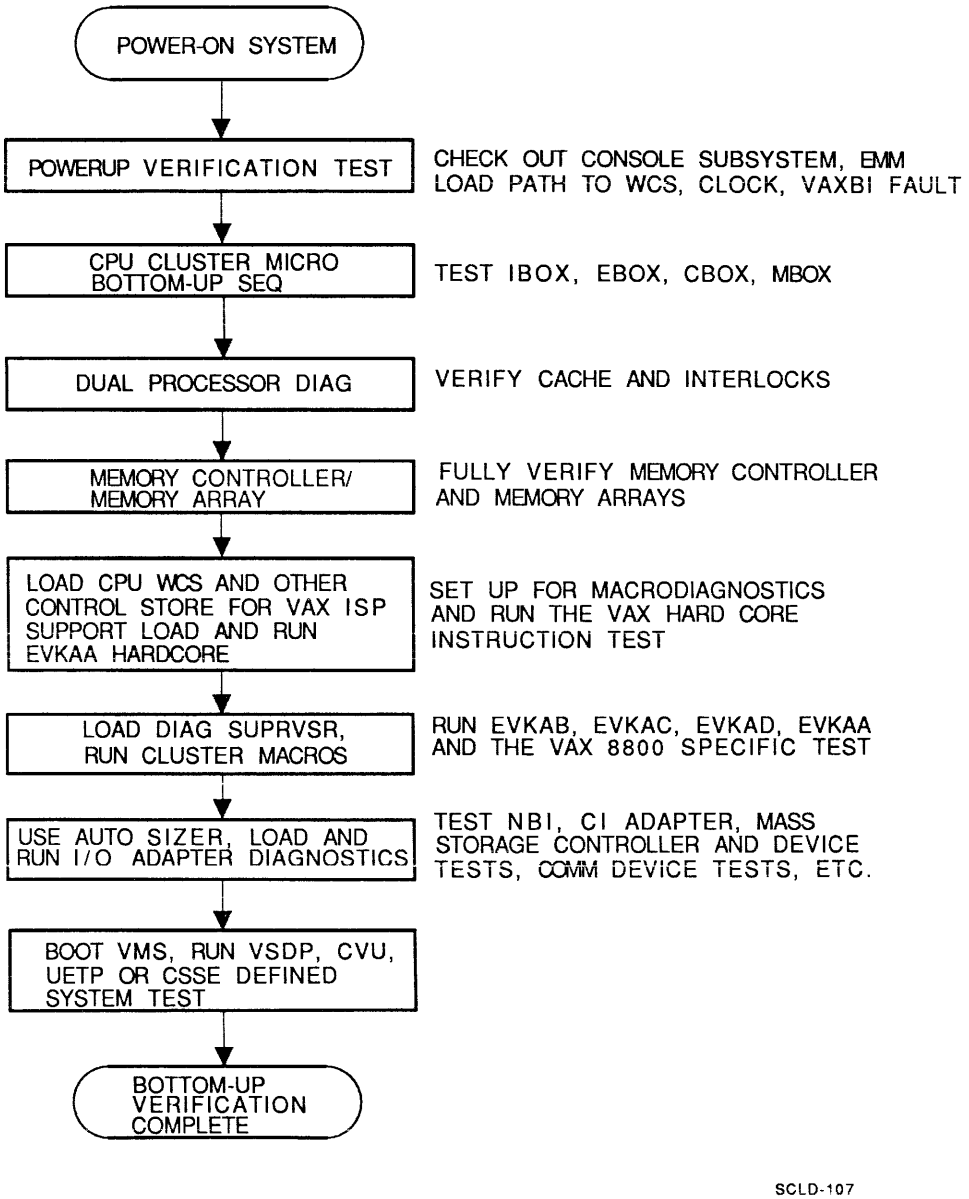


Figure 4-1 Bottom-Up Testing

4.3 DIAGNOSTICS

The diagnostics for the VAX 8800 system are divided into the following major areas:

- Console selftest
- Microdiagnostics
- Macrodiagnostics
- Customer runnable diagnostics

Each of the major diagnostic areas tests a certain section or sections of the VAX 8800 system.

4.3.1 Console Selftest

The console selftest is initiated automatically when the console power is applied and requires no intervention by the console operator. A failure during the console selftest results in a graphic identification of the failed area on the operator's console display. The console selftest checks:

- That the console hardware will run P/OS (professional operating system)
- Disk drives (floppy and hard)
- The RTI (real-time interface)

Refer to the PRO Series manuals for additional information concerning console error analysis.

4.3.2 Microdiagnostics

The VAX 8800 specific microdiagnostics are initiated from the console by entering commands at the console keyboard when in the micromonitor (MICMON) mode.

The micromonitor program is part of the console software and runs in the console processor. It is used to load, control, and monitor both console-based and WCS-based microdiagnostics. The microdiagnostics test:

- The left and right CPUs
- The clock/console interface
- The NMI
- Memory

4.3.2.1 CSM Commands -- A subset of console commands requiring console support microcode (CSM) is available while running diagnostics. If CSM commands are executed between tests, the test cannot be continued. Because some CSM code may use functions that have not yet been tested, some tests may not function correctly after executing CSM commands from the console.

To execute console CSM commands while running diagnostics, enter the START/C 3000 command following the MIC> prompt. This will force the UPC to the start of diagnostic CSM code and turn the clocks ON. The following commands will be valid:

- DEPOSIT/PHYSICAL/LONG
- DEPOSIT/PHYSICAL/BYTE
- EXAMINE/PHYSICAL/LONG
- EXAMINE/PSL
- DEPOSIT/PSL
- EXAMINE/SDF
- DEPOSIT/SDF
- EXAMINE/TEMP
- DEPOSIT/TEMP
- DEPOSIT/CACHE

Executing the EXAMINE/DEPOSIT/TEMP commands while running diagnostics (using diagnostic CSM code) requires using different addresses for the TEMP registers. Table 4-1 lists the TEMP register addresses to be used for diagnostic CSM.

Table 4-1 TEMP Register Addresses for use with Diagnostic CSM

Temp Register	Address (Hex)
WDR	0
T0	1
T1	2
T2	3
T3	4
T4	5
MT1	6
MT2	7
MT3	8
MD0	9
MD1	A
MD2	B
MD3	C
MD4	D
MD5	E
MD6	F
MM.MDR	10
VA	11
BACK.UP.PC	12
INVALID	13 - 1F

The microcode error registers can be read when running diagnostic CSM code, by using the EXAMINE/IPR command with the error register addresses listed in Table 4-2.

Table 4-2 Microcode Error Register Addresses

Register	Address	Format of Returning Register Data
Cache Register	0	Bits<2-0> = NMI ON, CACHE ON, and MM ENABLE, respectively
Cache Err Reg Byte 2	1	Bits<5-0> = CER Byte 2
Cache Err Reg Byte 1	2	Bits<5-0> = CER Byte 1
Cache Err Reg Byte 0	3	Bits<5-0> = CER Byte 0
NMI Error Address Reg	4	Bits<29-1> = PA<29-1>
NMI Fault Reg Byte 1	5	Bits<4-0> = NFSR Byte 1
NMI Fault Reg Byte 0	6	Bits<4-0> = NFSR Byte 0
NMI Silo	7	Bits<19-0> = NMI SILO Byte 2, Byte 1, and Byte 0, respectively
EBox Error Reg	8	EBox Error Reg
IBox Error Reg	9	IBox Error Reg

The NBI diagnostic routines test for error conditions and detect the condition of the NMI. The cursor prompt for the microdiagnostics is 'MIC>'.

4.3.2.2 Status and Error Information -- During the execution of microdiagnostics, as much information as possible is made available to the operator. Both section and test tracing information is provided.

Section tracing displays the name and revision level of each section that is being executed. Test tracing displays the name and a brief description of each test being executed.

An extented error printout is available that provides the SYNCH UPC and CONT UPC fields for hardware debug. SYNCH UPC provides the WCS address at the top of the loop, and CONT UPC provides the WCS address of the microinstruction just beyond the error loop.

Example 4-1 shows a sample micromonitor/microdiagnostic display output with section and test trace enabled. Example 4-2 shows a sample output with an error message.

```
MIC> DIAGNOSE

TESTING LEFT & RIGHT CPU
EZKBA - REV X.Z

TEST 1 (Description of Test 1)
TEST 2 (Description of Test 2)
 ..
 ..
 ..
EXKBB - REV X.Z

TEST 1 (Description of Test 1)
 ..
 ..
END OF PASS  [NO ERRORS DETECTED]

MIC>
```

Example 4-1 Sample Microdiagnostic Display Output

```
MIC> SELECT RIGHT
MIC> DIAGNOSE

TESTING RIGHT CPU
EZKBA - REV X.Z

TEST 1 (Description of Test 1)
TEST 2 (Description of Test 2)

*** RIGHT CPU FAILED *** EZKBA *** TEST 02 *** ERROR 01 ***

     FAILING HW: M1234, M5678, M9012 ....
     SYNCH UPC:  aaaa      CONT UPC: bbbb

MIC>
```

Example 4-2 Sample Microdiagnostic Error Display

4.3.2.3 Micromonitor Error Messages -- The following is a list of possible micromonitor error messages:

- Unexpected trap occurred (type of trap)
- Unexpected interrupt occurred (type of interrupt)

NOTE

An unexpected error detected by a diagnostic usually indicates that the WCS code went off to some illegal location.

- Illegal error number received from WCS-based test
- Illegal message received from WCS-based diagnostic
- Failure sending message to diagnostic
- Power not on
- Illegal section name(s)
- No section currently loaded
- Illegal test number(s)
- Maximum test number exceeded
- Micmon protocol error
- Unable to load console-based section
- Unable to load WCS-based section
- Unable to load diagnostic CSM

4.3.3 Macrodiagnostics

The macrodiagnostics are both VAX 8800 system specific and generic diagnostics. Macrodiagnostics (except EVKAA) are run under the VAX diagnostic supervisor (DS>).

The macrodiagnostics are used to test and isolate problems in the:

- CPU kernel (both left and right CPU)
- I/O subsystem
- I/O adapters
- I/O devices such as the NBIA, NBIB, VAXBI, SI

The macrodiagnostics are also used to exercise various portions of the VAX 8800 system as well as repair verification. The cursor prompt for the macrodiagnostics is: DS>.

NOTE

In order to fully test the VAX 8800 system, both the macrodiagnostics and the microdiagnostics must be run.

Table 4-3 lists and describes the major tests available in the macrodiagnostic programs.

Table 4-3 Macrodiagnostic Tests

Test	Description
EVKAA	Function Verification of the VDS Kernel Instruction Set
EVKAB	Basic Instruction Exerciser. Runs Native Mode Instructions in: Integer Arithmetic Variable-Length Bit Fields Control Instructions Queue Instructions Character Strings Decimal Strings
EVKAC	VAX Floating-Point Exerciser
EVKAE	VAX Privileged Architecture Diagnostic
EZKAX	VAX 8800 System-Specific CPU Cluster Exerciser. EZKAX tests: Internal Processor Register Access Processor Power Failure Processor Halts User Writable Control Store Machine Checks Dual-Processor Communication
EZCJA	NMI-to-VAXBI Adapter Diagnostic Test
EZXCA	NMI Activity Diagnostic Test

4.3.4 Customer Runnable Diagnostics

Customer runnable diagnostics (CRD) are a packaged set of user-friendly diagnostics designed to enable the user to isolate system problems to the failing option. CRDs are designed with the customer in mind, and are fast and easy to use. All test results are printed in English.

When the diagnostic media is loaded and the TEST command is given, the test proceeds automatically under the control of the micromonitor. The CRDs will test the system and inform the operator of the status of the option being checked, the time required to complete the test, and whether that particular option passed or failed.

Customer runnable diagnostics provide the user with two options:

1. Auto-Test mode
2. Menu mode

4.3.4.1 Auto-Test Mode -- The auto-test mode verifies the operation of the CPU internal options, VMS system disk, and the diagnostic load disk. The test requires approximately 20 minutes to complete.

4.3.4.2 Menu Mode -- Menu mode is an extension of auto-test that allows verification of all system supported options. The test run time is dependent upon the number of installed system options.

4.3.5 Remote Diagnostics

A COMM port on the rear of the console PRO-38N is available for remote access to the VAX 8800 system. From a console software point of view, the remote port is identical to the local operator's console. For safety reasons, the POWER ON command is not accepted from the remote console. The DISABLE PRINTER and DISABLE REMOTE MONITORING commands will not be executed from the remote terminal.

When a remote port is connected and is operating in the console mode, the local port cannot execute commands that can change the state of the VAX CPUs. This restriction prevents a local operator from inadvertently changing the machine state while remote debugging is taking place. The local operator is limited to control and status commands during a remote diagnostic operation.

4.4 POWER/ENVIRONMENTAL SYSTEM

When the main circuit breaker of the VAX 8800 system is turned to the ON position, power is applied to the cooling blower and the NBox. The NBox supplies a dc voltage to the EMM for the module keying test prior to application of power to the backplane.

4.4.1 Module Placement Verification

Module placement verification is performed in two steps. The first step uses a module key test to ensure that there are no modules inserted in a slot that will cause damage to the module when power is applied. The second step verifies that the correct module is installed in the intended slot.

4.4.1.1 Module Key Test -- The key test verifies that VAX 8800 modules are installed correctly in one of the following four module groupings:

1. CPU modules
2. Clock module
3. Memory control logic module
4. NBIA module

The modules within a group have similar power requirements and cannot be damaged, provided that the module is installed in the correct grouping, even if it is in the wrong slot (for example, a CPU decoder module installed the in shifter module slot).

There is no requirement to test memory array or VAXBI modules, as they will not fit into the other slots. The memory array modules use three segments of pins, and the VAXBI modules use a five-segment card. CPU, clock, MCL, and NBIA modules use two segment cards.

The EMM parallel key circuit uses the key sense input shown in Figure 4-2 to enable it to turn on the power to the system. The key sense signal is a single wire looped from the EMM to the computer backplane and back to the EMM.

The EMM compares the Key Sense input with an internal reference voltage and asserts a signal that holds the EMM in a reset state if an error is detected. Holding the EMM in a reset state, keeps the EMM from receiving POWER ON commands from the console. Failure of the module key test illuminates the KEY FAULT LED on the front of the EMM. The "Electrical Key Override" switch on the front of the EMM is not used in the VAX 8800 system. Figure 4-3 shows how the Key Source signal is looped through the four module groups and returned to the EMM as Key Sense.

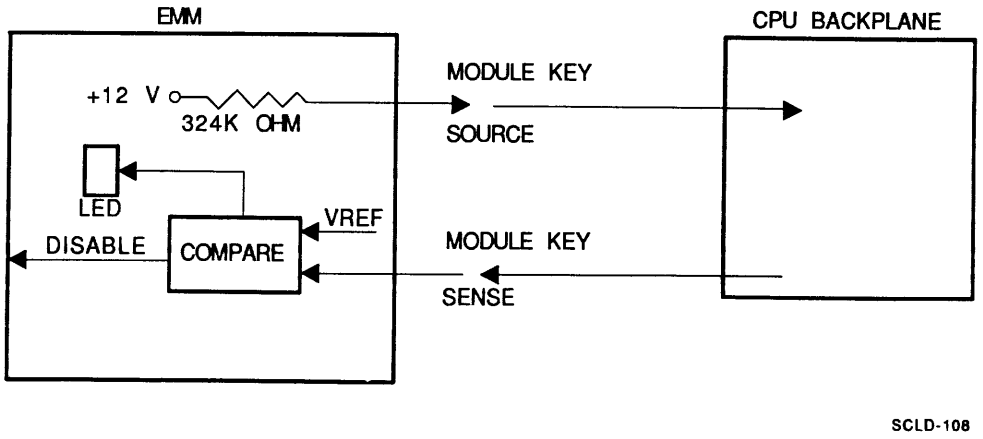


Figure 4-2 Module Keying Test Simplified Block Diagram

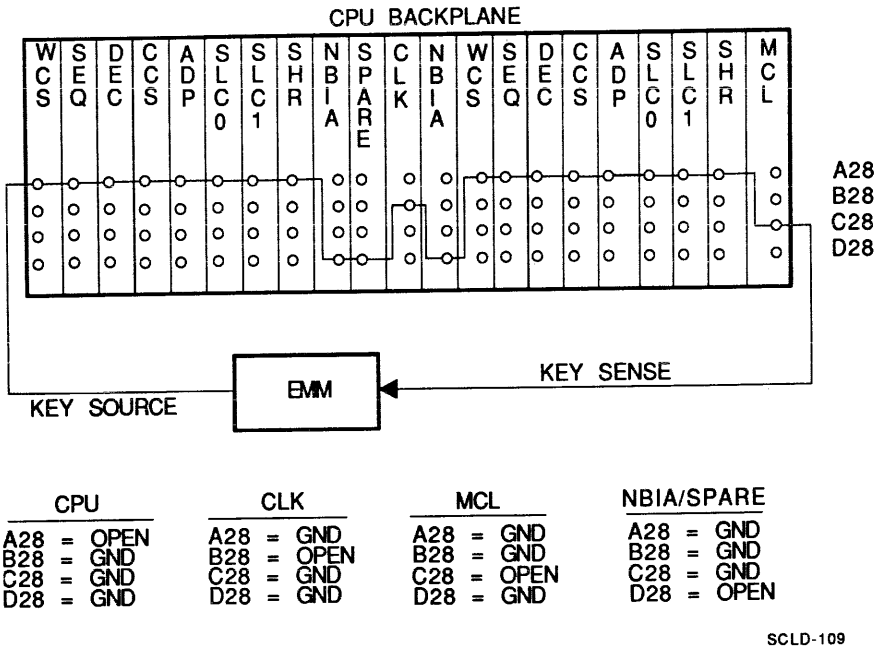


Figure 4-3 Module Key Test Connections

4.4.1.2 Module Placement -- When the key test verifies that the modules are installed in a correct grouping and power is applied, the Vbus is used to determine that the correct module is inserted in the clock and CPU slots.

The Vbus test that is used to test for correct module placement within the CPU group will not work if the clock module is not installed in the correct slot. (The Vbus control registers are located on the console interface portion of the clock module.)

Console support microcode is used to check for the presence of the MCL and NBI modules.

4.4.2 Power Monitoring/Error Reporting

The environmental monitoring module in the power system complex has sensors for monitoring cooling system airflow, cabinet temperatures, and line and system operating voltages. The EMM also contains actuators that control the application and removal of the system power and the battery backup unit in response to console commands.

The EMM has a microprocessor that communicates with the console processor and performs a power shutdown when the environmental parameters are not within allowable specifications established by the console.

4.4.2.1 Default Mode Error Reporting -- When system power is first applied and the console has not yet established communications with the EMM, the EMM uses a set of default parameters to monitor power system performance. The default parameters are stored in the EMM ROM and are loaded into the EMM RAM during the EMM boot procedure.

If the EMM detects an error during the EMM selftest, a single character error code is sent to the console to inform the console of a test failure. The EMM repeats the failing test until it passes successfully or until the console reaches the point in the power-up sequence where the console is ready to communicate with the EMM.

When the console receives the error message, or fails to establish communications with the EMM, the console software prints an error message on the operator display and stops the power-up sequence.

The following default parameter error codes are used by the EMM when the console has not reached the point in the power-up sequence to pass monitoring parameters:

Voltage	DMCODE 0-11; Regulator A-L
Temperature	DMCODE 12-15; Therm T1-T4
Temperature	DMCODE 16-18; Diff D12-D14
Module OK Transition	DMCODE 19
Air Flow Fault	DMCODE 20-21; AFF1-2
Battery Backup Fault	DMCODE 22

4.4.2.2 Operational Error Reporting -- During the power-up and initialization process, the console passes a set of parameters to the EMM to replace the default parameters used prior to establishing communications.

When one of the parameters monitored by the EMM no longer meets specifications, the EMM sends an unsolicited warning message to the console display and begin a five minute timer. If the problem is not resolved in five minutes, the EMM asserts the TOTAL OFF command and shuts down the system power. Example 4-3 shows a sample EMM warning message display.

```
**** WARNING, ENVIRONMENTAL MONITORING MESSAGE RECEIVED

VOLTAGE EXCEPTION REPORTED BY EMM %067, 5-AUG-85  10:00:00
REG A transition from 'In Range' to 'Below Range'
pol: +        mea: 0001       input: 2
mar: 0000     lolim: 0000     hilim: 0000

AUTOMATIC SHUTDOWN IN PROGRESS (BLINKING)
reg:   A   B   C   D   E   F   G   H   I   J   K   L   KAC LAC KEY
mok:   ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok

sig:   AF1 BBF CBF AF2 ACL DCL PEN PER
val:    0   0   1   0   1   0   1   1
```

Example 4-3 EMM Warning Message

4.4.3 Voltage Margining

The voltage regulators in the VAX 8800 system can be margined up or down by the console operator, using the SET MARGIN command.

The Margin Enable (MARGEN) and Margin Hi Lo (MARHILO) Registers shown in Figure 4-4 are EMM registers used to margin the Module Power Supply Regulators. The MARGEN Register enables voltage margining for the appropriate regulator when the corresponding bit is set. The same bit in the MARHILO register determines if the regulator voltage is raised or lowered.

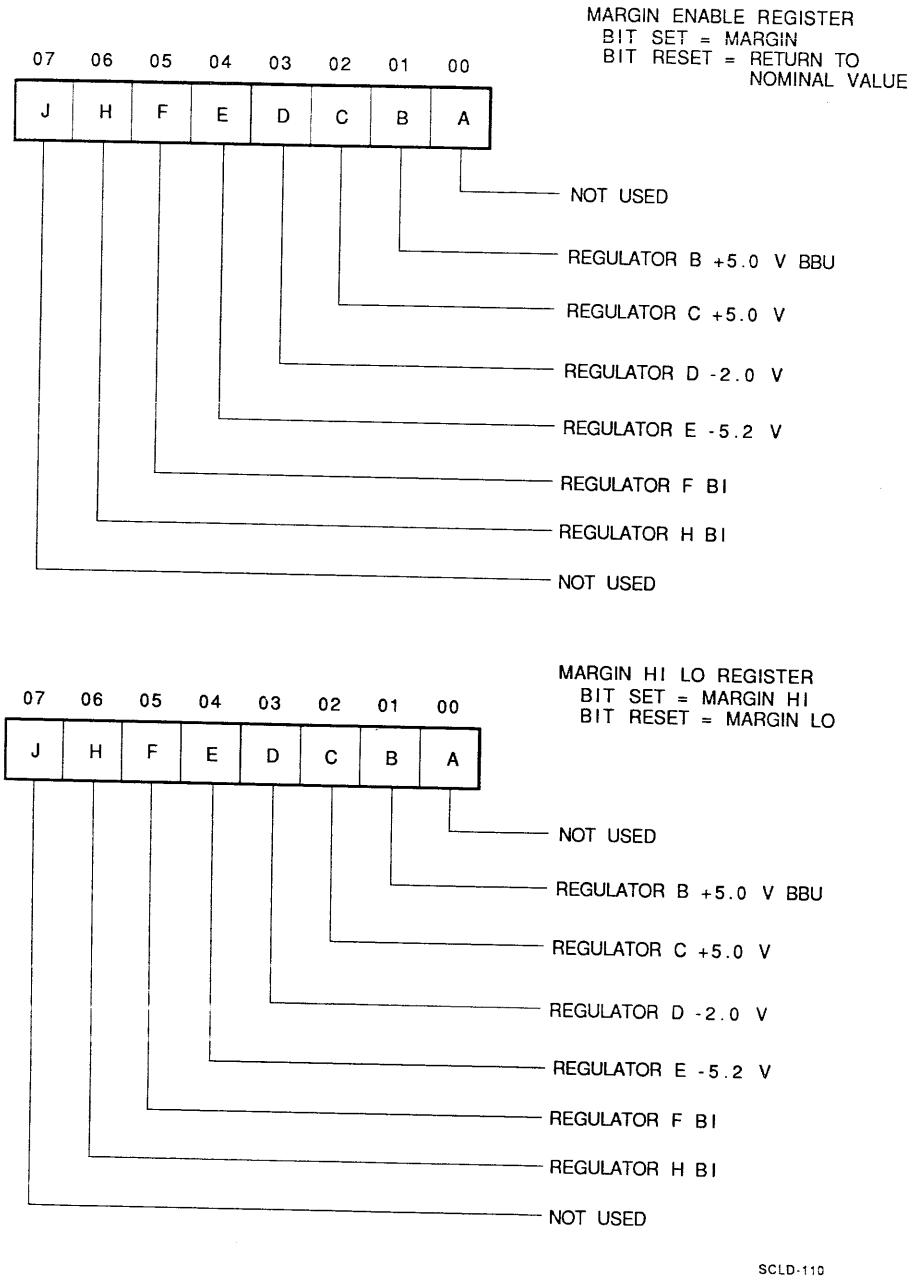


Figure 4-4 Margin Enable and Margin Hi Lo Registers

4.5 MAINTENANCE AIDS

4.5.1 Machine Check Logout Stack

A machine check exception is taken any time the processor detects an internal error. Each machine check exception must be detected and reported to VMS in an established way. Parameters from the environment in which the error occurred have to be collected from various locations in the machine and pushed onto a Machine Check Logout Stack. The data in the stack is used by VMS to determine whether to continue to run normally or halt and return to Console Mode. Figures 4-5 through 4-15 show the format of the Machine Check Logout Stack and associated error registers.

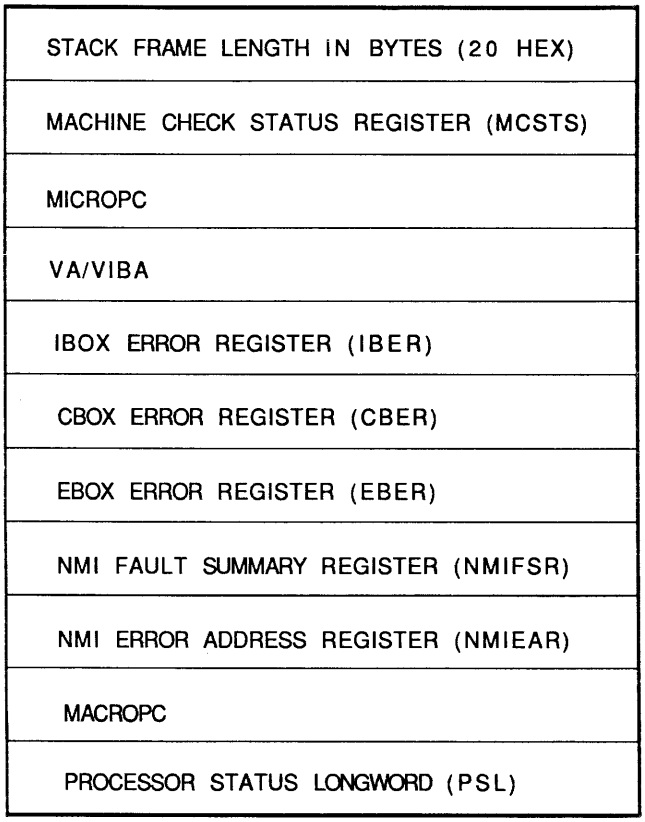


Figure 4-5 Machine Check Logout Stack

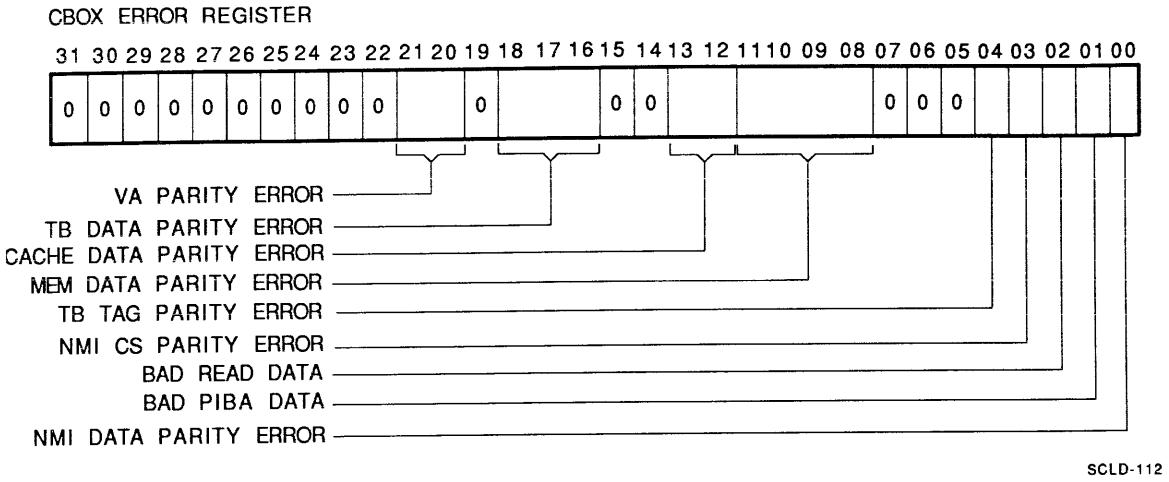


Figure 4-6 CBox Error Register

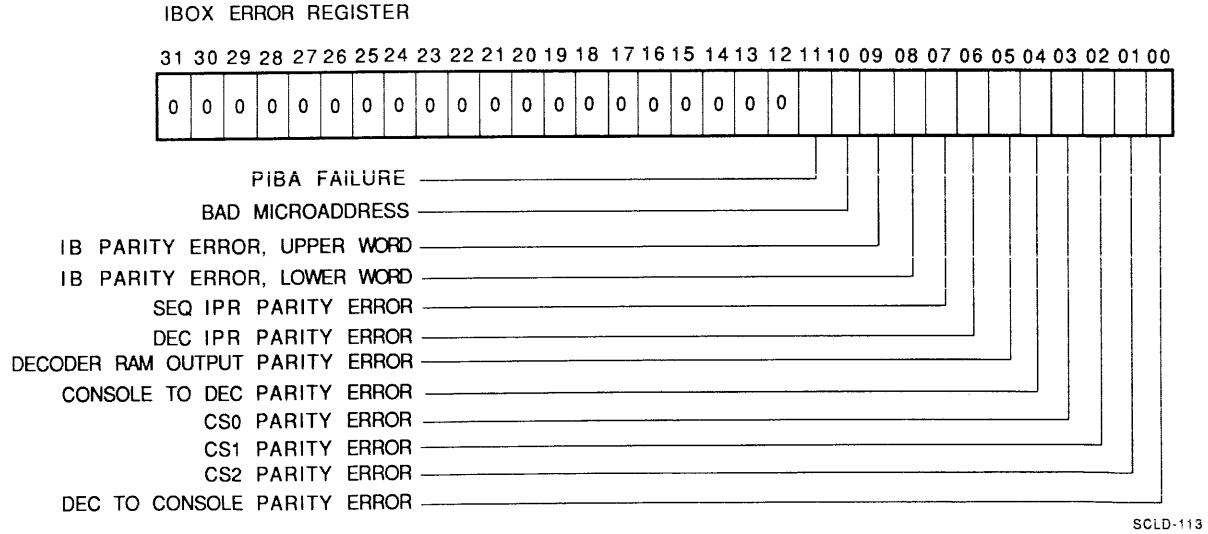


Figure 4-7 IBox Error Register

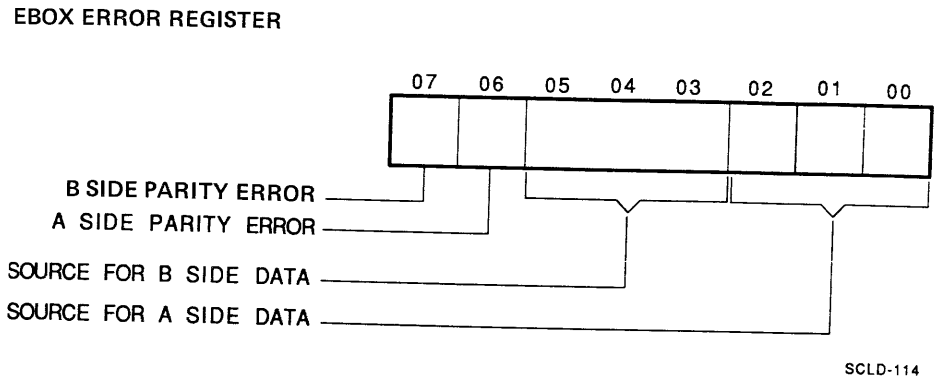


Figure 4-8 EBox Error Register

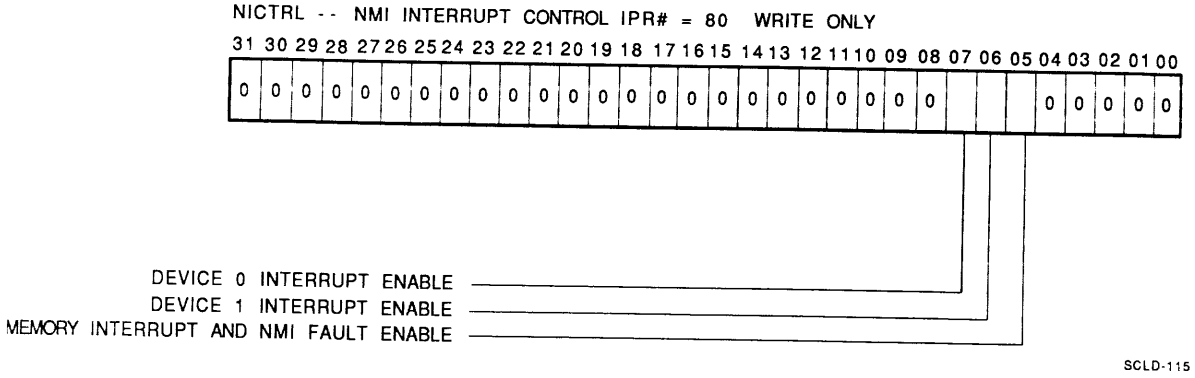
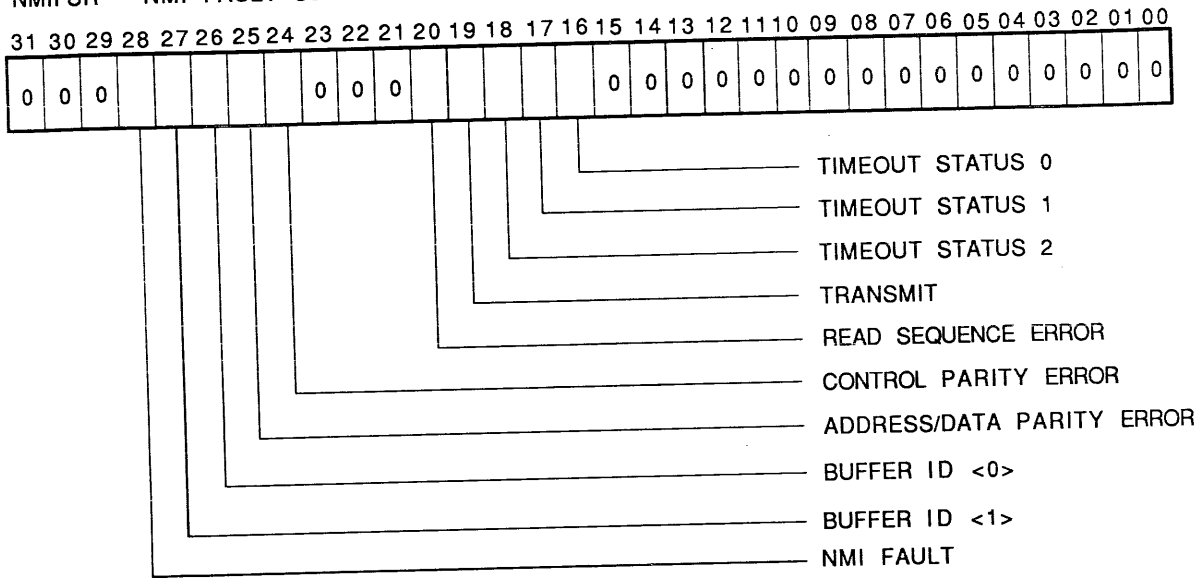


Figure 4-9 NMI Interrupt Control Register

NMIFSR - NMI FAULT SUMMARY IPR# = 82 READ ONLY



BUF ID <1>	<0>	BUFFER CODE
0	0	NO TIMEOUT
0	1	WRITE TIMEOUT
1	0	READ TIMEOUT
1	1	PIBA TIMEOUT

TIMEOUT CODE VALID IF BUF ID INDICATES TIMEOUT

<2>	<1>	<0>	TIMEOUT CODE
0	0	0	NO TIMEOUT
0	0	1	RESERVED
0	1	0	INTERLOCK TIMEOUT
0	1	1	NO RETURN READ DATA
1	0	0	NO ACCESS - NO RESPONSE
1	0	1	NO ACCESS TO BUS
1	1	0	NO ACCESS - INTERLOCKED
1	1	1	NO ACCESS BUSY

SCLD-116

Figure 4-10 NMI Fault Summary

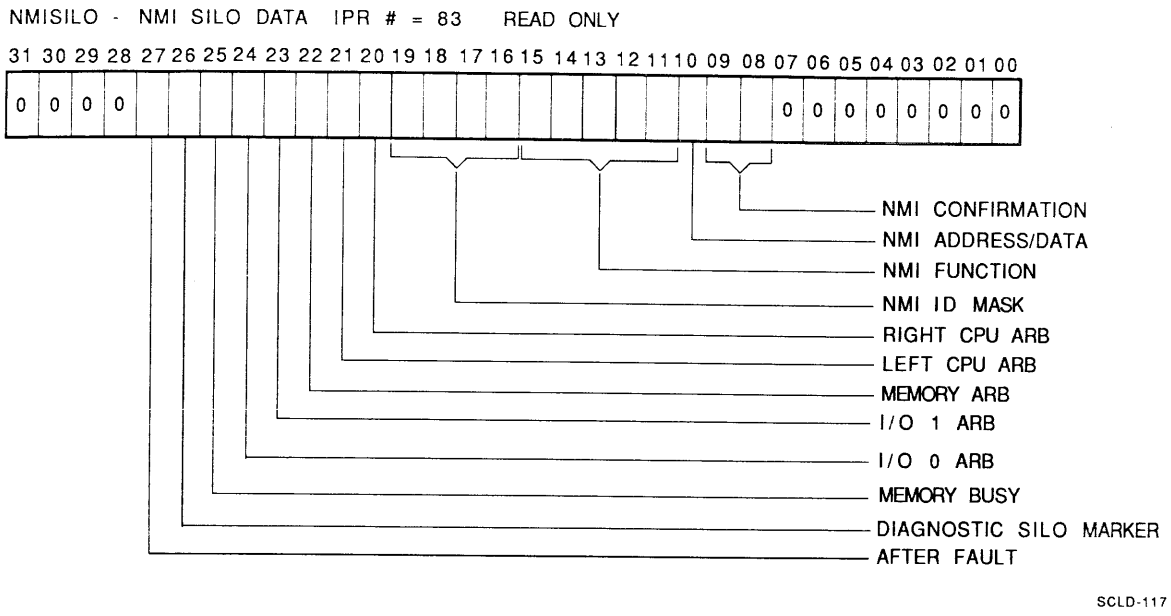


Figure 4-11 NMI Silo Data

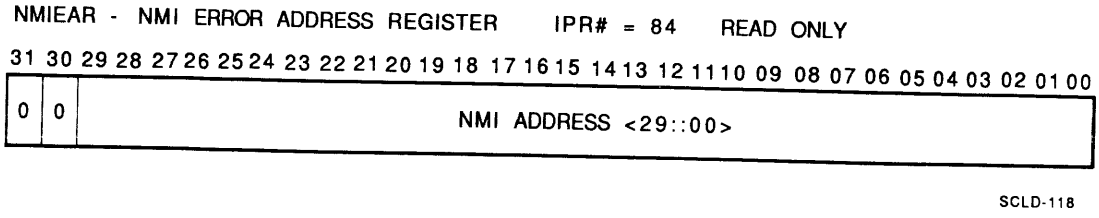


Figure 4-12 NMI Error Address Register

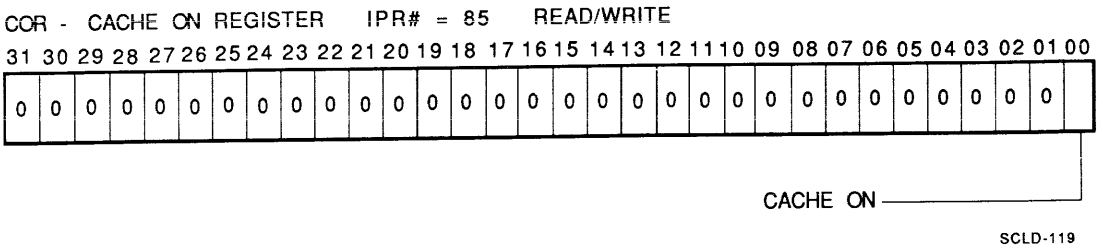


Figure 4-13 Cache ON Register

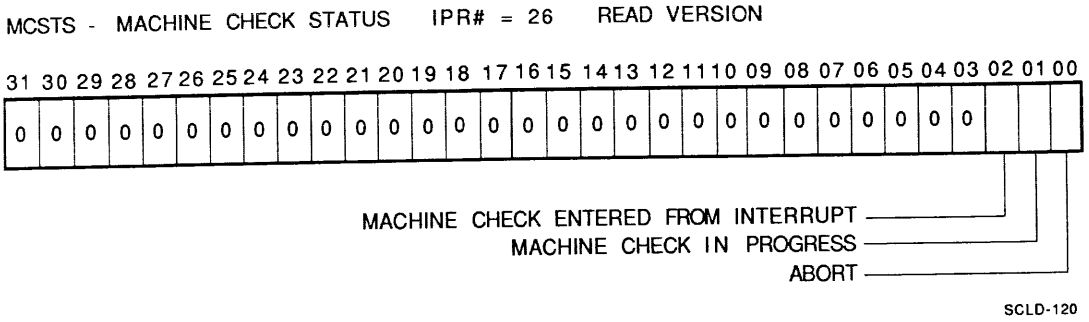


Figure 4-14 Machine Check Status

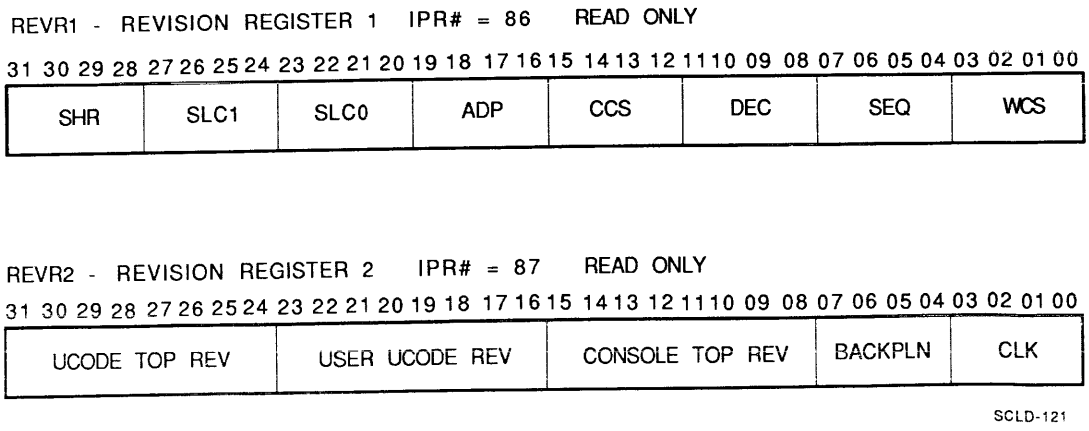


Figure 4-15 Revision 1/2 Registers

SECTION 2
SYSTEM BUS SUMMARY

CHAPTER 1
MEMORY INTERCONNECT (NMI)

1.1 INTRODUCTION

The Memory Interconnect (NMI) is a synchronous backplane bus that interconnects the VAX 8800 primary and attached CPUs, memory, and I/O adapters as shown in Figure 1-1. Terms used in the NMI description that follows are defined in Table 1-1.

The NMI has 32 multiplexed address/data lines plus control lines and system clocks. It also has address/data and control line parity. Bus arbitration is centralized (bus arbitrator in left CPU) and bus signals are ECL, except for FET-driven AC LO and DC LO signals.

The primary CPU services all interrupts generated on the NMI. (The primary CPU may be either the left or right CPU in dual-processor systems.) It also performs all I/O data transfers on the NMI.

The NMI supports the following read/write transactions:

- Write transactions: longword, quadword, and octaword
- Read transactions: longword, octaword, and hexword

Longword, quadword, octaword, and hexword transactions are used to transfer memory data. Only longword transactions are used for I/O data transfers. If desired, memory data transfers may be hardware interlocked by the memory nexus (interlocked reads/unlock writes).

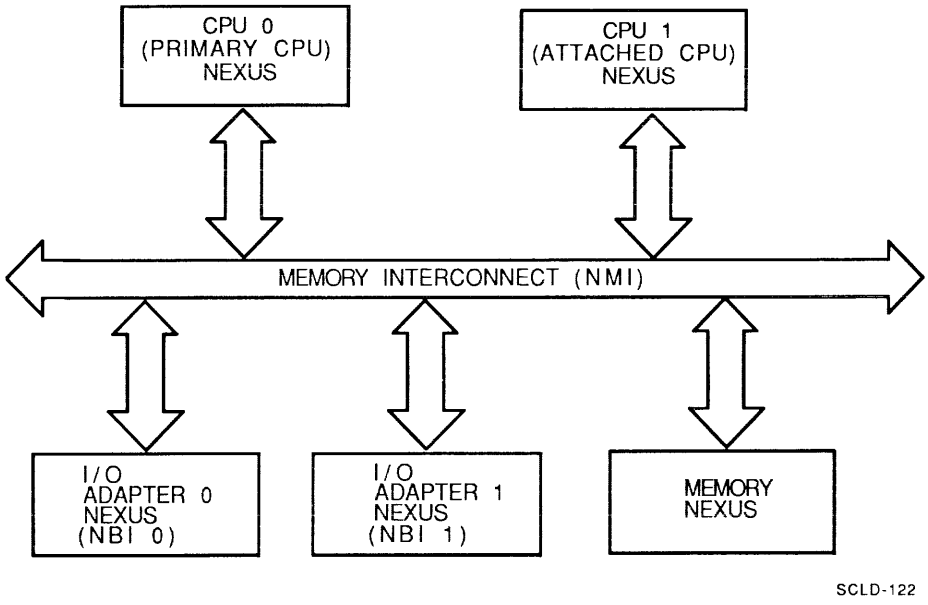


Figure 1-1 Memory Interconnect (NMI)

Table 1-1 Glossary of NMI Terms

Term	Definition
Nexus	A hardware block that physically connects to the NMI.
Transaction	A task performed on the NMI (read/write). A transaction consists of one or more transfers. For example, a write transaction consists of one transfer (command/address and data). A read consists of two or three transfers (command/address followed by one or two transfers of return read data).
Transfer	The discrete transfer of data that occurs after a nexus gets control of the NMI and before it relinquishes the NMI. A transfer may consist of one or more bus cycles. For example, (1) the command/address cycle of a read transaction or (2) the command/address cycle and the data cycle(s) of a write transaction.
Commander	The nexus that initiates a transaction. For example, if the CPU initiates a read from memory, the CPU is the commander.
Responder	The nexus that is the object of a transaction. For example, if the CPU initiates a read from memory, the memory is the responder.

1.2 BASIC FUNCTIONS

The NMI performs six major functions.

1. Memory read/write operations -- By means of bus read/write transactions, allows the CPUs and I/O adapters to access memory. (Memory read/write operations by the I/O adapters are DMA operations initiated by the I/O devices connected to the adapters.)

2. I/O register read/write operations -- Again by means of bus read/write transactions, allows the primary CPU to access control/status registers in the memory, I/O adapters, and the I/O devices connected to the adapters.

3. Interrupt handling -- Transmits interrupt requests generated by the memory and I/O adapters to the primary CPU. (Interrupt requests by an I/O adapter may be generated by the I/O adapter itself or by the I/O devices connected to the adapter.)

4. System synchronization -- Provides system clocks to all nexus.

5. System initialization -- Allows console to initialize all nexus. (UNJAM console command asserts UNJAM signal on NMI.)

6. Power loss warning -- Provides AC LO and DC LO signals to all nexus.

1.3 NMI SIGNALS AND TIMING

Generally, bus signals are asserted and negated at the beginning of a bus cycle, which is phase B of the system clock. Also, bus signals are received and latched using phase A of the system clock. Refer to Figure 1-2.

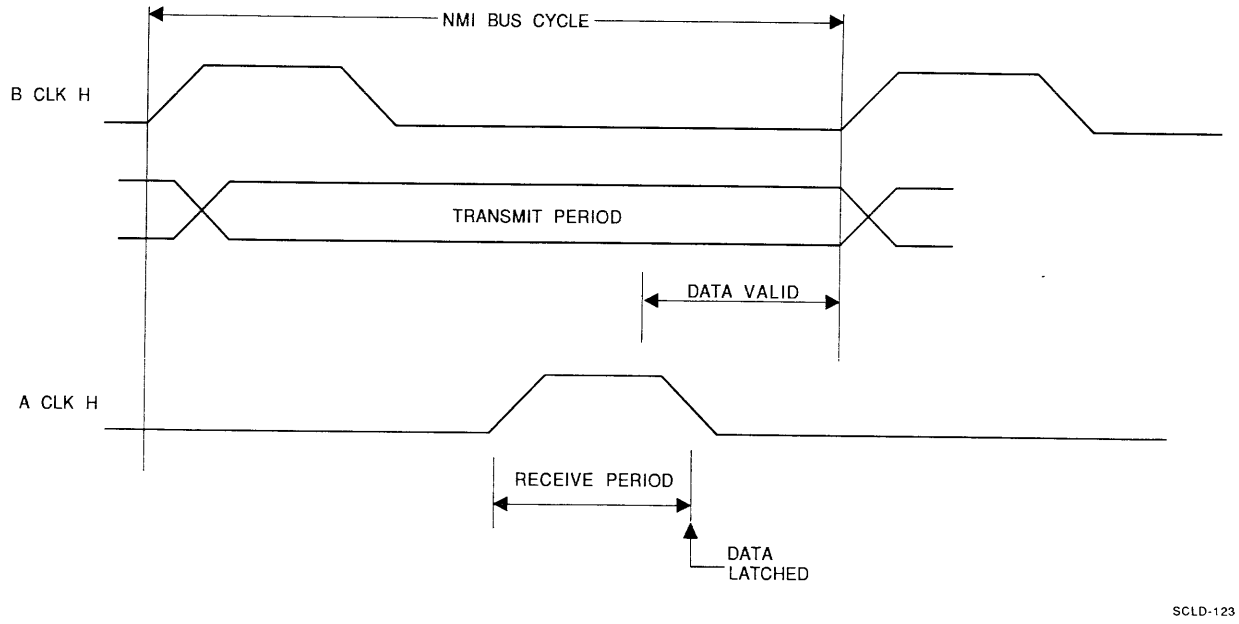
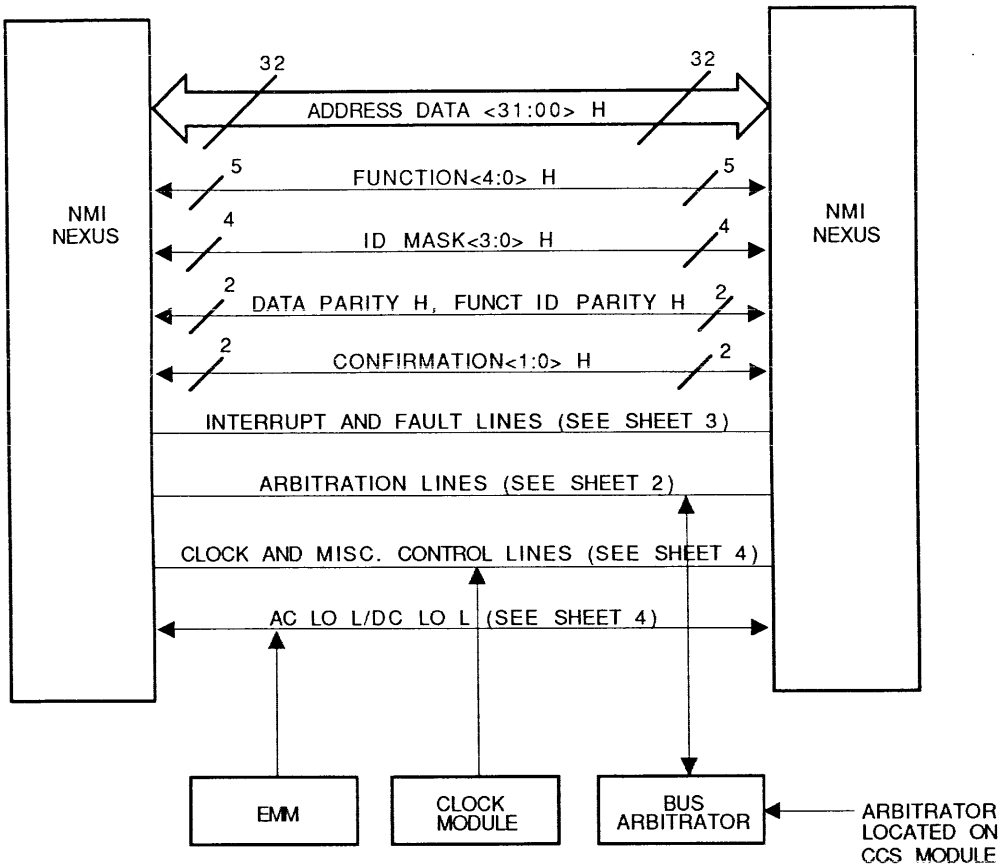


Figure 1-2 Basic NMI Timing

All the NMI signals are shown in Figure 1-3 and defined in Table 1-2. Most signals are generated by (and only connect to) the NMI nexus. However, the system clocks and some miscellaneous control signals originate on the clock module, and the bus arbitration signals connect to the bus arbitrator in the left CPU (in the CCS module). AC LO and DC LO signals are generated by the EMM in the power subsystem.



ARBITRATION LINES (17)

MEMORY HOLD H (1)	MEMORY ARB H (1)	MCL BUS EN (1)
LEFT CPU HOLD H (1)	I/O 0 ARB H (1)	I/O 0 BUS EN (1)
RIGHT CPU HOLD H (1)	I/O 1 ARB H (1)	I/O 1 BUS EN (1)
I/O 0 HOLD H (1)	LEFT CPU ARB H (1)	LEFT CPU BUS EN (1)
I/O 1 HOLD H (1)	RIGHT CPU ARB H (1)	RIGHT CPU BUS EN (1)
MEMORY BUSY H (1)	MEMORY BUSY ARB H (1)	

INTERRUPT LINES (14)

LCPU MEM INTR H (1)	RCPU MEM INTR (1)
DEV0 LINTR H (1)	DEV0 RINTR H (1)
DEV1 LINTR H (1)	DEV1 RINTR H (1)
DEV0 LINTR LVL<1:0> H (2)	DEV0 RINTR LVL<1:0> H (2)
DEV1 LINTR LVL<1:0> H (2)	DEV1 RINTR LVL<1:0> H (2)

MISC. CONTROL LINES (10)

UNJAM <2:0> H (3)	
SLOW CLOCK ENABLE H (1)	
SLOW MODE H (1)	
HARBINGER <2:0> H (3)	
RESET <1:0> H (2)	

FAULT LINES (5)

FAULT DETECT<3:0> H (4)
FAULT H (1)

SYSTEM CLOCKS

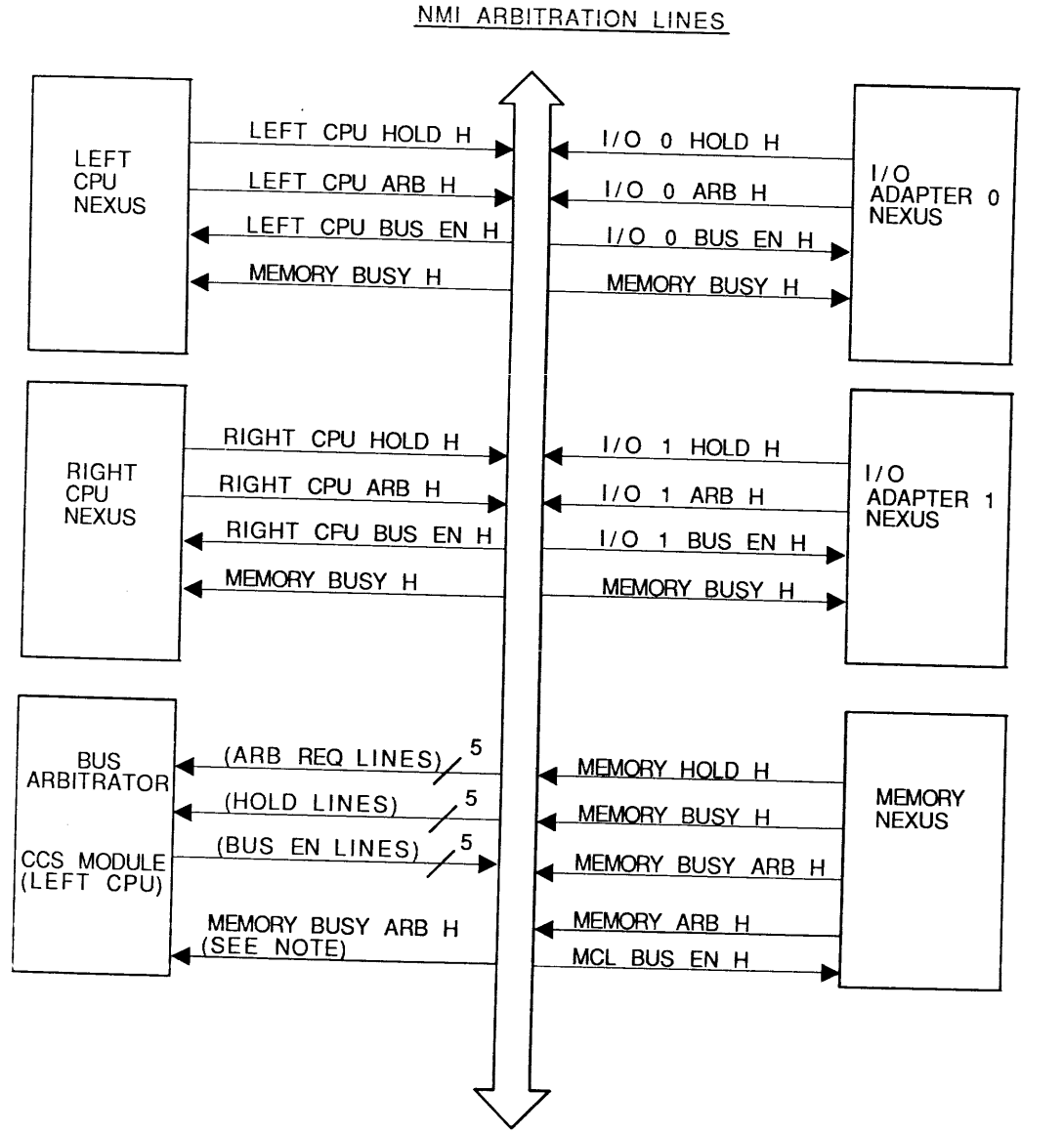
(SEE SHEET 4)

BACKPANEL SELECT LEVELS

I/O SEL <1> H, ONE LEVEL H

SCLD-124

Figure 1-3 NMI Signals (Sheet 1 of 4)



NOTE: MEMORY BUSY ARB IS NOT AN ARBITRATION REQUEST LINE. IT IS A COPY OF MEMORY BUSY THAT CONNECTS TO THE BUS ARBITRATOR.

SCLD-125

Figure 1-3 NMI Signals (Sheet 2 of 4)

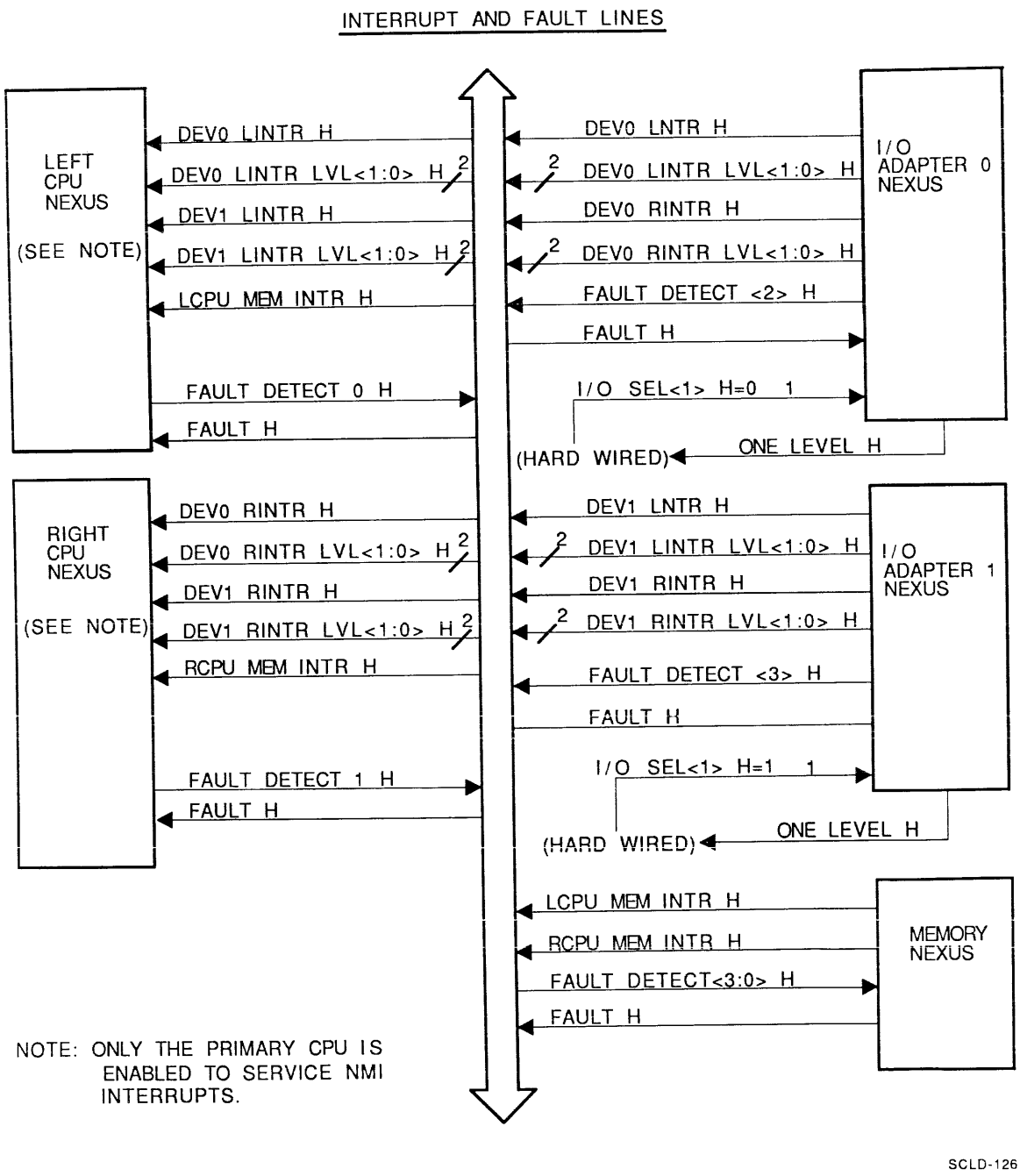
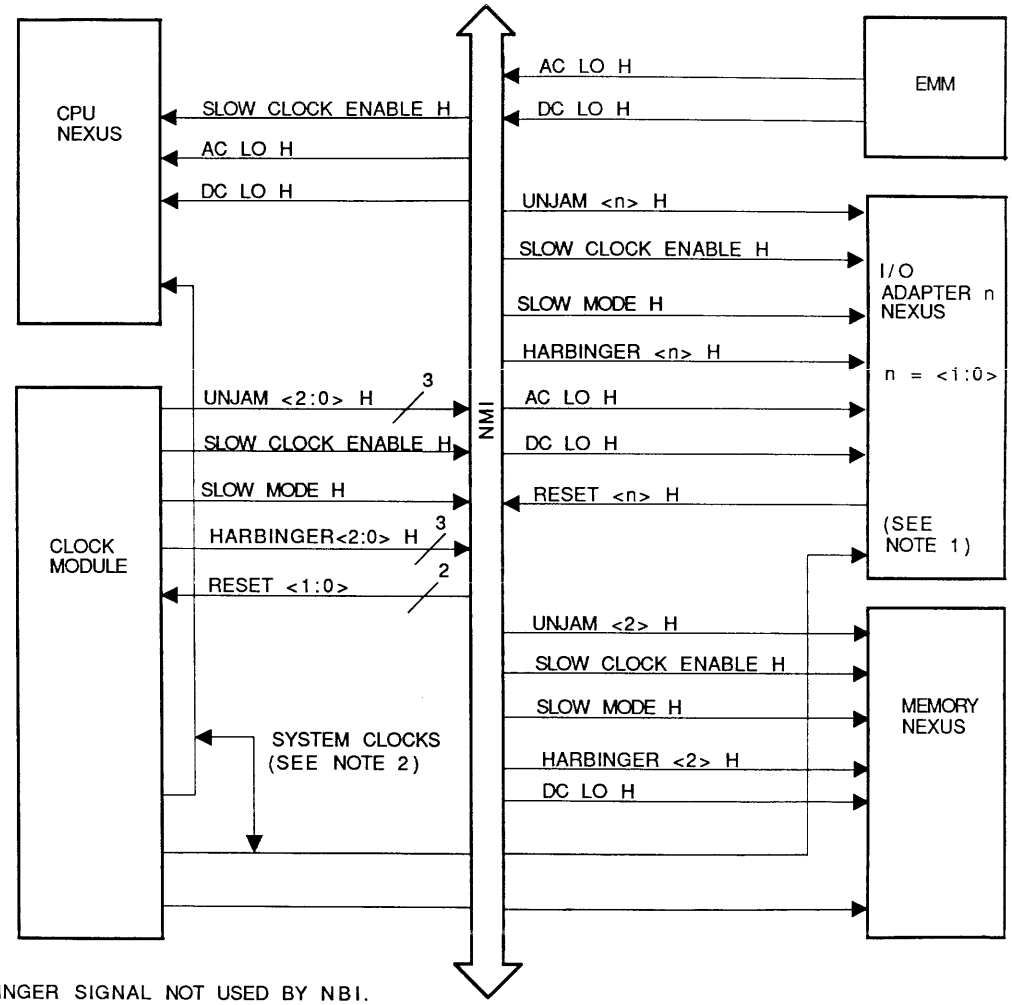


Figure 1-3 NMI Signals (Sheet 3 of 4)

SYSTEM CLOCKS AND MISCELLANEOUS CONTROL SIGNALS



NOTE 1: HARBINGER SIGNAL NOT USED BY NBI.

NOTE 2:

NEXUS	SYSTEM CLOCKS USED
CPU	A CLK H/L, B CLK H/L
MEMORY	F A CLK H/L, F B CLK H/L
I/O	A CLK H/L, B CLK H/L
	F A CLK H/L, F B CLK H/L (THESE CLOCKS NOT USED BY NBI)

SCLD-127

Figure 1-3 NMI Signals (Sheet 4 of 4)

Table 1-2 NMI Signal Descriptions

Signal Line(s)	Number	Description
ADDRESS DATA<31:00> H	32	Transfer 30-bit read/write address during NMI command/address cycles and 32-bit longword of read/write data during NMI data cycles.
FUNCTION<4:0> H	5	Specify type of bus transaction (command type) during command/address cycle and type of data during NMI data cycles.

FUNCTION<4:0>

(Hex)	Command/Data Type
10	Read Longword
12	Read Octaword
13	Read Hexword
14	Read Longword (Interlocked)
16	Read Octaword (Interlocked)
17	Read Hexword (Interlocked)
1B	Write Longword
1F	Write Octaword
18	Write Longword Masked
19	Write Quadword Masked
1A	Write Octaword Masked
1C	Write Longword Unlock Masked
1D	Write Quadword Unlock Masked
1E	Write Octaword Unlock Masked
00	No Op
04	Memory Pause (Not Used)
0A	Return Read Data (Good Data)
0E	Return Read Data (Bad Data)
08	Read Continuation (Good Data)
0C	Read Continuation (Bad Data)
09	Write Data

Table 1-2 NMI Signal Descriptions (Cont)

Signal Line(s)	Number	Description
ID MASK<3:0> H	4	Specify ID of commander during NMI command/address cycles and return read data cycles. No ID is required for the memory nexus because it is never the commander during a bus transaction. ID — Nexus 0100, 0101 — CPU 0 (Primary CPU) 0110, 0111 — CPU 1 (Attached CPU) 1000, 1001 — NBI 0 (VAXBI 0) 1010, 1011 — NBI 0 (VAXBI 1) 1100, 1101 — NBI 1 (VAXBI 0) 1110, 1111 — NBI 1 (VAXBI 1) During NMI write data cycles of masked write transactions, specify the bytes in the longword (four bytes) of write data to be written. MASK BIT 3 2 1 0 X X X 1 — Write byte 0 X X 1 X — Write byte 1 X 1 X X — Write byte 2 1 X X X — Write byte 3
DATA PARITY H	1	Transfers even parity bit for the 32 ADDRESS DATA lines during NMI command/-address and data cycles. Parity computed by transmitter. Parity checked by all nexus.

Table 1-2 NMI Signal Descriptions (Cont)

Signal Line(s)	Number	Description
FUNCTION ID PARITY H	1	Transfers even parity bit for the five FUNCTION lines and four ID MASK lines during NMI command/address and data cycles. Parity computed by transmitter. Parity checked by all nexus.
CONFIRM-ATION<1:0> H	2	Specify response (by responder) to function sent by commander.

CONFIRMATION <1:0>	Response
00	No response
01	Responder accepts data
10	Responder interlocked
11	Responder busy

Arbitration Lines

MEMORY ARB H	1	Asserted by memory nexus to request use of the bus. This bus arbitration request line has the highest priority.
IO 0 ARB H	1	Asserted by I/O adapter 0 nexus (NBI 0) to request use of the bus.
IO 1 ARB H	1	Asserted by I/O adapter 1 nexus (NBI 0) to request use of the bus.
LEFT CPU ARB H	1	Asserted by left CPU nexus to request use of the bus.
RIGHT CPU ARB H	1	Asserted by right CPU nexus to request use of the bus.

NOTE

Arbitration requests by I/O adapter and CPU nexus have the same (lowest) priority and are granted on an alternating basis. That is, if there are both CPU and I/O adapter requests, a CPU will get the bus if an I/O adapter was given the bus last (and vice versa). Also, if there are two CPU requests when a CPU is given the bus, CPU 0 gets the bus if CPU 1 was given the bus last (and vice versa). Similarly, if there are two I/O adapter requests when an I/O adapter is given the bus, I/O adapter 0 gets the bus if I/O adapter 1 was given the bus last (and vice versa).

Table 1-2 NMI Signal Descriptions (Cont)

Signal Line(s)	Number	Description
MEMORY BUSY H	1	Asserted by memory nexus when its command/ data buffers are full. Causes current transmitter to abort transaction and rearbitrate for the bus when asserted during command/address cycle of memory read, or during command/address or first data cycle of memory write.
MEMORY BUSY ARB H	1	Not a bus arbitration request line. It is a copy of MEMORY BUSY that connects only to the bus arbitrator. Asserted by memory nexus when its command/data buffers are full. This line disables the bus arbitrator so that it will not arbitrate requests by the CPU and I/O adapter nexus. (The arbitrator will arbitrate requests from the memory nexus.)
MEMORY HOLD H	1	Asserted by memory nexus when it requires another (more than one) bus cycle after winning the bus.
LEFT CPU HOLD H	1	Asserted by left CPU nexus when it requires another (more than one) bus cycle after winning the bus.
RIGHT CPU HOLD H	1	Asserted by right CPU nexus when it requires another (more than one) bus cycle after winning the bus.
IO 0 HOLD H	1	Asserted by I/O adapter 0 nexus when it requires another (more than one) bus cycle after winning the bus.
IO 1 HOLD H	1	Asserted by I/O adapter 1 nexus when it requires another (more than one) bus cycle after winning the bus.
		NOTE HOLD signals disable bus arbitrator so that it cannot arbitrate requests by any nexus.
MCL BUS EN H	1	Asserted by bus arbitrator to grant memory nexus use of the bus.

Table 1-2 NMI Signal Descriptions (Cont)

Signal Line(s)	Number	Description
Interrupt Lines		
IO 0 BUS EN H	1	Asserted by bus arbitrator to grant I/O adapter 0 nexus (NBI 0) use of the bus.
IO 1 BUS EN H	1	Asserted by bus arbitrator to grant I/O adapter 1 nexus (NBI 1) use of the bus.
LEFT CPU BUS EN H	1	Asserted by bus arbitrator to grant left CPU nexus use of the bus.
RIGHT CPU BUS EN H	1	Asserted by bus arbitrator to grant right CPU nexus use of the bus.
LCPU MEM INTR PT H	1	Asserted by memory nexus to interrupt the left CPU. (Left CPU enabled as the primary CPU.)
RCPU MEM INTR PT H	1	Asserted by memory nexus to interrupt the right CPU. (Right CPU enabled as the primary CPU.)
DEV0 LINTR H	1	Asserted by I/O adapter 0 nexus (NBI 0) to interrupt the left CPU. (Left CPU enabled as the primary CPU.)
DEV0 RINTR H	1	Asserted by I/O adapter 0 nexus (NBI 0) to interrupt the right CPU. (Right CPU enabled as the primary CPU.)
DEV0 LINTR LVL<1:0>	2	Asserted by I/O adapter 0 nexus (NBI 0) to specify the interrupt request level to left CPU. (Left CPU enabled as primary CPU.)
DEV0 RINTR LVL<1:0>	2	Asserted by I/O adapter 0 nexus (NBI 0) to specify the interrupt request level to right CPU. (Right CPU enabled as primary CPU.)

DEV0 (L/R)INTR LVL <1:0>

	Interrupt Request Level
00	BR4
01	BR5
10	BR6
11	BR7

Table 1-2 NMI Signal Descriptions (Cont)

Signal Line(s)	Number	Description
Fault Lines		
DEV1 LINTR H	1	Same as DEV0 LINTR except asserted by I/O adapter 1 (NBI 1) nexus.
DEV1 RINTR H	1	Same as DEV0 RINTR except asserted by I/O adapter 1 (NBI 1) nexus.
DEV1 LINTR LVL<1:0>	2	Same as DEV0 LINTR LVL<1:0> except asserted by I/O adapter 1 (NBI 1) nexus.
DEV1 RINTR LVL<1:0>	2	Same as DEV0 RINTR LVL<1:0> except asserted by I/O adapter 1 (NBI 1) nexus.
FAULT DETECT<3:0>	4	Each line asserted by corresponding I/O adapter or CPU nexus when it has detected a fault. Signals are ORed by fault-handling logic in memory nexus. FAULT DETECT — Fault Detected By 0 — Left CPU nexus 1 — Right CPU nexus 2 — I/O adapter 0 nexus (NBI 0) 3 — I/O adapter 1 nexus (NBI 1)
FAULT H	1	Asserted by fault-handling logic in memory nexus to signal that a FAULT DETECT line has been asserted. Used to generate a CPU interrupt request, freeze the NMI transaction silo in the CBox, and latch the fault status registers in all nexus.
Miscellaneous Control Signals		
UNJAM H	1	Initializes all nexus without clearing fault status registers. If I/O nexus is an NBI, UNJAM also causes a power-fail sequence on the VAXBI. Asserted by clock module in response to the console command.

Table 1-2 NMI Signal Descriptions (Cont)

Signal Line(s)	Number	Description
RESET <1:0> H	2	Each line asserted by corresponding NBI adapter if VAXBI RESET is asserted. Causes console to stop both CPUs and then boot the system (a cold start). The boot does not occur until RESET is negated. RESET — Asserted By 0 — I/O adapter 0 nexus (NBI 0) 1 — I/O adapter 1 nexus (NBI 1)
SLOW CLOCK ENABLE H	1	Asserted by clock module and used to increment timeout counters in all nexus.
SLOW MODE H	1	Early-warning signal asserted by clock module to indicate that the normal system clocks (A CLK and B CLK) are about to start or stop. Asserted a minimum of 4 microseconds before HARBINGER.
HARBINGER<2:0> H	3	All lines asserted by clock module during last B PHASE clock when normal system clocks are stopped. Used as clock blocking signal by nexus using free-running clocks (F A CLK and F B CLK) to simulate stopped-clock condition. HARBINGER — Connects To 0 — I/O adapter 0 nexus 1 — I/O adapter 1 nexus 2 — Memory nexus
AC LO H	1	FET driven and received power-loss warning signal asserted by EMM when ac power is below specified limits.
DC LO H	1	FET driven and received power-loss warning signal asserted by EMM when dc power is below specified limits.

Table 1-2 NMI Signal Descriptions (Cont)

Signal Line(s)	Number	Description
Backpanel Select Levels		
I/O SEL<1> H	1	Backplane-generated (hard-wired) signal that determines I/O address space for each I/O adapter nexus. A logical one is generated by backplane wiring that connects I/O SEL input to the ONE LEVEL output of the nexus. (No connection is equivalent to a logical zero.)
ONE LEVEL H	1	Logic level output (logical one) asserted by each I/O nexus. Used to assert I/O SEL input signal on the backplane.
System Clocks		
A CLK H/L	2	Phase A of normal system clock. Generated on clock module. Stalled A CLK is generated in a nexus by gating A CLK with STALL. The STALL signal is asserted by the CBox.
B CLK H/L	2	Phase B of normal system clock. Generated on clock module. An NMI bus cycle is defined as the period between the leading edge of one B CLK and the next.
F A CLK H/L	2	Phase A of free-running system clock.
F B CLK H/L	2	Phase B of free-running system clock.

I/O SEL <1>	I/O Adapter	NMI Base Address (Hex)
0	0	2000 0000
1	1	2400 0000

NOTE

A nexus may receive more than one copy of a specific system clock from the clock module. Each copy of a system clock is a pair of differentially driven and received signals connecting to only one destination module.

1.4 NMI ADDRESS SPACE

The 1 Gbyte of NMI address space is allocated as shown in Figure 1-4. One half, 512 Mbytes, is allocated to physical memory. The other half is allocated to I/O register space.

I/O register space includes a 64-Mbyte block allocated to each of the two I/O adapters (128 Mbytes total), a reserved block of 352 Mbytes, and a 32-Mbyte block allocated to memory controller control/status registers (CSRs). When an I/O adapter is an NBI, the 64 Mbytes of address space is VAXBI I/O space and is allocated evenly between the two VAXBIs that may connect to the adapter (32 Mbytes for each VAXBI). The I/O registers in the memory controller (the CSRs) and the NBI are listed in Table 1-3. NBI registers are either VAXBI node registers or NMI nexus registers (for example, the NBI CSRs). The VAXBI node registers, one set for each VAXBI, are in the node register space for VAXBI 0 and VAXBI 1. The NMI nexus registers are in the private address space for VAXBI 0. VAXBI address space is discussed in Chapter 2.

The NMI address, which is 30 bits, is asserted on the ADDRESS DATA lines at the beginning of a read or write transaction (during the command/address cycle). The address is asserted on the 30 low-order lines (ADDRESS DATA<29:00>). The two high-order lines (ADDRESS DATA<31:30>) are not used for addressing purposes.

As shown in Figure 1-5, not all of the NMI address bits are significant. Their significance depends upon the transaction type.

During write transactions, write data is transferred to naturally aligned addresses (even longword addresses, quadword addresses that are a multiple of eight, octaword addresses that are a multiple of 16). This means that the following low-order bits can be assumed to be zeros and, thus, are not significant.

1. Two low-order bits of a longword address
2. Three low-order bits of a quadword address
3. Four low-order bits of an octaword address

During read transactions, transfers are also from naturally aligned blocks of data, but multilongword transfers may be wrapped. Thus, the address for read octaword and read hexword transactions must identify the longword to be transferred first; only the two low-order bits are not significant. Also, as for a write longword address, the two low-order bits of a read longword address are not significant except for longword-oriented devices (memory and most I/O devices). For word-oriented devices (some UNIBUS devices that may be connected to an I/O adapter), address bit <01> is significant. This is because there is no explicit NMI read word transaction and the read longword transaction must specify the word address. Low words (address bit <01> = 0) are returned over the 16 low-order ADDRESS DATA lines during read data cycles. (The 32 ADDRESS DATA lines normally transfer a longword of data at a time.) High words (address bit <01> = 1) are returned over the 16 high-order ADDRESS DATA lines.

Basic NMI address selection is shown in Figure 1-6. Address bit <29> determines if the address is a memory or an I/O address. If it is an I/O address, bits <28:26> specify the I/O adapter (0 or 1). In addition, when the I/O adapter is an NBI, bit <25> selects the appropriate VAXBI (0 or 1). All these bits (<29:25>) must be equal to one to select a memory controller CSR.

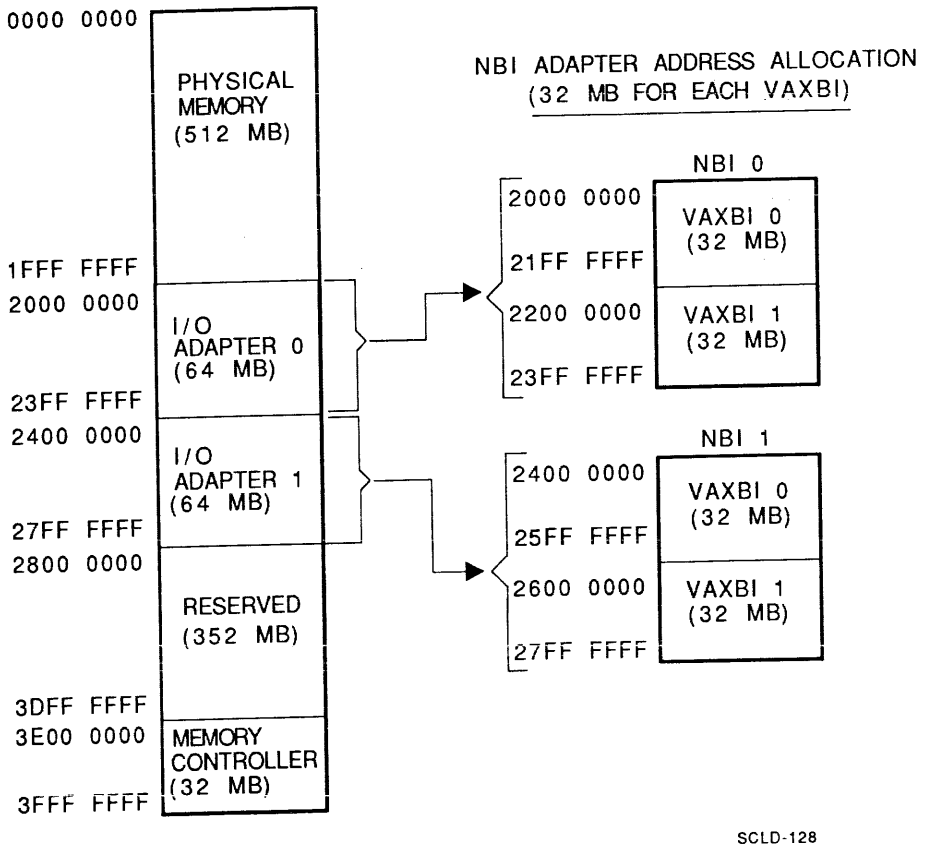


Figure 1-4 NMI Address Space

Table 1-3 I/O Registers in NBI and Memory Controller

Address (Hex)*	Registers
NBI (VAXBI Node) Registers	
bb + 0	Device Register (DTYPE)
bb + 4	VAXBI Control/Status Register (BICSR)
bb + 8	Bus Error Register (BER)
bb + C	Error Interrupt Control Register (EINTRCSR)
bb + 10	Interrupt Destination Register (INTRDES)
bb + 14	IPINTR Mask Register (IPINTRMSK)
bb + 18	IPINTR/STOP Destination Register (FIPSDES)
bb + 1C	IPINTR Source Register (IPINTRSRC)
bb + 20	Starting Address Register (SADR)
bb + 24	Ending Address Register (EADR)
bb + 28	BCI Control/Status Register (BCICSR)
bb + 2C	Write Status Register (WSTAT)
bb + 30	Force IPINTR/STOP Command Register (FIPSCMD)
bb + 40	User Interrupt Control Register (UINTRCSR)
bb + F0	General-Purpose Register 0 (GPR0) -- Not used
bb + F0	General-Purpose Register 1 (GPR0) -- Not used
bb + F0	General-Purpose Register 2 (GPR0) -- Not used
bb + F0	General-Purpose Register 3 (GPR0) -- Not used
NBI (NMI Nexus) Registers	
2x08 0000	Control/status register 0
2x08 0004	Control/status register 1
2x08 0008	Reserved
2x08 000C	Reserved
2x08 0010	BR4 vector register
2x08 0014	BR5 vector register
2x08 0018	BR6 vector register
2x08 001C	BR7 vector register

Table 1-3 I/O Registers in NBI and Memory Controller (Cont)

Address (Hex)*	Registers
Memory Controller Registers	
3E00 0000	Control/status register 0
3E00 0004	Control/status register 1
3E00 0008	Control/status register 2
3E00 000C	Control/status register 3
3E00 0010	Control/status register 4
3E00 0014	Control/status register 5
3E00 0018	Control/status register 6

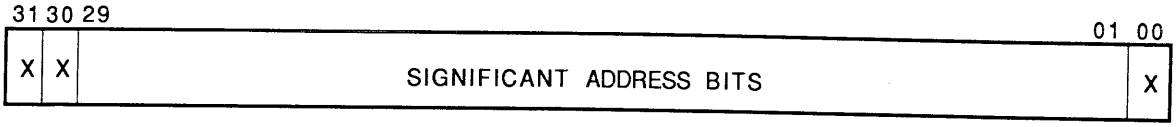
*NOTE:

For NBI (VAXBI Node) Registers

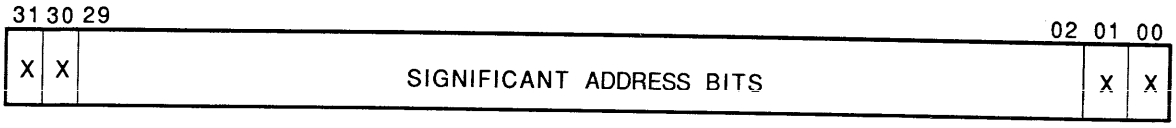
For NBI (NMI Nexus) Registers

bb = 2000 0000 (hex) + 2000 (hex) * node ID (NBI 0, VAXBI 0)
bb = 2200 0000 (hex) + 2000 (hex) * node ID (NBI 0, VAXBI 1)
bb = 2400 0000 (hex) + 2000 (hex) * node ID (NBI 1, VAXBI 0)
bb = 2600 0000 (hex) + 2000 (hex) * node ID (NBI 1, VAXBI 1)

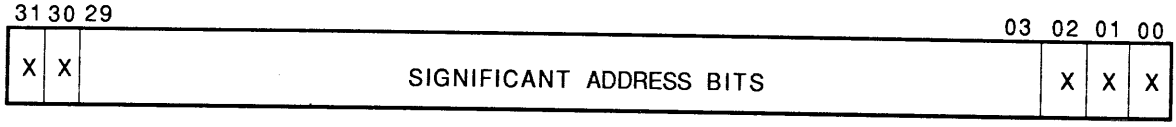
READ LONGWORD (TO WORD-ORIENTED DEVICE)



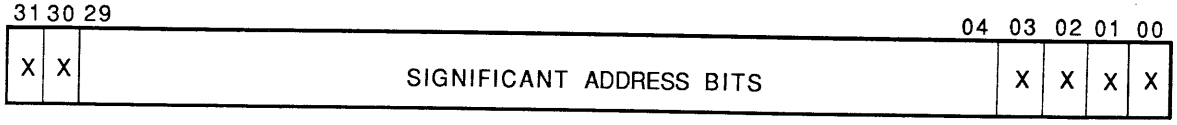
READ LONGWORD, READ OCTAWORD, READ HEXWORD, WRITE LONGWORD



WRITE QUADWORD



WRITE OCTAWORD



NOTE: AN X INDICATES BIT IS NOT USED FOR ADDRESSING PURPOSES.

SCLD-129

Figure 1-5 NMI Address Bits

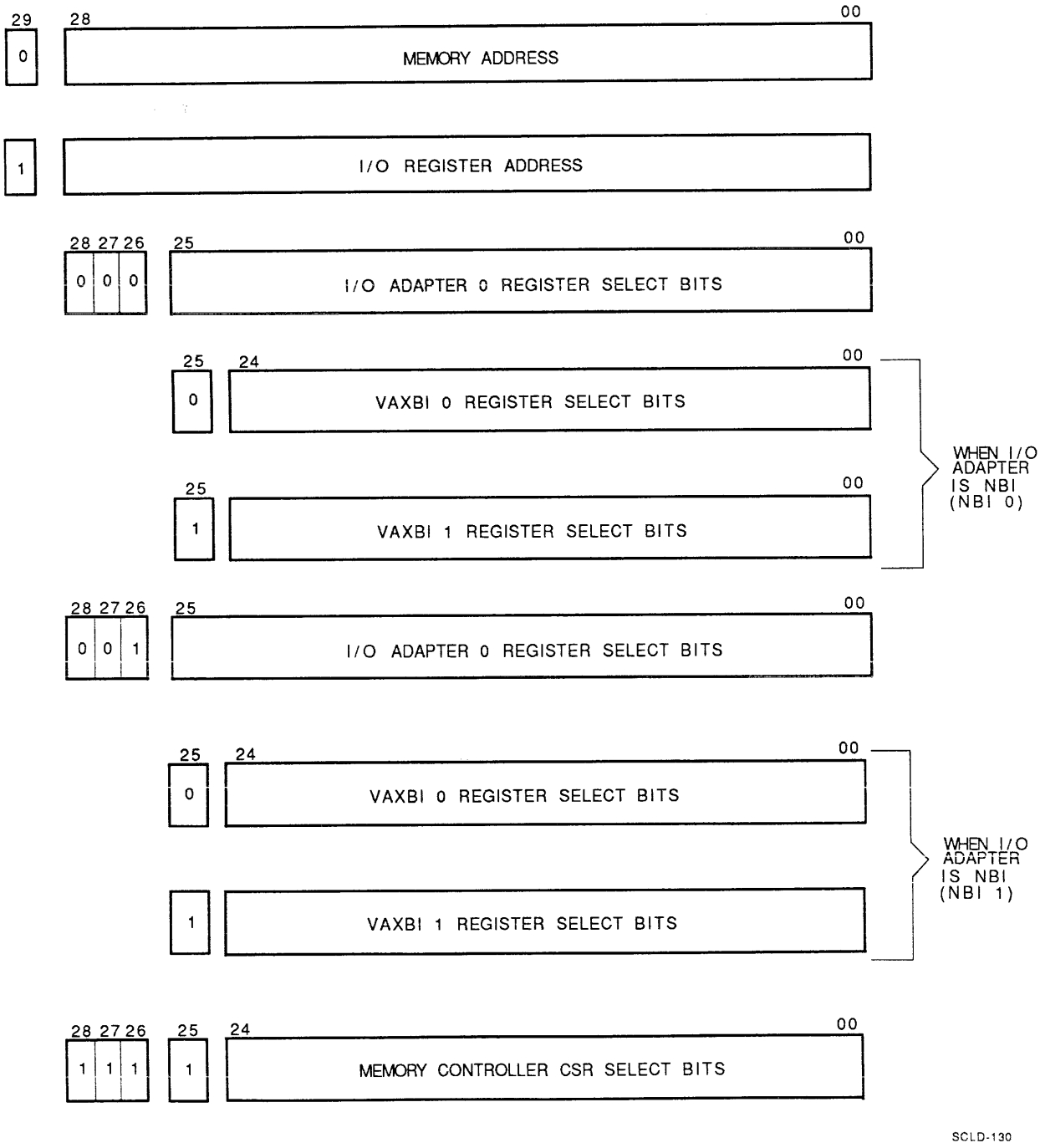


Figure 1-6 NMI Address Selection

1.5 READ/WRITE TRANSACTIONS

Before any nexus can perform a transfer of data on the NMI, it must first request and then be granted use of the bus. It does this by asserting its NMI arbitration (ARB) request line. The bus arbitrator monitors all requests, resolves request priority if more than one nexus is requesting the bus, and (when the bus is free) grants the bus by asserting a bus enable (BUS EN) line to the requesting nexus. The bus is granted for one bus cycle only. If a transfer is to take more than one cycle, the nexus must assert a HOLD signal during each bus cycle of the transfer except the last cycle. (The HOLD signal from a nexus disables the bus arbitrator.) Bus arbitration is discussed in Section 1.7.

A bus write transaction requires only one transfer. That is, the nexus initiating the transaction (the commander) arbitrates for the bus and then transfers command/address information followed by write data. This ends the transaction. No transfer by the nexus addressed by the commander (the responder) is required.

A bus read transaction requires more than one transfer. First, the commander arbitrates for the bus and transfers the command/address. The responder then must arbitrate for the bus and return the read data to the commander during a second transfer. A third transfer of read data may be necessary for some read hexword transactions. This occurs when the second octaword of data is not ready to be transferred following transfer of the first octaword.

Timing for a write transaction and a read transaction are shown in Figures 1-7 and 1-8. The information transmitted on the NMI signal lines is indicated.

During the first bus cycle of a transaction, called the command/address cycle, the commander transmits the memory or I/O address on the NMI ADDRESS DATA lines. It also transmits the transaction type (for example, write octaword and read longword) on the FUNCTION lines and its own ID number on the ID MASK lines.

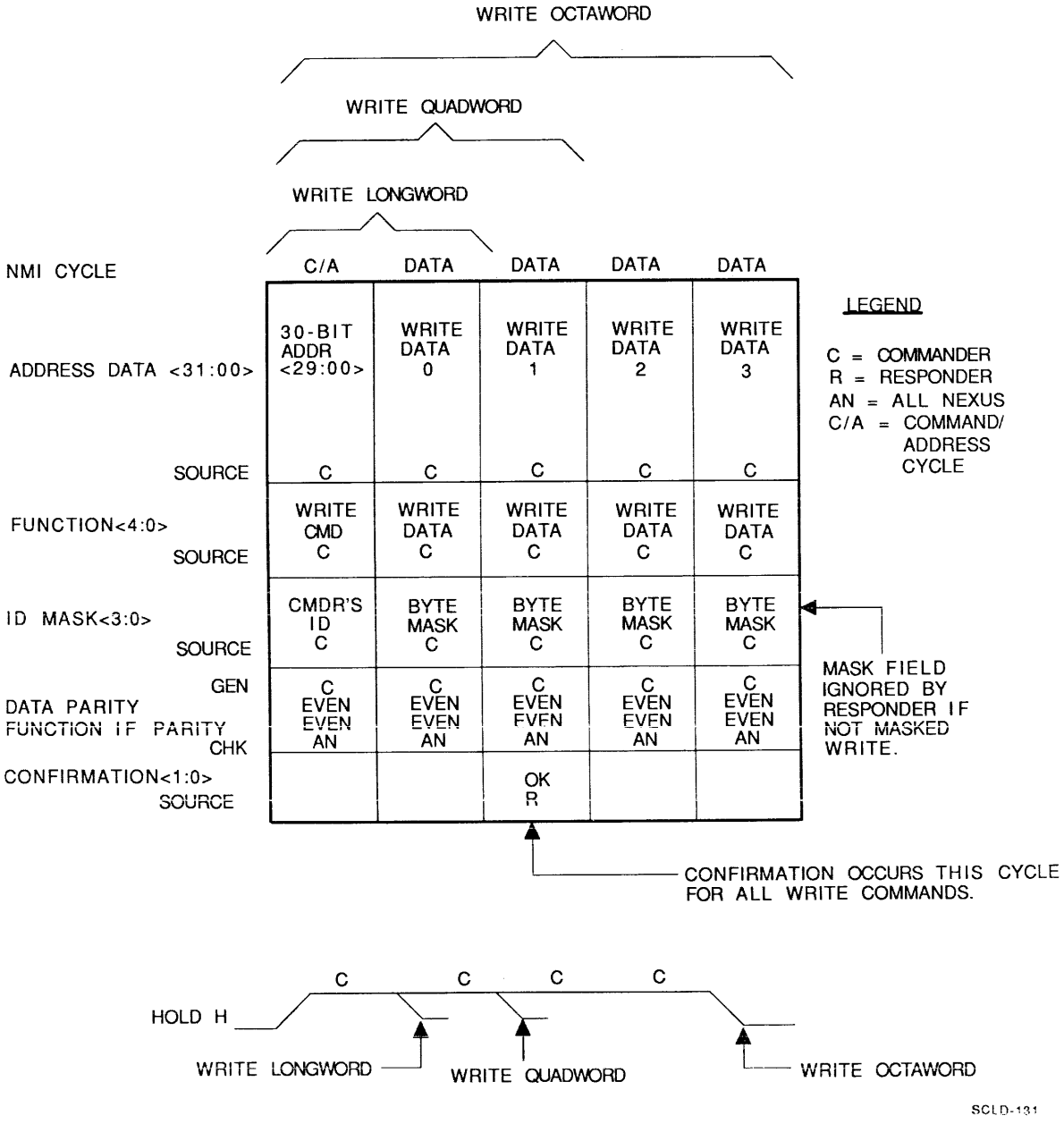
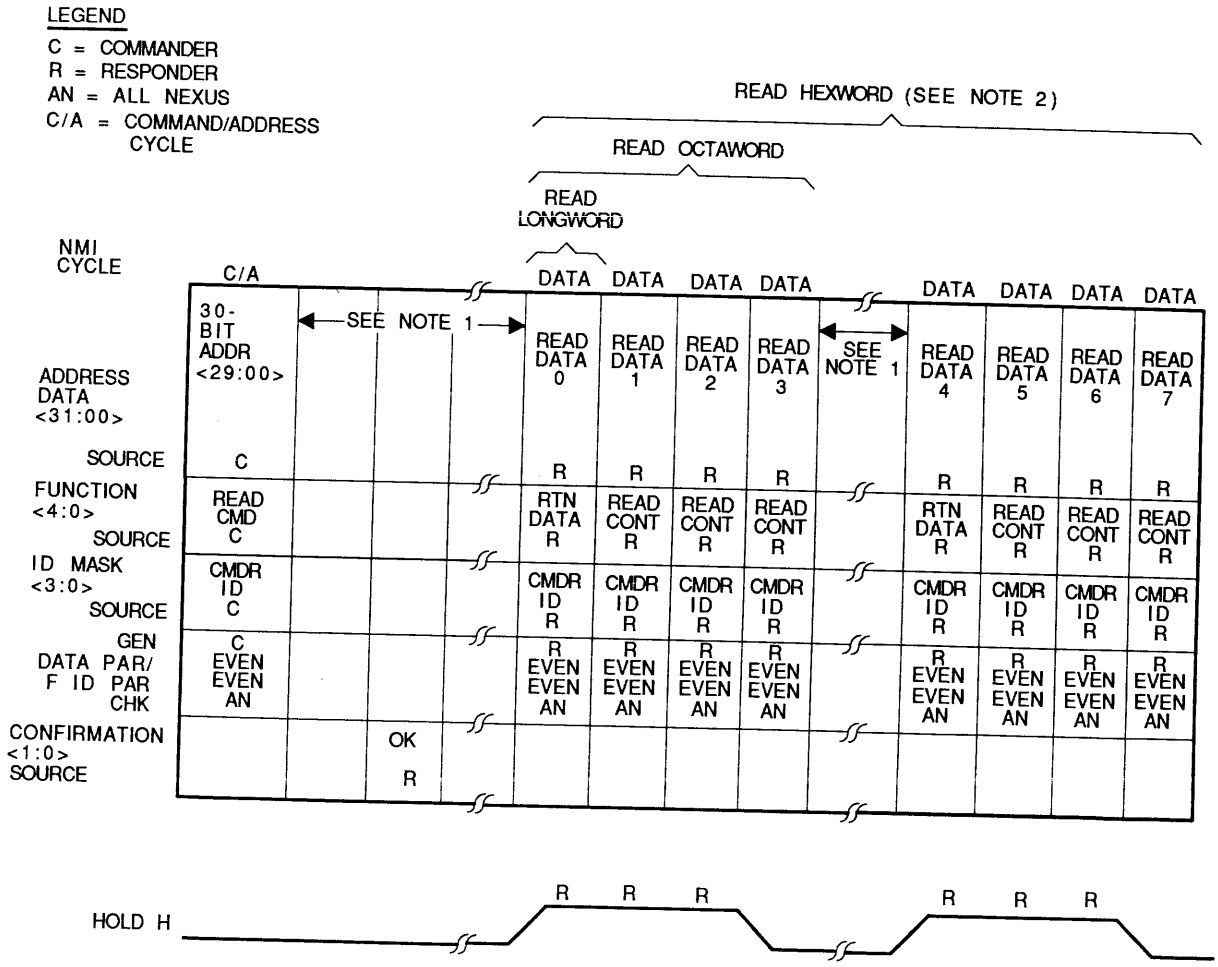


Figure 1-7 NMI Write Transaction



NOTE 1: NMI AVAILABLE FOR OTHER TRANSACTIONS.

NOTE 2: RESPONDER (MEMORY) RELEASES NMI AFTER TRANSFER OF FIRST OCTAWORD IN THE READ HEXWORD TRANSACTION SHOW. RESPONDER (MEMORY) MAY ALSO HOLD BUS AND TRANSFER ALL THE DATA (EIGHT SUCCESSIVE DATA CYCLES) IF SECOND OCTAWORD IS READY FOR TRANSMISSION.

SCLD-132

Figure 1-8 NMI Read Transaction

Following the command/address cycle and if the transaction is a write, the commander transmits the write data on the ADDRESS DATA lines during the next bus cycle or cycles. The number of these data cycles depends on the transaction type. (One longword of data may be transferred over the 32 ADDRESS DATA lines each bus cycle.) Also, during the data cycles, the commander asserts the FUNCTION lines to identify the data as write data, and it transmits a byte mask on the ID MASK lines if the transaction is a masked write. (The byte mask indicates which bytes in the longword of write data are to be written.) If the transaction is a read, no data cycles immediately follow the command/address cycle.

All nexus monitor the NMI during each bus cycle. When the responder is addressed during a command/address cycle, it will accept the transaction by transmitting a "responder accepts data" (OK) response on the NMI CONFIRMATION lines. The CONFIRMATION lines are asserted two bus cycles after the command/address cycle. These steps complete a write transaction with the responder taking the write data as it is received during each data cycle.

A read transaction requires the responder to arbitrate for the NMI and return the read data to the commander over the ADDRESS DATA lines in one or more data cycles. (Again, the number of data cycles depends on the transaction type. Two separate transfers of data may be required as stated previously.) The FUNCTION lines identify the data as return read data. The first data cycle of a transfer has a return data function specified; all others have a read continuation function specified. Also, the function specifies if the data is good or bad. Bad data is data that cannot be corrected by memory.

During all read data cycles, the ID MASK lines specify the commander and the commander completes the transaction by taking the read data as it is transmitted by the responder. It does not notify the responder that the data has been taken.

Figures 1-9 and 1-10 show the specific format for each read and write transaction type.

Two separate even parity bits are generated by the transmitter (either commander or responder) during command/address and data cycles. One parity bit, which is generated for the information on the ADDRESS DATA lines, is transmitted on the DATA PARITY line. The other parity bit, which is generated for the information on both the FUNCTION and ID lines, is transmitted on the FUNCTION ID PARITY line. The two parity bits are checked by all nexus. If bad (odd) parity is detected, it causes an NMI fault. Parity and other errors that can occur during NMI read/write transactions are discussed in Section 1.9.

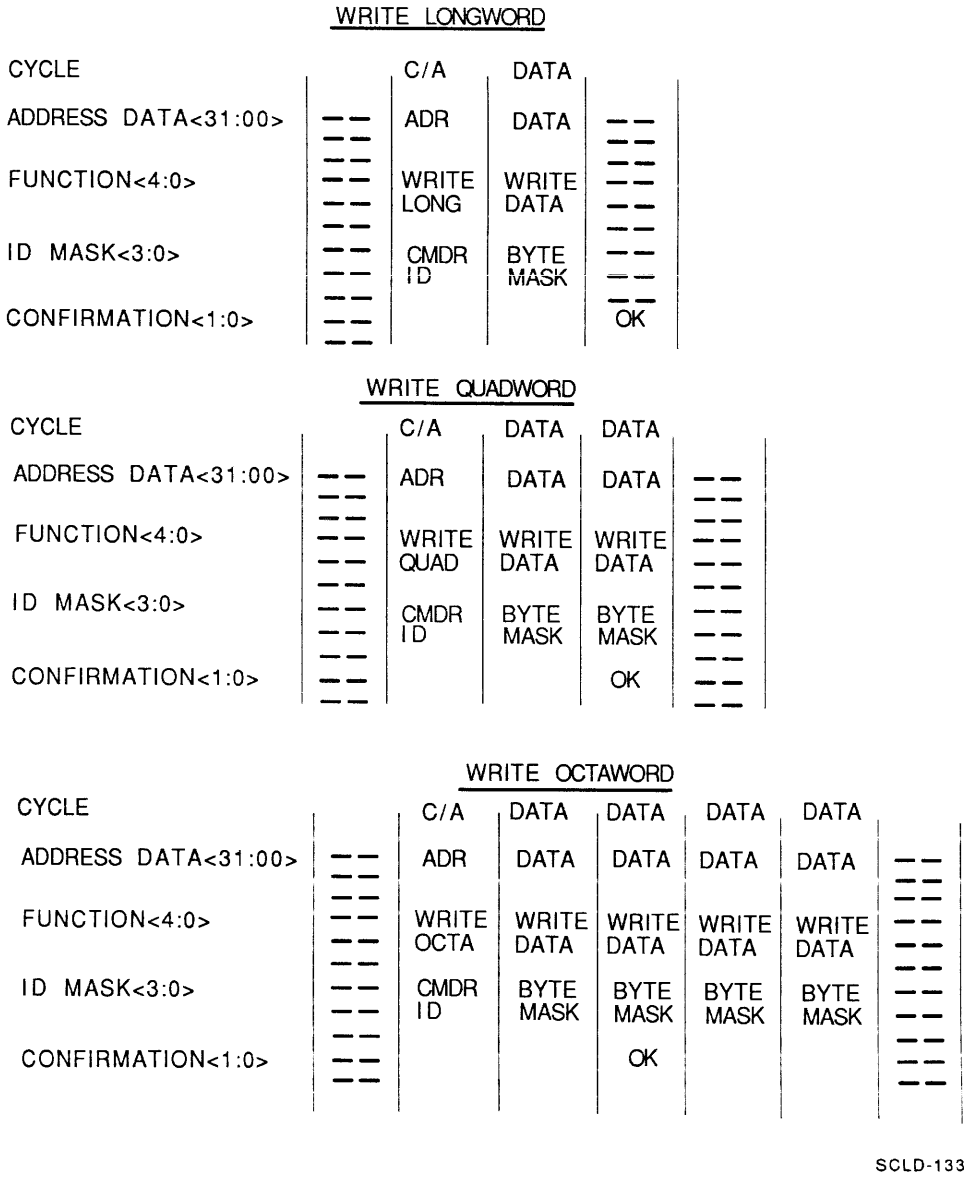


Figure 1-9 NMI Write Transaction Types

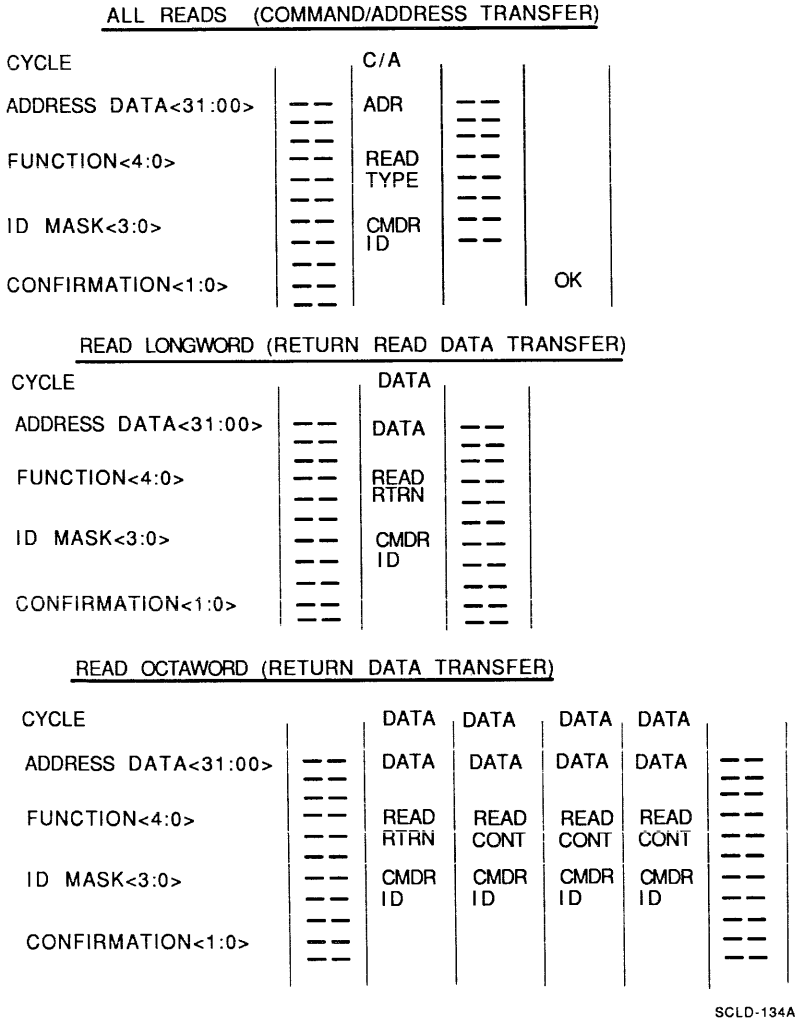


Figure 1-10 NMI Read Transaction Types (Sheet 1 of 2)

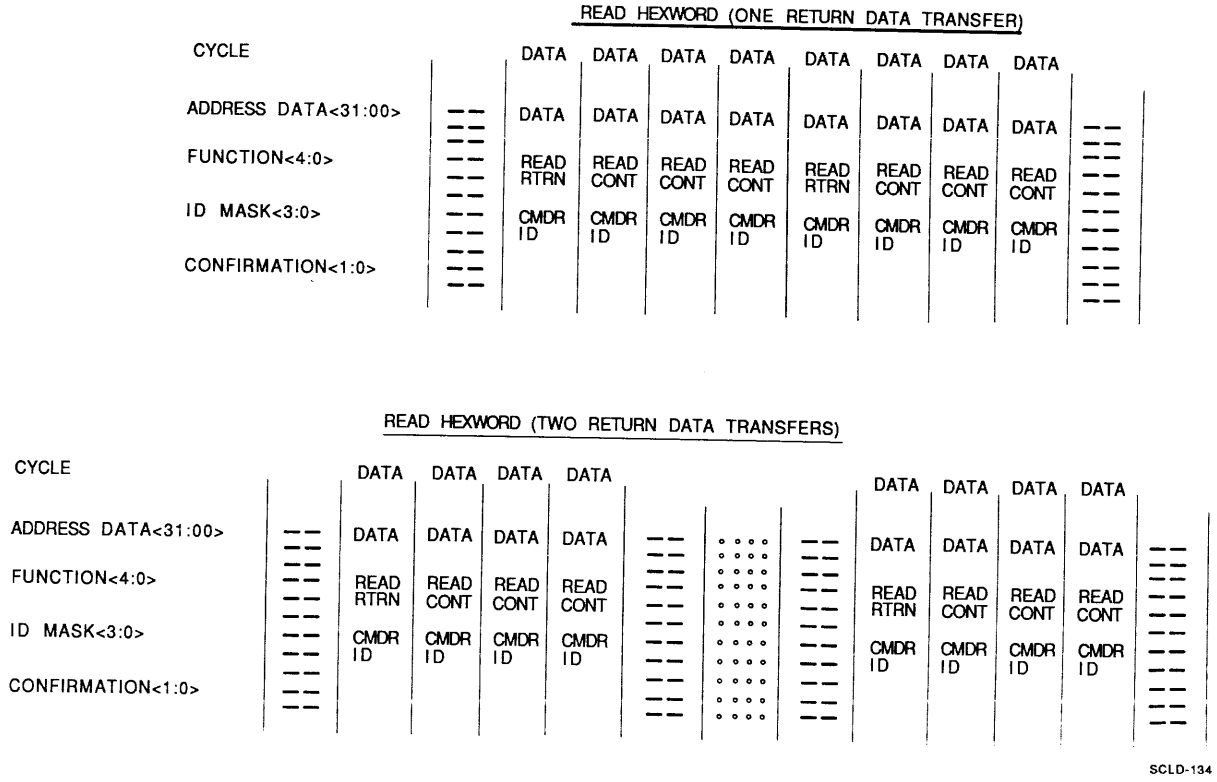


Figure 1-10 NMI Read Transaction Types (Sheet 2 of 2)

1.6 INTERLOCKED OPERATIONS

The memory nexus has hardware that allows memory to be interlocked by a read transaction and unlocked by a write transaction. This is to allow synchronization of processes in dual-CPU configurations that access shared areas of memory.

When the memory executes an interlocked read transaction, the normal read function (longword, octaword, or hexword) is performed. However, it will not accept any further interlocked read transactions until the interlock has been removed by a write unlock transaction. Also, to limit the time a nexus can hold a hardware interlock, a counter in the memory will remove the interlock after a timeout period. If the time limit is exceeded, the memory interrupts the (primary) CPU.

During the time memory is interlocked, it will accept normal read transactions and all write transactions. However, a "responder interlocked" response is returned to the commander on the CONFIRMATION lines when a read interlock transaction is attempted.

1.7 BUS ARBITRATION

To control bus arbitration on the NMI, each nexus has a bus arbitration (ARB) request line, a bus HOLD line, and a bus enable (BUS EN) line connecting to the bus arbitrator in the left CPU. The memory nexus has two ARB lines connecting to the arbitrator. However, only one (MEMORY ARB) is a bus arbitration request line. The other (MEMORY BUSY ARB) is a copy of MEMORY BUSY. The MEMORY BUSY line connects from the memory nexus to the CPU and I/O adapter nexus. Basic arbitration line timing is shown in Figure 1-11.

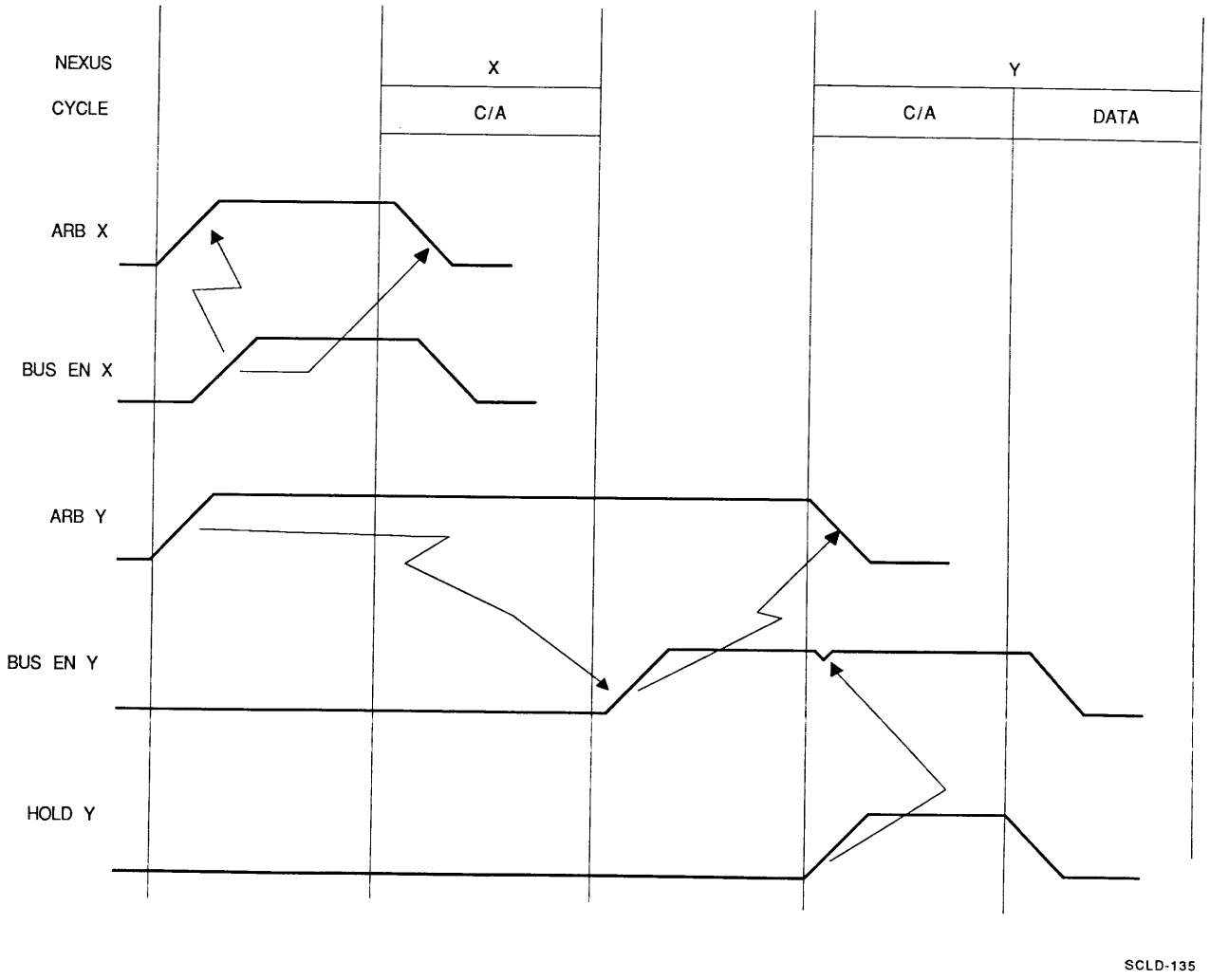


Figure 1-11 Basic NMI Arbitration Line Timing

To request the use of the bus, each nexus must assert its ARB request line. The request line is asserted at the beginning of a bus cycle. More than one nexus may request the bus at a time. As a result, the bus arbitrator in the left CPU resolves request priority and asserts the BUS EN line for the winning nexus. The BUS EN line is asserted immediately if the bus is not currently in use. If the bus is in use, the BUS EN line is not asserted until the bus is free. This ensures a dead cycle between transfers on the NMI, which is necessary due to electrical constraints. When a nexus receives its BUS EN signal, it negates its ARB request line at the beginning of the next bus cycle.

If a nexus wins the bus and requires more than one bus cycle for a transfer, it must assert its HOLD line to keep the bus. The HOLD line is asserted at the beginning of the first bus cycle in the transfer. It is negated at the beginning of the last bus cycle in the transfer. In the bus arbitrator, the asserted HOLD line disables any further bus arbitration. It also causes the BUS EN line to the nexus holding the bus to remain asserted.

The memory nexus has the highest priority when requesting the bus. The CPUs and I/O adapters have the lowest priority and share the bus on an alternating basis. Operation of the bus arbitrator is shown in Figure 1-12.

The CPUs and I/O adapters share the bus in the following manner. If both a CPU and an I/O adapter are requesting the bus, and if a CPU rather than an I/O adapter won the bus last, then the I/O adapter wins the bus. Conversely, if a CPU rather than an I/O adapter had the bus last, the CPU wins. Furthermore, if both I/O adapters are requesting the bus and an I/O adapter wins, it will be the one that did not use the bus last. Similarly, if both CPUs are requesting the bus and a CPU wins, it will be the one that did not use the bus last. Typical arbitration line sequencing is shown in Figure 1-13.

In addition to having the highest priority, the memory nexus may assert MEMORY BUSY, which causes the arbitrator to completely ignore requests by the CPUs and I/O adapters. MEMORY BUSY causes the arbitrator to ignore requests by means of the MEMORY BUSY ARB line (the copy of MEMORY BUSY connecting to the bus arbitrator). That is, like the HOLD lines, MEMORY BUSY ARB disables the arbitrator but only for CPU or I/O adapter requests. Bus requests by the memory itself can still be granted. Timing is shown in Figure 1-14.

The memory asserts MEMORY BUSY when its command/data buffers are full. It does this to prevent the other nexus (the CPUs and I/O adapters) from using the bus until it can accept more commands or data. Also, if the buffers contain read data, the memory nexus has unrestricted use of the bus to return the data to the commander and, thus, empty the buffers as soon as possible.

If MEMORY BUSY is asserted during the command/address cycle of a memory read/write transaction, it indicates to the current transmitter that the command will not be accepted by the memory. In this respect, MEMORY BUSY acts as an early-warning status line in addition to its function of temporarily preventing further memory read/write transactions on the bus. That is, the current transmitter does not have to wait for the "responder busy" code on the CONFIRMATION lines before aborting and then retrying the transaction. This allows the nexus to retry before another lower priority nexus gets the bus, should memory become available in the shortest possible time. (MEMORY BUSY is asserted for only two bus cycles in some cases.)

When MEMORY BUSY is asserted after the command/address cycle but during the first data cycle of a memory write transaction, it also indicates the command will not be accepted by the memory. Again, the current transmitter does not have to wait for the "responder busy" on the CONFIRMATION lines before retrying the transaction. When MEMORY BUSY is asserted during a data cycle following the first of a multilongword memory write transaction (write quadword or write octaword), it is ignored by the current transmitter. In this case, the command and data will be accepted by memory unless otherwise indicated by the code returned on the CONFIRMATION lines. These lines are valid during the second data cycle of the transaction.

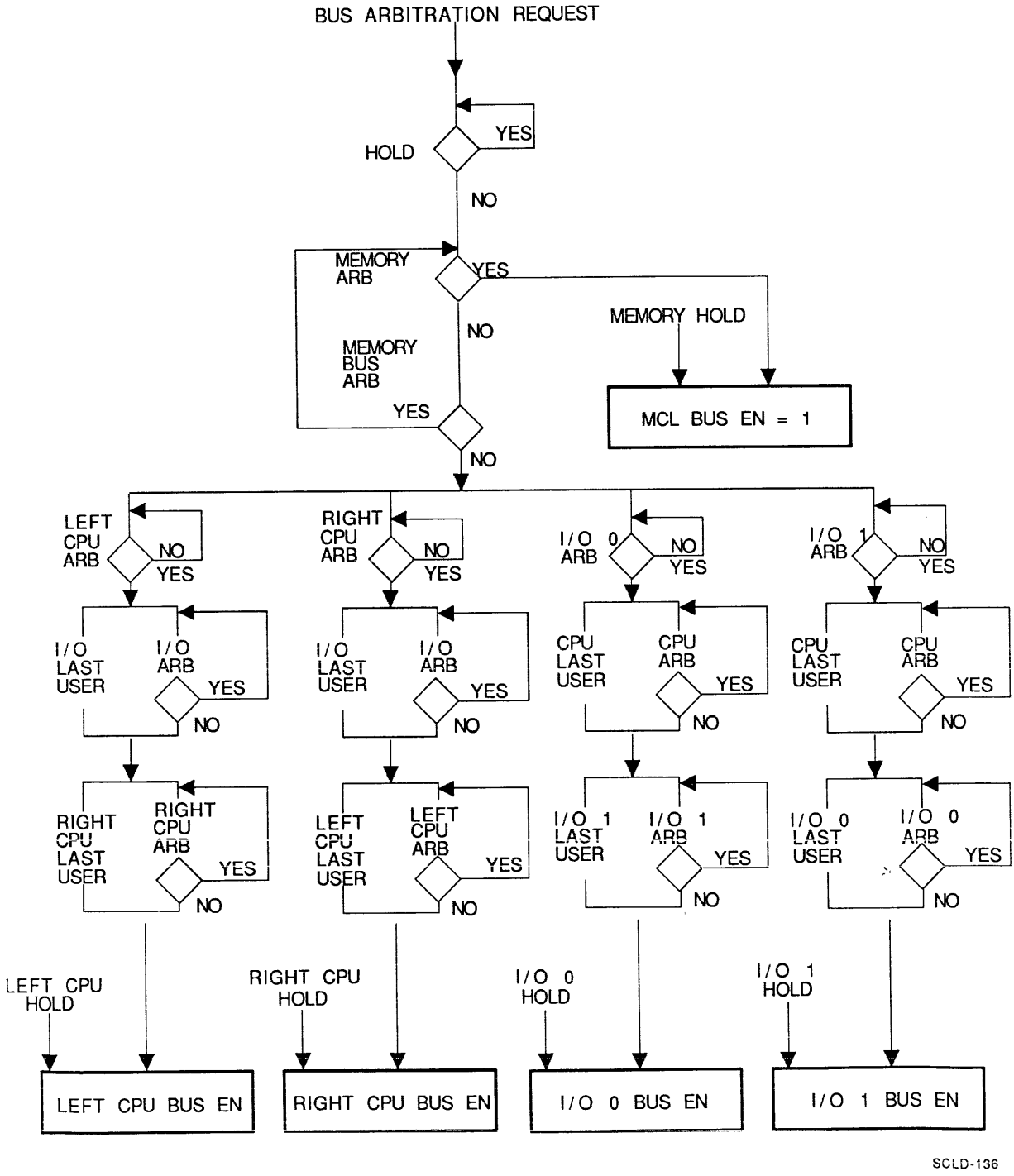


Figure 1-12 NMI Arbitrator Operation

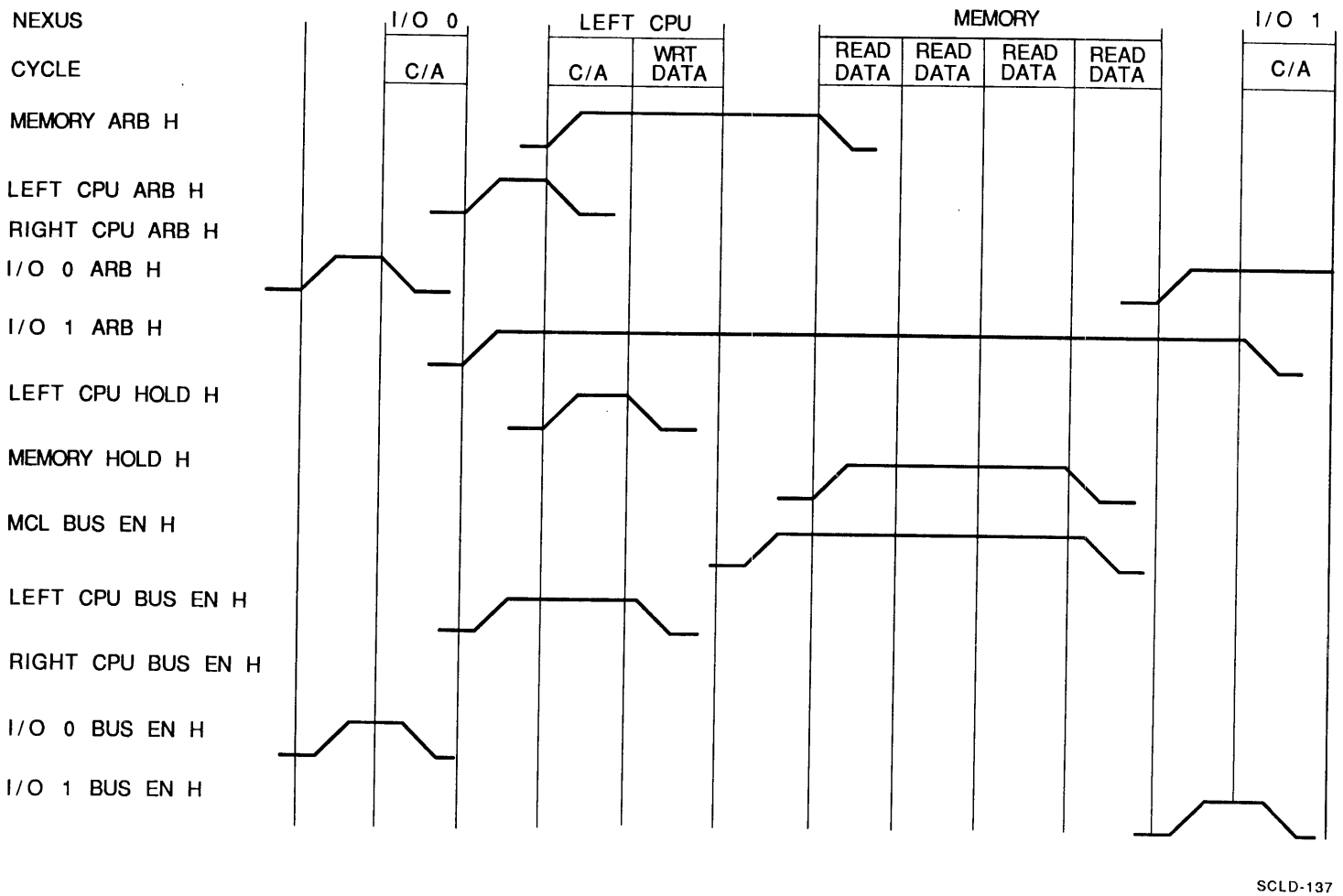
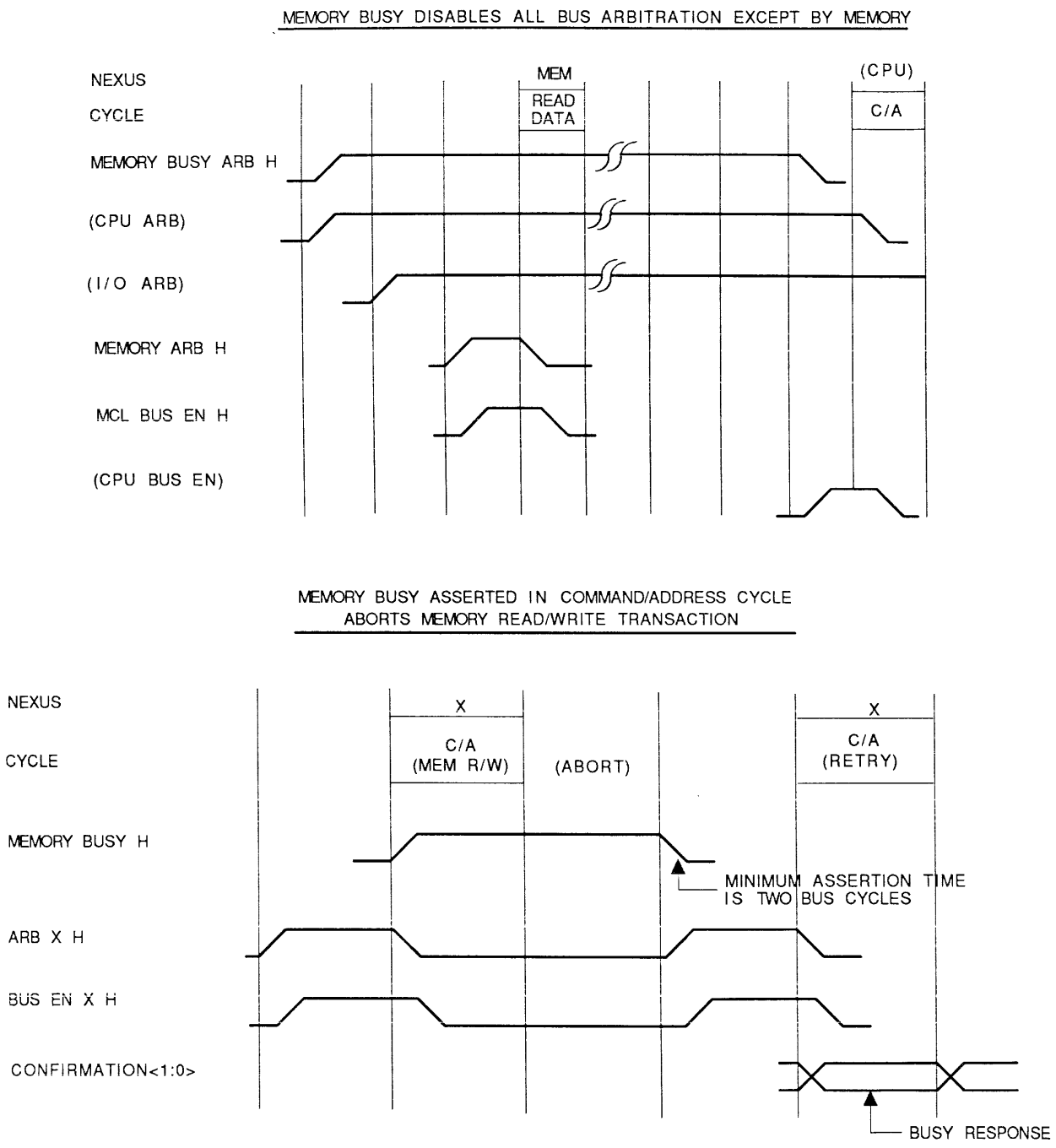


Figure 1-13 Detailed NMI Arbitration Line Timing (Typical)



NOTE: MEMORY BUSY ASSERTED IN FIRST DATA CYCLE OF MEMORY WRITE.
ALSO ABORTS TRANSACTION AND CAUSES RETRY.

SCLD-138

Figure 1-14 MEMORY BUSY Timing

1.8 INTERRUPTS

The interrupt lines on the NMI connect to both CPUs. However, only the primary CPU is enabled to accept NMI interrupts. A CPU is enabled by setting bit <00> in its NMI interrupt enable register.

There are two types of NMI interrupts.

1. Conventional interrupts by an I/O device (memory or an I/O adapter nexus) when it has an error or other condition to report to the CPU.

2. A special interrupt caused by the assertion of the NMI FAULT line. All nexus, including the CPUs, can assert FAULT when certain types of bus errors are detected. That is, FAULT not only interrupts the primary CPU, it freezes an NMI transaction silo in the CBox. This silo holds the state of selected bus signals for the faulting and preceding bus cycles.

1.8.1 NMI Interrupt Priority Levels

Table 1-4 lists the interrupt priority level (IPL) assigned to each NMI interrupt. As can be seen, an NMI fault has the highest IPL (1C) and thus the highest priority. Interrupts by an I/O adapter may be at one of four possible bus request levels: BR 4, 5, 6, or 7. The BR7 request level has the highest priority corresponding to an IPL of 17. The BR4 level has the lowest priority corresponding to an IPL of 14. The memory will always interrupt at a BR5 request level (IPL = 15).

Table 1-4 NMI Interrupt Priority Levels (IPLs)

Device or Condition	IPL (Hex)
NMI Fault	1C
I/O Adapter 0 (BR7)	17
I/O Adapter 1 (BR7)	17
I/O Adapter 0 (BR6)	16
I/O Adapter 1 (BR6)	16
I/O Adapter 0 (BR5)	15
I/O Adapter 1 (BR5)	15
Memory (BR5)	15
I/O Adapter 0 (BR4)	14
I/O Adapter 1 (BR4)	14

1.8.2 Device Interrupts

Seven NMI lines connect to the primary CPU to request device interrupts. There is an interrupt request line from the memory (CPU MEM INTRPT) and one from each of the I/O adapters (DEV0 INTR and DEV1 INTR). In addition, there are two interrupt request level lines from each adapter (DEV0 INTR LVL<1:0> and DEV1 INTR LVL<1:0>). There are two sets of these seven lines. One set connects to the left CPU and one to the right CPU. However, only one CPU (the primary CPU) is enabled to accept interrupts as explained previously.

The interrupt sequence is as follows.

When ready to interrupt, the interrupting device simply asserts its interrupt request line and (if it is an I/O adapter) transmits a request level code on the INTR LVL lines. More than one device may be requesting an interrupt at any time.

INTR LVL<1:0>	BR Level
0 0	BR4
0 1	BR5
1 0	BR6
1 1	BR7

When ready to service the highest priority device interrupt request, and if the request is by an I/O adapter, the CPU microcode reads an interrupt vector from an adapter register. The register is read by means of an NMI read transaction. The CPU microcode does not read a vector from the memory when the memory interrupts.

The CPU microcode must read a vector from an I/O adapter to identify the interrupting device. That is, the interrupt can be locally generated by the adapter itself (one vector value) or it can be generated by any one of the I/O devices connected to the adapter (vector value depending on the device). The vector is used by the CPU microcode to generate an address in the SCB. (It is added to the SCBB.) The contents of the SCB location, in turn, are used by the CPU microcode to dispatch to the appropriate interrupt service routine.

The vector addresses for locally generated interrupts by each of the two I/O adapters are in page 0 of the SCB. In this case, an adapter is interrupting as an NMI nexus like memory, whose vector is also in page 0. Vectors for devices connected to the adapters are allocated starting with page 1.

1.8.3 NMI Faults

The bus errors causing an NMI fault condition are as follows.

- Bus parity errors
- Write sequence errors
- Read sequence errors

These errors, and the nexus checking for each error, are discussed in Section 1.9.

When a CPU or I/O adapter nexus detects a fault, it asserts and negates a FAULT DETECT line. (Refer to Figure 1-15.) There is a FAULT DETECT line for each CPU and I/O adapter, and the signals are logically ORed in the memory nexus causing a single FAULT line to be asserted at the beginning of the next bus cycle. (When the memory nexus detects a fault, it also asserts the FAULT line.) The FAULT line connects from the memory to all other nexus. It does the following.

1. In the primary CPU, the asserted FAULT line causes an interrupt request as stated previously, and it remains asserted until cleared by the CPU microcode (to dismiss the interrupt). FAULT is cleared by reading CSR5 in the memory controller. The vector address for an NMI fault condition is in page 0 of the SCB.

2. In all nexus, FAULT latches the NMI error status bits so they may be examined during the interrupt sequence.

3. In the primary CPU, FAULT freezes (prevents further loading of) the NMI transaction silo. This silo, containing 256 locations, holds a history file of the last 256 bus cycles including the faulting cycle. The states of the following NMI lines are stored for each cycle.

   - Arbitration lines
   - FUNCTION lines
   - ID MASK lines
   - CONFIRMATION lines
   - ADDRESS DATA <29>

The NMI transaction silo is an IPR (NMISILO register) and can be accessed by error logging programs. A "last in/first out" addressing scheme allows the bus information for the most recent (faulting) cycle to be read first, followed by the older information.

During microdiagnostics, the silo address can be cleared to zero and the silo conditioned to freeze when the address overflows (increments from the highest address back to zero). This allows silo information to be captured without having to induce an NMI fault condition.

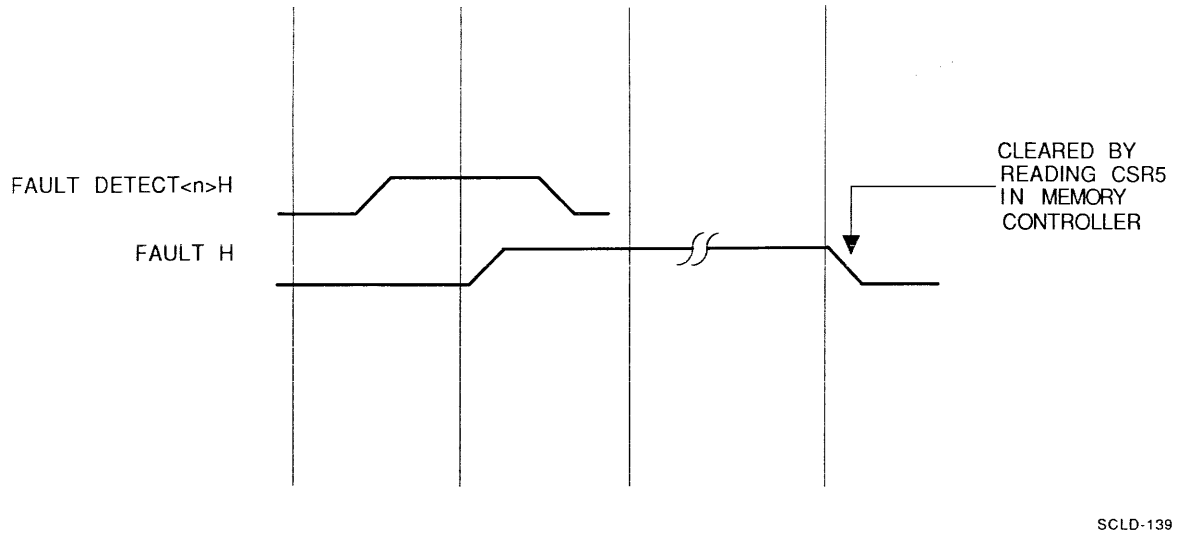


Figure 1-15 Fault Signal Timing

1.9 NMI ERRORS

There are two classes of NMI errors.

1. Interrupt (Only) -- These errors only cause a CPU interrupt request by the nexus detecting the error. Memory and I/O nexus generate the interrupt (a device interrupt) request by means of the interrupt lines on the NMI. Interrupt requests by CPU nexus are internally generated.

2. Faults -- A fault causes a CPU interrupt request but it also freezes the NMI transaction silo. Thus, this class of errors are those for which a recent history of NMI bus transactions are useful in determining the cause of error.

NMI errors are listed in Table 1-5.

Table 1-5 NMI Errors

NMI Error	Interrupt/ Fault	Nexus Checking Error		
		CPU	I/O	Memory
No Return Read Data (Timeout)	I	Yes	Yes	No
No Access To Bus (Timeout)	I	Yes	Yes	No
No Access, Busy (Timeout)	I	Yes	Yes	No
No Access, Interlocked (Timeout)	I	Yes	Yes	No
No Access, No Response (Timeout)	I	Yes	Yes	No
Interlock, No Unlock (Timeout)	I	No	No	Yes
Bus Parity Error	F	Yes	Yes	Yes
Write Sequence Error	F	No	Yes	Yes
Read Sequence Error	F	Yes	Yes	No

Errors causing conventional interrupts by the individual nexus are all timeout errors. That is, a timeout counter in a CPU or I/O adapter nexus generates the interrupt when the nexus does not receive return read data or gain access to the bus or another nexus after a certain length of time. Also, a timeout counter in the memory generates an interrupt request when it has been interlocked for too long a time.

Specifically, a bus access timeout occurs when the arbitrator delays granting the bus for a transfer. This condition is not checked by the memory because it requests the bus only to transfer return read data. (The absence of return read data will cause a timeout error in the CPU or I/O adapter initiating the read transaction.) The other access errors occur when a CPU or I/O adapter continues to receive a "no response" code, a "busy" response code, or an "interlocked" response code on the CONFIRMATION lines following a command/address cycle. The memory does not check these error conditions because it is never the commander during a bus transaction.

Errors causing an NMI fault are bus parity errors and read and write sequence errors. A write sequence error is when a responder does not receive enough write data. That is, there are not enough write data cycles to complete the specified transaction. A read sequence error is when a commander does not receive enough read data (too few read data cycles) to complete the transaction. Also, a read sequence error can be caused when a "read continuation" code is received on the FUNCTION lines during a read data cycle, and there was no "return read data" code received (flagging the first longword of read data) during a previous data cycle in the transfer.

NMI errors detected by a CPU nexus will set error bits in the NMI fault/status register (NMIFSR). NMI errors detected by a memory or I/O adapter nexus will set error bits in the nexus first control/status register (CSR0).

CHAPTER 2
VAX BUS INTERCONNECT (VAXBI)

2.1 INTRODUCTION

The VAX Bus Interconnect (VAXBI) is the I/O bus for the system. A VAXBI connects to the system through the NMI to VAXBI (NBI) adapter as shown in Figure 2-1. Each NBI adapter may interface up to two VAXBIs allowing a maximum of four VAXBIs to be connected to the system when both NBI adapters are installed. The terms used in the following VAXBI description are defined in Table 2-1.

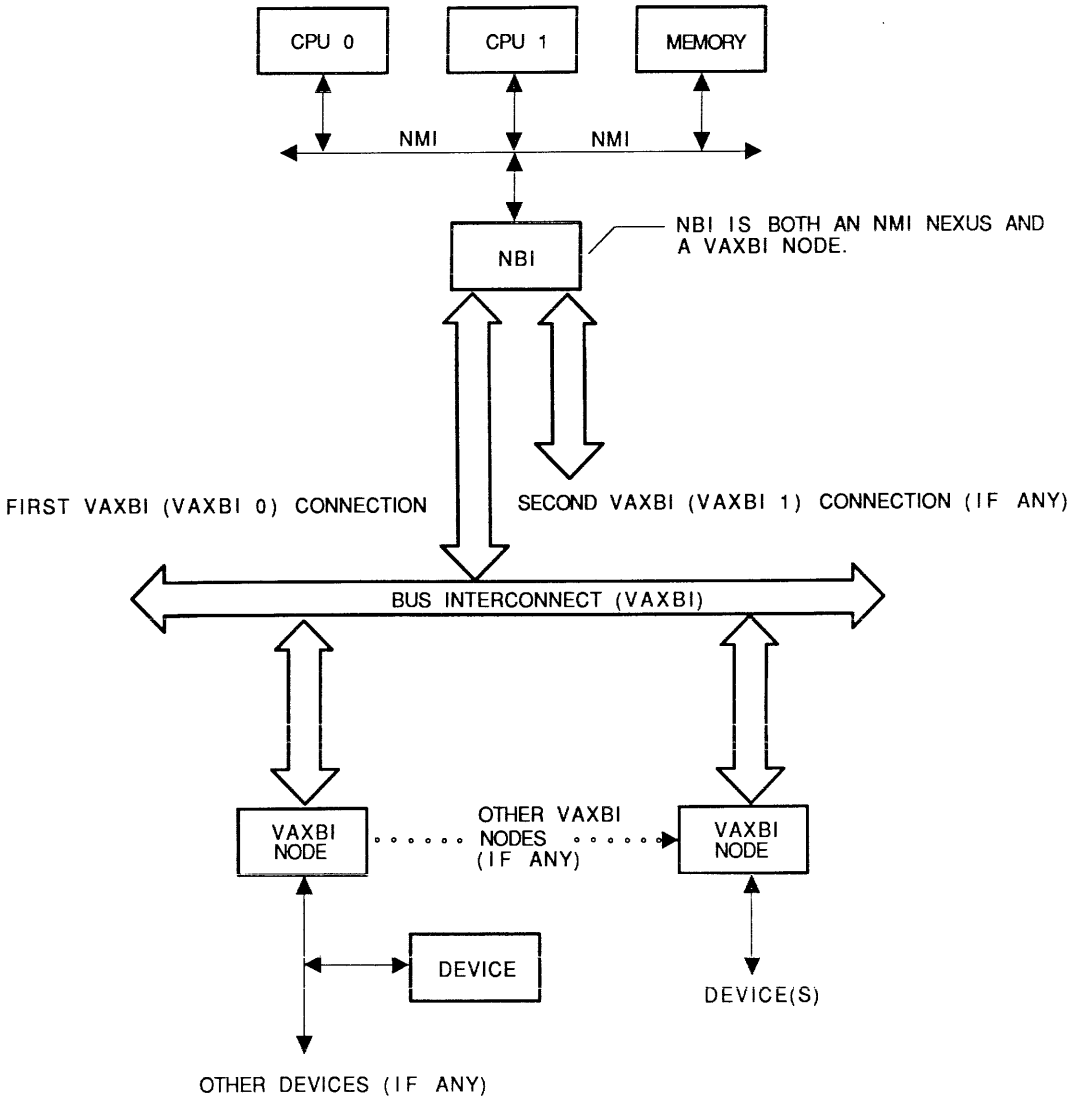


Figure 2-1 VAX Bus Interconnect (VAXBI)

Table 2-1 Glossary of VAXBI Terms

Term	Definition
Node	A hardware block (a VAXBI Interface) physically connecting to (and occupying one of sixteen logical locations on) the VAXBI. A VAXBI node consists of one or more VAXBI modules.
Transaction	The execution of a VAXBI command on the VAXBI. Also, the term applies to the special mode execution of a loopback request. A loopback operation transfers data within a node without using the VAXBI data lines.
Master	The node that gains control of the VAXBI and initiates a VAXBI transaction.
Slave	A node that responds to a transaction initiated by the master.

The VAXBI is a 32 bit wide synchronous bus with parity that interconnects up to 16 VAXBI interfaces (VAXBI nodes) having logical addresses 0 through 15. (The address of a node is determined by an ID plug inserted on the backplane.) The NBI associated with the VAXBI is one node. The other nodes are the I/O device controllers or I/O bus adapters interfacing the system's I/O devices to the VAXBI. An example of an I/O bus adapter is the VAXBI to UNIBUS adapter (DWBUA) that allows the family of UNIBUS I/O devices to be used as peripherals on the system. Processor nodes (I/O processors) can also be connected to a VAXBI.

Bus arbitration for the VAXBI is not controlled by any one node. A distributed arbitration scheme is used where each node requesting use of the bus samples all bus requests. The node with the highest priority then assumes control of the bus during the next bus cycle (when the bus is inactive) or following the current bus transaction (when the bus is busy). To allow bus arbitration when the bus is busy, an embedded arbitration cycle is included as part of every bus transaction.

The VAXBI supports the following read/write transactions:

- Write transactions: longword, quadword, octaword
- Read transactions: longword, quadword, octaword

All read/write transaction types (longword, quadword, and octaword) are used for memory data transfers. Only longword transactions are used to transfer I/O register data.

If desired, memory read/write transactions may be interlocked (interlocked reads/unlock writes). Also, memory read/write transactions may be specified as having cache intent following cache misses. This facilitates the invalidating of cache locations in cached multiprocessor configurations on the VAXBI (not applicable to this system, because any processors that may be on a VAXBI must have their caches turned off as explained in Section 2.8).

All nodes using the VAXBI must use a ZMOS integrated circuit called the VAXBI interface chip (BIIC) as the bus interface. This chip initiates and responds to all bus transactions in response to input commands by the node's internal logic. BIIC operation is discussed in the NBI Technical Description.

2.2 BASIC FUNCTIONS

A VAXBI on the VAX 8800 system performs the following major functions.

1. Memory read/write operations -- By means of bus read/write transactions directed to the NBI node, allows DMA data transfers between an I/O device on the VAXBI and the system's main memory (on the NMI). (Main memory may also be accessed by an I/O processor node on the VAXBI.)

2. I/O register read/write operations -- By means of bus read/write transactions originated by the NBI node, allows the primary CPU (on the NMI) to access I/O registers (for example, CSRs) in the I/O devices on the VAXBI. (Device I/O registers may also be accessed by an I/O processor node on the VAXBI.)

3. Interrupt handling -- By means of bus INTR transactions directed to the NBI node, allows the I/O devices on the VAXBI to interrupt the primary CPU (on the NMI). Also, in response to the interrupts and by means of bus IDENT transactions originated by the NBI node, allows the primary CPU to read interrupt vectors from the I/O devices on the VAXBI. (Interrupts by I/O devices and the NBI may also be fielded by an I/O processor node on the VAXBI.)

4. System synchronization -- Provides clocks to synchronize operation of all nodes. Clocks are generated by the NBI node.

5. System initialization -- By asserting a RESET line, allows a node to initiate a simulated VAXBI power-fail (AC LO/DC LO) sequence by the NBI node. The NBI also asserts RESET on the NMI causing the VAX console to halt both CPUs and initiate a system bootstrap (a cold start).

6. Power loss warning -- Provides AC LO and DC LO signals to all nodes.

2.3 VAXBI SIGNALS AND TIMING

VAXBI signals are shown in Figure 2-2 and defined in Table 2-2. All signals are ZMOS-driven (TTL voltage levels), except the FET-driven AC LO and DC LO signals and the ECL clocks.

The clocks and basic bus timing are shown in Figure 2-3. Data path and synchronous control signals are asserted and negated at the beginning of a bus cycle. The signals are received and latched near the end of the cycle. The data path signals are the data, parity, and information signals. The synchronous control signals are the confirmation, NO ARB, and BSY signals. The other control signals (AC LO, DC LO, RESET, STF, and BAD) are asserted and negated asynchronously with respect to the bus cycle.

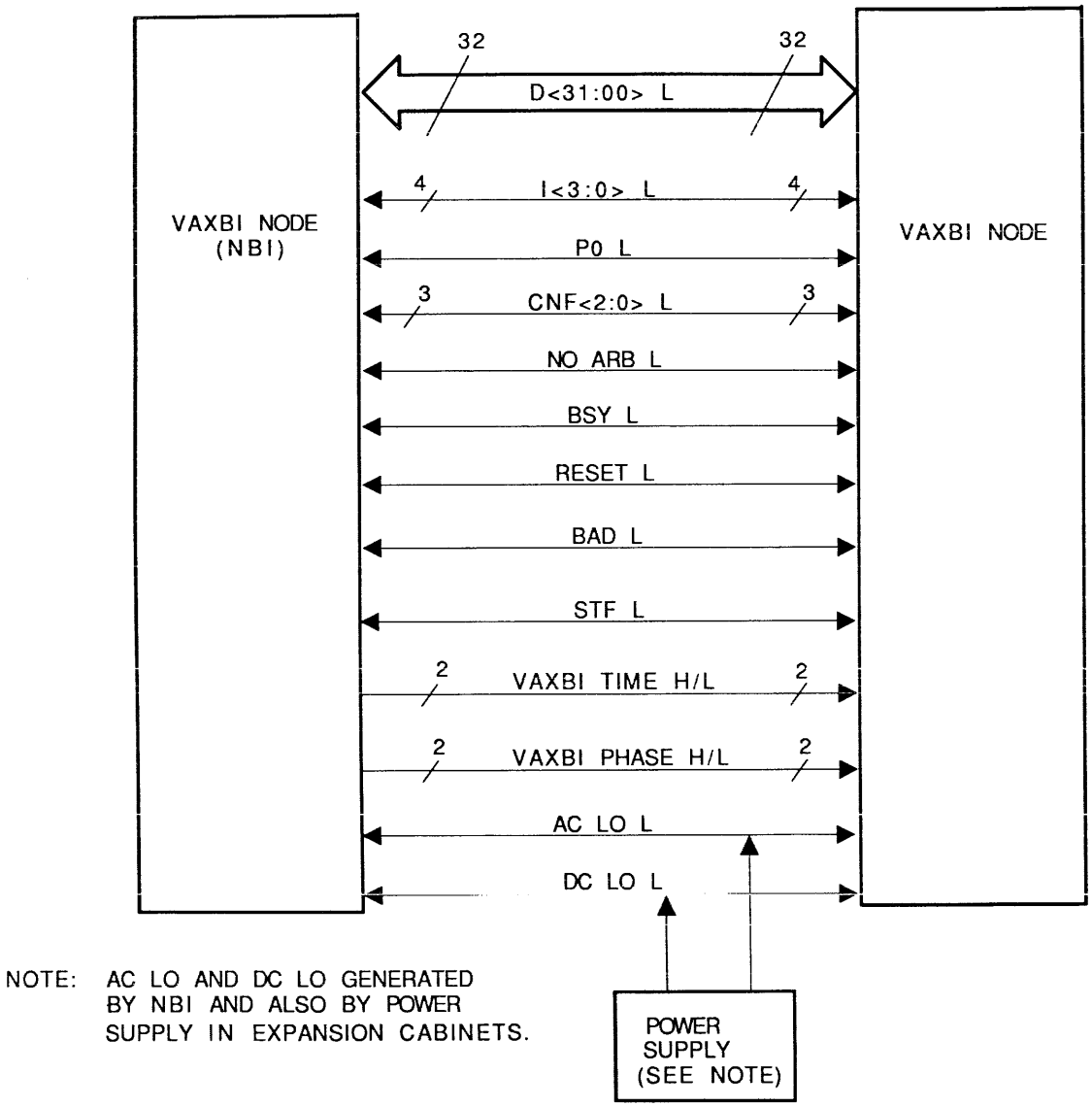


Figure 2-2 VAXBI Signals

Table 2-2 VAXBI Signal Descriptions

Signal Line(s)	Number	Description
D<31:00> L	32	Data lines -- Specify length of transfer and 30-bit address during command/address cycles of read, write, and invalidate transactions. (Also specify interrupt level and/or destination mask information during command/address cycles of other transactions.) Transfer write, read, or vector data during data cycles. Specify decoded IDs of arbitrating nodes during arbitration cycles.
I<3:0> L	4	Information lines -- Specify VAXBI command during command/address cycles, byte mask during write data cycles, and data status during read and vector data cycles. Also specify the master's ID during embedded arbitration cycles.

I<3:0> (Hex)	Command
0	Reserved
1	Read (READ)
2	Interlocked read with cache intent (IRCI)
3	Read with cache intent (RCI)
4	Write (WRITE)
5	Write with cache intent (WCI)
6	Unlock write masked with cache intent (UWMCI)
7	Write masked with cache intent (WMCI)
8	Interrupt (INTR)
9	Identify (IDENT)
A	Reserved
B	Reserved
C	Stop (STOP)
D	Invalidate (INVAL)
E	Broadcast (BDCST)
F	Interprocessor interrupt (IPINTR)

Table 2-2 VAXBI Signal Descriptions (Cont)

Signal Line(s)	Number	Description
		I<3:0> Byte Mask ------ --------- XXX1 Write byte 0 XX1X Write byte 1 X1XX Write byte 2 1XXX Write byte 3 I<3:0> Data Status ------ ----------- 0X00 Reserved 0X01 Read data 0X10 Corrected read data 0X11 Read data substitute 0X00 Reserved 1X01 Read data, don't cache 1X10 Corrected read data, don't cache 1X11 Read data substitute, don't cache
P0 L	1	Parity line -- Transfers odd parity bit for the I lines during embedded arbitration cycles, and for the D and I lines during command/address cycles, and during data cycles when read/write (or vector) data is being transmitted.
CNF<2:0> L	3	Confirmation lines -- Specify response by slave(s) to command sent by master, by slave(s) during the data cycles of a transaction, and by node receiving data during the two bus cycles following the data cycles of a transaction. CNF <2:0> Response ----- -------- 000 No Acknowledgment (NOACK) 001 Illegal 010 Illegal 011 Acknowledgment (ACK) 100 Illegal 101 STALL 110 RETRY 111 Illegal

Table 2-2 VAXBI Signal Descriptions (Cont)

Signal Line(s)	Number	Description
NO ARB L	1	No arbitration line -- Asserted by arbitrating nodes, master, pending master, or slave to inhibit arbitration by nodes during the next bus cycle. The pending master is the node winning the bus after an embedded arbitration cycle.
BSY L	1	Busy line -- Asserted by master or slave to indicate that a transaction is in progress. May also be asserted by any node to extend the current transaction, or (when asserted together with NO ARB) delay the start of the next transaction in order to perform special mode operations such as a loopback request.
RESET L	1	Reset line -- Causes the NBI to simulate a VAXBI power-fail (AC LO/DC LO) sequence. Also causes NBI to assert RESET on the NMI causing the VAX console to halt both CPUs. The VAX console then boots the system (a cold start) when VAXBI RESET is negated.
BAD L	1	Bad line -- Indicates one or more nodes detected a selftest or other error.
STF L	1	Selftest fast line -- Enables the fast selftest mode in all nodes.
TIME H/L	2	20 MHz differentially driven clock lines (ECL) -- Used in conjunction with PHASE H/L to provide reference for VAXBI cycle timing in all nodes. Generated by the NBI.
PHASE H/L	2	5 MHz differentially driven clock lines (ECL) -- Used in conjunction with TIME H/L to provide reference for VAXBI cycle timing in all nodes. Generated by the NBI.
AC LO L	1	AC LO line -- Indicates ac power is below specified limits. Asserted by NBI and expander cabinets.
DC LO L	1	DC LO line -- Indicates dc power is below specified limits. Asserted by NBI and VAXBI expander cabinets.

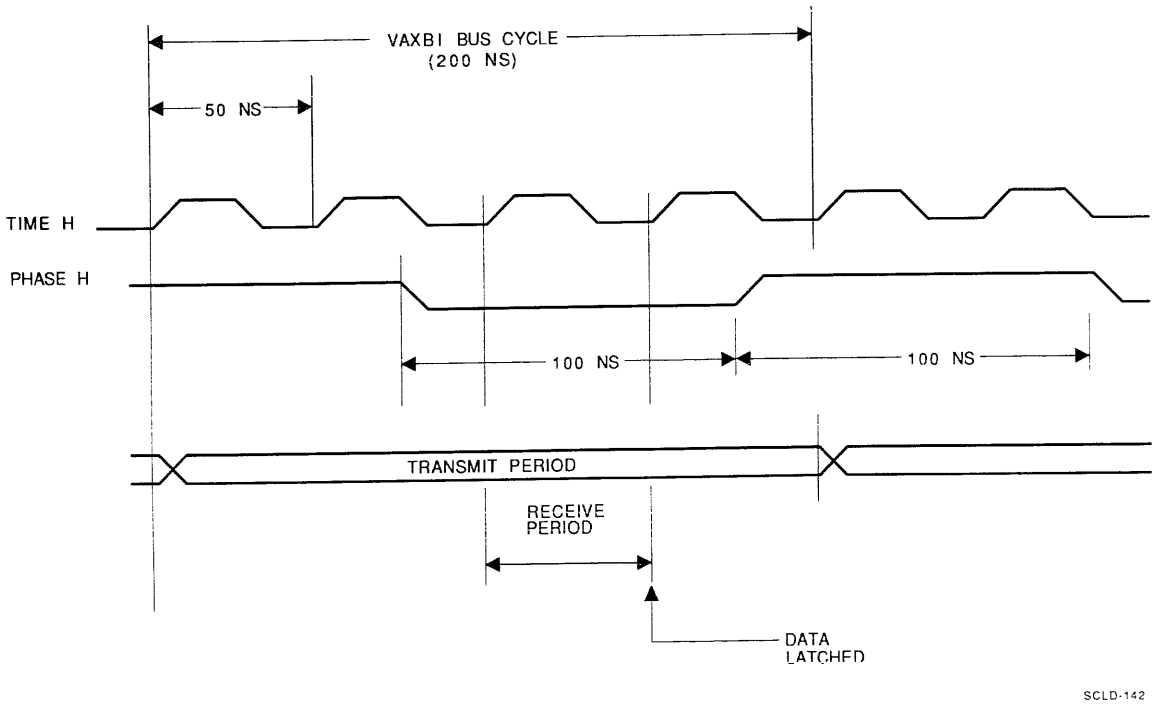


Figure 2-3 Basic VAXBI Timing

2.4 VAXBI ADDRESS SPACE

The VAXBI address space is 1 Gbyte. As shown in Figure 2-4, one half (512 Mbytes) is physical memory space. The other half is I/O space.

2.4.1 Memory Address Space

Except when a memory is connected to the VAXBI to support an I/O processor node, the VAXBI memory space in the system is actually NMI memory space. That is, memory data transfers by I/O devices on the VAXBI are to/from the system's main memory (on the NMI) through the NBI node. The transfers, initiated by the I/O device nodes, are DMA transfers that take place independently of the CPU(s). System memory on the VAXBI is not supported. (The NBI cannot initiate memory data transfers on the VAXBI in response to an NMI transaction.) The only VAXBI memory supported is that associated with an I/O processor. In this case, transfers are initiated by the processor and are local to the VAXBI (directly between the processor node and its associated VAXBI memory node).

2.4.2 I/O Address Space

The allocated I/O space for a VAXBI is 32 Mbytes. The range of VAXBI I/O addresses, 2000 0000 to 21FF FFFF (HEX), is the same for each of the (up to four) VAXBIs in the system. This means that when VAXBI I/O space is accessed by the NBI in response to an NMI transaction, the NBI must clear address bits <26:25> during the NMI to VAXBI address translation as shown below.

NMI I/O Addresses (Hex)	VAXBI	VAXBI I/O Addresses(Hex) [Bits <26:25> Cleared by NBI]
2000 0000 - 21FF FFFF	VAXBI 0 (NBI 0)	2000 0000 - 21FF FFFF
2200 0000 - 23FF FFFF	VAXBI 1 (NBI 0)	2000 0000 - 21FF FFFF
2400 0000 - 25FF FFFF	VAXBI 0 (NBI 1)	2000 0000 - 21FF FFFF
2600 0000 - 27FF FFFF	VAXBI 1 (NBI 1)	2000 0000 - 21FF FFFF

The 32 Mbytes of I/O space for each VAXBI consist of register space for each node, multicast space, node private space, and adapter window space. (Refer again to Figure 2-4.) Multicast space contains addresses for which more than one node can respond. Node private space contains registers that are not accessed from the VAXBI. For example, the NMI nexus registers in the NBI node (the CSRs, and vector offset registers) have node private space addresses. Window space is used for address mapping by adapter nodes interfacing another bus and its devices to the VAXBI. That is, the VAXBI addresses are converted to addresses specific to the other bus (UNIBUS addresses, for example). A block of window space is allocated to each node.

The register space for each node (8K bytes) is shown in Figure 2-5. The first group of registers, called the VAXBI required registers, must be implemented by every node on the VAXBI. The second group are the BIIC-specific registers. A register in this group may or may not be used depending upon the node design.

The VAXBI required and BIIC-specific registers are contained in the node's BIIC. These registers are shown in Figures 2-6 and 2-7 and are described in the NBI Technical Description. The BIIC-specific registers include four general-purpose registers. If more registers are needed, they may be implemented outside the BIIC but these addresses are still within the 8K bytes of register space allocated for the node.

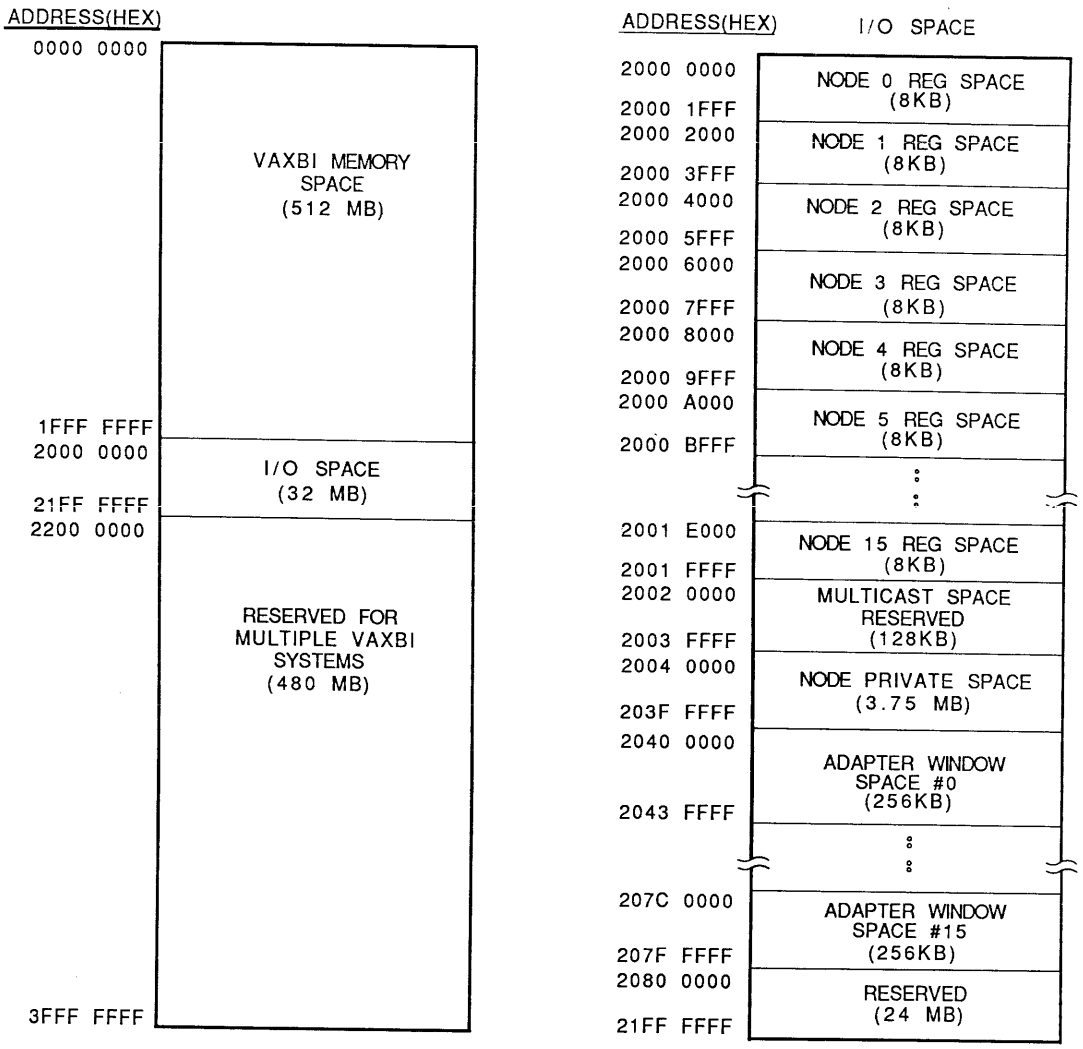
2.4.3 Address Selection

A VAXBI memory or I/O register address (30 bits) is asserted on the data lines during the first bus cycle of a read or write transaction. As shown in Figure 2-8, the two high-order data lines are not used for addressing, but are used to specify the length of the transaction: longword, quadword, or octaword. As stated previously, only longword transactions are used to transfer I/O data. Thus, only a longword length is specified when the address is an I/O address.

Because data is transferred to/from naturally aligned addresses, some number of the low-order address bits are not significant depending upon the transaction type.

For longword transactions, the two low-order bits are not significant except when the transaction is directed to a word or byte-oriented I/O device. (A longword transaction is used to address these devices because there is no specific word or byte transaction type.) Word and byte references are made only to devices on other buses through an adapter and are restricted to node window space.

Multilongword write transfers are not wrapped, so the three low-order address bits of write quadword transactions and the four low-order address bits of write octaword transactions are not significant. However, multilongword read transactions may be wrapped. Thus, the address must specify the first longword to be transferred and only the two low-order address bits are not significant.



NOTE: VAXBI MEMORY SPACE IN A VAX 8800 SYSTEM IS THE SYSTEM'S MAIN MEMORY SPACE (ON THE NMI) EXCEPT FOR ANY MEMORY SPACE ALLOCATED TO AN I/O PROCESSOR'S VAXBI MEMORY NODE.

SCLD-426

Figure 2-4 VAXBI Address Space

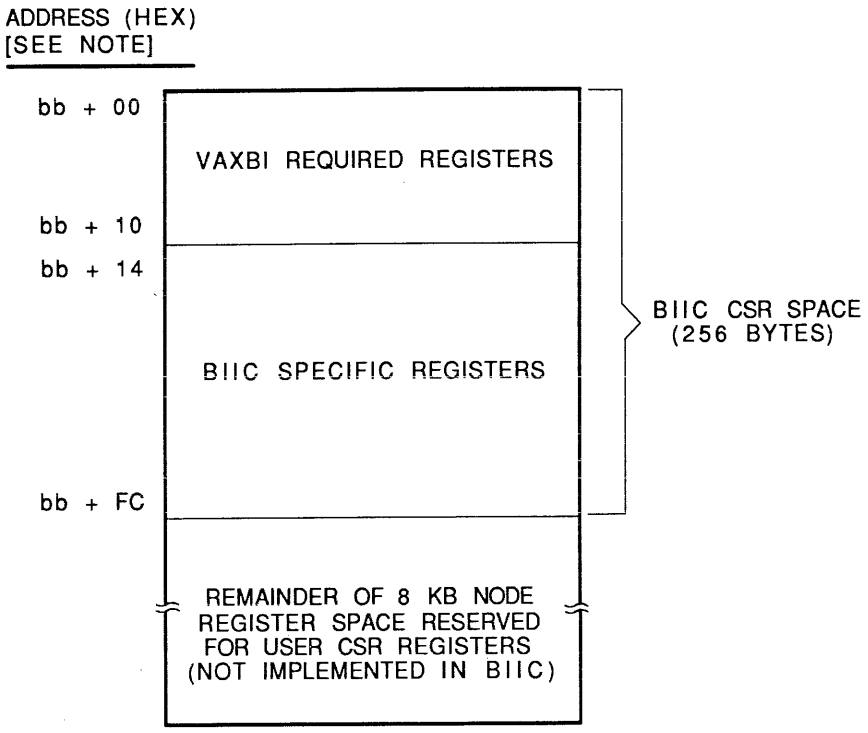


Figure 2-5 VAXBI Node Register Space

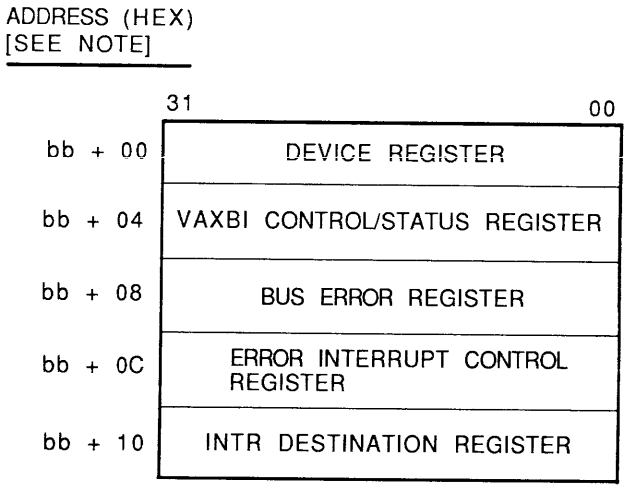


Figure 2-6 VAXBI Required Registers

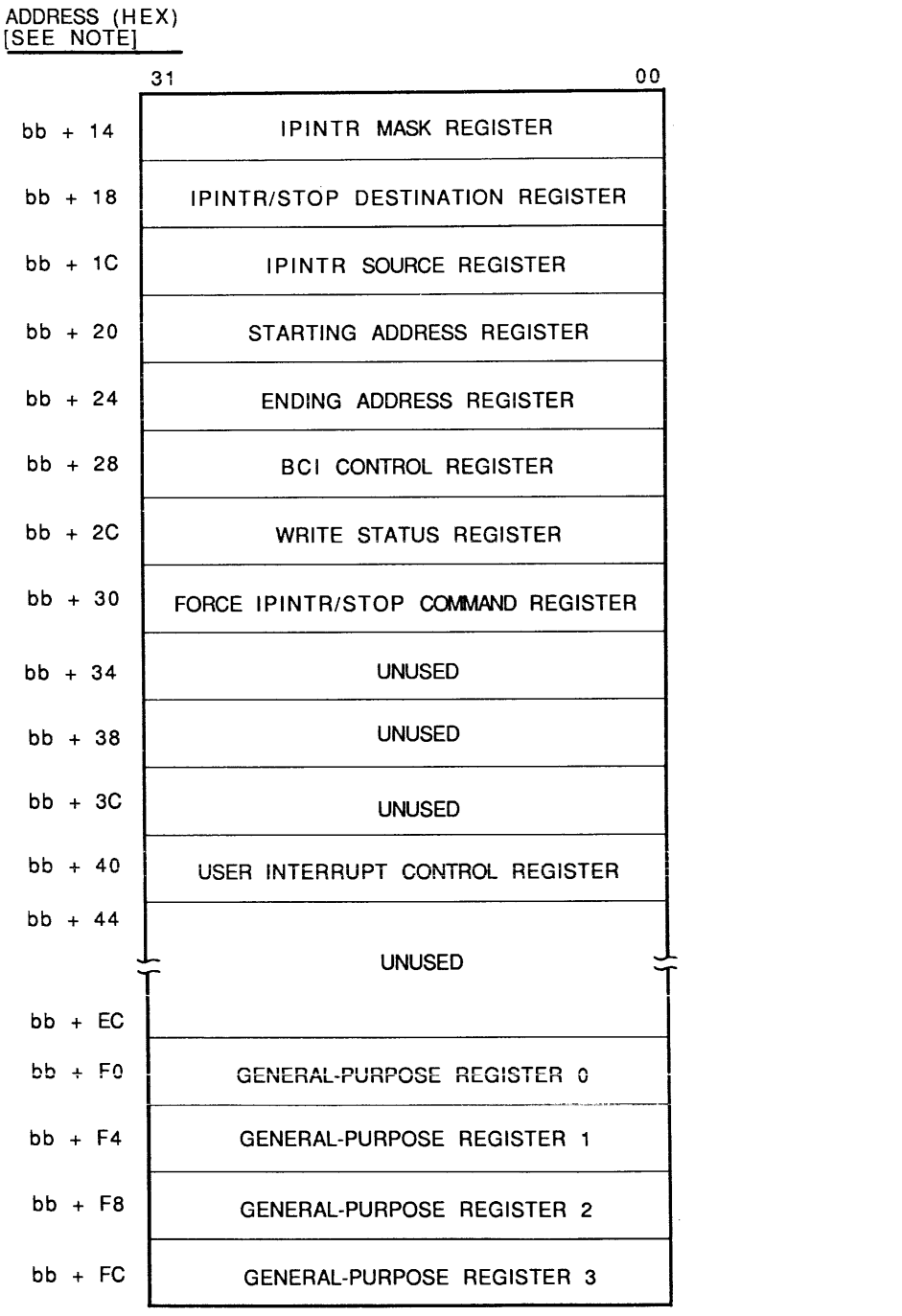
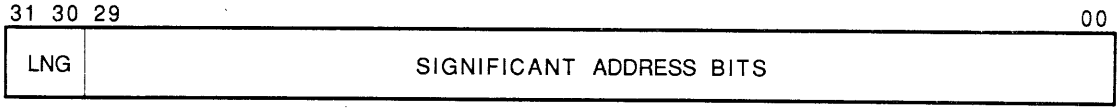
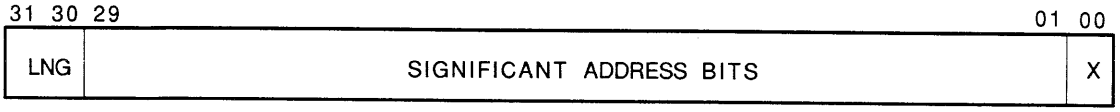


Figure 2-7 BIIC-Specific Device Registers

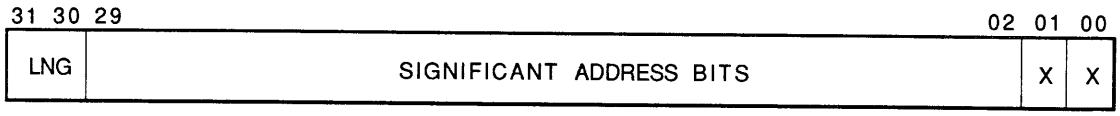
READ/WRITE LONGWORD (TO BYTE-ORIENTED DEVICE, WINDOW SPACE ONLY)



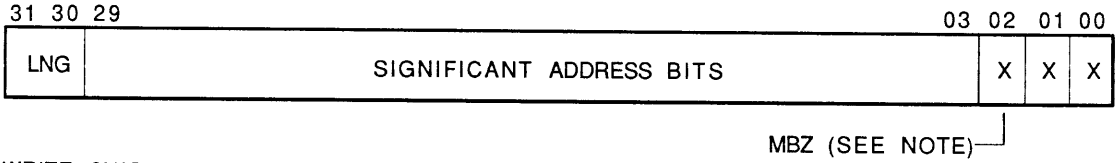
READ/WRITE LONGWORD (TO WORD-ORIENTED DEVICE, WINDOW SPACE ONLY)



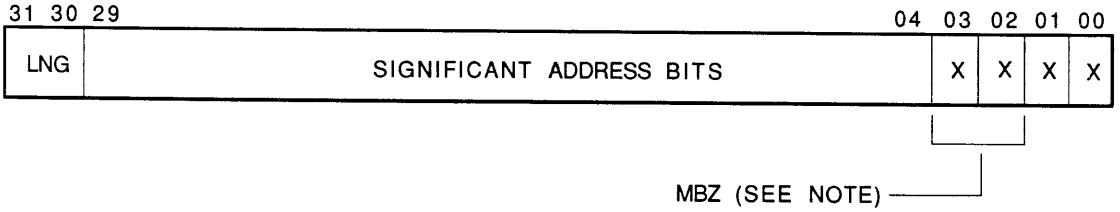
READ/WRITE LONGWORD, READ QUADWORD, READ OCTAWORD



WRITE QUADWORD



WRITE QUADWORD



NOTES:

1. AN X INDICATES BIT IS NOT SIGNIFICANT AND SHOULD BE IGNORED BY SLAVE. HOWEVER, BIT <02> IN WRITE QUADWORD ADDRESS AND BITS <03:02> IN WRITE OCTAWORD ADDRESS MUST BE ZERO(MBZ) TO PREVENT WRAPPED WRITES IN SOME NODES.
2. BITS <31> AND <30> SPECIFY LENGTH OF DATA TRANSFER.

<31:30>	DATA LENGTH
00	RESERVED CODE
01	LONGWORD
10	QUADWORD
11	OCTAWORD

SCLD-183

Figure 2-8 VAXBI Read/Write Address Bits

2.5 BASIC VAXBI TRANSACTION FORMAT

All VAXBI transactions consist of a command/address cycle followed by an embedded arbitration cycle followed by at least one data cycle. Refer to Figure 2-9.

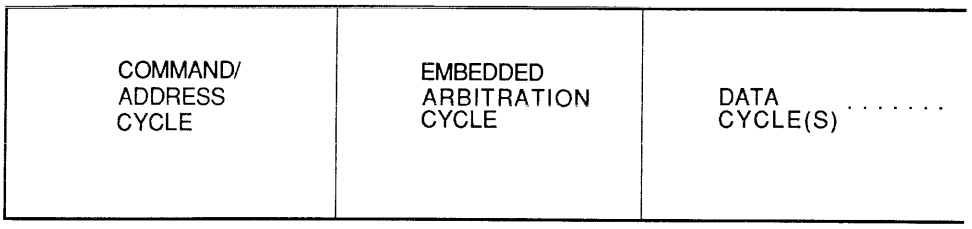


Figure 2-9 Basic VAXBI Transaction Format

2.5.1 Command/Address Cycle

During the command/address cycle, the bus master must select a slave (or slaves) and specify the command type. To select the slave(s), it transmits a read/write address or other selection information such as a destination mask on the data (D) lines. To specify the command type (read, write, etc.), it transmits a 4-bit command code on the information (I) lines. For read/write transaction types, it also transmits a 2-bit length code along with the 30-bit read/write address on the D lines. As discussed previously, this length code specifies the data size (longword, quadword, or octaword) of the read/write transaction.

2.5.2 Embedded Arbitration Cycle

The embedded arbitration cycle is the second cycle of every VAXBI transaction. This cycle provides an additional opportunity for bus arbitration (when the bus is busy). To request use of the bus, a node asserts a D line corresponding to its node ID. The D line may be one of the low-order 16 lines (a high priority request) or it may be one of the high-order 16 lines (a low priority request). Also, during the embedded arbitration cycle, the current bus master transmits its node ID on the I lines. Bus arbitration is discussed in Section 2.11.

2.5.3 Data Cycles

For single-responder transactions (never more than one responding slave), the data cycle or cycles following the embedded arbitration cycle transfer read/write (or vector) data between the master and slave over the D lines. Write data is also transferred for the broadcast transaction, which is a multiresponder transaction (more than one possible responding slave). The other multiresponder transactions contain a data cycle, but no data is actually transferred over the D lines. During this cycle, as in all data cycles, the responding slave(s) transmits a response code to the bus master on the confirmation (CNF) lines.

2.5.4 Bus Parity

Odd parity is generated and checked on the VAXBI during the command/address and embedded arbitration cycles of all transactions. It is also generated and checked during the data cycles of transactions when read/write (or vector) data is being transmitted. The parity bit is asserted on the P0 line.

Except for the embedded arbitration cycle, parity is generated for the information on both the D and I lines. During an embedded arbitration cycle, only I-line parity is generated.

The bus master generates the parity bit during command/address and embedded arbitration cycles. Parity is checked by all nodes (including the master) and each node sets an error flag if bad (even) parity is detected. Nodes detecting a parity error in the command/address cycle will not respond to the command/address. This aborts the transaction. A transaction is not aborted by parity errors detected during the embedded arbitration cycle. Parity generation and checking during data cycles is discussed in the following sections.

2.6 READ/WRITE TRANSACTIONS

Examples of write type and read type transactions are shown in Figures 2-10 and 2-11. Except for data length (octaword in the examples), all write transaction formats are basically the same and all read transaction formats are basically the same. A write transaction may be a normal write, a write with cache intent, a write masked (with cache intent), or an unlock write masked (with cache intent). A read transaction may be a normal read, a read with cache intent, or an interlocked read (with cache intent).

The number of data cycles following the command/address and embedded arbitration cycles of a write or read transaction depends upon the length of the transfer. It also depends upon whether there are any stalls or a retry (refer to Sections 2.6.4 and 2.6.5). A minimum of four data cycles are required for an octaword data transfer. Correspondingly, a quadword transfer requires a minimum of two data cycles and a longword transfer requires a minimum of one.

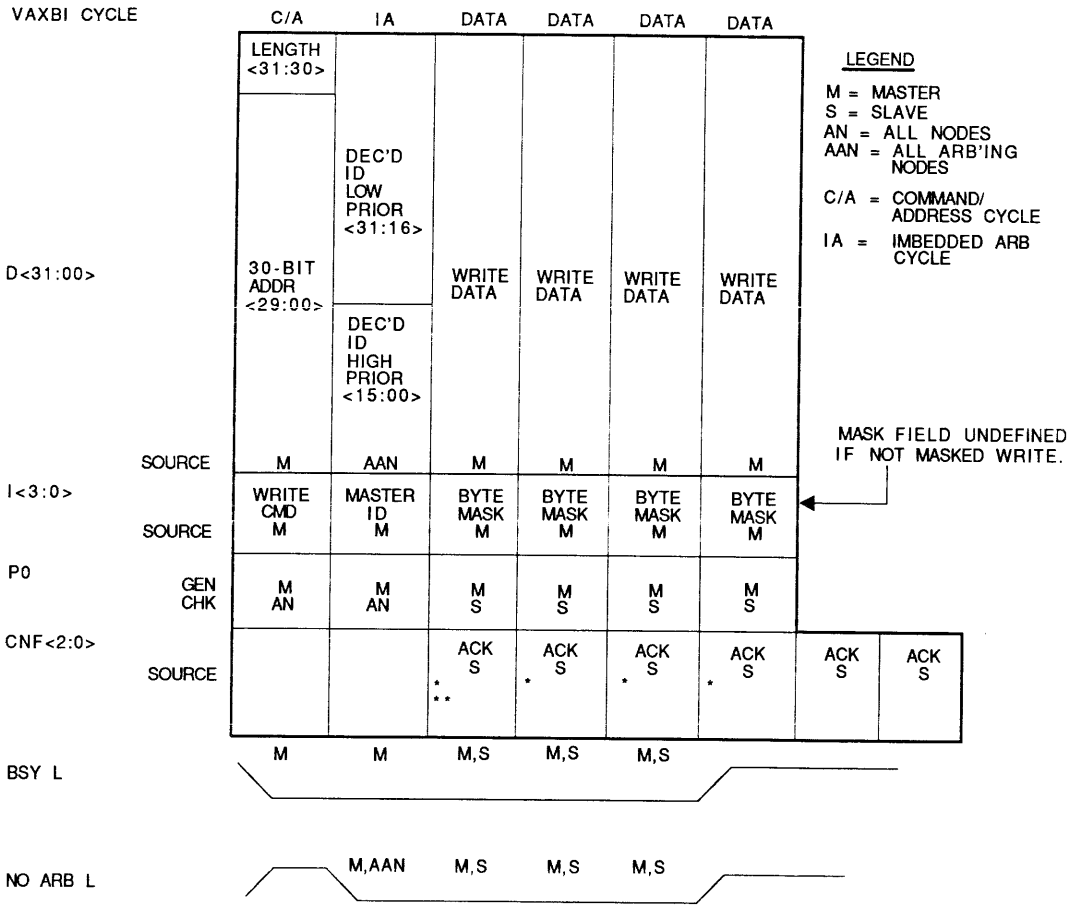
2.6.1 Write Data Cycles

During each data cycle of a write transaction, the bus master transmits the write data to the selected slave on the D lines and transmits a 4-bit byte mask on the I lines if the write transaction is a masked write. (The byte mask indicates which byte or bytes in the longword of write data are to be written.) The bus master also transmits odd parity for the D and I lines on the P0 line. The selected slave (if it is ready and if it has detected no errors) takes the write data and transmits an acknowledgment (ACK) code back to the master on the CNF lines. An ACK response is also generated by the slave for the two cycles following the last data cycle to indicate that no error was detected when executing the transaction.

If an error (such as a parity error) is detected by the slave during a data cycle, it asserts no CNF lines (a NO ACK response) during any remaining data cycles in the transaction and for at least two cycles following the last data cycle. The NO ACK response causes the master to set an error flag and terminate the transaction.

2.6.2 Read Data Cycles

During each data cycle of a read transaction, the selected slave (if it is ready and if it has detected no errors) transmits the read data back to the master on the D lines, together with an ACK response on the CNF lines. It also transmits a read data status code on the I lines as well as the D and I-line odd parity bit on the P0 line.

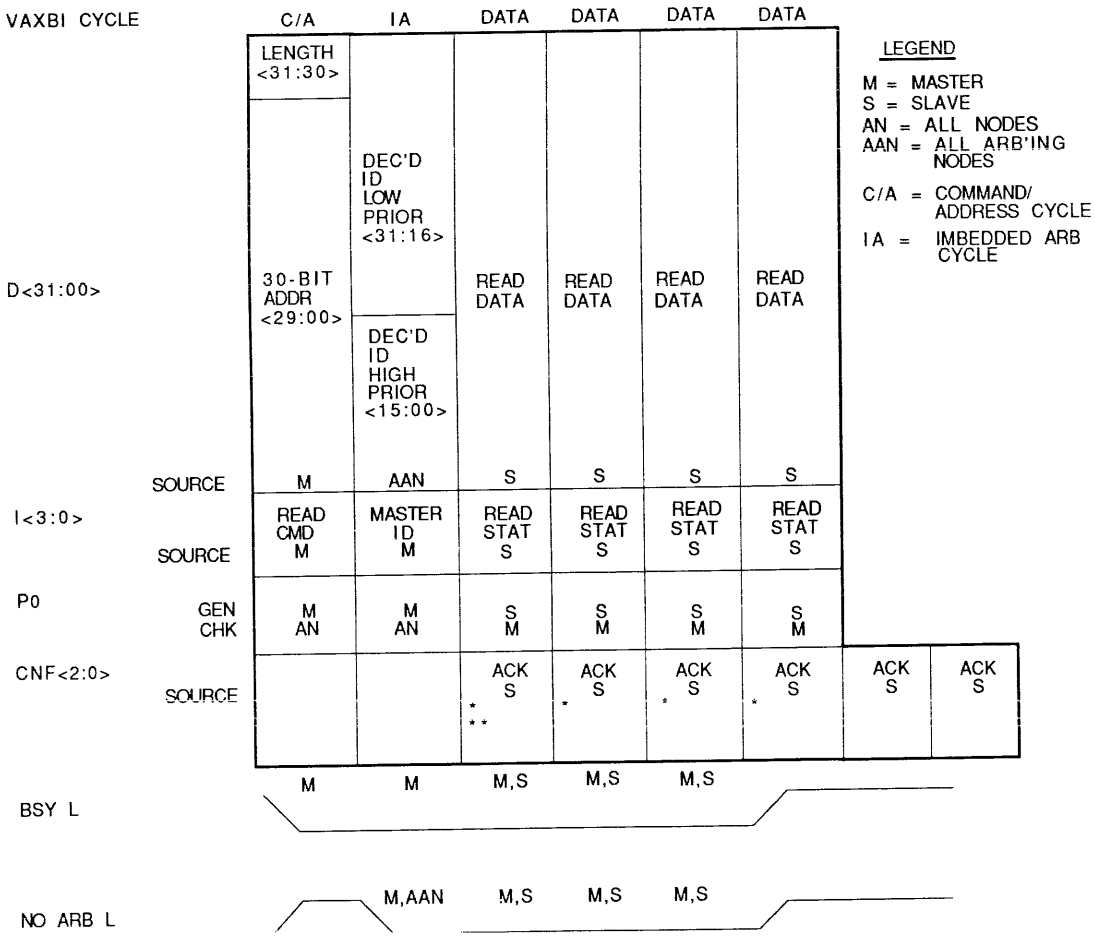


NOTES: 1. AN ASTERISK (*) INDICATES SLAVE MAY STALL (GIVE STALL RESPONSE) FOR ONE OR MORE BUSY CYCLES BEFORE TAKING DATA (ACK RESPONSE).

2. A DOUBLE ASTERISK (**) INDICATES SLAVE MAY REQUEST RETRY OF TRANSACTION (GIVE RETRY RESPONSE).

SCLD-185

Figure 2-10 VAXBI Write Transaction (Octaword Length)



NOTES: 1. AN ASTERISK (*) INDICATES SLAVE MAY STALL (GIVE STALL RESPONSE) FOR ONE OR MORE BUSY CYCLES BEFORE TAKING DATA (ACK RESPONSE).

2. A DOUBLE ASTERISK (**) INDICATES SLAVE MAY REQUEST RETRY OF TRANSACTION (GIVE RETRY RESPONSE).

SCLD-186

Figure 2-11 VAXBI Read Transaction (Octaword Length)

The status code indicates if the data is valid, valid but corrected, or uncorrectable (READ DATA SUBSTITUTE code). (A READ DATA SUBSTITUTE code causes the master to set an error flag.) Also, to limit the caching and subsequent invalidation of data in some systems, the status code indicates when the read data is not to be cached.

Like the write, an ACK response is transmitted on the CNF lines (this time by the master) for two cycles after the last data cycle if the transaction completed successfully. A NO ACK response by the master indicates that an error (a parity error, for example) was detected during transaction execution causing the slave to set an error flag. Also, as for a write, a NO ACK response by the slave during (and not after) the data cycles of a read transaction is an error condition. The error condition causes the master to set an error flag and end the transaction.

2.6.3 Nonexistent Addresses

If there is a NO ACK response (no responding slaves) during the first cycle of a read or write transaction, it probably indicates the master transmitted a nonexistent memory or I/O address. As for any NO ACK response during a data cycle, the master sets an error flag and ends the transaction.

2.6.4 Stalls

If the selected slave is not ready to accept write data or return read data during any data cycle, it can assert a STALL code on the CNF lines.

During stall data cycles for a write transaction, the master continues to transmit the same longword of write data on the D lines until the slave takes the data and generates an ACK response. For a read transaction, the D lines are undefined (may be in any state) until the stalled data and an ACK response are transmitted by the slave.

If a slave has to generate 128 consecutive STALL responses, it sets an error flag, generates a NO ACK response, and ends the transaction.

2.6.5 Retries

A selected slave not ready to accept write data or send read data may not just stall. It may also cause the master to retry the transaction.

A slave may cause a retry in the first data cycle or following a stall data cycle (if no data has previously been transferred) by transmitting a RETRY response on the CNF lines. The master, or the slave following a stall data cycle, then ends the transaction. The master can retry the transaction after arbitrating again for the bus. If a slave causes a transaction to be retried 4096 times, the master sets an error flag.

2.7 BROADCAST TRANSACTIONS

The broadcast transaction, not currently used during normal bus operation, allows more than one node to be written at a time. It provides a means of announcing events to a number of nodes without using interrupt requests.

Format for the broadcast transaction is shown in Figure 2-12. An octaword data length is shown. The transaction is similar to a write, except that the bus master transmits no memory or I/O address on the D lines during the command/address cycle. Instead, it transmits a destination mask on the 16 low-order D lines. The D lines asserted correspond to the node IDs of the slaves to be selected.

During each data cycle of a broadcast transaction, at least one slave must generate an ACK response on the CNF lines. A NO ACK response during a data cycle (or during the two cycles after the last data cycle) causes the master to set an error flag during a write transaction. Unlike a write, and because more than one slave can respond to the master, a single slave cannot stall or cause a retry of the broadcast transaction. Thus, the master sets an error flag if it receives a STALL or RETRY response.

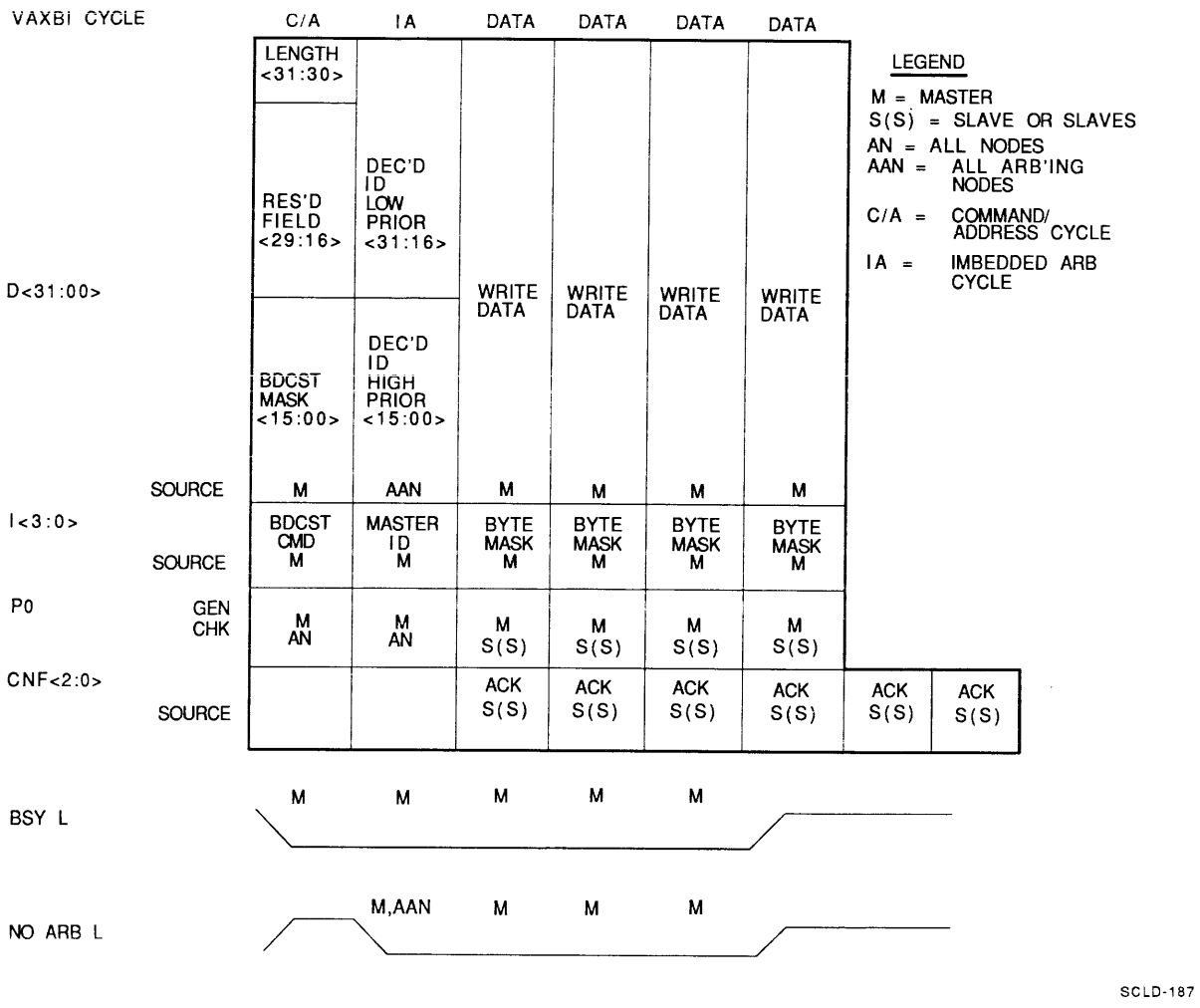


Figure 2-12 VAXBI Broadcast (BDCST) Transaction (Octaword Length)

2.8 INVALIDATE TRANSACTIONS

The invalidate transaction (Figure 2-13) allows a processor node to signal other nodes that they may have cached data that is no longer valid. The bus master transmits the address and data length of the invalid block of data in the command/address cycle. More than one slave may respond with an ACK response on the CNF lines during the transaction's single data cycle. (No data is actually transferred and parity is not generated or checked during this cycle.) If there is no response by any node, an error flag is set in the master.

The invalidate transaction is not used on a VAXBI in this system. Any processor nodes must have their caches turned off as stated previously. This is because the addresses of memory transactions local to the NMI are not passed to the VAXBI; thus, caches on the VAXBI (if turned on) could contain invalid data.

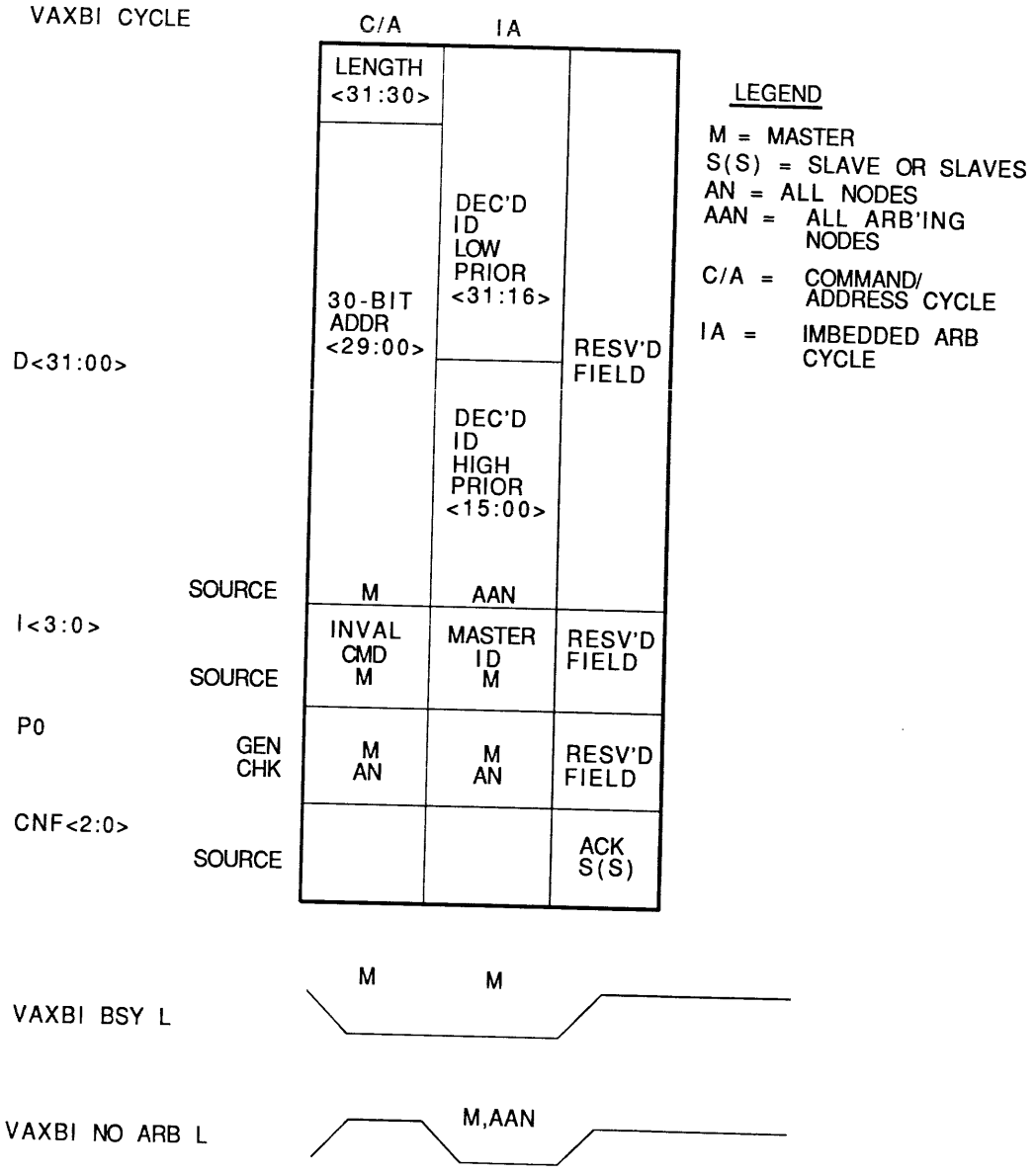


Figure 2-13 VAXBI Invalidate (INVAL) Transaction

2.9 INTERRUPT OPERATION (INTR, IDENT, AND IPINTR TRANSACTIONS)

The interrupt (INTR) and identify (IDENT) transactions are used to signal and service conventional device type interrupts on the VAXBI. That is, a node can send an interrupt request to one or more nodes using the INTR transaction. (As for the NMI, there are four request priority levels; BR4, 5, 6, and 7.) Then, when an interrupt fielding node is ready to service the interrupt requests at a specific request level, it uses an IDENT transaction to read an interrupt vector from the interrupting node. Because more than one node may be interrupting at that request level, the vector is read from the highest priority device based on the node ID.

The VAXBI also allows one processor node to interrupt another processor node. This is done using the interprocessor interrupt (IPINTR) transaction. Transaction format is similar to the INTR transaction. However, there is no need for a node responding to the interrupt request to follow with an IDENT transaction. This is because the interrupt fielding node stores the vector (and also the request level) information for this type of interrupt.

The interrupt fielding nodes on a VAXBI in the system are the NBI and any I/O processors that are installed. The NBI node services INTR transactions by passing the interrupt requests on to the NMI and the primary CPU. It then issues an IDENT transaction to collect the vector information when the primary CPU reads an NBI vector register over the NMI. (There are four vector registers in the NBI, one for each request level.) The interrupt requests generated on the VAXBI are normally from I/O device nodes but may also be from an I/O processor node.

The NBI can also field IPINTR transactions allowing an I/O processor to interrupt the primary CPU using an interprocessor interrupt request. (An I/O processor can also interrupt another I/O processor using an IPINTR.) Not only can the NBI field an IPINTR, it can generate one. It can also generate an INTR, allowing the primary CPU to interrupt an I/O processor with either type of interrupt request.

2.9.1 Interrupt (INTR) Transactions

Format for the INTR transaction is shown in Figure 2-14. During the command/address cycle, the master (the interrupting node) transmits a 16-bit destination mask on the low-order D lines. Each bit in the mask corresponds to one of the 16 possible nodes on the VAXBI, allowing the master to select one or more slaves to field the interrupt. For example, the master may signal both the NBI and an I/O processor that it is interrupting.

The master also transmits the interrupt request level on four of the high-order data lines during the command/address cycle. There is a bit for each level as shown below. This allows a master (an adapter node, for example) to make an interrupt request at more than one request level with a single INTR transaction. (More than one I/O device attached to the adapter can be interrupting at the same time and at different assigned request levels.) When more than one request level is specified, the interrupt fielding node (when it is ready) will service the highest priority request only. Requests at other levels have to be made again using another INTR transaction.

D<19:16>	Request Level
1XXX	BR7 (Highest Priority)
X1XX	BR6
XX1X	BR5
XXX1	BR4 (Lowest Priority)

During the single data cycle following the command/address and embedded arbitration cycles, any selected slave that intends to service the interrupt request asserts an ACK response on the CNF lines. This is essentially a command confirmation cycle with no data being transferred between the master and the selected slave(s). If no slave responds (a NO ACK response), the master sets an error flag.

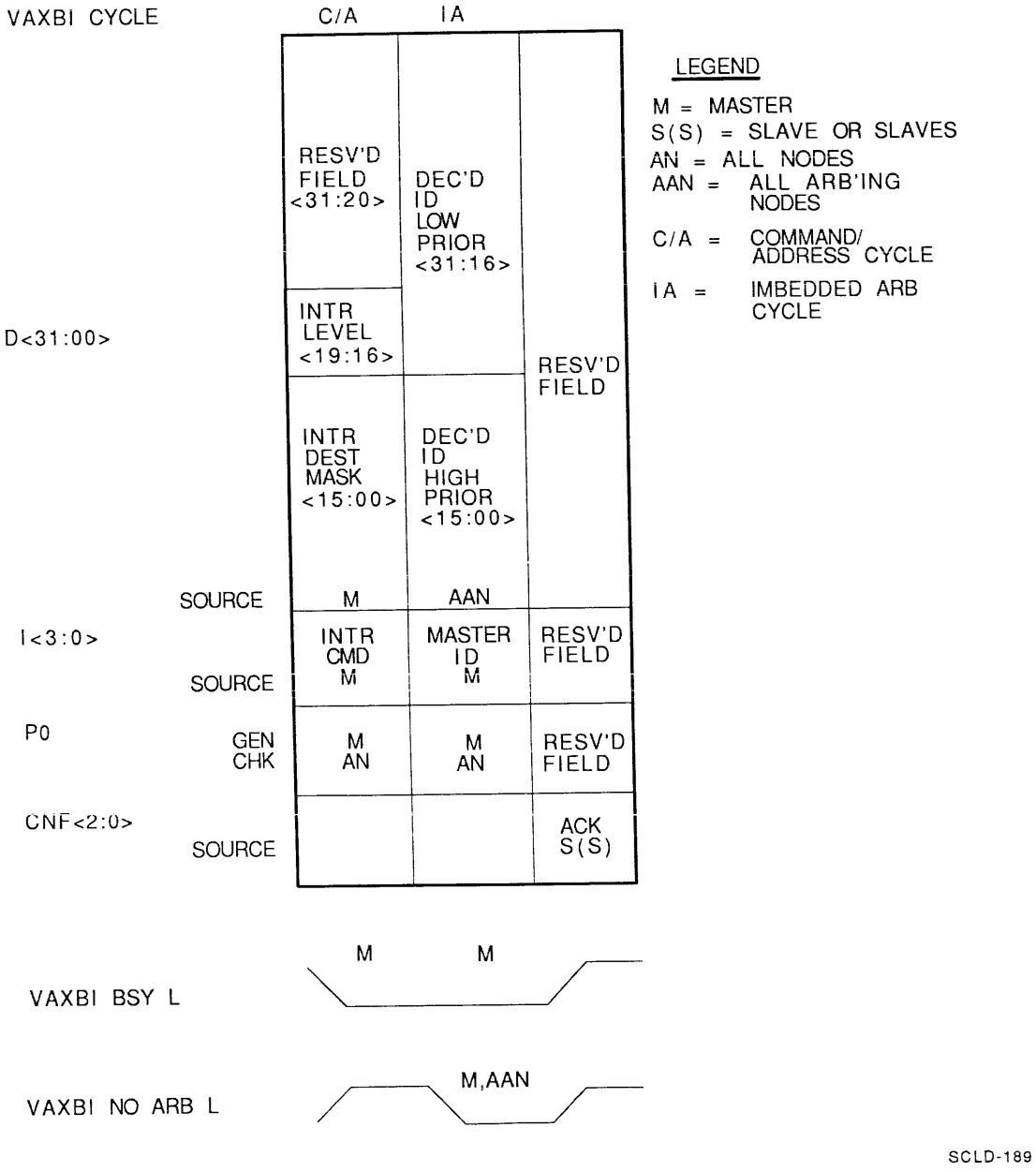


Figure 2-14 VAXBI Interrupt (INTR) Transaction

2.9.2 Identify (IDENT) Transactions

As stated previously, an interrupt fielding node responding to an INTR transaction does an IDENT transaction to read an interrupt vector. IDENT format is shown in Figure 2-15.

During the command/address cycle and similar to the INTR transaction, the master transmits the interrupt level on four of the high-order D lines. However, this is the only select information transmitted and only one of the four lines should be asserted. The line asserted specifies the single interrupt request level to be serviced.

D<19:16>	Request Level
1000	BR7 (Highest Priority)
0100	BR6
0010	BR5
0001	BR4 (Lowest Priority)

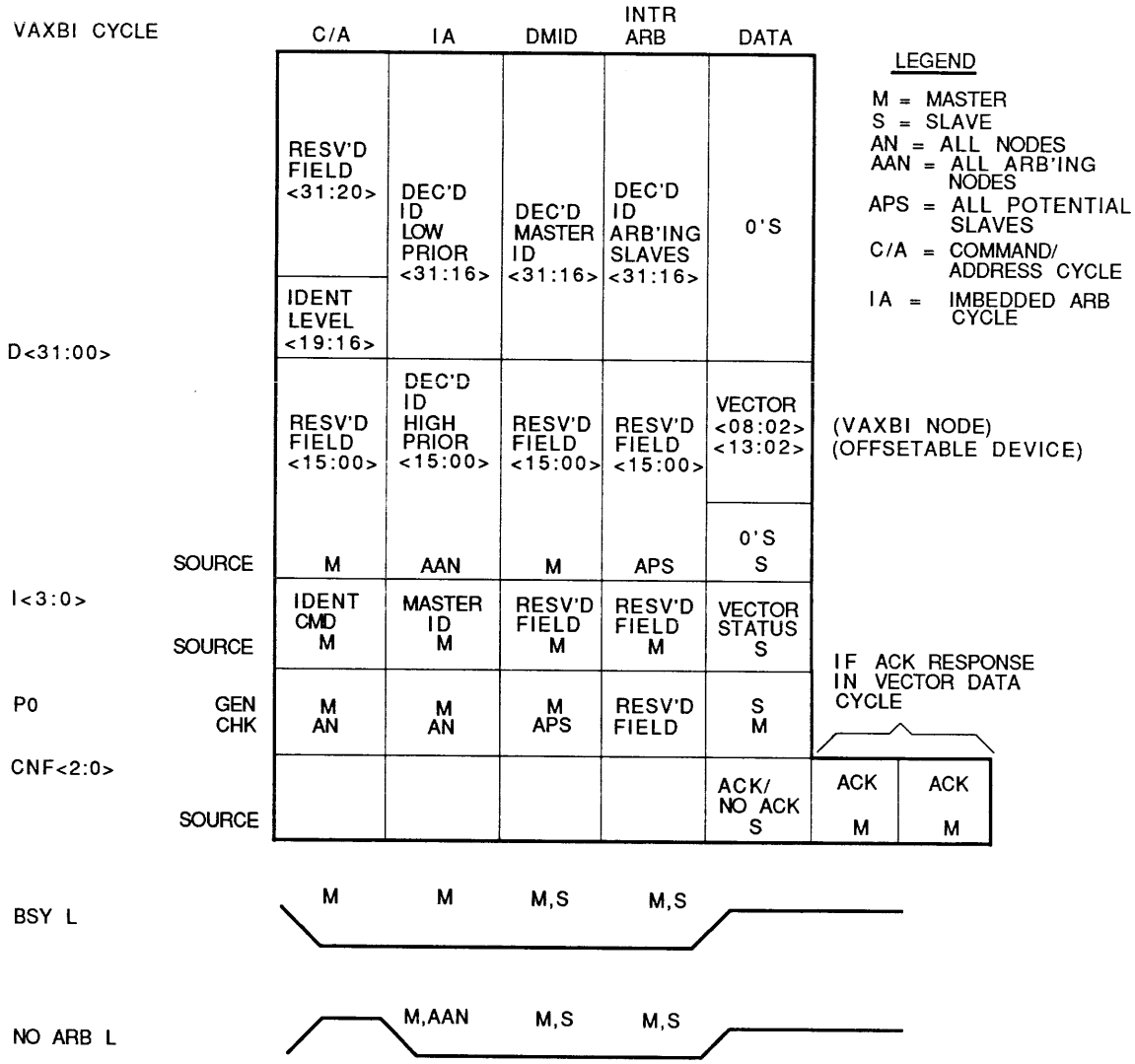
During the cycle following the command/address cycle and embedded arbitration cycle, the master transmits its decoded ID on the 16 high-order D lines. Only the D line corresponding to the master's node ID (plus 16) will be asserted. The transmission of the master's ID is necessary because there can be more than one interrupt fielding node on the bus, and nodes with an interrupt pending at the specified request level may not want service by the node doing the IDENT.

During the next bus cycle of the IDENT, the nodes wanting interrupt service by the master must arbitrate for the bus as when arbitrating to become bus master (Section 2.11). (However, decoded IDs are transmitted only on the high-order data lines.) The nodes must arbitrate because only one may be serviced by the IDENT. That is, only one node may become the slave and return an interrupt vector to the master. Nodes not winning the arbitration must make another interrupt request (INTR transaction) unless they are serviced by another IDENT before a request can be made.

After winning the bus, the slave (if it is ready) asserts an ACK response on the CNF lines and transmits the vector on the D lines during the next bus cycle. At the same time, and like a read transaction, the slave transmits a data status code on the I lines, and it transmits D and I-line parity on the P0 line. (A READ DATA SUBSTITUTE status code causes the master to set an error flag.) Also, similar to a read transaction, the slave may transmit a STALL response on the CNF lines for one more cycle until the vector is ready to transmit. (An adapter node would do this if the vector had to be first read from an attached I/O device.) Again, 128 stall cycles cause the slave to set an error flag and end the transaction. Once the vector is sent by the slave ending the transaction, the master transmits an ACK response on the CNF lines for the next two bus cycles if the transaction was executed correctly. If not (a parity error detected by the master, for example), a NO ACK response causes an error flag to be set in the slave.

A slave serviced by an IDENT may not necessarily have made a previous interrupt request. The interrupt condition may have occurred just before it could arbitrate for the bus and transmit an INTR transaction. For this reason, a node cannot arbitrate for the bus to do an INTR during the embedded arbitration cycle of the IDENT. It may win the bus and become pending bus master, and then it may win the following arbitration for service by the IDENT. It would then have to abort the INTR because the request had already been serviced.

A NO ACK response occurring during (not after) an IDENT transaction is not an error condition. This can happen if the IDENT is the result of an interrupt request to more than one interrupt fielding node, and the request has already been serviced by another node.



NOTE: A NO ACK BY SLAVE IS A VALID RESPONSE INDICATING A TRANSIENT INTERRUPT OR THE INTERRUPT HAS BEEN SERVICED BY ANOTHER NODE. ALSO, SLAVE MAY STALL (GIVE STALL RESPONSE) FOR ONE OR MORE BUS CYCLES BEFORE RETURNING VECTOR (ACK RESPONSE).

SCLD-190

Figure 2-15 VAXBI Identify (IDENT) Transaction

2.9.3 Interprocessor Interrupt (IPINTR) Transaction

The format for the IPINTR transaction (Figure 2-16) differs from the INTR transaction only in the selection information transmitted by the master in the command/address cycle. A destination mask is transmitted on the low-order D lines, but no interrupt level is transmitted on the high-order D lines. This is because the interrupt fielding node (the slave) holds the interrupt request level. Instead, the master's decoded ID is transmitted on the high-order D lines as in the third cycle of an IDENT. This is necessary because a slave may not be enabled to accept interprocessor interrupt requests from the current master.

As for the INTR, the single data cycle following the embedded arbitration cycle is essentially a command confirmation cycle. (No data is transferred between master and slave.) At least one slave must respond with an ACK during the cycle or an error flag is set in the master.

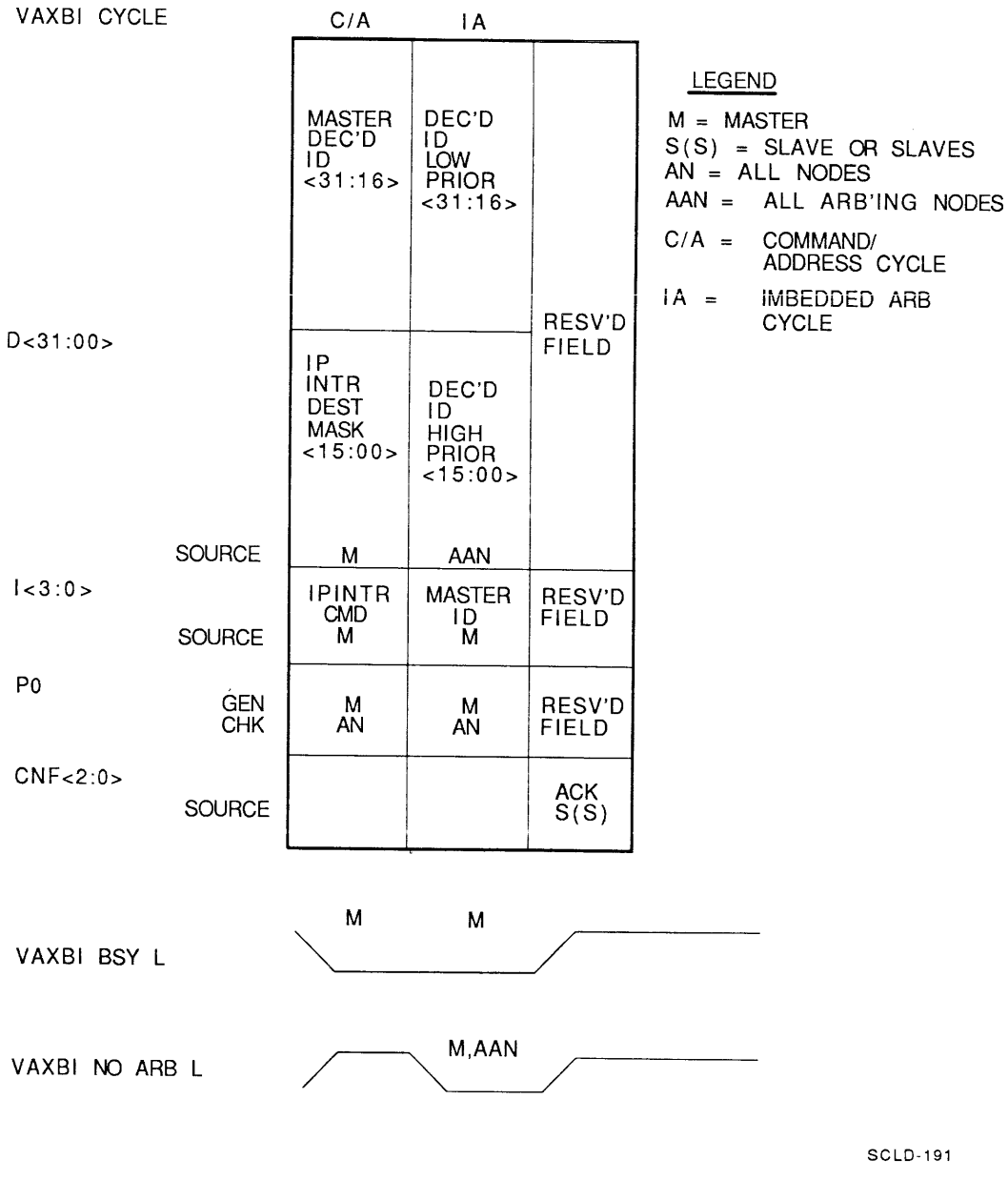


Figure 2-16 VAXBI Interprocessor Interrupt (IPINTR) Transaction

2.10 STOP TRANSACTIONS

The STOP transaction is used to force nodes to a state where they cannot issue any more VAXBI transactions while retaining as much error and other status information as possible. However, nodes must still be able to respond to VAXBI transactions so that the retained status information can be examined by another node.

A processor node can use the STOP transaction to first stop all bus activity after an error condition. It can then read status registers in one or more nodes for error logging or diagnostic purposes. A node can be returned to normal operation by forcing a selftest operation. (A DC LO indication is generated by the BIIC during the selftest sequence, which initializes the node.)

STOP transaction format (Figure 2-17) is similar to INTR and IPINTR format, except that the master transmits only a destination mask on the D lines during the command/address cycle. More than one slave may be addressed and at least one must generate an ACK response on the CNF lines or the master sets an error flag. Like the INTR and IPINTR, the response is generated in the single data cycle following the embedded arbitration cycle. (No data is actually transferred over the D lines in this cycle.)

If a responding slave cannot enter STOP mode by the end of the transaction's single data cycle, it still generates an ACK response. However, until the STOP sequence is complete, it must either hold the bus (by asserting BSY) or it must generate RETRY responses to single-responder commands and NO ACK responses to multiresponder commands. (The BSY signal is discussed in Section 2.11.)

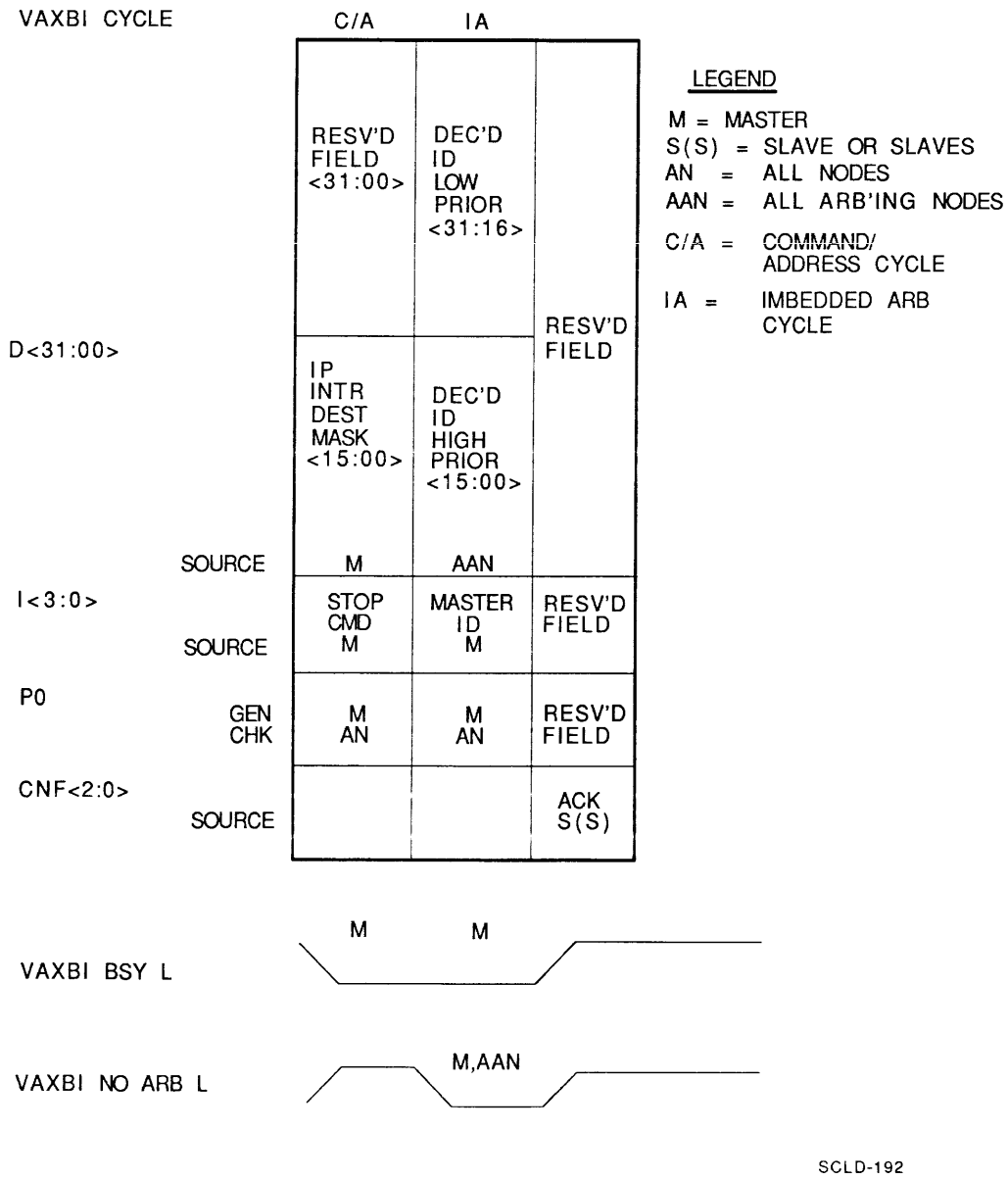


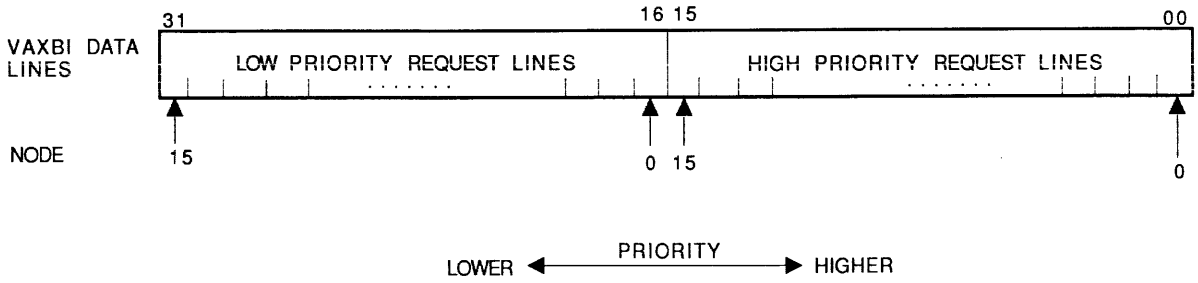
Figure 2-17 VAXBI STOP Transaction

2.11 BUS ARBITRATION AND CONTROL

A node may request the bus during any bus cycle when the bus is inactive. It may also request the bus during the embedded arbitration cycle of any VAXBI transaction when the bus is busy. Each requesting node monitors all requests (distributed arbitration), and if it has the highest priority, assumes control of the bus as bus master, either in the next cycle (when there is no transaction in progress) or at the end of the current transaction. When a node wins the bus during the embedded arbitration cycle of the current transaction, it is called the pending bus master until it assumes control of the bus when the current transaction ends.

2.11.1 Bus Requests

Nodes request use of the bus by asserting a D line that corresponds to its node ID. As shown in Figure 2-18, the line asserted may be one of the 16 high-order lines or one of the 16 low-order lines. The low-order lines are high priority requests. The high-order lines are low priority requests. Within each group of high or low priority requests, the node with the lowest ID (lowest numbered D line asserted) has the highest priority. Of course, any high priority request always has a higher priority than any low priority request.



NOTE: NODE ASSERTS EITHER A HIGH OR LOW REQUEST LINE TO ARBITRATE FOR VAXBI.

SCLD-193

Figure 2-18 Bus Arbitration Request Lines

2.11.2 Arbitration Modes

Whether the node asserts its high priority line or its low priority line depends on the arbitration mode. There are three modes.

1. Dual round robin
2. Fixed high priority
3. Fixed low priority

Nodes are normally programmed to operate in dual round-robin mode. In this mode, a requesting node asserts its high priority request only if the previous bus master has a lower ID (higher priority) than it does. At any other time, it asserts its low priority line. All nodes store the ID of the previous bus master, which is asserted on the I lines during the embedded arbitration cycle of the previous transaction.

On the average, dual round-robin mode assures equal access to the bus for all nodes. If a node requires rapid access to the bus as in some special real-time applications, it can be programmed to operate in fixed high-priority mode. That is, it will always assert its high-priority request line when requesting the bus. Also, access may be further enhanced by programming other nodes to operate in fixed low-priority mode (only low-priority requests will be asserted).

2.11.3 Arbitration Control

The arbitration for the bus by any VAXBI node (as described above) is shown in Figure 2-19. As can be seen, arbitration is controlled by two bus signals, NO ARB and BSY.

NO ARB is asserted by any nodes requesting the bus. It is also asserted by the current and pending bus masters and the slave at various times during a transaction depending upon the transaction type, its length, and the responses generated. The net result is that NO ARB is always asserted except during the following bus cycles.

1. Null cycles (no bus arbitration or transaction in progress)
2. Command/address cycles
3. The last data cycle in transactions unless there is a pending bus master

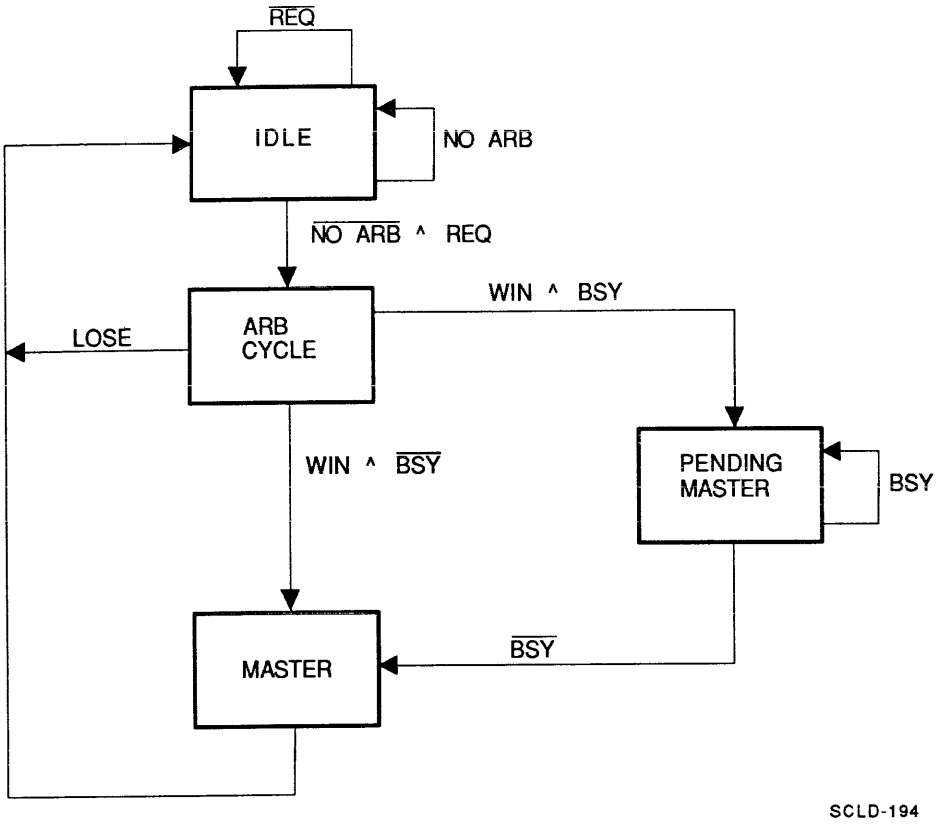


Figure 2-19 Arbitration State Diagram

BSY is asserted only by the current master and slave during a transaction. The master asserts BSY during the command/address and embedded arbitration cycles. Both the master and slave can assert BSY during data cycles as for the NO ACK signal. BSY can also be asserted to extend a transaction and, together with NO ARB, during special mode functions as well. BSY is not asserted during the following bus cycles.

1. Null cycles

2. Arbitration cycles (when there is no transaction in progress)

3. The last cycle of transactions or special mode functions (see Section 2.11.5)

When NO ARB is not asserted, nodes may arbitrate for the bus during the next cycle. When BSY is not asserted, a node may begin a transaction in the next cycle. For example, a node winning a bus arbitration when BSY = 0 becomes bus master and begins the transaction in the next cycle. (Refer again to Figure 2-19.) Similarly, if a node wins the bus during the embedded arbitration cycle when BSY = 1, it must wait as pending bus master until BSY = 0 before it can begin its transaction in the next cycle.

Timing for NO ARB and BSY during a typical bus operation is shown in Figure 2-20. As can be seen, when there is little bus activity and no nodes arbitrating in the embedded arbitration cycle of a transaction, there is at least one dead cycle before the next transaction can begin. However, with increased bus activity and nodes arbitrating during embedded arbitration cycles, one transaction immediately follows the next, raising the data transfer rate for the bus.

2.11.4 Extending a Transaction

BSY may be asserted by a node to extend a transaction one or more cycles beyond its normal length. The additional cycles, called busy stall cycles, stop bus activity until the node is ready to respond properly to another transaction. For example, a node responding to the previously discussed STOP transaction may assert BSY to delay the start of the next transaction until it can enter STOP mode.

2.11.5 Special Mode Functions

Both BSY and NO ARB may be asserted by a node during execution of some special mode functions. A bus transaction between other nodes may or may not be in progress. As when extending a transaction, the assertion of BSY delays the start of the next bus transaction until the special mode function is completed. (The NO ARB signal is asserted to identify the operation as a special mode function.) For example, a node doing a BIIC loopback operation cannot respond to bus transactions and it asserts BSY and NO ARB to prevent any transaction until the loopback is complete. A loopback operation is when the node is reading or writing one of its own BIIC registers using internal data paths only.

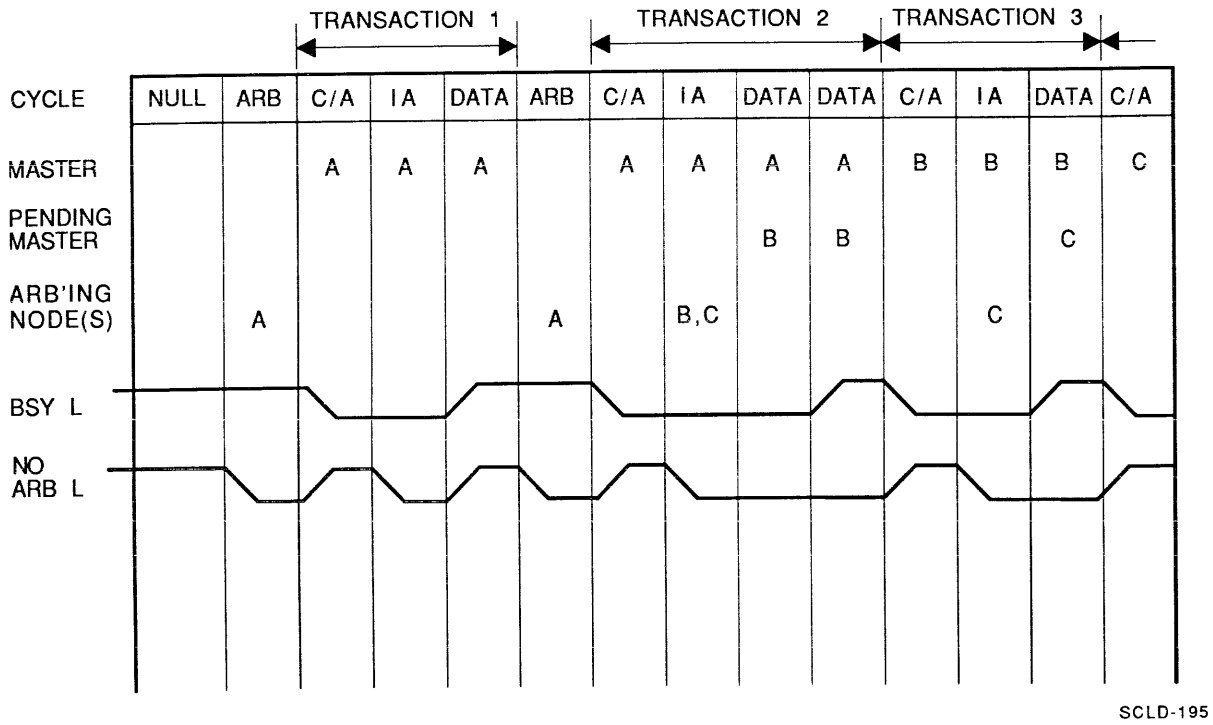


Figure 2-20 VAXBI Arbitration (Example)

2.12 VAXBI ERRORS

The error detection by VAXBI nodes (done by the BICC) consists of the following:

1. Parity Checking
2. Transmit Check Error Detection
3. Protocol Checking

2.12.1 Parity Checking

The VAXBI has a single parity line (P0). Except for embedded arbitration cycles, the parity bit (when it is generated) is for the information on the D and I lines. During embedded arbitration cycles, only I-line parity is generated. The nodes generating and checking parity during VAXBI cycles are listed below.

Cycle	Parity Generating Node	Parity Checking Node(s)	Transaction Aborted
Command/Address	Master	All	No
Embedded Arbitration	Master	All	No
Decoded ID (IDENT)	Master	Potential Slave(s)	No
Write Data	Master	Slave	Yes
Read (Vector) Data	Slave	Master	Yes
Null	N/A	All (see note)	N/A

NOTE
All nodes check that no D or L lines are asserted during null cycle.

When a node detects a parity error, it sets an error flag in its bus error register (in the BIIC) and generates an interrupt request if error interrupts are enabled. Nodes detecting bad parity during command/address cycles do not acknowledge (ACK) the command/address. Also, interrupt nodes that detect a parity error during the decoded ID cycle of an IDENT transaction do not participate in the IDENT arbitration cycle.

2.12.2 Transmit Check Error Detection

There are two types of transmit check errors. One type is when the information transmitted by a master on the D, I, and P0 lines does not compare with the information received (by the same node). The check is made during command/address, write data, and decoded ID (IDENT) cycles when the master is the only node transmitting information on the D, I, and P0 lines. When the information transmitted and received does not compare, the master sets an error flag in its bus error register and generates an interrupt request (if enabled). The transaction is also aborted. The other type of transmit check error is when a master or a slave should be asserting BSY or NO ARB, and it does not detect the asserted state on the VAXBI lines. Again, the node sets an error flag in its bus error register and generates an interrupt request (if enabled). However, if a transaction is in progress, it is not aborted.

2.12.3 Protocol Checking

The following errors in VAXBI transaction execution are checked. When an error is detected by a node, it sets an error flag in its bus error register and generates an interrupt request (if enabled).

1. NO ACK to Multi-Responder Command Received - A master received a NO ACK response for an INVAL, STOP, INTR, or IPINTR command.

2. Interlock Sequence Error - A node successfully completed an unlock write transaction that was not preceded by a corresponding read interlocked transaction.

3. IDENT Vector Error - An ACK response was not received by the master indicating the vector was not correctly received by the slave.

4. Read Data Substitute Error - A read data substitute (or reserved) status code was received with read (or vector) data and no parity error was detected.

5. Retry Timeout - The master received 4096 consecutive RETRY responses from the slave for the same transaction.

6. Stall Timeout - The slave transmitted 128 consecutive STALL responses. Causes the slave to abort the transaction.

7. Bus Timeout - A node was unable to start a pending bus transaction after 4096 consecutive bus cycles.

8. Nonexistent Address - A master received no response to a read/write type command, and it detected no parity error or transmit check error for the command/address information. Causes the master to abort the transaction.

9. Illegal Confirmation Error - A master or slave detected a reserved or illegal response code.

CHAPTER 3
VISIBILITY BUS (VBUS)

3.1 INTRODUCTION

The visibility bus (VBus) is a slow speed bus consisting of 16 data lines and two control lines that allow the PRO-38N console to read selected logic levels in the CPU modules. It is used mainly during the execution of microdiagnostics and during system initialization. Normally, the VBus is used when the system clocks are stopped.

The PRO-38N controls and reads the VBus by means of two registers located on the clock module's console interface. These registers, the VBus control register and the VBus access register (actually the outputs of a VBus data multiplexer), are accessed by the PRO-38N over the RTI. Both the VBus and the VBus control on the console interface are shown in Figure 3-1.

3.2 BASIC FUNCTIONS

The VBus allows the PRO-38N console to perform the following major functions:

1. Monitor the state of the CPU(s) during the execution of microdiagnostics or in response to commands entered at the console during system debug. CPU signals are usually examined in the interval between single-stepped clocks or clock bursts.

2. Verify during system initialization (with the clock stopped) that the CPU modules are installed correctly. Module revision numbers are also read at this time.

3. Check that no control store parity errors occur when microcode is loaded during system initialization. In this case, VBus data bits (parity error flags) are read with the system clock running at full speed.

3.3 VBUS SIGNALS

The VBus signals, which are ECL driven and received, are defined in Table 3-1.

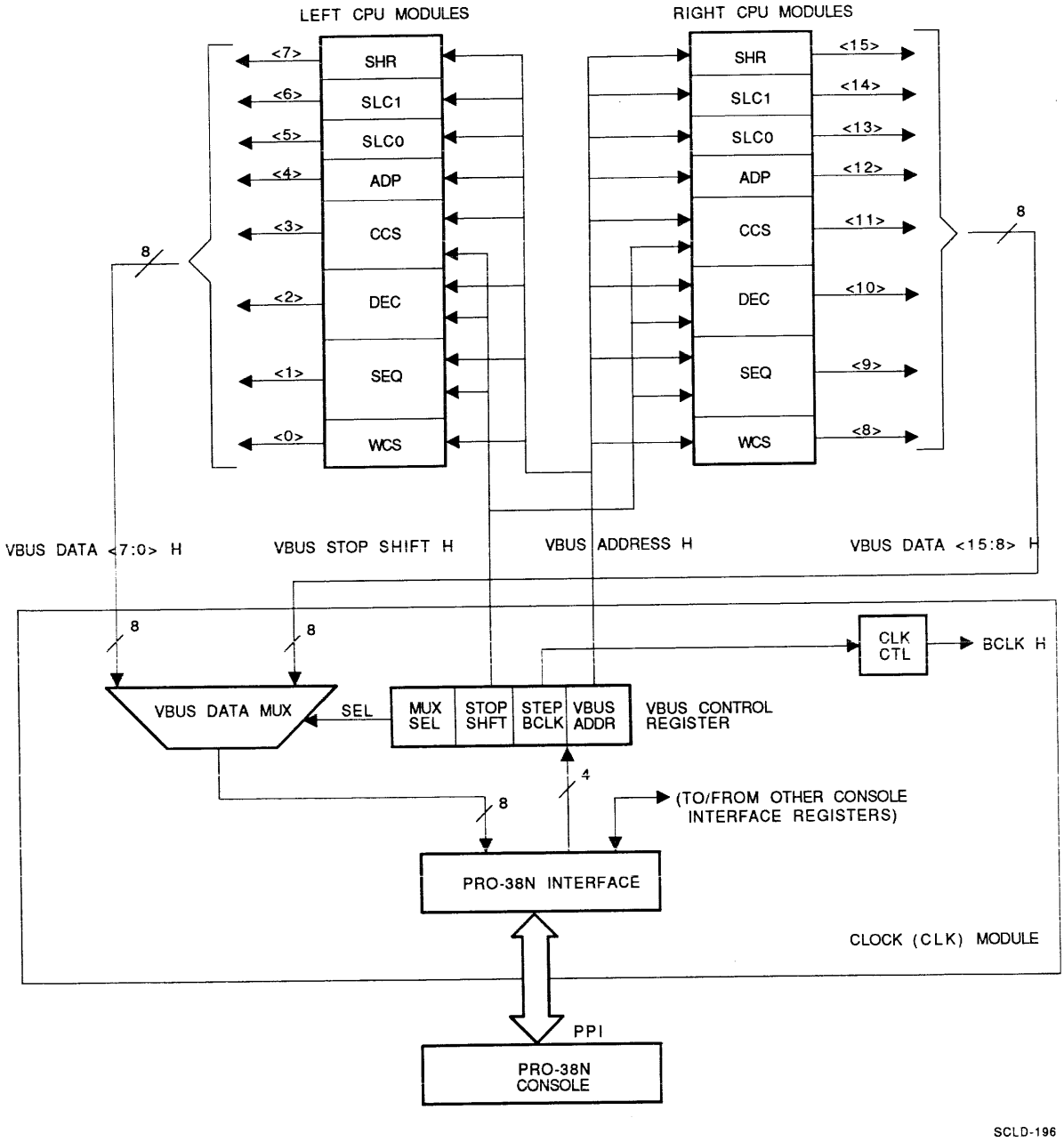


Figure 3-1 Visibility Bus (VBus) and VBus Control (on CLK Module)

Table 3-1 VBUS Signal Descriptions

Signal Line	Number	Description
VBUS DATA <15:0> H	16	Transfer VBus data from CPU modules to console interface on clock (CLK) module. Each data line connects to one CPU module and corresponds to one VBus channel.

VBus DATA	Left/Right CPU	CPU Module
<15>	R	SHR
<14>	R	SLC1
<13>	R	SLC0
<12>	R	ADP
<11>	R	CCS
<10>	R	DEC
<9>	R	SEQ
<8>	R	WCS
<7>	L	SHR
<6>	L	SLC1
<5>	L	SLC0
<4>	L	ADP
<3>	L	CCS
<2>	L	DEC
<1>	L	SEQ
<0>	L	WCS

VBUS ADDRESS H	1	Transfer VBus data address from console interface to all CPU modules. Address is shifted into VBus address registers (shift registers) on each module one bit at a time.
VBUS STOP SHIFT H	1	Asserted by console interface to disable the shifting of VBus address registers in some CPU modules (CCS, DEC, and SEQ modules). The registers hold their current contents (the current VBus data address) as long as the line is asserted.

The 16 data lines allow 16 separate visibility channels to be read by the console. That is, a data line connects from each of the 16 CPU modules to the VBus data multiplexer on the console interface. (There are eight modules in the left CPU and eight in the right CPU.) Eight VBus channels, either the eight left CPU channels or the eight right CPU channels, can be read by the console at a time.

The two VBus control lines connect to more than one module. One, the VBus ADDRESS line, connects to all modules. The console reads VBus data from a module by first shifting an address onto this line one bit at a time. The address is held in the module (in a shift register) and it selects the single VBus data bit that is transmitted on the module's VBus data line. Because the address line connects to all modules, a VBus data bit may be selected in more than one module by a single address. As mentioned above, the VBus data bits from all modules can be read eight bits at a time once an address is loaded.

The second VBus control line, VBus STOP SHIFT, connects to only the CCS, DEC, and SEQ modules in each CPU. When asserted by the console, this signal causes the shift registers holding the VBus data address bits in the modules to hold that address even if the system clocks are started. The selected VBus data bits may then be examined during normal operation when clocks are running at full speed. This is done during the loading of microcode when the VBus is used to monitor control store parity error flags. (These parity error flags in the CCS, DEC, and SEQ modules are all selected by the same VBus data address bits: 100110, where the right-most or least significant bit is the last bit shifted into the VBus address path.)

VBus STOP SHIFT can also be asserted during system debug so that selected VBus signals may be scoped on the backpanel while the system clock is running.

3.4 VBUS REGISTERS

Register bit formats for the two VBus registers on the console interface are shown in Figures 3-2 and 3-3. The VBus control register is a write-only register. The VBus access register, which is the output of the VBus data multiplexer, is a read-only register.

The bits in the VBus control register are defined in Table 3-2.

The VBus ADDRESS and STOP SHIFT control bits (latch outputs) drive the two VBus control lines. These control bits hold their current state unless changed when the VBus control register is written again.

The STEP BCLK control bit causes one system BCLK to be generated. It is set only momentarily (for one clock cycle) and must be set every time a BCLK is to be generated. It is used during VBus operations to clock the VBus data address into the modules. (The shift registers that hold the VBus data address in the modules are clocked by BCLK). When the console loads a VBus data address bit, it first loads the bit in the VBus ADDRESS latch, which transmits it on the VBus ADDRESS line. The console then shifts the address bit into the module's shift register by setting STEP BCLK. This is repeated until all bits of the address are shifted into the modules.

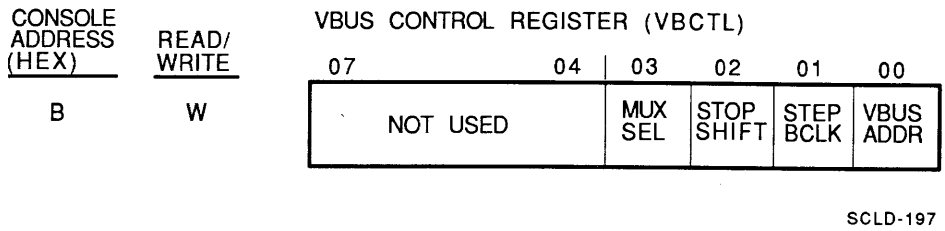
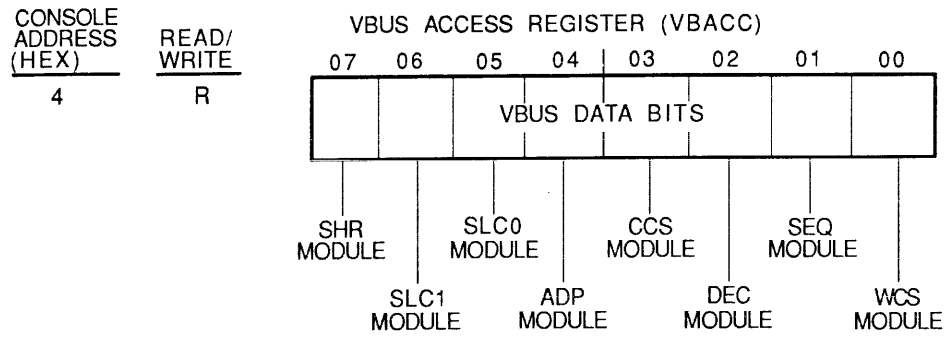


Figure 3-2 VBus Control Register

Table 3-2 VBus Control Register Bit Descriptions

Bit(s)	Description
<7:4>	Not used.
<3>	Multiplexer select. Selects which VBus channels are read by console when it reads the VBus access register. Cleared by CPU INIT. MUX SEL — SELECTED VBUS CHANNELS 0 — Eight channels from left CPU 1 — Eight channels from right CPU
<2>	Stop shift. Prevents BCLK from shifting registers that hold VBus address in CCS, DEC, and SEQ modules. Used to freeze address of control store parity error flags in these modules when microcode is loaded. Cleared by CPU INIT.
<1>	Step BCLK. Causes one system BCLK to be generated. Set for only one clock cycle.
<0>	VBus address. This address bit is shifted into serial VBus address path (in each module) by BCLK. Cleared by CPU INIT.



NOTE: VBUS DATA BITS ARE FROM LEFT CPU IF BIT<3> IN VBUS CONTROL REGISTER IS CLEARED. IF BIT<3> IS SET, VBUS DATA BITS ARE FROM RIGHT CPU MODULES.

SCLD-198

Figure 3-3 VBus Access Register

The MUX SELECT control bit (another latch output) simply drives the select line for the VBus data multiplexer and determines which eight VBus channels are read by the console when it reads the VBus access register.

The console uses the VBus registers as follows during a typical read of one or more VBus channels (system clock turned off).

1. Loads VBus ADDRESS bit and STOP SHIFT = 0

2. Loads same VBus ADDRESS bit, STOP SHIFT = 0, and STEP BCLK = 1 (to shift address bit into modules)

3. Repeats steps 1 and 2 (until all address bits are shifted into modules)

4. Loads MUX SELECT bit and reads VBus data bits

Note that a copy of the VBus ADDRESS bit is loaded when the BCLK is stepped. This is because the console is not allowed to change the state of the VBus ADDRESS bit when STEP BCLK is set. It also cannot change the state of the STOP SHIFT bit when STEP BCLK is set. These programming restrictions are necessary due to the electrical properties of the VBus.

3.5 MODULE VBUS CHANNEL CIRCUITRY

The VBus channel circuitry in a CPU module is basically the following:

1. One or more shift registers (clocked by BCLK) that hold the VBus data address shifted into the module by the console. The same number of address bits are not used by all modules.

2. One or more data multiplexers that select the VBus data bit specified by the address. The bit is transmitted on the module's single VBus data line.

3.5.1 Minimum Configuration

The minimum amount of VBus channel circuitry required for a CPU module is shown in Figure 3-4. That is, all CPU modules must allow the console to read the module's revision number and also an "eat the tail" bit to test VBus channel operation. The console reads these VBus data bits during system initialization. Modules having the minimum amount of VBus circuitry are the SHR, SLC0, SLC1, and WCS modules. (In the WCS module, two control store parity error flags can be read in addition to the revision number and tail bit.)

The tail bit is a bit in the serial address stream that is not used for selection purposes but, when shifted out of a module's shift register and followed by its own VBus data address bits, may be read by the console. When testing the VBus channel, the console shifts both a 1 and a 0 tail bit through the shift register and reads each back to verify that the VBus channel is operating correctly. A successful test also verifies that a module is installed in the slot. Furthermore, because the tail bit address is different for each module, the test verifies that the correct module is installed in the slot.

3.5.2 Expanded Configuration

Some CPU modules allow a large number of internal logic levels to be examined using the VBus. Thus, additional shift registers and data multiplexers are required in the module and these, for the most part, are located within the MCAs. A typical expanded configuration is shown in Figure 3-5. Modules having an expanded VBus capability are the ADP, CCS, DEC, and SEQ modules.

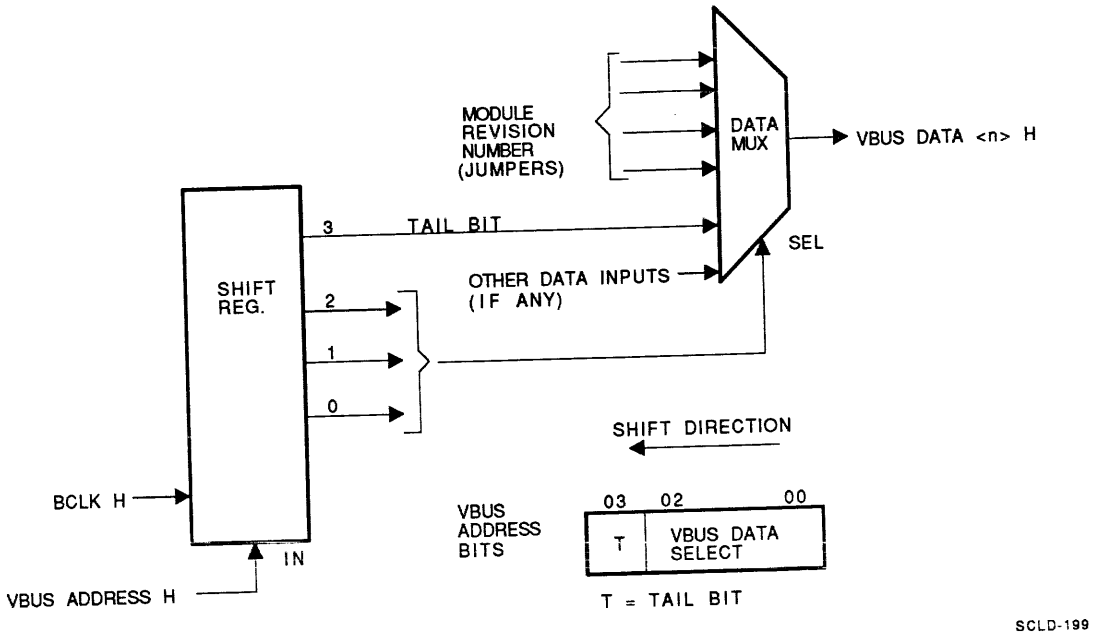


Figure 3-4 VBus Channel in CPU Module (Minimum Configuration)

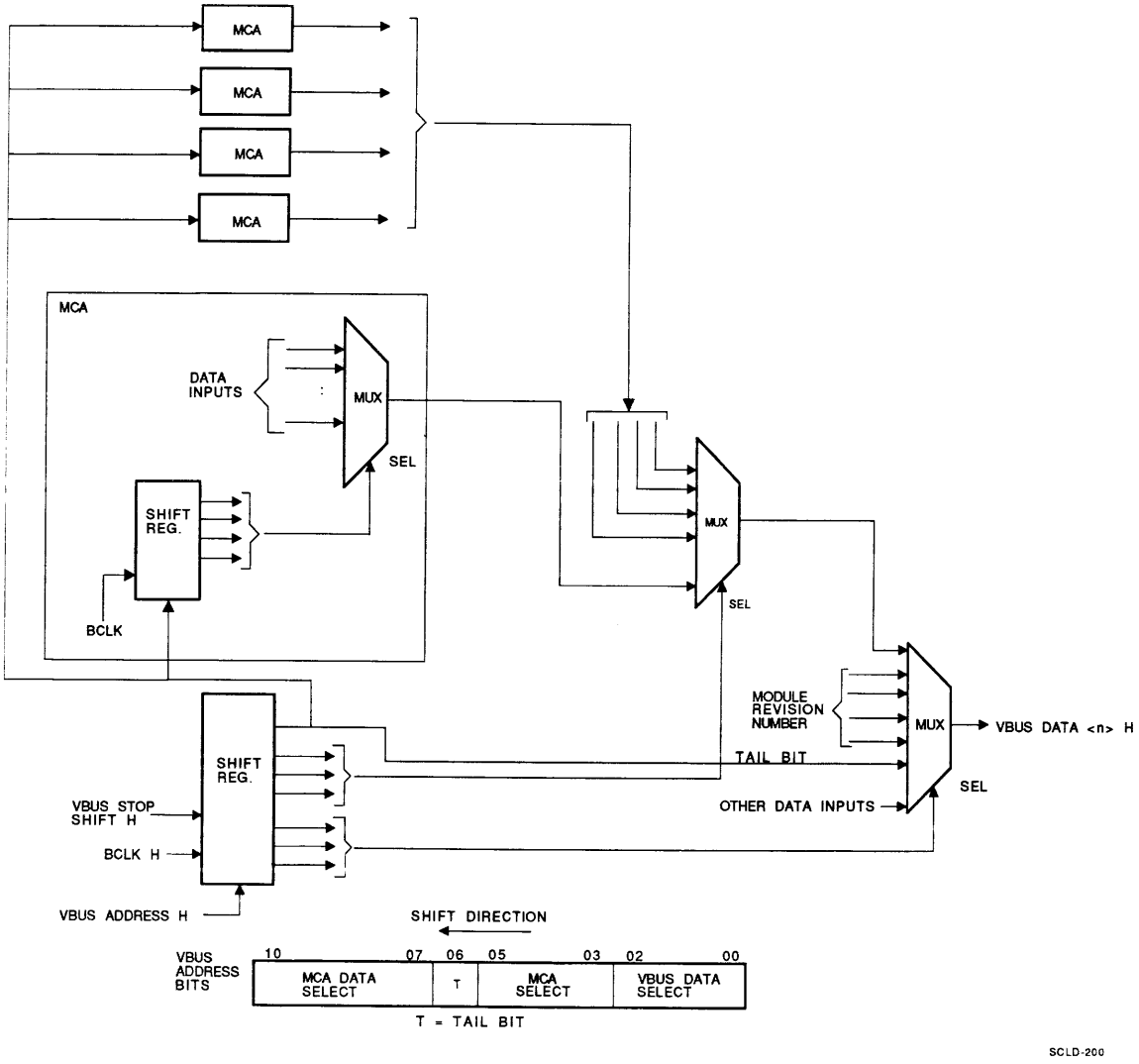


Figure 3-5 VBus Channel in CPU Module (Expanded Configuration)

3.6 VBUS ADDRESS/DATA SUMMARY

The VBus directory in the System Maintenance Guide lists the logic levels in each module that can be read using the VBus. (An excerpt from the beginning of the directory is given in Table 3-3.) The minimum number of VBus data address bits required for each module (each channel) varies from three bits to ten bits (not including the tail bit).

3.7 VBUS CONSOLE COMMANDS

There are three VBus console commands:

1. PROBE VBUS <address> <accumulator bit>
2. SHOW ACCUMULATOR
3. CLEAR ACCUMULATOR

The PROBE VBUS command selects a single VBus data bit and loads it into (actually ORs it with) one of the bits in a 32-bit accumulator. The use of an accumulator allows a string of these commands to assemble several bits of VBus data into a useful format before the data is finally examined with the SHOW ACCUMULATOR command. All bits in the 32-bit accumulator can be cleared at any time with the CLEAR ACCUMULATOR command.

The address for the PROBE VBUS command consists of a channel number and a VBus data address. This is followed by the bit number in the 32-bit accumulator. The command causes the console to shift the VBus data address out to all modules, read the selected VBus data (eight channels), and then load the data bit for the specified channel into the specified bit position in the accumulator.

Use of the VBus commands is illustrated below. The example is a command file that reads the address of the first microword in the pipeline. The address bits, stored in the microPC silo on the SEQ module, are assembled in bits <13:00> of the accumulator.

```
!MICROPC.CMD
!assembles microPC bits from VBus and displays them
CLEAR ACCUMULATOR
PROBE VBus %X1007 00            !read bit 0
PROBE VBus %X100F 01            !read bit 1
PROBE VBus %X1017 02            !read bit 2
        :                            :
        :                            :
PROBE VBus %X1117 12            !read bit 12
PROBE VBus %X111F 13            !read bit 13
SHOW ACCUMULATOR
```

Table 3-3 VBus Directory (Excerpt)

Channel	Module	Module Type
0	WCS	F1009
1	SEQ	F1008
2	DEC	F1007
3	CCS	F1006
4	ADP	F1005
5	SLC0	F1004
6	SLC1	F1003
7	SHR	F1002

Channel	PROBE VBus Address (Hex)	VBus Data Address (Binary)*	Signal	Remarks
0	0000	0 000	TAIL BIT H = 0	Tail bit should be 0
0	0008	1 000	TAIL BIT H = 1	Tail bit should be 1
0	0002	010	INT CS1 PE H	CS1 Parity Error
0	0003	011	INT CS2 PE H	CS2 Parity Error
0	0004	100	MODULE REVISION <0> H	
0	0005	101	MODULE REVISION <1> H	
0	0006	110	MODULE REVISION <2> H	
0	0007	111	MODULE REVISION <3> H	
1	1000	000	INT CS0 PE H	CS0 Parity Error
1	1001	0 XXX 001	TAIL BIT H = 0	Tail bit should be 0
1	1041	1 XXX 001	TAIL BIT H = 1	Tail bit should be 1
1	1002	010	MODULE REVISION <0> H	
1	1003	011	MODULE REVISION <1> H	
1	1004	100	MODULE REVISION <2> H	
1	1005	101	MODULE REVISION <3> H	
1	1006	110	CS PARITY ERROR H	CS0, CS1, or CS2 PE
1	1007	00 00X 000 111	DIG UADDR 3 <0> H	Address of first microword in pipeline (address in uPC silo)
1	100F	00 00X 001 111	DIG UADDR 3 <1> H	
1	1017	00 00X 010 111	DIG UADDR 3 <2> H	
1	101F	00 00X 011 111	DIG UADDR 3 <3> H	
1	1027	00 00X 100 111	DIG UADDR 3 <4> H	
1	1087	00 01X 000 111	DIG UADDR 3 <5> H	
1	108F	00 01X 001 111	DIG UADDR 3 <6> H	
1	1097	00 01X 010 111	DIG UADDR 3 <7> H	
1	109F	00 01X 011 111	DIG UADDR 3 <8> H	
1	10A7	00 01X 100 111	DIG UADDR 3 <9> H	
1	1107	00 10X 000 111	DIG UADDR 3 <10> H	
1	110F	00 10X 001 111	DIG UADDR 3 <11> H	
1	1117	00 10X 010 111	DIG UADDR 3 <12> H	
1	111F	00 10X 011 111	DIG UADDR 3 <13> H	

* The right-most bit of the VBms data address (binary) is the least significant bit and the last bit shifted into the serial address path.

SECTION 3
CONSOLE SUBSYSTEM

CHAPTER 1
INTRODUCTION

1.1 GENERAL

This section provides a technical description of the VAX 8800 console subsystem. The section is divided into three chapters containing the following information:

Chapter 1 -- An introduction and overview of the console subsystem at the simplified block diagram level.

Chapter 2 -- A functional description of the console and how it interacts with the VAX 8800 system.

Chapter 3 -- Detailed analysis of the console interface and a description of the input/output ports, timing signals, MCA logic, and register definitions.

1.2 RELATED DOCUMENTATION AND REFERENCES

This technical description does not contain detailed information pertaining to the Professional Series 380 computer. The following references have been provided for operation, installation, and maintenance of the PRO-38N:

Professional 300 Series Owner's Manual (AA-N587A-TH)

Professional 300 Installation Guide (AZ-N626A-TH)

Professional 300 User's Guide for Hard Disk System (AA-N603A-TH)

Professional 300 Communications Manual (AA-N602B-TH)

Professional 300 Pocket Service Card (EK-PC3XX-PC)

Professional 300 Technical Manual (EK-PC3XX-TM

Professional 300 Field Maintenance Print Set (MP-01394-00)

VR210 Field Maintenance Print Set (MP-01410-00)

LK201 Field Maintenance Print Set (MP-01395-00)

KEF11 Field Maintenance Print Set (MP001473-00)

VAX 8800 System Hardware User's Guide (EK-8800H-UG-001)

VAX 8800 System Installation Guide (EK-8800I-IN-001)

1.3 FUNCTION AND PURPOSE

The console subsystem is an independent computer system that provides primary control of the VAX 8800 system. The software of the console subsystem controls power switching, loading of microcode, and the general operation of a dual-CPU VAX 8800 system.

1.4 SUBSYSTEM COMPONENTS

The console subsystem shown in Figure 1-1 uses a Professional Series 380 computer (PRO-38N) as a console device. The PRO-38N is a self-contained system that consists of a Winchester disk drive, dual-floppy disks, display monitor, and terminal keyboard.

A real-time interface (RTI) I/O module located in slot six of the PRO-38N, provides the communication path between the console and the VAX 8800 CPU(s). All of the console/VAX 8800 communications are transmitted and received through the RTI. The RTI uses a programmable peripheral interface (PPI) and a serial line unit (SLU) as I/O devices for communication.

The PPI communicates by means of an 8-bit bus to the VAX 8800 cabinet, and handles the transfer of read/write data, system status, and control signals. The SLU is the console's interface to the VAX 8800 power subsystem. Control and status signals between the console and a microprocessor in the environmental monitoring module (EMM) provide the console with the ability to monitor specific parameters in the VAX 8800 cabinet such as voltage, temperature, and airflow.

The PPI bus is connected to a console interface residing on the VAX 8800 clock module. The clock module's connection to the VAX 8800 backplane provides the console with a point of communication with the VAX 8800 CPUs.

An optional printer can be connected to the console subsystem through a serial printer port on the rear of the PRO-38N. A remote diagnostic link can be established through an existing port on the PRO-38N that is set up for modem control.

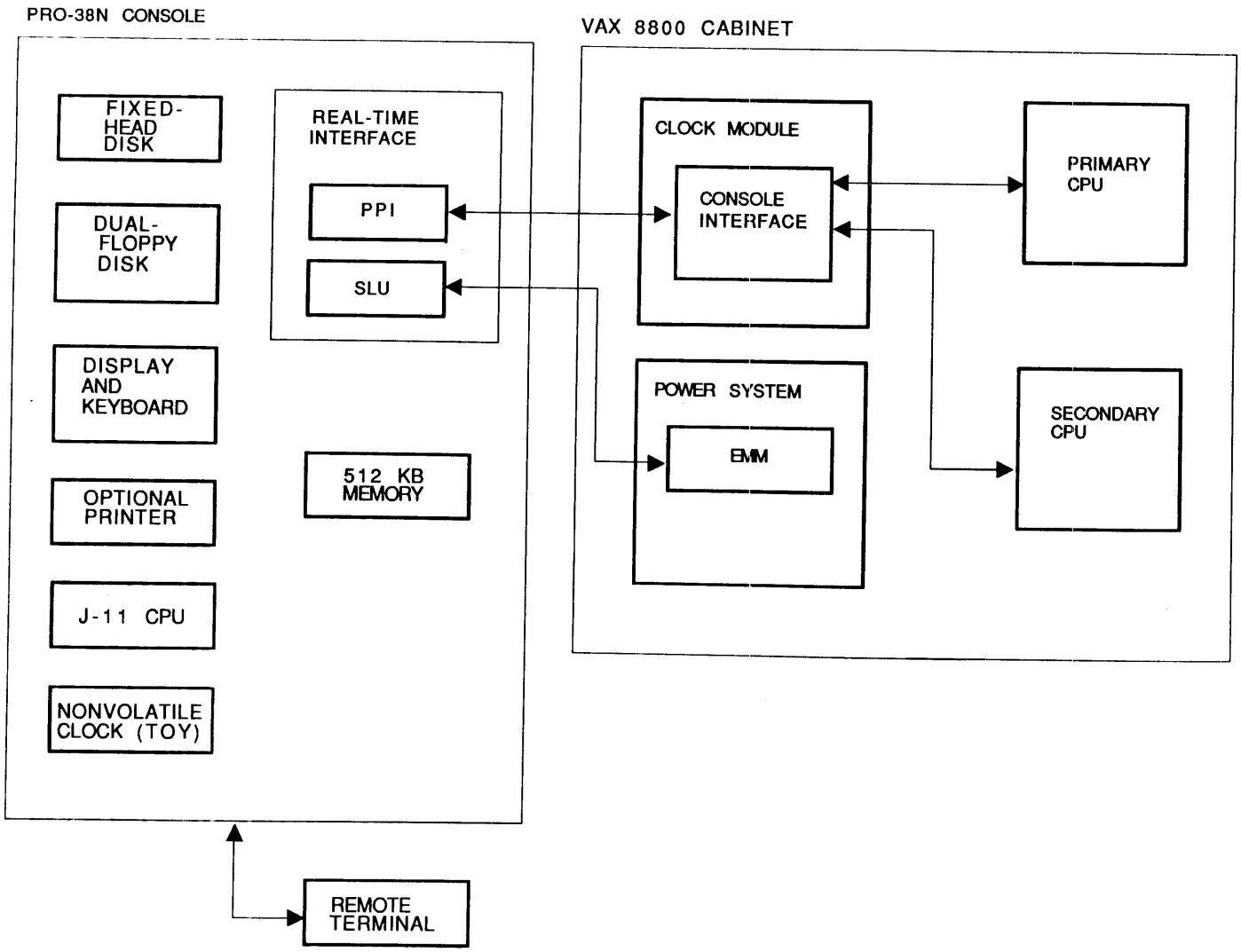


Figure 1-1 Simplified Block Diagram of the Console Subsystem

1.5 CONSOLE/VAX 8800 INTERACTION

The console operates in two modes as shown in Figure 1-2. The power-up and boot mode is included in the discussion of operating modes, because of the important role that the console has in the power-up sequence.

1.5.1 Power-Up Mode

During the power-up mode of operation, the console is responsible for the sequencing and control of the following VAX 8800 functions:

- Power application/sequencing
- Monitoring environmental status
- Performing loopback testing of the interface
- Verifying correct module placement
- Performing revision compatibility comparisons
- Providing system identification
- Loading VAX 8800 firmware and booting the system

1.5.2 Console I/O

The console mode can be in effect during either the VAX 8800 halt or run mode. During halt mode, the console can perform, examine, and deposit, microcode/macrocode stepping and clock control functions. The VAX software must be halted during these functions to allow the console support microcode (CSM) to communicate with the console. Commands not requiring microcode assistance are valid in either mode.

Placing the console in console mode during the VAX 8800 CPU run mode with a CTRL/P limits the amount of VAX 8800 commands available to the operator, but allows the console to run independently of the VAX.

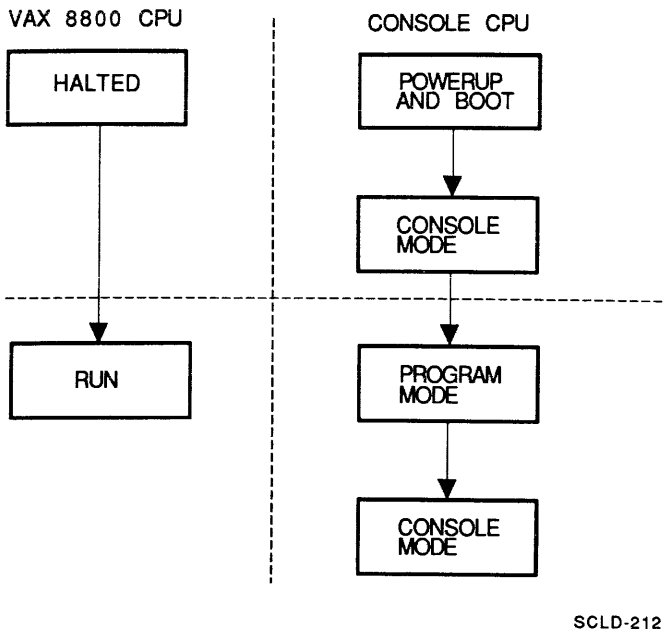


Figure 1-2 Modes of Operation

Program mode places the console in a state similar to any VAX terminal. Communication with the VAX 8800 CPU is under interrupt control using the ready and done conditions. During program mode, the console is free to interact with the VAX 8800 at the instruction set processor (ISP) level and perform normal operator interactive functions.

1.5.3 VAX 8800 State Description

The VAX 8800 CPU exists in one of the following four hardware/operational states:

1. Power off
2. Clock stopped/WCS invalid
3. Clock running/WCS invalid
4. Clock running/WCS valid

1.5.3.1 Power Off -- All regulators turned off.

1.5.3.2 Clock Stopped/WCS Invalid -- The VAX 8800 regulators have been turned on and the clock is stopped while the console checks module and software revisions. Microcode has not been loaded, and volatile RAM locations remain invalid. The clocks are then started to load microcode.

1.5.3.3 Clock Running/WCS Invalid -- There are two conditions in which this state can be true: during powerup and loading of the WCS. The clock is running during the power-up sequence and all RAM areas are volatile and invalid. During WCS and IBox loading, the clock is running.

1.5.3.4 Clock Running/WCS Valid -- Microcode has been loaded into the WCS; the IBox decoder RAMs and cache control store are loaded. The VAX system is executing a NOP microinstruction and the VAX 8800 CPU is in the ready state.

1.5.4 Console State Description

The console CPU exists in one of the following operational states:

Power Off

The power switch on the PRO-38N is in the OFF position.

Stuck

Console selftests have failed. Power-up and boot sequence will not proceed.

P/OS

The console is running the PRO operating system. This condition can be the result of an intentional exit from the VAX 8800 system to use P/OS, or the result of a console-only power failure.

> **NOTE**
>
> **The P/OS state will exist when the system has been assembled and the console applications have not been installed.**

Console Applications

Console applications have begun. The console is initializing the data base and spawning the RTI driver. Testing of the real-time interface, environmental monitoring module, and VAX 8800 hardware have not been performed. Commands for testing and interrogation are valid during this state.

Test

EMM, RTI, and clock module loopback testing have been completed successfully. Commands to the EMM to begin VAX 8800 powerup are valid.

Power

VAX 8800 CPU power has been applied and the clock is NOT running. Commands to start the clock and load microcode, IBox decoder RAMs, and cache control store are valid. The POWERDOWN command is valid.

Micromonitor Mode

Micromonitor is a subset of the console program that allows loading, control, and monitoring of console-based and WCS-based microdiagnostics.

The commands that are valid include the conventional console commands plus a set of micromonitor commands.

Console Mode

The VAX 8800 ISP with console support microcode has been loaded. Commands that are allowable when the clock is running are valid.

Microbreak

Clock stopped, microcode execution halted. This state is a result of one of the following:

- MicroPC match and stop on micromatch set
- Clock burst of (n) cycles complete
- SET CLOCK OFF command

Intermediate

This state exists during the transition from console mode to program mode. Characters are not passed from the terminal and a limited amount of commands are valid.

Program Mode

Normal mode of operation for the VAX 8800 run mode.

Boot

The console is in the process of booting the VAX 8800 system.

Restart

The console is in the process of restarting the VAX 8800 system.

1.6 CONSOLE SOFTWARE COMPONENTS

The console software consists of the following three distinct processes to implement control of the VAX 8800 system:

1. Control program
2. Logical block server program
3. Real-time interface driver

1.6.1 Control Program

The control program is the main console program and is responsible for implementing console modes (program or console I/O), logfiles, micromonitor, EMM support, and remote terminal access. The control program uses RSX system directives SET, CLEAR, TIMED REQUESTS, WAIT EVENT FLAGS, and QUEUED I/O WAIT.

During system initialization, the control software installs and initializes the RTI driver, spawns the logical block server and disk access programs, reads nonvolatile console state from the console disk drive, and initializes the console database.

1.6.2 Logical Block Server Program

The logical block server program enables the VAX 8800 CPU to read or write files to and from the console floppy disks. The console presents the floppies to the VAX 8800 as large virtual floppies by creating a file on the Winchester drive.

1.6.3 Real-Time Interface Driver

The real-time interface driver provides the software link between the RTI circuitry in the PRO-38N, and the hardware independent functions of the control, logical block server, and file transfer programs. Separate subdrivers are used for communications with either the programmable peripheral interface, or the serial line unit. The PPI subdriver provides mapping from high-level functions, to machine-specific operations (for example: Examine or Deposit requests require a correct sequencing of reads, writes, and acknowledge operations).

1.7 CONSOLE/VAX 8800 POWER SEQUENCE

The description of the console and VAX 8800 power sequencing is divided into five functional areas:

1. Powerup
2. EMM/Console initialize
3. Restart/Boot/Halt
4. Power fail
5. Powerdown

Each of the five functional areas is illustrated with a flowchart in Figures 1-3 through 1-7. The flowcharts are intended as a reference for the brief overview of the VAX 8800 power sequencing, and do not contain sufficient information for a detailed analysis. Refer to Section 10 of the manual for information about the power subsystem.

1.7.1 Powerup (Refer to Figure 1-3)

The power-up sequence involves two independent processors to bring the VAX 8800 system to ready state. The microprocessor in the environmental monitoring module, and the console processor perform numerous cooperative tasks that ensure the correct sequencing of power and initialization of the system.

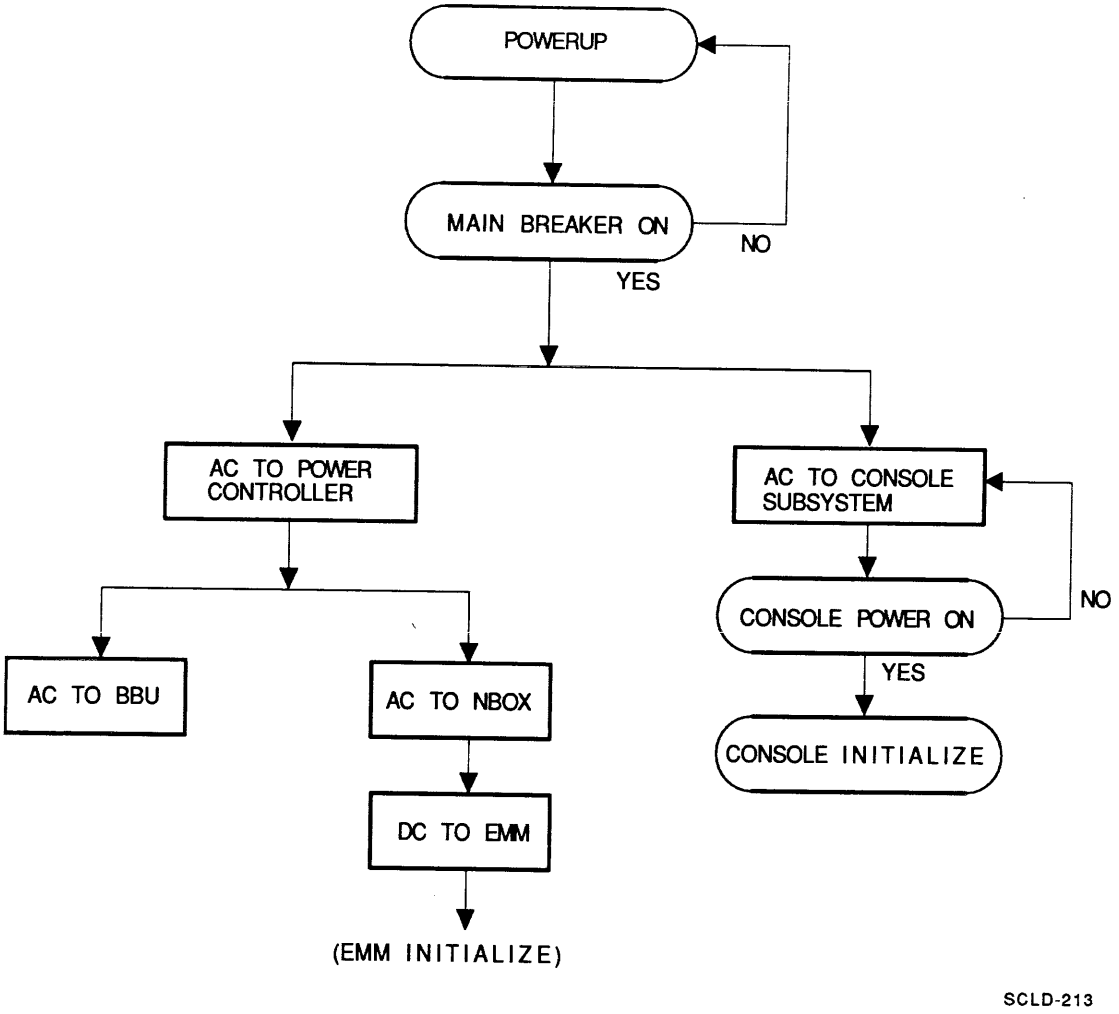


Figure 1-3 Power-Up Sequence

When the main power circuit breaker on the VAX 8800 cabinet is turned to the ON position, ac voltage is applied to the following:

- 876 power controller
- Console subsystem

The 876 power controller applies ac voltage to the NBox (port conditioner) and the battery backup unit (BBU). The NBox provides a 12-volt and 5-volt dc to the EMM, enabling the EMM to begin the power-on sequence.

1.7.2 EMM/Console Initialize (Refer to Figure 1-4)

The dc voltages applied to the EMM reset the microprocessor and begin the initialize sequence by performing a module keying test. If the keying test is satisfactory, the EMM conducts a selftest. Failure of either the keying test or the selftest will result in an error message to the console and a reset of the EMM. The reset will prevent the EMM from responding to console commands. Upon completion of the selftest, the EMM will load its own default monitoring parameters, initiate auto shutdown monitoring, and wait for the console to be switched "ON".

When the console operator places the PRO-38N power switch to the "1" position, the console processor performs a selftest of the PRO-38N's major components. Satisfactory completion of the selftest results in the loading and running of the console software. The console software initializes the database and the RTI driver so that it can communicate with the EMM in the VAX 8800 system.

As soon as the console software verifies that it can communicate with the EMM, the console sends specific monitoring parameters to the EMM and enables failure monitoring. The specific parameters replace the default parameters loaded by the EMM during the powerup without the console.

The console software checks the INITIALIZE IN PROGRESS flag to determine if a previous powerup attempt had failed prior to completion. If the flag indicates that a previous attempt had been aborted, the console is set to the console I/O mode and the power sequence is aborted. Power supply status is checked to determine if it was a console-only power failure. If a console-only power failure had occurred, the console clears the CONSOLE GONE flag, and sets either console or program mode depending on the state when the console failed.

The console executes SYSINIT.COM and the command file checks AUTO POWER ON to determine if the console should continue with the automatic power sequence or abort and wait for further instructions.

The console performs a series of loopback tests to determine that all required communication paths with the VAX 8800 system are operational. Successful completion of the loopback tests allows the console to verify correct module placement and revision status.

When the console is satisfied that all modules are correctly placed, and the revisions are compatible, the console loads the WCS, cache control store, and IBox decoder RAMs. IPRs and memory are initialized and an UNJAM is performed to initialize I/O.

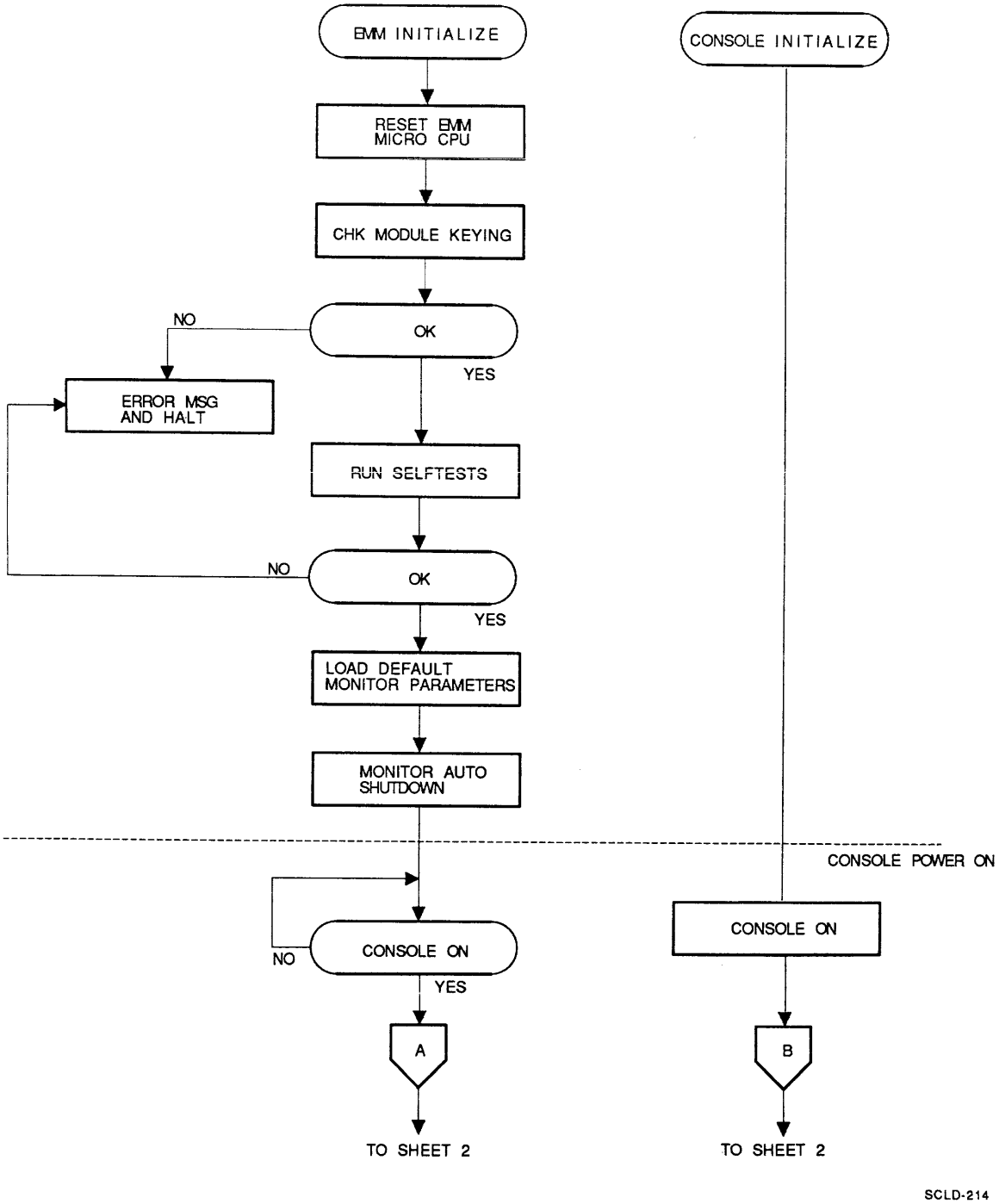


Figure 1-4 EMM/Console Initialize (Sheet 1 of 4)

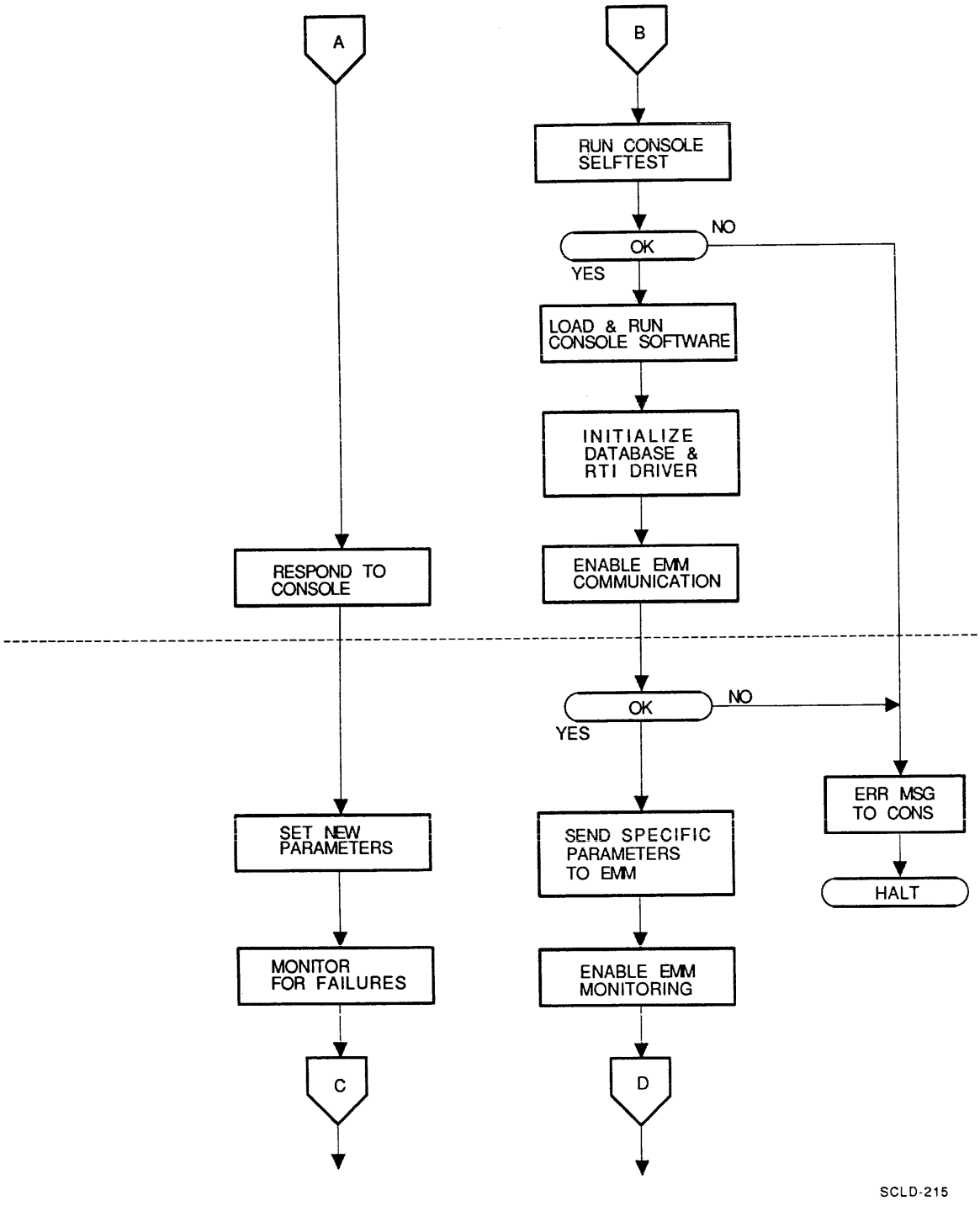
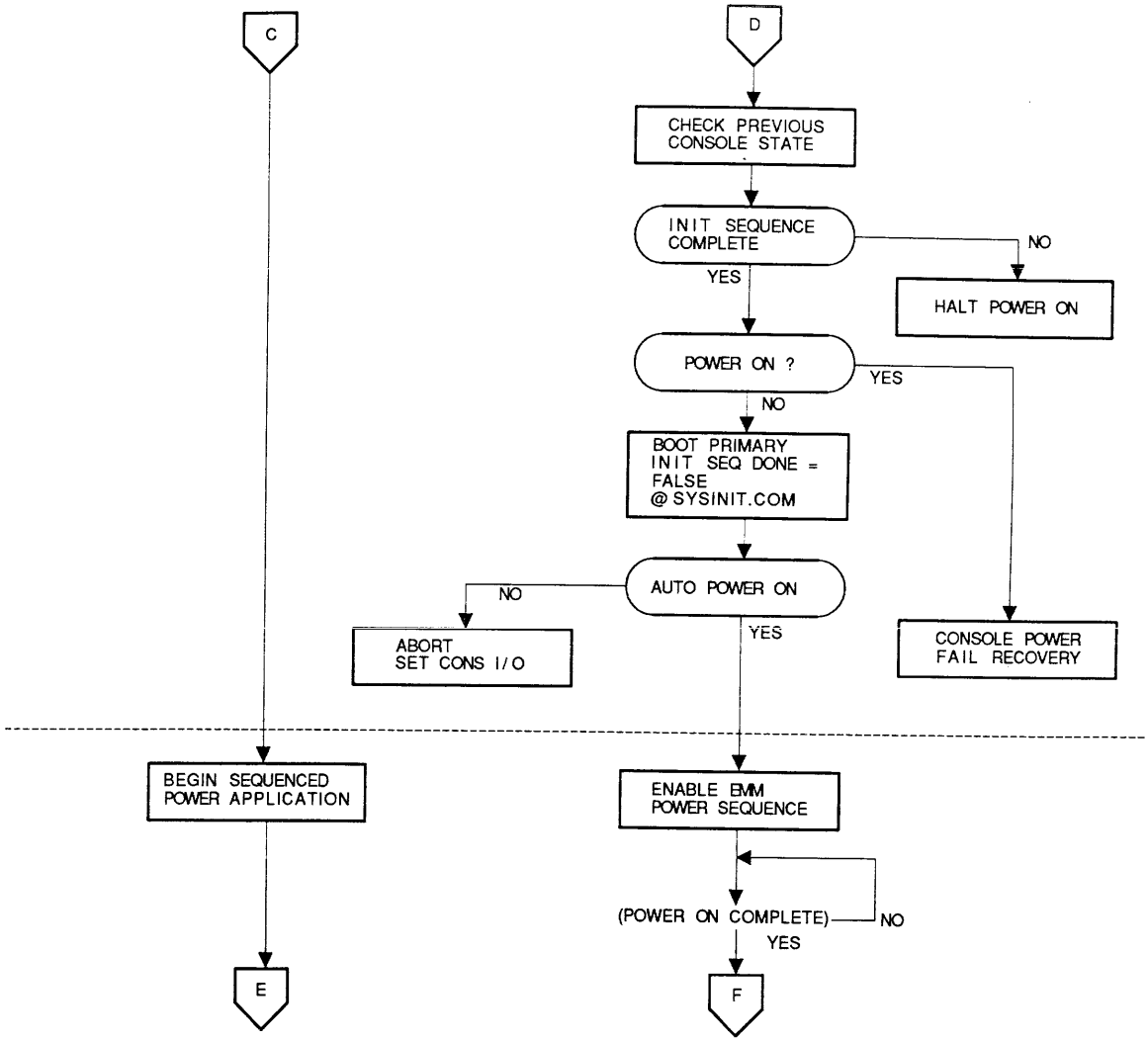


Figure 1-4 EMM/Console Initialize (Sheet 2 of 4)



SCLD-216

Figure 1-4 EMM/Console Initialize (Sheet 3 of 4)

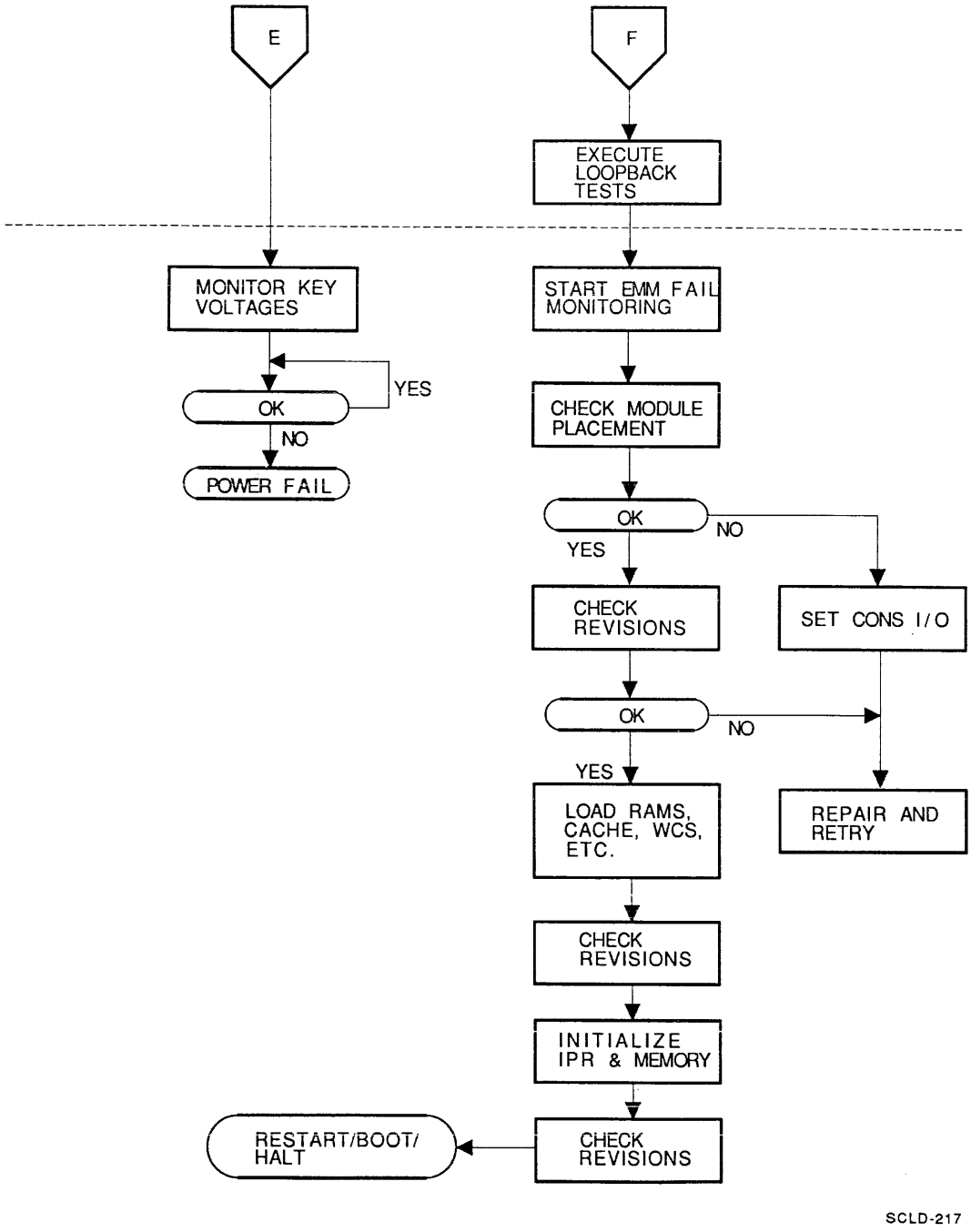


Figure 1-4 EMM/Console Initialize (Sheet 4 of 4)

1.7.3 Restart/Boot/Halt (Refer to Figure 1-5)

When the power-on sequence is completed, the VAX 8800 system is ready to enter either the restart, boot, or halt mode, depending on the position of the software mode keyswitch. The flowchart, shown in Figure 1-5, only refers to the primary VAX 8800 CPU. The secondary CPU is restarted using an RPB saved in memory by the primary CPU. The secondary CPU is restarted by the console when requested by the primary CPU.

If the software keyswitch indicates boot, the console uses the console support microcode to locate 64 Kbytes of contiguous memory, and loads and executes VMB. A successful boot results in a continue to program I/O mode.

The FIND MEMORY command requests the microcode to search main memory, starting at address 0, for a page-aligned 64-Kb block of good memory. If a block is found, the starting address plus 512 is left in the SP and the cold restart flag is set.

The FIND RPB command initiates a microcode search of physical memory for a valid restart parameter block. If an RPB is found, the warm restart flag is set. A warm restart states that memory has been backed up and a restart parameter block is available.

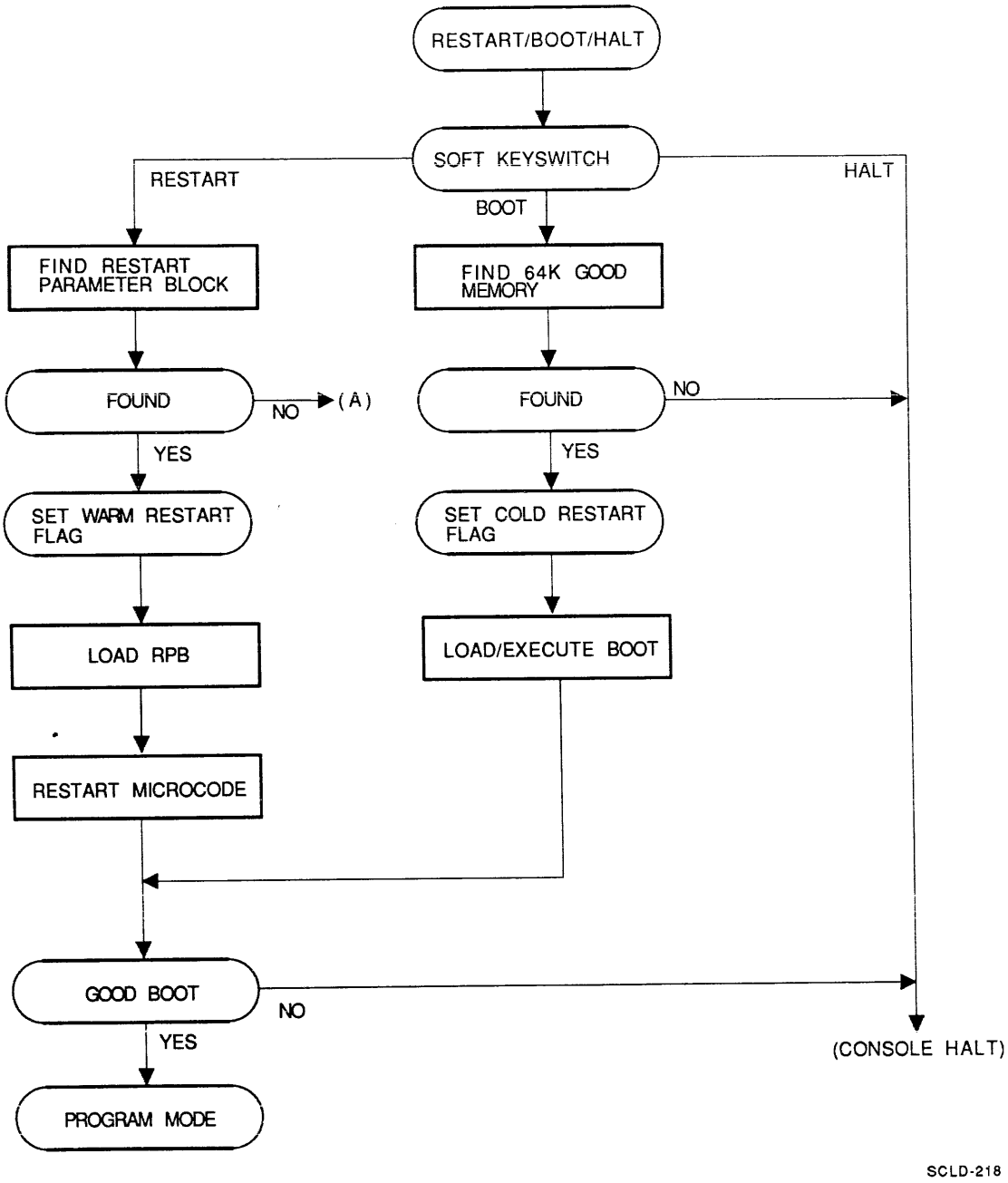


Figure 1-5 Restart/Boot/Halt

1.7.4 Power Fail (Refer to Figure 1-6)

The environmental monitoring module is responsible for recognizing a pending power fail and informing the VAX 8800 CPU and the console. When the EMM senses a decrease in a monitored voltage, it asserts a signal called ACLO. The ACLO signal generates a power-fail interrupt to the VAX by means of the console interface and informs the console that a sensed voltage has fallen below acceptable limits.

At least five milliseconds after the ACLO signal, the EMM asserts the DCLO signal. DCLO asserts the CPU INIT signal to the primary and secondary CPUs and disables microcode execution. If the console subsystem power is still active, the console recognizes the interrupt and executes the POWER OFF command

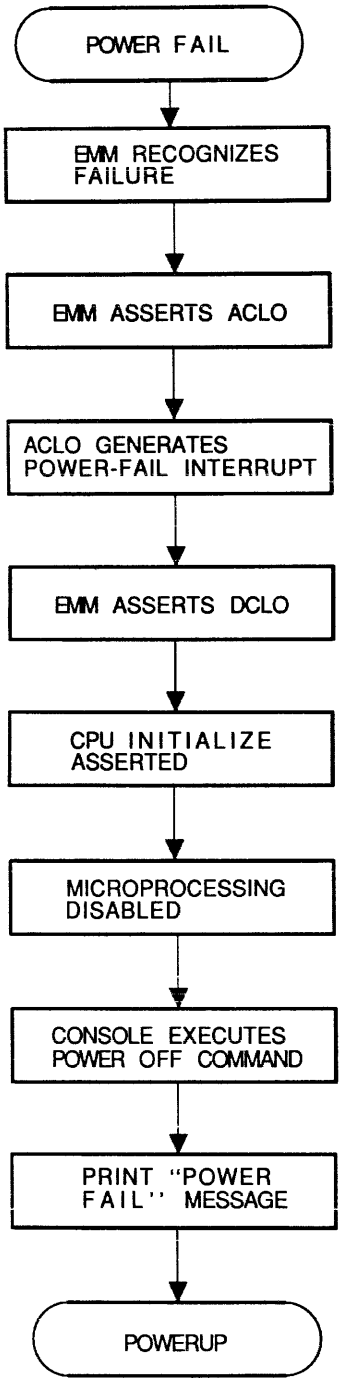


Figure 1-6 Power-Fail Sequence

1.7.5 Powerdown (Refer to Figure 1-7)

An intentional powerdown is initiated by a console command. When the command is recognized, the console software checks the status of the battery backup unit.

If the BBU is enabled or the operator wishes to continue without backup power, the EMM is directed by the console to begin sequential shutdown of the voltage regulators.

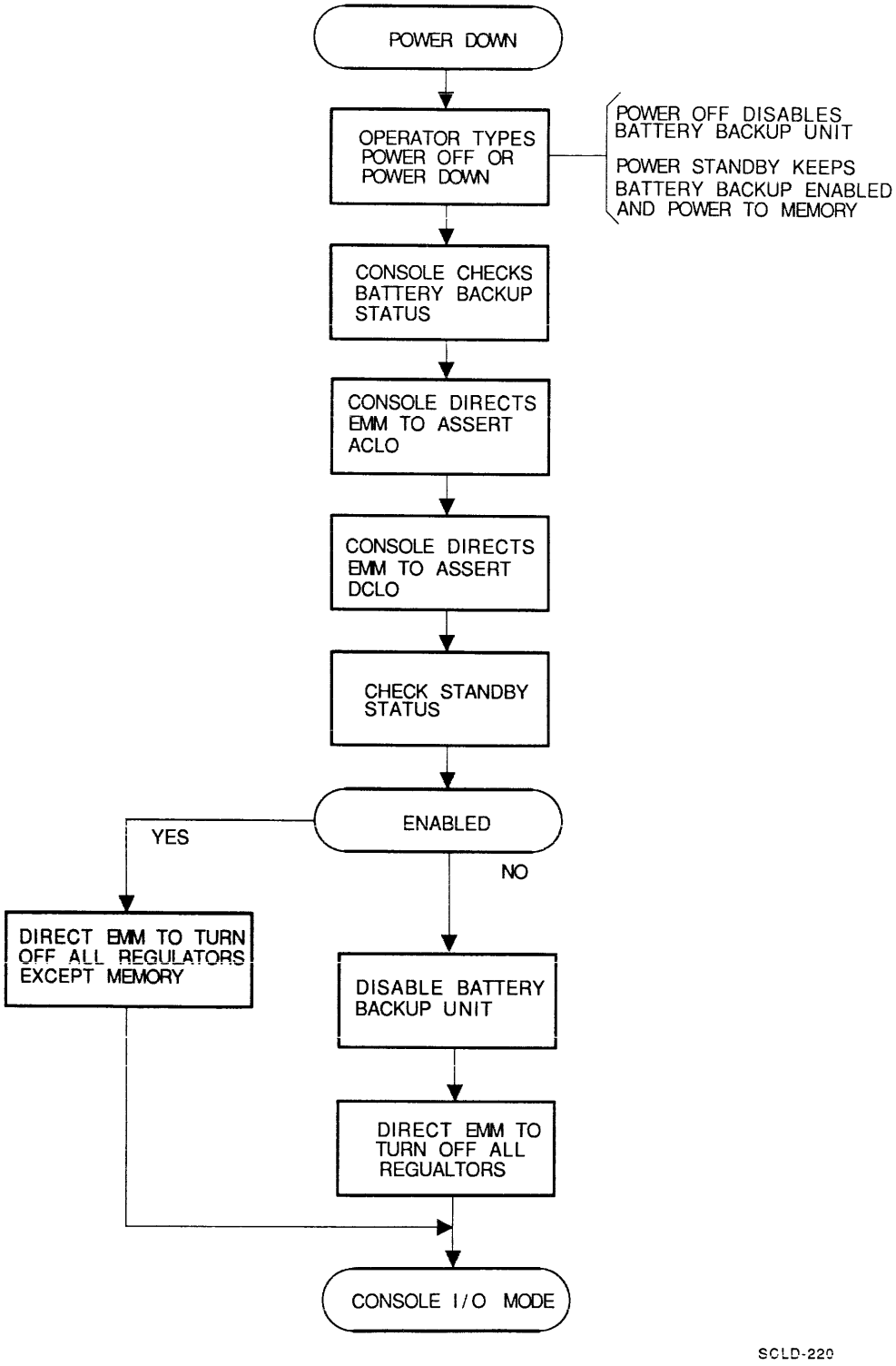


Figure 1-7 Powerdown Procedure

CHAPTER 2
FUNCTIONAL DESCRIPTION

2.1 GENERAL

This chapter contains a functional description of the console subsystem and how it interacts with the VAX 8800 system. Simplified block diagrams illustrate the process of initialization, status and control, and data transfers.

Figure 2-1 is a functional block diagram of the console subsystem. It is designed to show the interaction between the console and the VAX 8800 system at a functional flow level, by indicating key signals. Refer to the block diagram for the following system description.

2.2 REAL-TIME INTERFACE (RTI)

The real-time interface is the only data link between the PRO-38N console computer and the console interface located on the VAX 8800 clock module. The RTI is connected to the console computer through the PRO-38N backplane and provides the VAX 8800 system CPUs with a data path to the console storage, display, and control components.

The RTI consists of a programmable peripheral interface (PPI), an IEEE port, and two serial line ports. The PPI is the primary data transfer point to and from the VAX 8800 system. Serial line port "A" is connected to the environmental monitoring module in the VAX 8800 power system, and provides the console with a means of controlling and monitoring the power and environmental parameters of VAX 8800. Serial line port "B" is connected to a spare connector at the VAX 8800 bulkhead. The IEEE port is not used in the present configuration.

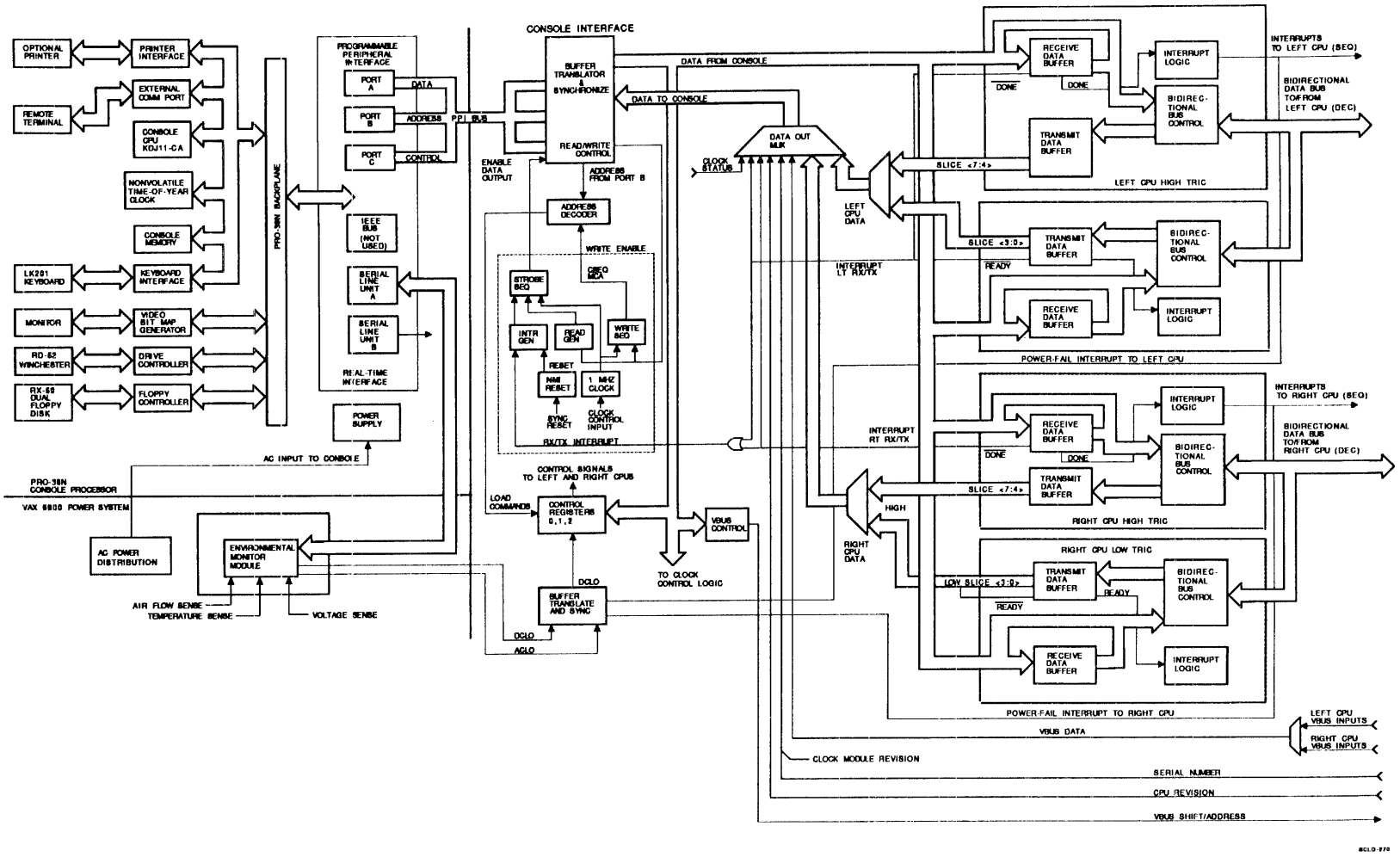


Figure 2-1 Console Subsystem Functional Block Diagram

2.2.1 Programmable Peripheral Interface (PPI)

The PPI contains three 8-bit ports that are used to transfer data, address, and control signals between the console and the VAX 8800 system. The PPI ports are connected to the buffer, translator, and synchronize circuitry on the console interface board by means of the PPI bus. Figure 2-2 shows the format of PPI Port A.

2.2.1.1 Port A -- Eight-bit bidirectional data port. Data bits to and from the VAX 8800 CPU are transferred by means of this port.

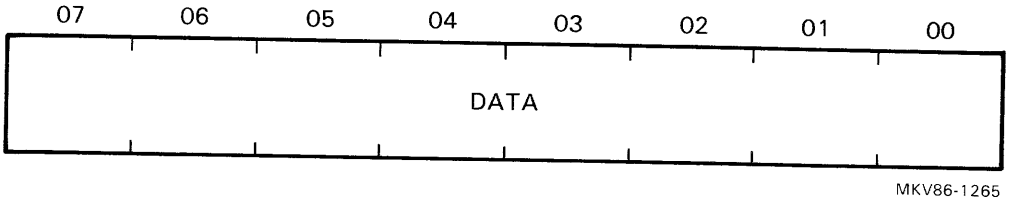


Figure 2-2 PPI Port A Format

2.2.1.2 Port B -- Eight-bit address and control port (refer to Figure 2-3 and Table 2-1). Bits <3:0> contain address information that is passed to the console address decoder. Bits <7:4> enable/disable console-to-CPU communications, enable reads from the interface, and mask interrupts to the console.

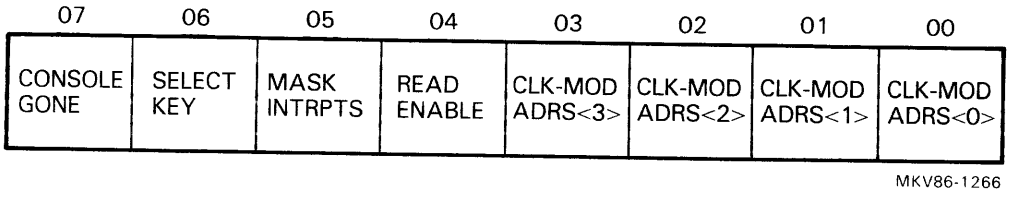


Figure 2-3 PPI Port B Format

Table 2-1 PPI Port B Bit Description

Bit	Name	Description
<07>	CONSOLE GONE	0 = Console power is ON. 1 = Pulled up to "1" by clock module, when console power is OFF, or when any of the cables connecting the RTI to the clock module are disconnected.
<06>	SELECT KEY	Provides multiple use of MASK INTRPTS (mask interrupts) and READ ENABLE bits. 0 = MASK INTERPTS and READ ENABLE bits perform normal functions. 1 = MASK INTERPTS serves the KEY DATA function, and READ ENABLE serves the KEY CLOCK function. The key data and key clock signals are used in the unlock sequence of console isolation.
<05>	MASK INTERPTS (Mask Interrupts)	Function determined by SELECT KEY bit. If the SELECT KEY bit is a "0", this bit masks all interrupts from the console interface to the console. If the SELECT KEY bit is a "1", this bit and the READ ENABLE bit enable console-to-CPU communications. SELECT KEY = 0 0 = Interrupts are not masked. 1 = Interrupts are masked. The mask bit must be set prior to a read/write from console interface operation. SELECT KEY = 1 0 = Normal mode, no action. 1 = Provides a serial data input to the shift registers that enable the console sequencer following a console power failure or cable disconnect.

Table 2-1 PPI Port B Bit Description (Cont)

Bit	Name	Description
<04>	READ ENABLE	Function determined by SELECT KEY bit. If the SELECT KEY bit is a "0", this bit initiates a read sequence from the console interface. If the SELECT KEY bit is a "1", this bit, in conjunction with the MASK INTERRUPTS bit, unlocks the console interface and enables console-to-CPU communications. SELECT KEY = 0 Read Enable is toggled from 0 to 1 to initiate read sequencing control of the console interface. SELECT KEY = 1 Clock input to the shift registers used to enable the console sequencer following a PRO-38N power failure or cable disconnect.
<03>	CLK_MOD ADRS <3:0> (Clock Module Address)	Four-bit address to the console address decoder on the console interface.

2.2.1.3 Port C -- Eight-bit bidirectional status and control port (refer to Figure 2-4 and Table 2-2). Bits <3, 5, and 7> are read-only bits that display PPI status information. Bits <6 and 4> are used by the console's I/O driver to control interrupts during data transfers to and from the console interface. Bits <2:0> are programmable bits used for the detection of RTI power-up diagnostic execution. Bit 0 enables the console interconnect test (LOOPBACK A). Bit 1 enables the interface data path test (LOOPBACK B).

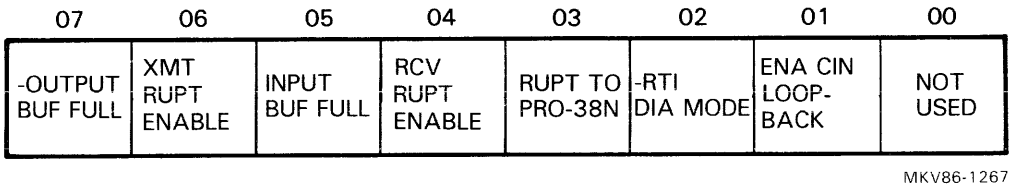


Figure 2-4 PPI Port C Format

Table 2-2 PPI Port C Bit Description

Bit	Name	Description
<07>	-OUTPUT BUF FULL (Output Buffer Full)	0 = PPI Port "A" output buffer contains data for the console interface. 1 = PPI Port "A" output buffer is empty.
<06>	XMT RUPT ENABLE (Transmit Interrupt Enabled)	0 = Disables interrupts to the console, when PPI Port "A" output buffer is empty. 1 = Enables interrupts to the console, when the PPI Port "A" output buffer is empty.
<05>	INPUT BUF FULL (Input Buffer Full)	0 = PPI Port "A" input buffer is empty. 1 = PPI Port "A" input buffer contains data from the console interface.
<04>	RCV RUPT ENABLE (Receive Interrupt Enable)	0 = Disables interrupts to the console when the PPI Port "A" input buffer is full. 1 = Enables interrupts to the console when the PPI Port "A" input buffer is full.

Table 2-2 PPI Port C Bit Description (Cont)

Bit	Name	Description
<03>	RUPT TO PRO-38N (Interrupt to PRO-38N)	0 = Inactive interrupt request to the PRO-38N. 1 = Active interrupt request.
<02>	-RTI DIA MODE (RTI Diagnostic Mode)	0 = Real-time interface power-up diagnostics are executing. 1 = Normal VAX 8800 console operation.
<01>	ENABLE LOOPBACK	0 = Loopback disabled. 1 = Loopback enabled.

2.2.1.4 PPI Control -- The PPI is controlled by programming an 8-bit control word register. The control word register performs two functions selected by the MSB of the register (refer to Figure 2-5 and Table 2-3). When the MSB equals 0, the CWR controls the bit set and reset functions of PPI Port "C". If the MSB equals 1, the CWR selects the modes of operation of Ports "A", "B", and "C".

CAUTION

The control word register should never be written to by anything other than the console's I/O driver or VAX 8800 diagnostics.

Writing to this register using either the BIT SET/RESET function or the MODE SELECT function can cause damage to the RTI or clock module hardware.

Reading the control word register in accordance with the PPI specification drives the data bus with undefined data and places the clock handshaking signals into an unknown state.

The contents of the control word register are not returned when the register is read.

Figure 2-5 shows the control word register as it should appear after the system has been booted. This configuration should not be altered except by the console's I/O driver and the VAX 8800 diagnostics.

CONTROL WORD REGISTER (WRITE ONLY)
PRO-38N ADDRESS = 17775216 (OCTAL)

07	06	05	04	03	02	01	00
CW FUNC	A MODE		A I/-O	CU I/-O	B MODE	B I/-O	CL I/-O
1	1	0	1	0	0	0	0

MKV86-1268

Figure 2-5 PPI Control Register Format

Table 2-3 PPI Control Register Bit Description

Bit(s)	Name	Description
<07>	CW FUNC (Control Word Function)	0 = Port "C" bit set and reset 1 = Mode select (Ports "A", "B", and "C")
<05:06>	A MODE (Port "A" Mode Control)	1 0 = Mode 2 (Strobed Bidirectional)
<04>	A I/-O (Port "A" Input/Output Mode)	Must = 1 This bit is used by the RTI ROM only
<03>	CU I/-O (Port "C" Upper 4 bits I/O)	0 = Output 1 = Input
<02>	B MODE (Port "B" Mode Control)	0 = Basic input/output (outputs latched, inputs unlatched, no handshaking) 1 = Strobed input/output
<01>	B I/-O (Port "B" Input/Output Mode)	0 = Output 1 = Input
<00>	CL I/-O (Port "C" Lower 4 bits I/O)	0 = Output 1 = Input

2.2.2 Serial Line Port

Serial line port "A" provides the console with a communication link to the environmental monitoring module in the VAX 8800 power supply. A major component of the serial line port is the enhanced programmable communications interface (EPCI). The ECPI controls the method of communication between the console and the EMM through three programmable registers. Three hardware registers are used for transmitting and receiving data and reading status from the EMM.

2.2.2.1 ECPI Registers -- The ECPI controls the communications between the console and the EMM with two mode registers and one command register. Mode register 1 (Figure 2-6 and Table 2-4) defines the required format of all messages to be transmitted or received. Mode register 2 (Figure 2-7 and Table 2-5) defines the clock source, internal baud rate generator frequency, and the receiver/transmitter clock baud rate factor. The command register (Figure 2-8 and Table 2-6) enables/disables loopback testing and data transfers, and resets error flags in the status register.

MODE REGISTER 1 (READ/WRITE)
PRO-38N ADDRESS = 17775244 (OCTAL)

07	06	05	04	03	02	01	00
STOP BITS		PAR TYP	PAR EN	CHAR LENGTH		M/B FACT	
1	0	X	0	1	1	0	1

MKV86-1269

Figure 2-6 ECPI Mode 1 Register

Table 2-4 ECPI Mode 1 Register Bit Description

Bit(s)	Name	Description
<06:07>	STOP BITS (Number of Stop Bits)	1 0 = 1.5 Stop bits
<05>	PAR TYP (Parity Type)	x = Don't care (Parity disabled by bit 4)
<04>	PAR EN (Parity Enable)	0 = Disable 1 = Enabled
<02:03>	CHAR LENGTH (Character Length)	1 1 = 8-bit characters
<00:01>	M/B FACT (Mode and Baud Rate Factors)	0 1 = Asynchronous communication and Unity baud rate

NOTE

The required Mode 1 format for EMM communications is 1.5 stop bits, parity disabled, 8-bit word length, asynchronous communications, and a unity baud rate factor. This configuration is programmed by writing 215 Octal to mode register 1.

MODE REGISTER 2 (READ/WRITE)
PRO-38N ADDRESS = 17775244 (OCTAL)

07	06	05	04	03	02	01	00
RCV/XMT CLOCK				BAUD RATE SEL			
0	0	1	1	1	1	1	0

MKV86-1270

Figure 2-7 ECPI Mode 2 Register

Table 2-5 ECPI Mode 2 Register Bit Description

Bits	Name	Description
<04:07>	RCV/XMT CLOCK (Receiver and Transmitter Clock)	0011 = Internal clock source Unity baud rate factor
<00:03>	BAUD RATE SEL (Baud Rate Select)	1110 = 9600 Baud

NOTE

Console communications with the EMM require setting the internal clock source at 9600 baud with unity baud rate factor. This is accomplished by writing 076 (octal) to mode register 2.

Mode registers 1 and 2 use the same address for register access. An alternating mode register pointer allows alternating access to the registers during normal read/write operations. The pointer can be initialized to point to mode register 1 with either a RESET command, or a read command register operation.

The command register is used to enable or disable the transmitter and receiver, reset error flags in the status register, and force key signals into the high or low state. Normal operation for the command register is to have loopbacks disabled and the break condition deasserted. The following drawing shows the command register set up for normal operation.

COMMAND REGISTER (ACCESS AS SPECIFIED)
PRO-38N ADDRESS = 17775246 (OCTAL)

07	06	05	04	03	02	01	00
OP MODE		FRC RTS	RST ERR	FRC BRK	RCV CTL	FRC DTR	XMT CTL
0	0	1/0	1/0	0	1/0	1/0	1/0

MKV86-1271

Figure 2-8 ECPI Command Register

Table 2-6 ECPI Command Register Bit Description

Bit(s)	Name	Description
<06:07>	OP MODE (Operating Mode)	0 0 = Normal operation 1 0 = Local Loopback enabled (all characters echo back)
<05>	FRC RTS (Force Request To Send)	0 = Disabled 1 = Enabled
<04>	RST ERR (Reset Error) write-only bit	0 = Reset not enabled 1 = Reset status register error flags
<03>	FRC BRK (Force Break)	0 = Normal operation
<02>	RCV CTL (Receiver Control)	0 = Receiver disabled 1 = Receiver enabled
<01>	FRC DTR (Force Data Terminal Ready)	0 = Disabled 1 = Enabled
<00>	XMT CTL (Transmit Control)	0 = Transmitter disabled 1 = Transmitter enabled

2.2.2.2 Data Transfer and Status Registers -- The serial line port uses three registers for transferring data and status between the RTI and the EMM (refer to Figure 2-9 and Table 2-7). The receive holding register is an 8-bit data buffer that holds data received from the EMM. The transmit holding register is an 8-bit data buffer that holds data to be transmitted to the EMM. The status register contains error status, data set ready and data carrier detect status, transmitter shift register status, and transmitter and receiver ready and done bits.

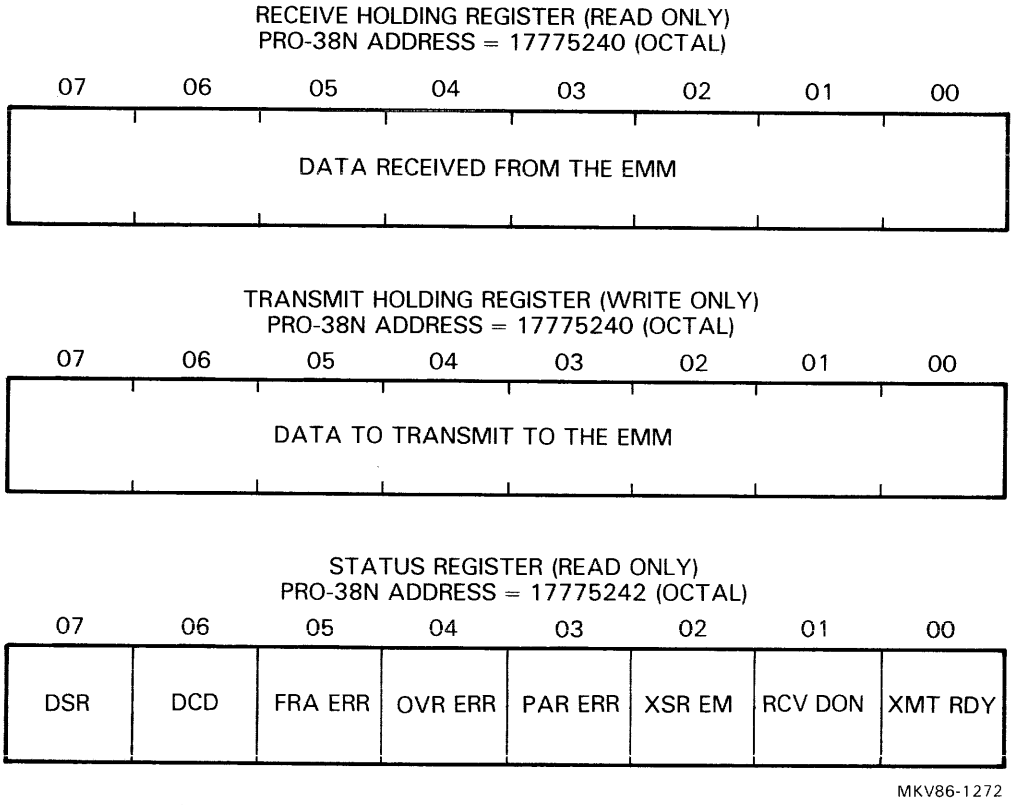


Figure 2-9 Serial Line Port Data and Status Registers

Table 2-7 Serial Line Port Data and Status Registers Bit Description

Bit	Name	Description
<07>	DSR (Data Set Ready)	0 = DCD Input is active 1 = DCD Input is not active
<06>	DCD (Data Carrier Detect)	0 = DSR Input is active 1 = DSR Input is not active
<05>	FRA ERR (Framing Error)	0 = No error 1 = Error
<04>	OVR ERR (Overrun Error)	0 = No Error 1 = Error
<03>	PAR ERR (Parity Error)	0 = No error 1 = Error
<02>	XSR EM (Transmitter Shift Register Empty/Change in DSR or DCD)	0 = Shift register not empty/No change in DSR or DCD 1 = Shift register empty/Change in DSR or DCD
<01>	RCV DON (Receiver Done)	0 = Rcvr holding register empty 1 = Rcvr holding register full
<00>	XMT RDY (Transmitter Ready)	0 = Xmt holding register full 1 = Xmt holding register empty

2.3 CONSOLE INTERFACE

The console interface provides the console and the VAX 8800 system with a medium for communication with, and control of, the VAX 8800 CPUs. Data and control signals are received from the real-time interface in the console and distributed to specific areas of the console interface for loading, control, and monitoring of the VAX 8800 system.

The primary functional areas of the console interface are:

- Buffer, translate, and synchronization circuitry
- Console address decode
- Console sequencer (CSEQ MCA)
- Terminal register interval clock (TRIC MCA)
- Data output mux
- CPU control registers
- Visibility bus
- Console interrupts
- Power status

2.3.1 Buffer Translate and Synchronize

The buffer translate and synchronize logic buffers the incoming signals from the console and provides translation of the console output signals from TTL to ECL logic. The translated data is applied to the TRIC MCAs, control registers, VBus control and clock control logic. The lower four bits of the translated port "B" address data are applied to the address decoder where they are used in the generation of control signals for the operation of the console interface. Most of the signals coming from port C or the upper four bits of port B are synchronized to the VAX 8800 CPU clocks after being translated to ECL logic levels. Control and data to the console is translated from ECL to TTL logic prior to being placed in the output drivers.

2.3.2 Console Address Decode

The address decoder logic uses the lower four bits <3:0> of the incoming port "B" address from the PPI to generate control signals for control registers <2:0>, VBus control, the clock registers, console receive data buffer, and the console load paths to the CPU. The address decoder is also used in the generation of transmit ready and receiver done signals for the VAX IPRs.

2.3.3 Console Sequencer (CSEQ MCA)

The console sequencer MCA is the centralized data input/output control point for the console interface. Synchronized strobe and acknowledge signals are generated for the proper sequencing of data transfers as well as creation of command flags for loading the VAX 8800 control store and RAMs. Major functions performed by the CSEQ MCA include:

- Console read sequencing
- Console interrupt generation
- Console write sequencing
- Control store load mechanism
- Console/VAX CPU isolation control
- CPU timeout
- 1 MHz clock generation
- CPU write address timing

2.3.4 Terminal Register/Interval Clock (TRIC) MCA

Each of the VAX 8800 CPUs use two TRIC MCAs to transfer data and control signals between the console interface and the CPU. The combination of both TRIC MCAs make up the VAX internal processor registers (IPR) required to implement the transfer process. The IPRs for one VAX CPU include:

Receive IPR

- Receive Data Buffer (RXDB <15:0>)
- Receive Control and Status (RXCS <7:0>)

Transmit IPR

- Transmit Data Buffer (TXDB <15:0>)
- Transmit Control and Status (TXCS <7:0>)

Interval Clock

- Interval Count Register (ICR <30:0>)
- Next Interval Count Register (NICR <31:0>)
- Interval Clock Control/Status Register (ICCS <31:0>)

Figure 2-10 shows the IPRs contained on the TRIC MCAs, and the bit configuration of each register.

Figure 2-10 RXDB, TXDB, and DBCS

The data buffer control/status register is formed by combining two bits of the RXCS register and two bits of the TXCS register into a common register. RXCS <7:6> are mapped into DBCS <7:6>, and TXCS <7:6> are mapped into DBCS <3:2>. The bits are reformatted by VAX 8800 microcode to be compatible with VMS.

Input and output data between the VAX CPU and the TRIC MCA is transferred over an 8-bit bidirectional bus connected to the CPU decoder. The control process is determined by the operating mode (program or console).

2.3.4.1 Program Mode -- Data from the console to the CPU is under interrupt control using the DONE bit from the receive data buffer in the High slice TRIC MCA. The transmit data buffer uses the READY bit in the low slice TRIC for interrupt control of data transfers to the console.

2.3.4.2 Console Mode -- During the console mode of operation, the CPU polls the READY and DONE bits to determine the status of the data transfer registers.

The READY and DONE bits from the receive and transmit data buffers are applied to the interrupt generator on the CSEQ MCA. These signals are used to create an interrupt strobe signal to the console for use in the data transfer process. The same signals are also applied to the data output mux as a flag to identify the source of the interrupt.

2.3.5 Data Output Mux

The data output mux is the centralized collection point for all of the data and status going to the console from the console interface. Console address bits <3:1> determine which signals are selected in the mux for output to the console. Input signals to the mux consist of:

- Left TXDB data
- Right TXDB data
- VBus data
- Interrupt status
- Clock status
- Left CPU timeout
- Right CPU timeout
- Clock module revision
- CPU backplane revision
- Serial number

The selected mux data is translated to TTL logic levels and sent to the console by means of the 8-bit bidirectional PPI port "A" bus.

2.3.6 Control Registers

The console interface uses three control registers (CR0, CR1, and CR2) to transfer control signals to the VAX 8800 CPUs. The write-only control registers receive the Port "A" data from the console and latch enable strobes, and the register load commands from the console address decoder logic. Figure 2-11 shows the configuration of the control registers and identifies the bit configurations.

CONTROL REGISTER 0 WRITE ONLY

07	06	05	04	03	02	01	00
DISABLE LFT CPU STALLS	DISABLE LFT CPU TRAPS	DISABLE LFT CPU NMI SEQ	LEFT CPU INIT	DISABLE RHT CPU STALLS	DISABLE RHT CPU TRAPS	DISABLE RHT CPU NMI SEQ	RIGHT CPU INIT

CONTROL REGISTER 1 WRITE ONLY

07	06	05	04	03	02	01	00
NMI UNJAM	ENABLE LFT RXDB INTRPT	NMI SLOW MODE	ENABLE RHT RXDB INTRPT	ENABLE TRAP ON LT MATCH	HALT LEFT CPU	ENABLE TRAP ON RT MATCH	HALT RIGHT CPU

CONTROL REGISTER 2 WRITE ONLY

07	06	05	04	03	02	01	00
NOT USED	NOT USED	NOT USED	ENABLE IDP LOOP-BACK	DISABLE LFT SDF WRITES	DISABLE RHT SDF WRITES	CLEAR LATCHED RESET	SET CPU TIMEOUT FLAG

MKV86-1274

Figure 2-11 Control Registers

2.3.7 Visibility Bus Control

The visibility bus provides the console with visibility into the VAX 8800 system and enables the console to monitor the sixteen modules using the VBus. Port "A" data from the console controls the selection of VBus signals through the use of the visibility bus control register (Figure 2-12). The VBus data is read from the VBus access register and muxed with the outgoing signals in the data output mux.

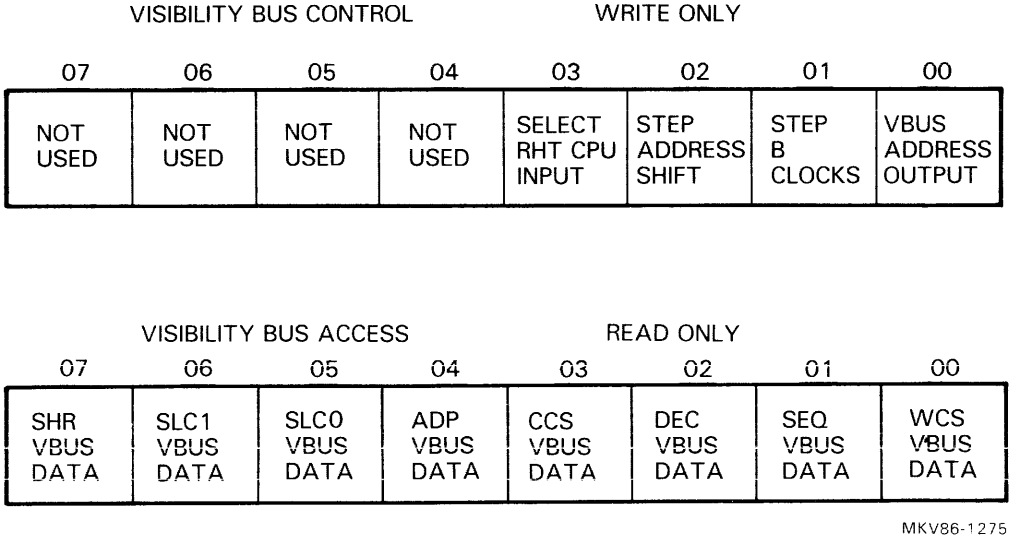


Figure 2-12 VBus Control and Access Registers

2.3.8 Console Interrupt Generation

Console interrupts can occur from any one of the following five events. The events are listed in priority sequence with NMI reset having the highest priority:

NMI Reset

A request from a processor external to the VAX 8800 system to halt and reboot the VAX 8800 processor(s).

TXDB Not Ready (Primary CPU)
TXDB Not Ready (Secondary CPU)

The VAX 8800 CPU has placed data for the console into the transmit data buffer.

RXDB Not Done (Primary CPU)
RXDB Not Done (Secondary CPU)

The data placed in the receive data buffer by the console has been accepted by the VAX 8800 CPU and the buffer is empty and ready for another data transfer.

Requests for a console interrupt are passed to the interrupt generator on the CSEQ module, which handles the sequencing of the interrupt request.

2.3.9 Power Status

The console interface receives a copy of the ACLO and DCLO signals from the EMM and translates the signals to ECL logic levels. The ACLO signal is synchronized with the VAX 8800 CPU clocks and is used to generate the power-fail interrupt to the VAX 8800 CPU.

The clock synchronized DCLO signal is used to assert all of the bits in control register 0, including the CPU hardware initialization signal CPU INIT.

2.4 CONSOLE/VAX 8800 INTERACTION

The following paragraphs describe the interaction of the console and the VAX 8800 system during initialization, read/write control, and data transfers.

2.4.1 Initialization

The console controls the operations required to get the VAX 8800 system powered up and running. Major tasks involved in this process include:

- Turn "ON" system power
- Reset EMM
- Console power ON
- Load and run console power software
- Sequenced power application
- Initialize hardware
- Test and checkout
- Load RAMs and DRAMs

The initialization process is presented in detailed flowchart form, and the events are shown in the sequence in which they occur. The flowcharts include a combination of operator commands, command files, and hardware actions and responses. They are intended to show the sequence of events that occur during a process. At the end of the flowcharts is a table listing some of the key signals and functions identified in the flowcharts. Included is a description of the signal and the logic circuit where the signal is processed.

2.4.1.1 Turn ON and Monitor System Power/Reset EMM -- The events involved in the initial turn-on of power to the VAX 8800 system, and the subsequent resetting of the environmental monitoring module are shown in Figures 2-13 and 2-14. These events are controlled by the VAX 8800 power system complex. Refer to the power system section of the manual for a complete description of the events shown in Figures 2-13 and 2-14.

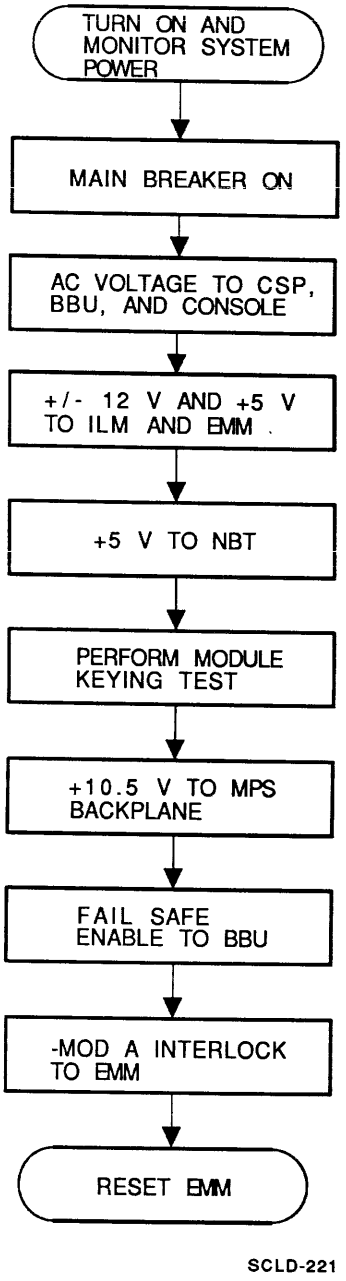


Figure 2-13 System Power-On Sequence

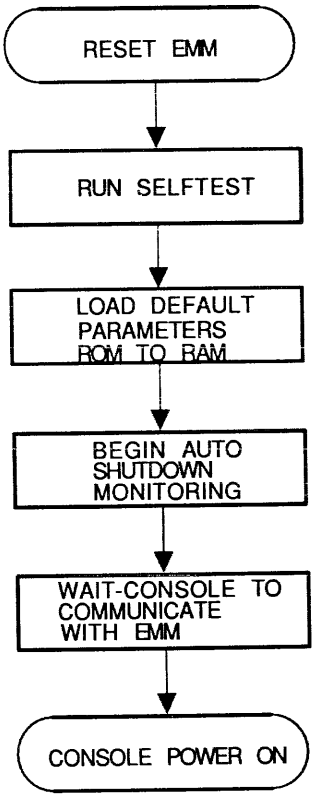


Figure 2-14 Environmental Monitoring Module Reset Sequence

2.4.1.2 Console Poweron -- When the console power switch is set to "1", the console begins a selftest of the PRO-38N before continuing with the VAX 8800 power sequence. Figure 2-15 shows the major events that occur when the console power is applied. Refer to the Professional 300 Series handbooks for additional information.

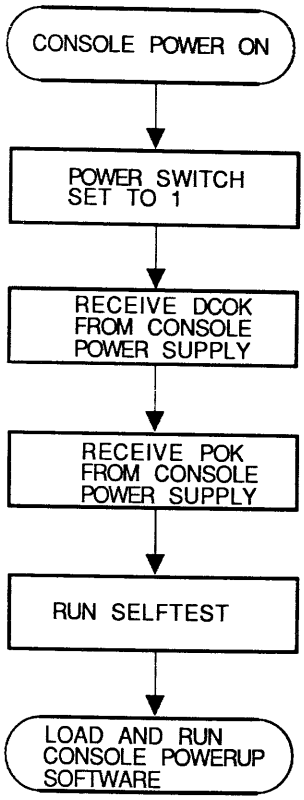


Figure 2-15 Console Power-On Events

2.4.1.3 Load and Run Console Power-Up Software -- When the VAX 8800 main cabinet power switch is turned on, the power supply and the environmental monitoring module are waiting for control signals from the console to proceed with the power-on sequence. Preliminary tasks of the console macrocode require communication with the EMM to:

- Establish and test communications
- Send environmental parameters to the EMM RAM
- Enable EMM monitoring tasks
- Enable sequenced power application to VAX 8800 modules

Serial data is transferred between the console and EMM using the transmit and receive holding registers in the serial line port of the RTI. Figure 2-16 shows a simplified drawing of the three key registers in the serial line port. Figure 2-17 identifies the events that occur during the loading and running of the console power-up software.

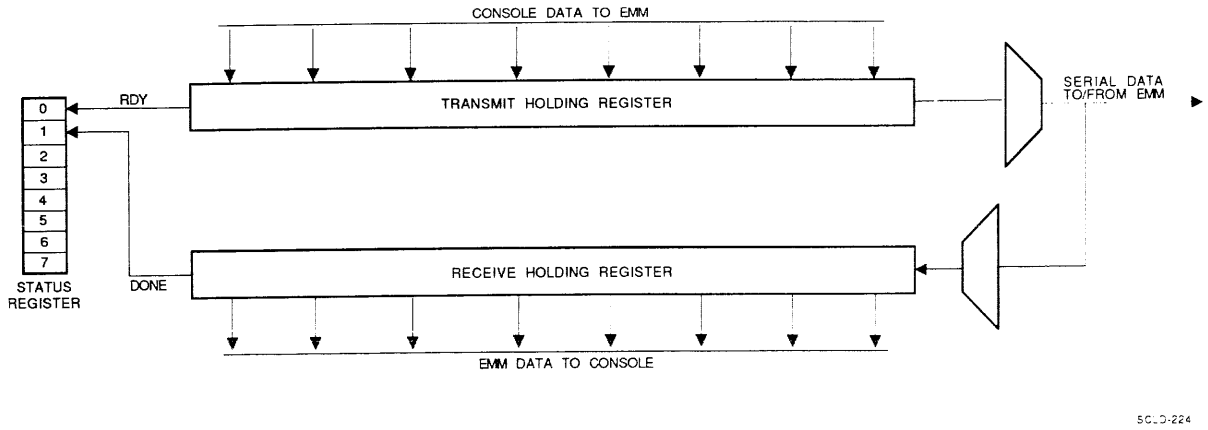


Figure 2-16 Serial Line Port Data Transfer Registers

Bits 0 and 1 of the status register inform the console of the status of the holding registers during data transfers (refer to Table 2-8).

Table 2-8 Serial Line Port Data Transfer Registers Bit Description

Bit	Name	Description
<01>	DONE	0 = Receiver holding register is empty and ready to accept another data character from the EMM. 1 = Receiver holding register still contains the last character placed in it by the EMM.
<00>	READY	0 = Transmit holding register is full. The EMM has not taken the last character yet. 1 = Transmit holding register is empty. The EMM has taken the last character and the register is ready to accept another character for transfer.

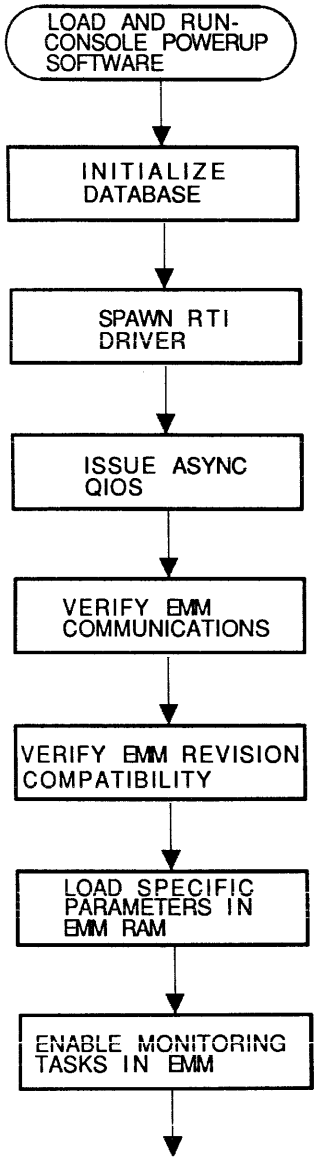


Figure 2-17 Load/Run Console Power-Up Software Events
(Sheet 1 of 2)

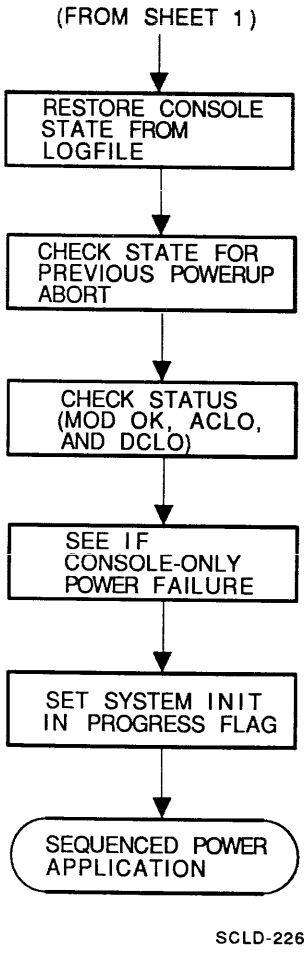


Figure 2-17 Load/Run Console Power-Up Software Events (Sheet 2 of 2)

2.4.1.4 Sequenced Power Application -- The sequenced power application consists of a series of software commands to the power system that enable the module regulators to apply voltages to the various VAX 8800 components. Response from the previous command is required to proceed to the next event in the sequence. Figure 2-18 identifies the events that occur during the power application phase.

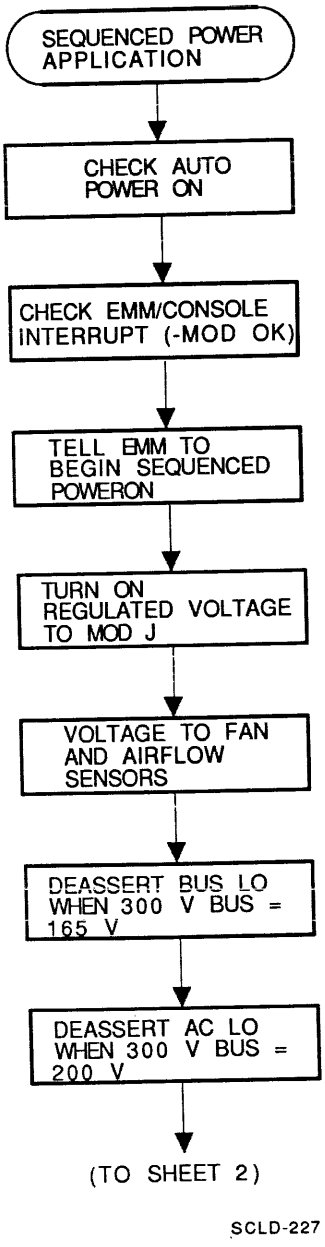


Figure 2-18 Sequenced Power Application Events (Sheet 1 of 4)

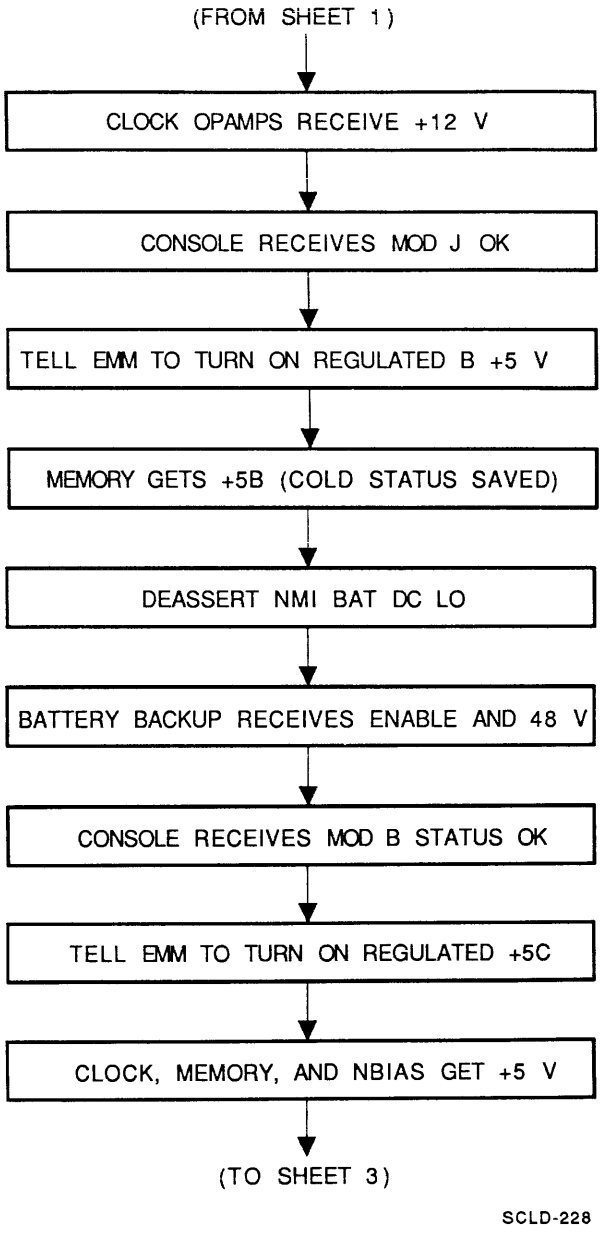


Figure 2-18 Sequenced Power Application Events (Sheet 2 of 4)

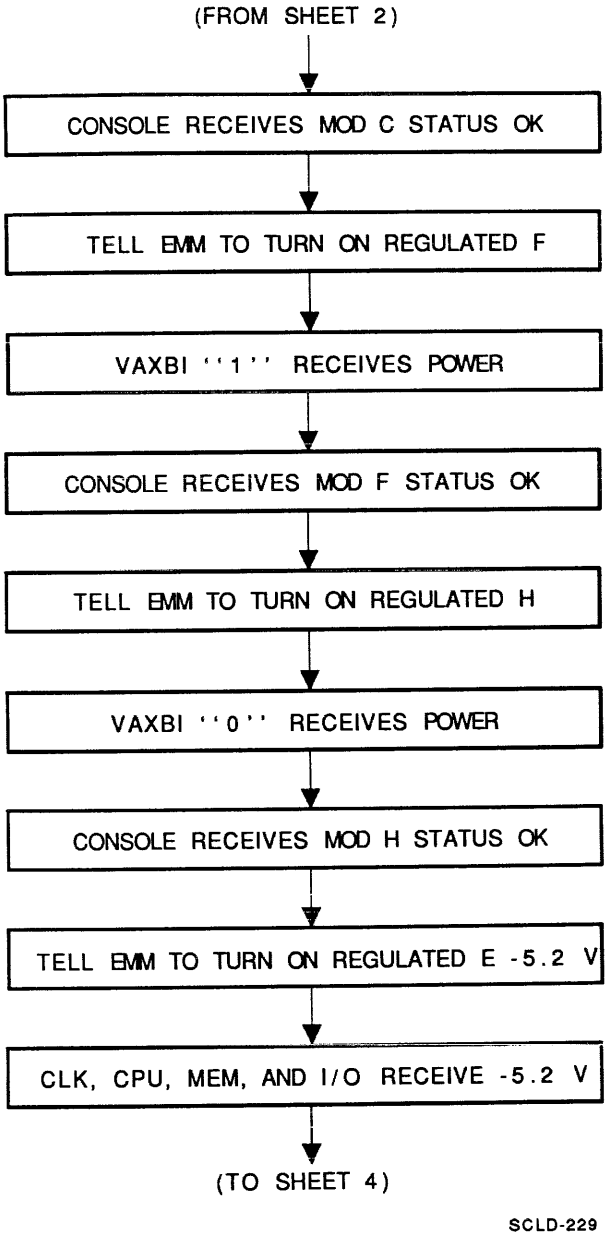


Figure 2-18 Sequenced Power Application Events (Sheet 3 of 4)

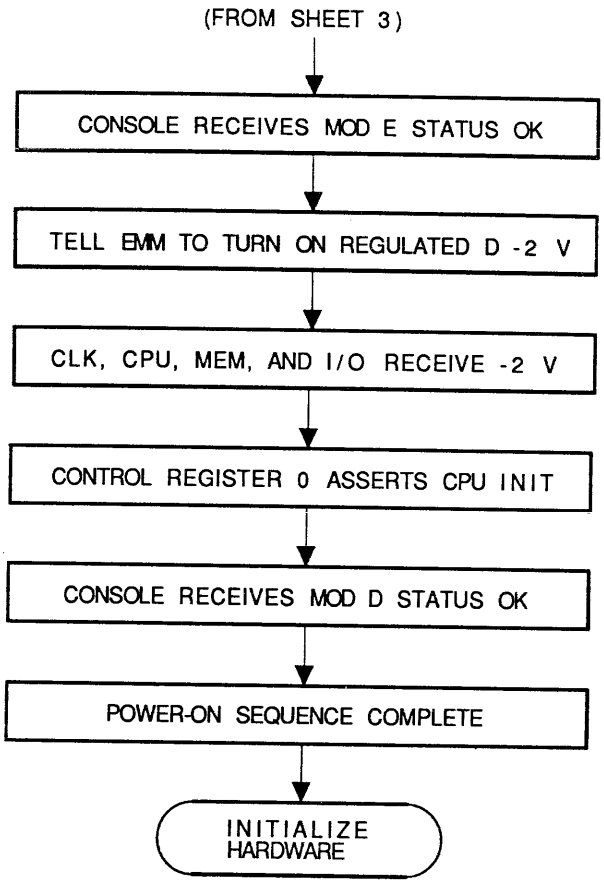


Figure 2-18 Sequenced Power Application Events (Sheet 4 of 4)

2.4.1.5 Initialize Hardware -- When the sequenced power application process is complete, the console software must ensure that the console interface is initialized and ready for communications between the console and the VAX 8800 CPUs. The first step in the initialization process involves verifying the communications path with a console interconnect loopback and an interface data path loopback.

The two tests allow the console to test data, address, and control paths between the console and the console interface, as well as the data path between the console and the VAX 8800 CPUs. Enabling the tests requires modification of the ECPI command register, the PPI control word register, bit 4 of control register 2, and bit 1 of PPI port "C".

CAUTION

The control word register should never be written to by anything other than the console's I/O driver or VAX 8800 diagnostics.

Writing to this register using either the BIT SET/RESET function or the MODE SELECT function can cause damage to the RTI or clock module hardware.

Reading the control word register in accordance with the PPI specification will drive the data bus with undefined data and place the clock handshaking signals into an unknown state.

The contents of the control word register are not returned when the register is read.

The console interconnect loopback test verifies the operation of the PPI ports through the console interface. Figures 2-19 and 2-20 show simplified block diagrams of the bits and signals that are verified.

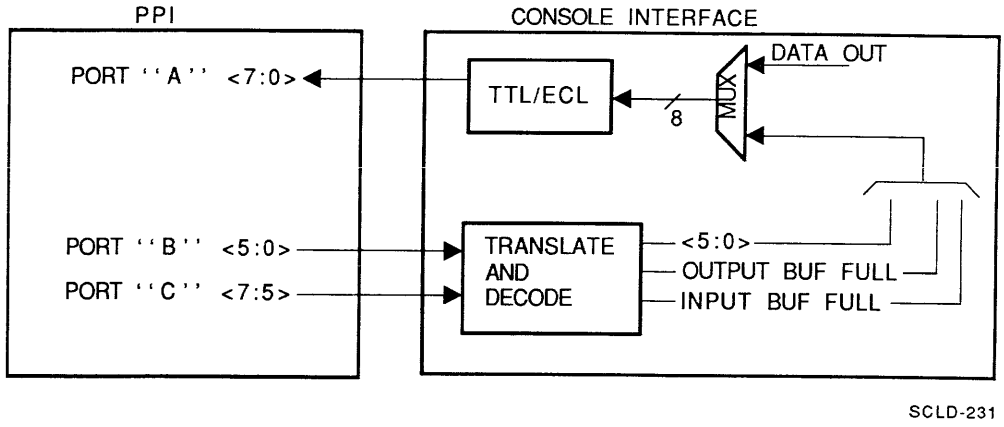


Figure 2-19 Console Interconnect Loopback Testing Through Ports A, B, and C

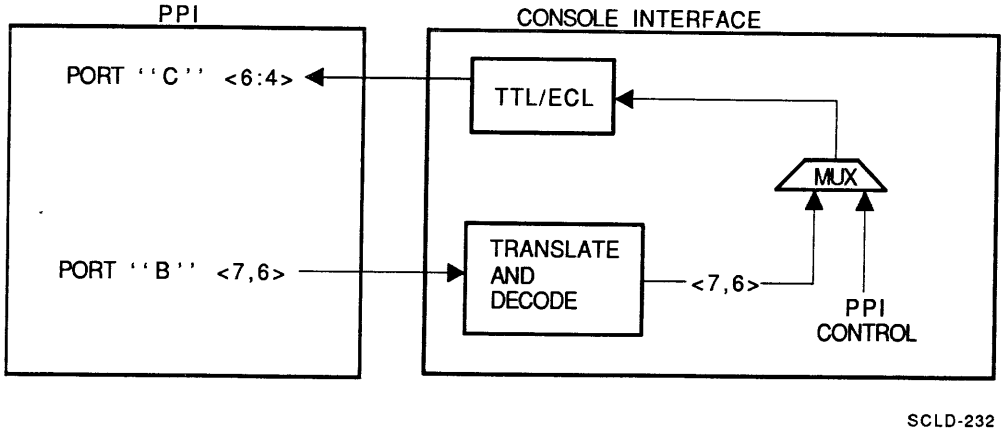


Figure 2-20 Console Interconnect Loopback Testing Through Ports B and C

The interface data path loopback test verifies the communication link between the console and the VAX 8800 CPUs. The test uses the same data path that is used in transferring both buffered and unbuffered data to and from the VAX 8800 system.

Figure 2-21 shows the two-step procedure for testing the unbuffered data path used in loading the VAX 8800 control store. Figure 2-22 shows the procedure for testing the buffered data path through RXDB and TXDB.

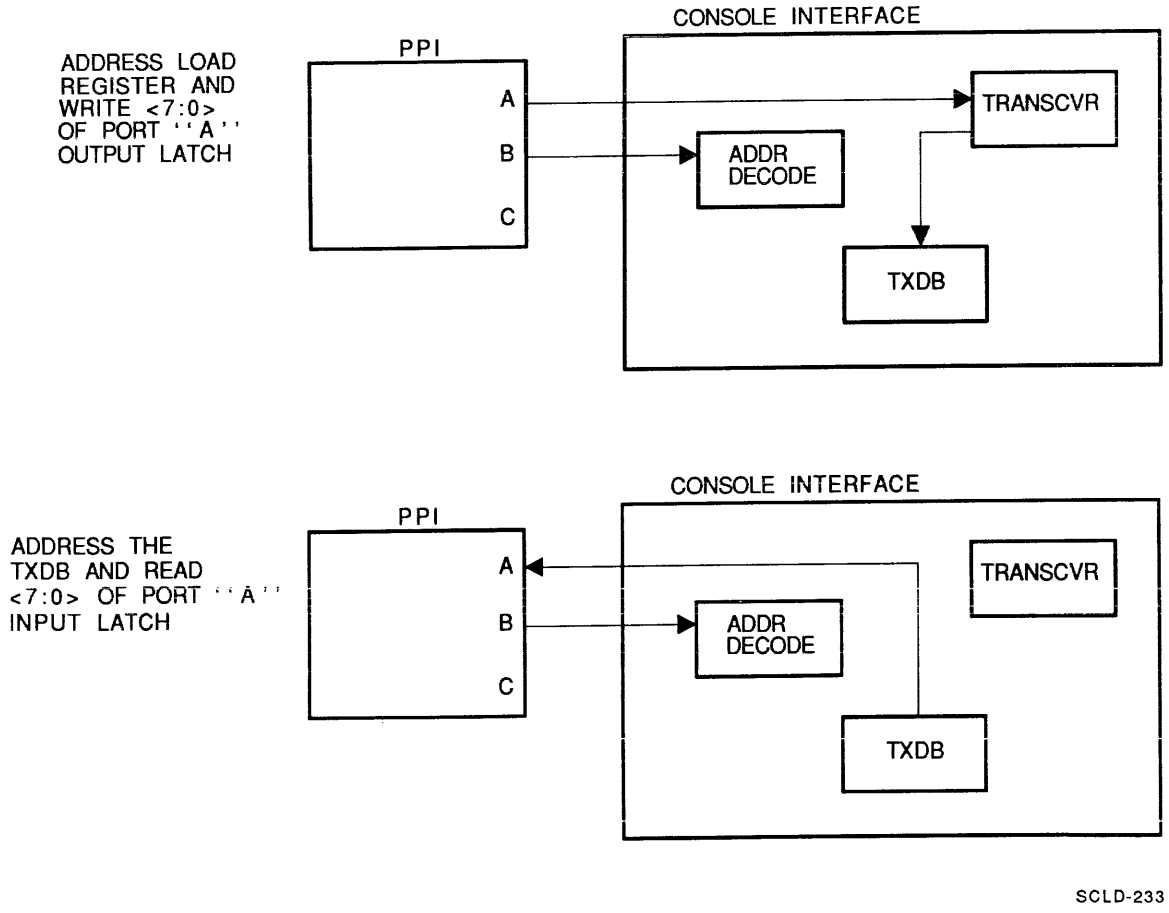


Figure 2-21 Interface Data Path Loopback Test of Unbuffered Data

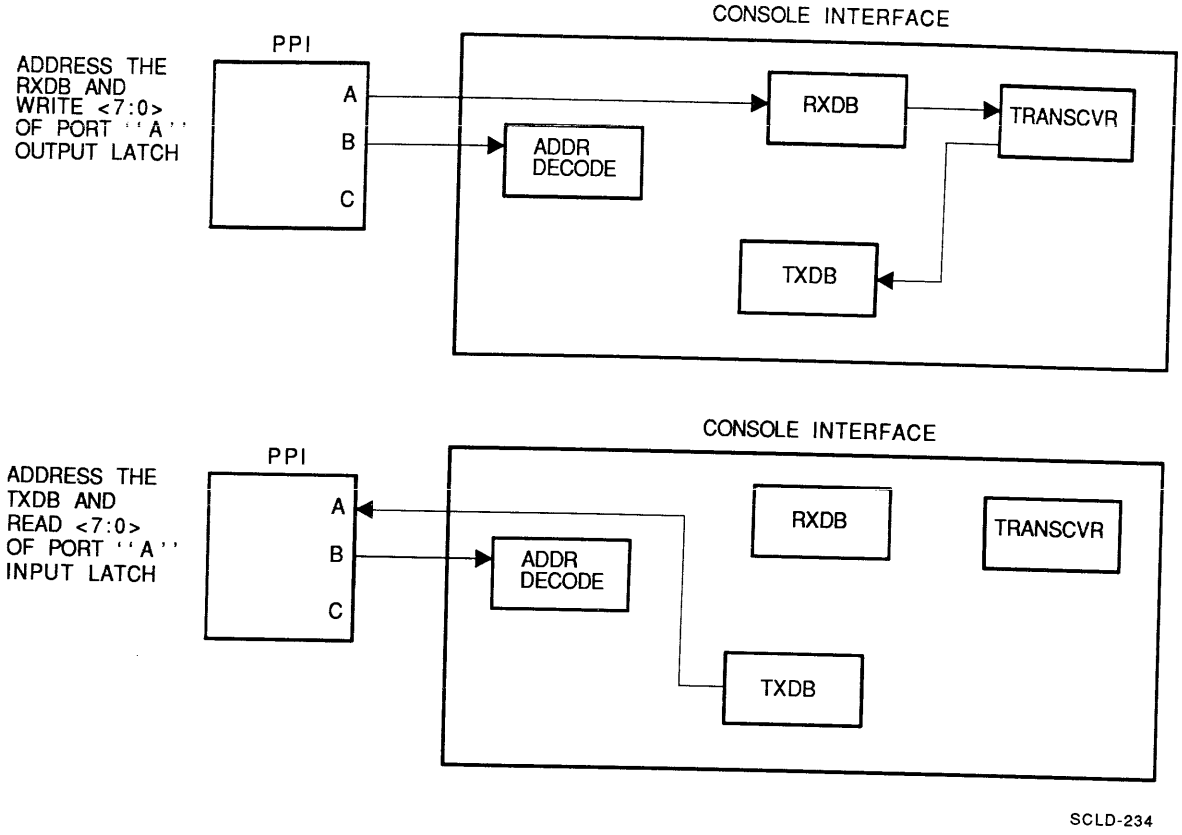


Figure 2-22 Interface Data Path Loopback Test of Buffered Data Through RXDB and TXDB

During powerup and testing of the console interface hardware, the console sequencer (CSEQ) is disabled to isolate the console subsystem from the VAX 8800 CPU.

During normal operation, bit 7 of PPI port B is driven low by the console. If the PRO-38N experiences a power failure or the console cable connecting the PPI to the clock module is disconnected, a pullup on the bit 7 line at the clock module asserts the CONSOLE GONE signal and disables all console operations that could affect the CPU.

When the PRO-38N and the RTI power-up diagnostics have been completed, the console enables communications with the CPU by sending a sequence of serial data (KEY DATA) to a demultiplexer on the CSEQ MCA. The SELECT KEY signal (port B bit 6) controls the two demultiplexers that allow the MASK/KEY DATA and READ/KEYCLOCK signals to be used for two functions.

Key data is clocked into the key circuit shift register by the KEY CLOCK signal and produces the unlock and INIT signals. Figure 2-23 shows a simplified drawing of the CSEQ enable logic.

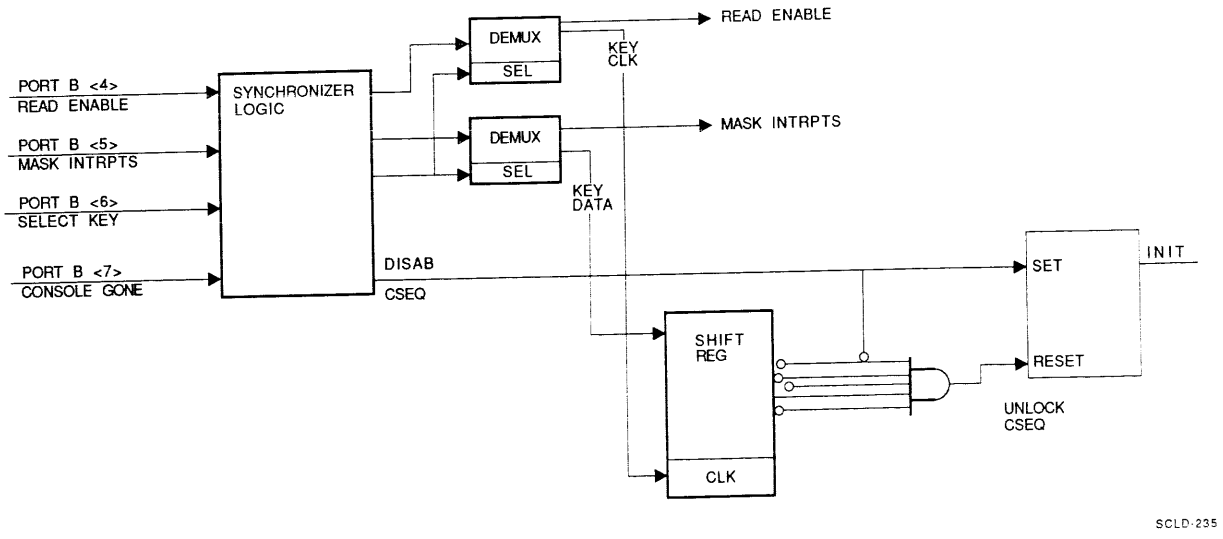


Figure 2-23 Console Sequencer Enable Logic

Hardware initialization of the VAX 8800 CPUs is performed by writing the CPU INIT bit of control register 0, or asserting DCLO on the EMM. Figure 2-24 shows a simplified block diagram of how the control register initialization signals are generated.

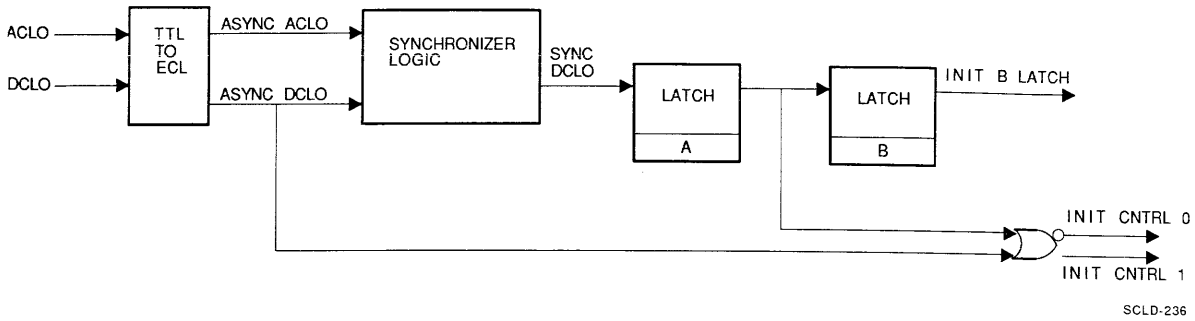


Figure 2-24 Control Register Initialization

The initialization of the hardware sets the clock, console, and VAX 8800 modules to a known state prior to the test and checkout of the system. Figure 2-25 identifies the events that occur during the hardware initialization process.

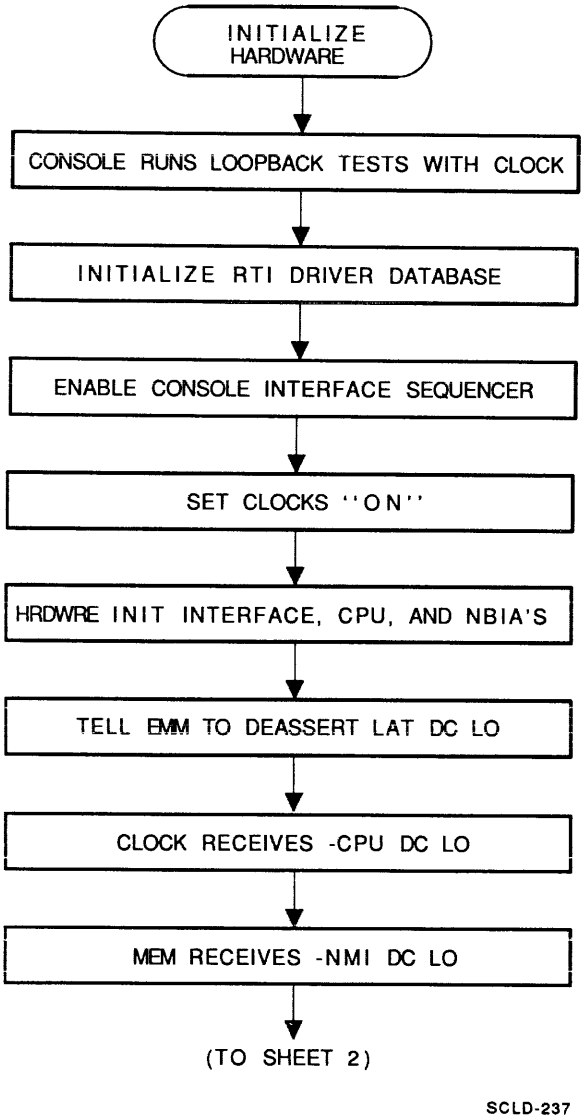


Figure 2-25 Hardware Initialization Events (Sheet 1 of 2)

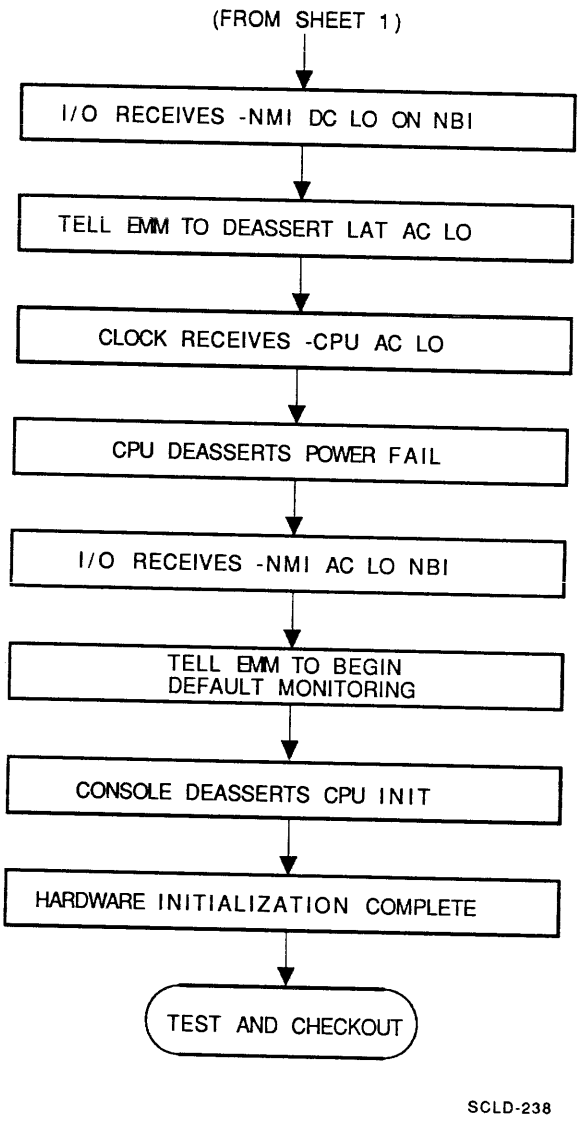


Figure 2-25 Hardware Initialization Events (Sheet 2 of 2)

2.4.1.6 Test and Checkout -- CPU module placement is checked using a callable diagnostic routine that writes the address of a module onto the VBus address path. The tail bit of the address is read back in the VBus data path and verified. This procedure is performed with both a "1" and "0" for all of the modules in both of the CPUs.

After completion of the module placement test, the console proceeds with the revision sensing test. Figure 2-26 identifies the events that occur during the test and checkout process.

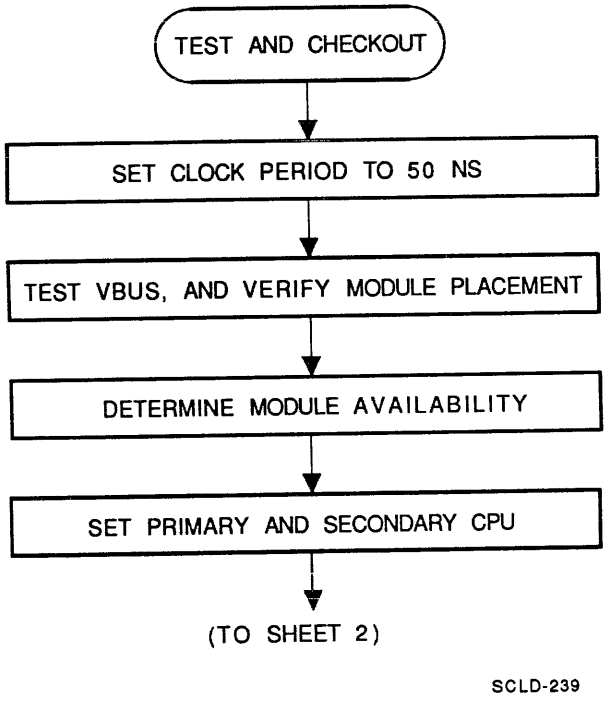


Figure 2-26 Test and Checkout Events (Sheet 1 of 2)

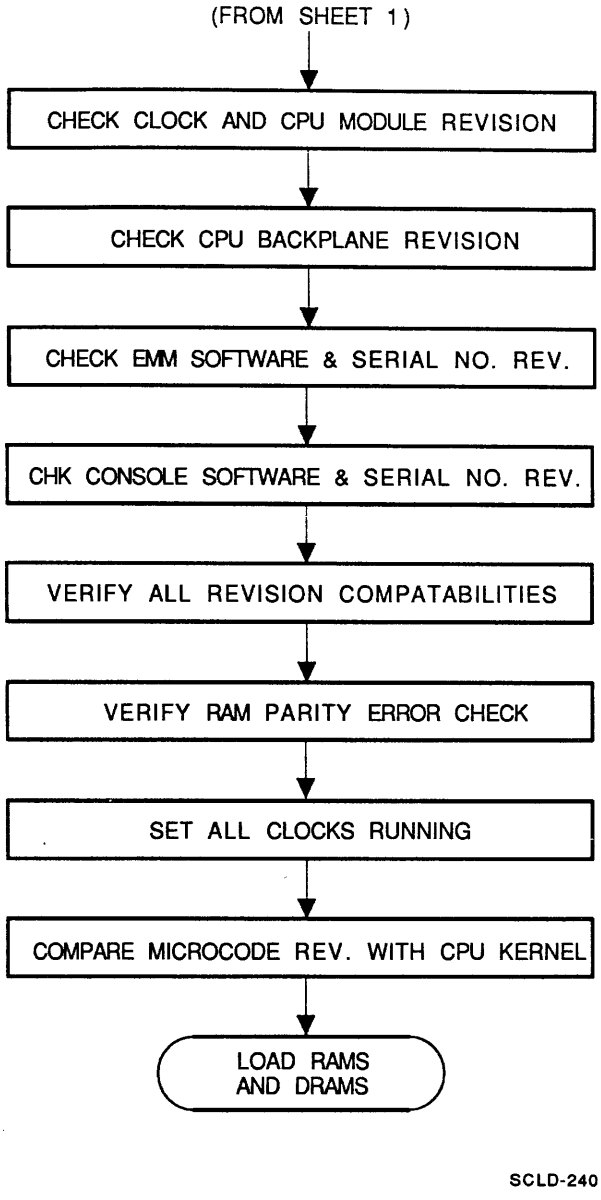


Figure 2-26 Test and Checkout Events (Sheet 2 of 2)

2.4.1.7 Load RAMs and DRAMs -- The console is responsible for loading the VAX 8800 microcode into the control store (CS2 - CS0) located on the WCS and SEQ modules, IBox decoder RAMs (DRAMs), and the NMI control store located on the CCS module.

The microcode to be loaded resides on the console Winchester disk and is loaded by means of the unbuffered data path to the DEC, SEQ, WCS, and CCS. The normal buffered data path through the IPRs cannot be used because of the lack of cooperating microcode.

Commands and data are transferred to each CPU through the use of a command load register and a data load register. Data written to either the command load or data load register is actually latched into the micromatch register on the decoder module of the IBox.

During the time that the RAMs are being loaded, clocks are running, stalls and traps are disabled, the NMI microsequencer is disabled, and CPU INIT is deasserted.

Parity error bits involved in loading the VAX 8800 RAMs can be read using the VBus.

Loading the Control Store

LOAD SEQUENCE

The sequence for loading the control store is:

```
    Write physical segment counts for CS2 - CS0
    Point VBus to parity error bits
+-> Write control store address
¶   Write control store data to the specified address (18 bytes)
¶   Check parity using the VBus
¶
¶                ¶
¶                ¶
+----REPEAT-------+
```

PHYSICAL SEGMENT COUNT

The physical segment count is the number of bytes to be loaded into each RAM address minus one. Because the number varies with the different RAMs loaded by the console, there are five segment counts that must be loaded (CS2, CS1, CSO, DRAM, and NMI CS).

The physical segment counts for loading the control store are:

Command Load Register = XXXX1100	MODE = Write CS0 PSEG Count
Data Load Register = CS0 PSEG Count	Currently equal to 5
Command Load Register = XXXX1101	MODE = Write CS1 PSEG Count
Data Load Register = CS1 PSEG Count	Currently equal to 5
Command Load Register = XXXX1110	MODE = Write CS2 PSEG Count
Data Load Register = CS2 PSEG Count	Currently equal to 5

RESETTING THE BYTE COUNTER

The RAM loading mechanism on the decoder module contains a counter that points to the byte currently being loaded. The counter must be reset prior to beginning the load process by writing XXXX1000(binary) to the command load register.

WRITE CONTROL STORE ADDRESS

An address of greater than 8 bits is loaded by means of the console bus in several slices. A slice ID field points to the destination of the bits being sent to the control store address latches. Address slices can be sent in any order.

The command load register is loaded with XXXX00001 to enter the SET CS ADDRESS mode. The CS address is loaded in three slices as follows:

Command Load Register = XXXX0001	Mode = Set CS Address
Data Load Register = 011AAAAA	AAAAA = CS Address <14:10>
Data Load Register = 010AAAAA	AAAAA = CS Address <9:5>
Data Load Register = 001AAAAA	AAAAA = CS Address <4:0>

WRITE CONTROL STORE DATA

The control store is partitioned into three sections (CS0, CS1, and CS2) and data for each microaddress is loaded into each of the sections in the following order:

First --- CS0
Second -- CS1
Third --- CS2

Each section is loaded in bytes, beginning with the most significant byte and ending with the least significant byte. The sequencer on the decoder module points to the byte currently being loaded.

The data load process begins by writing XXXX0101 to set the command load register mode, followed by 18 consecutive writes to the data load register. If a RAM data segment written to the data load register is less than 8 bits wide, the most significant bits are "don't care" bits.

Command Load Register = XXXX0101 Mode = Write CS Data

Data Load Register = Data for CS0 <47:40>
Data Load Register = Data for CS0 <39:32>
Data Load Register = Data for CS0 <31:24>
Data Load Register = Data for CS0 <23:16>
Data Load Register = Data for CS0 <15:08>
Data Load Register = Data for CS0 <07:00>

Data Load Register = Data for CS1 <95:88>
Data Load Register = Data for CS1 <87:80>
Data Load Register = Data for CS1 <79:72>
Data Load Register = Data for CS1 <71:64>
Data Load Register = Data for CS1 <63:56>
Data Load Register = Data for CS1 <55:48>

Data Load Register = Data for CS2 <142:136> -- (7 bits only)
Data Load Register = Data for CS2 <135:128>
Data Load Register = Data for CS2 <127:120>
Data Load Register = Data for CS2 <119:112>
Data Load Register = Data for CS2 <111:104>
Data Load Register = Data for CS2 <103:96>

CHECK PARITY

The VBus access register in the console interface contains the parity error bits for the NMI control store (VBA <3> from the CCS module), the decoder RAMs (VBA <2> from the DEC module), and the OR of the CS0, CS1, and CS2 (VBA <1> from the SEQ module). All of the parity error bits are selected by the VBus address 26(hex). The least significant digit of the VBus address is the last bit shifted into the CPU (refer to Figure 2-27).

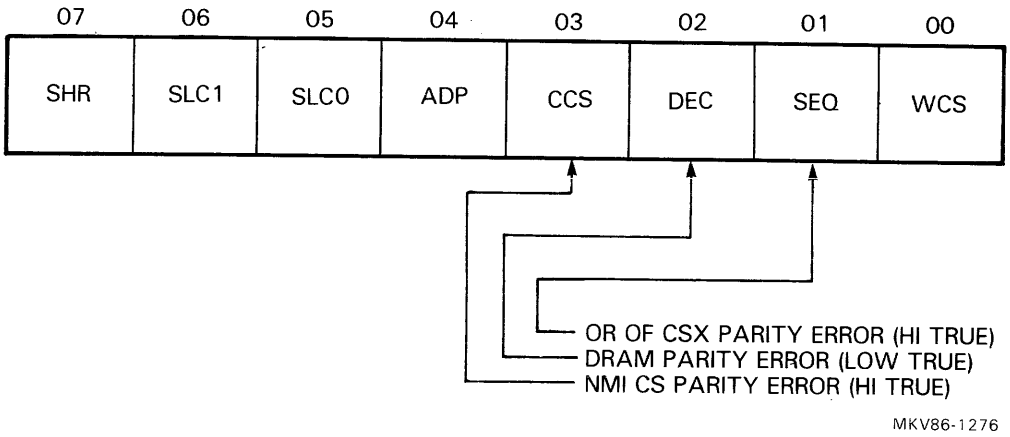


Figure 2-27 VBus Parity Bits

LOAD FUNCTION INACTIVE

When the control store data has been loaded, the loading mechanism on the decoder module must be left in the inactive state prior to starting the machine. This procedure is performed by writing XXXX0000 to the command load register.

Loading the Decoder RAMs

LOAD SEQUENCE

```
      Machine Executing Nonfunctional Loop

      Hardware initialization (CPU INIT)
      Write DRAM physical segment count
      Point VBus to parity error bits
      Reset byte counter

 +--> Write DRAM address
 ¶    Write DRAM data (3 bytes)
 ¶    Check parity using VBus
 ¶                ¶
 +-----------------+

      Zero the load address
      Set load function inactive
```

MACHINE EXECUTING NONFUNCTIONAL LOOP

The decoder RAM address is the OR of the load address from the console and the op code address field from the decoder. When one of these two inputs is used, the other must be zero. Prior to loading the DRAMs, a hardware initialization must be accomplished to force the op code address field to zero.

When the address field is to remain zero, the machine must be operating in a nonfunctional loop while the decoder RAMs are being loaded. This creates the requirement that the one microinstruction necessary to execute the loop be loaded previously into the control store.

PHYSICAL SEGMENT COUNT

Command Load Register = XXXX1011	Mode = Write DRAM PSEG Count
Data Load Register = DRAM PSEG Count	Currently equal to 2

WRITE DRAM ADDRESS (Figure 2-28)

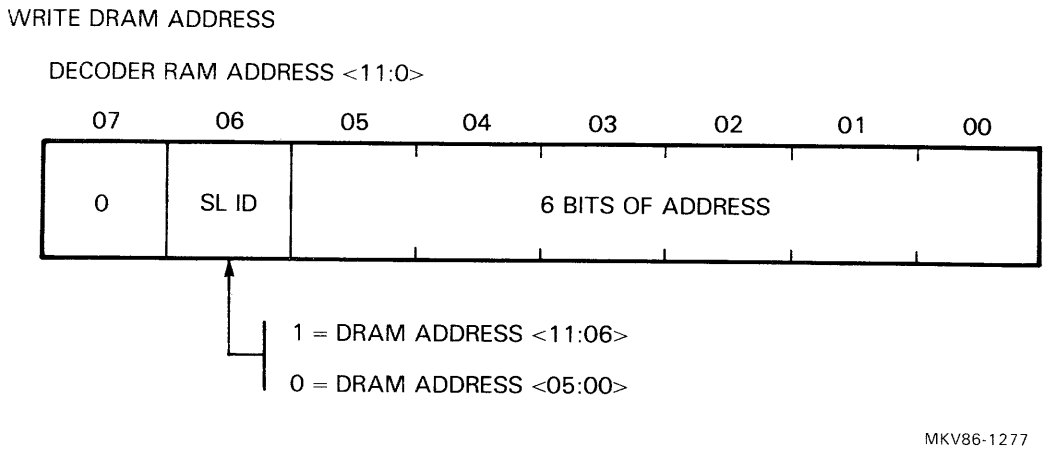


Figure 2-28 DRAM Address

The command load register is loaded with XXXX0010 to enter the SET DRAM ADDRESS mode and the address is loaded in two slices.

Command Load Register = XXXX0010 Mode = Set DRAM Address

Data Load Register = 01AAAAAA AAAAAA = DRAM Address <11:6>
Data Load Register = 00AAAAAA AAAAAA = DRAM Address <5:0>

WRITE DRAM DATA

The command load register is loaded with XXXX0110 to enter the SET DRAM DATA mode and the data is loaded in three consecutive slices.

Command Load Register = XXXX0110 Mode = Set DRAM Data

Data Load Register = Data for DRAM <16:12>
Data Load Register = Data for DRAM <11:06>
Data Load Register = Data for DRAM <05:00>

CLEAR THE LOAD ADDRESS

Command Load Register = XXXX0010	Mode = Set DRAM Address
Data Load Register = 01000000	DRAM Address <11:06> = 0
Data Load Register = 00000000	DRAM Address <05:00> = 0

LOAD FUNCTION INACTIVE

When the DRAM data has been loaded, the loading mechanism on the decoder module must be left in the inactive state prior to starting the machine. This procedure is performed by writing XXXX0000 to the command load register.

Loading the NMI Control Store

LOAD SEQUENCE

```
       Write NMI CS physical segment count
       Point VBus to parity error bits
       Reset byte counter

  +--> Write NMI CS address
  ¶    Write NMI CS data (4 bytes)
  ¶    Check parity using VBus
  ¶                ¶
  +----------------+
       Zero the load address
       Set load function inactive
```

PHYSICAL SEGMENT COUNT

Command Load Register = XXXX1111 Mode = Write NMI CS PSEG Count
Data Load Register = NMI CS PSEG Count Currently equal to 3

WRITE NMI CS ADDRESS (Figure 2-29)

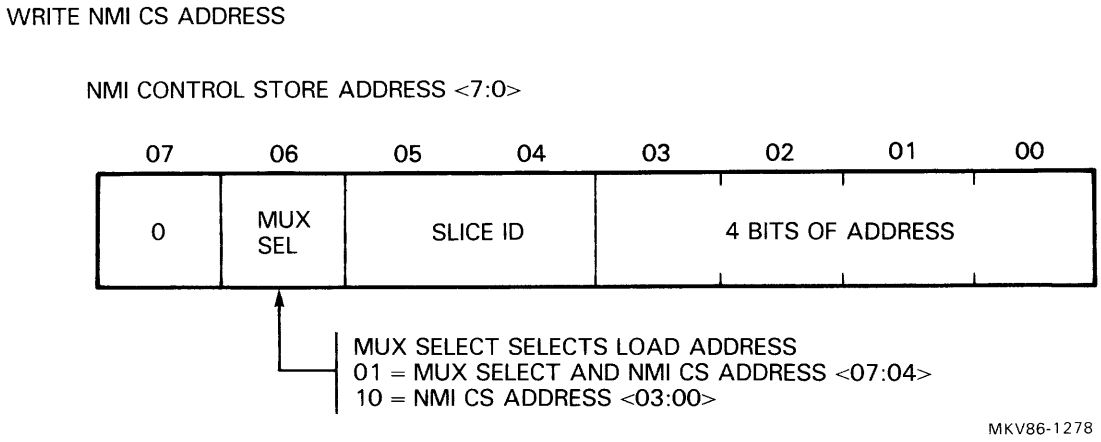


Figure 2-29 NMI Control Store Address

The command load register is loaded with XXXX0011 to enter the SET NMI CS ADDRESS mode and the address is loaded in two slices.

```
Command Load Register = XXXX0011     Mode = Set NMI CS Address

Data Load Register = 0101AAAA        AAAA = NMI CS Address <7:4>
Data Load Register = 0110AAAA        AAAA = DRAM Address <3:0>
```

WRITE NMI CS DATA

The command load register is loaded with XXXX0111 to enter the SET NMI CS WRITE DATA mode and the data is loaded in four consecutive slices.

```
Command Load Register = XXXX0111     Mode = Write NMI CS Data

Data Load Register = Data for NMI CS <27:24>
Data Load Register = Data for NMI CS <23:16>
Data Load Register = Data for NMI CS <15:08>
Data Load Register = Data for NMI CS <07:00>
```

CLEAR THE LOAD ADDRESS PATH

```
Command Load Register = XXXX0011     Mode = Set NMI CS Address

Data Load Register = 0001XXXX        MUX SELECT = 0
```

LOAD FUNCTION INACTIVE

When the NMI control store data has been loaded, the loading mechanism on the decoder module must be left in the inactive state prior to starting the machine. This procedure is performed by writing XXXX0000 to the command load register.

RAM/DRAM Load Circuitry

Figure 2-30 shows a simplified block diagram of the data path used in the transfer process. The gateway control (GWYC) on the decoder module functions as a distribution device, and routes the RAM data to the proper location. The console communicates with the GWYC by sending commands and data over the 8-bit bidirectional bus connecting the console interface with the decoder module. Commands must be written to the command load register, and data and addresses to the data load register for the appropriate CPU.

The flag and strobe signals used by the GWYC to load the RAMs are generated by the CSEQ MCA on the console interface.

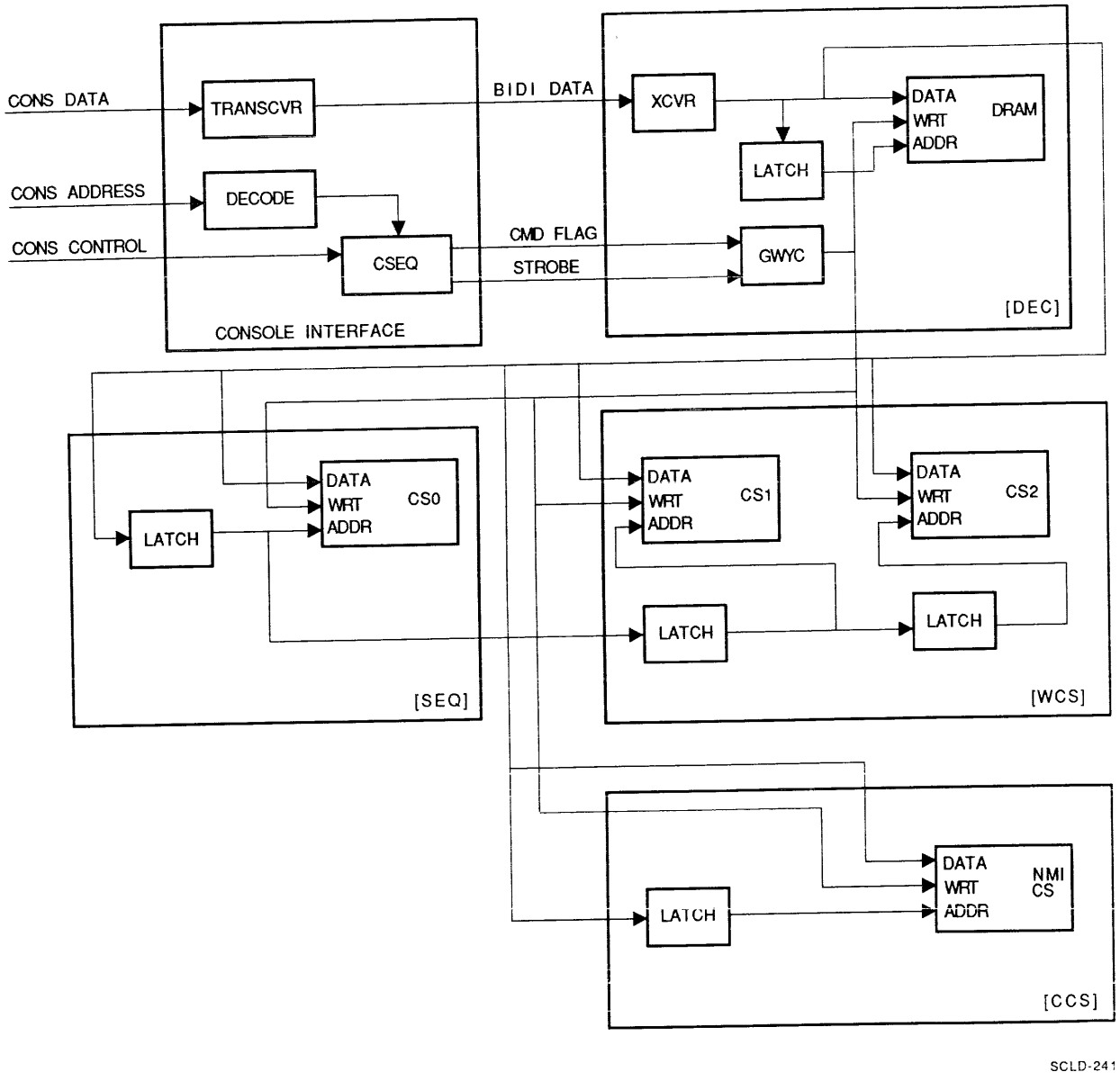


Figure 2-30 RAM Loading Simplified Block Diagram

Figure 2-31 identifies the sequence of events that occur during the process of loading the RAMs and DRAMs.

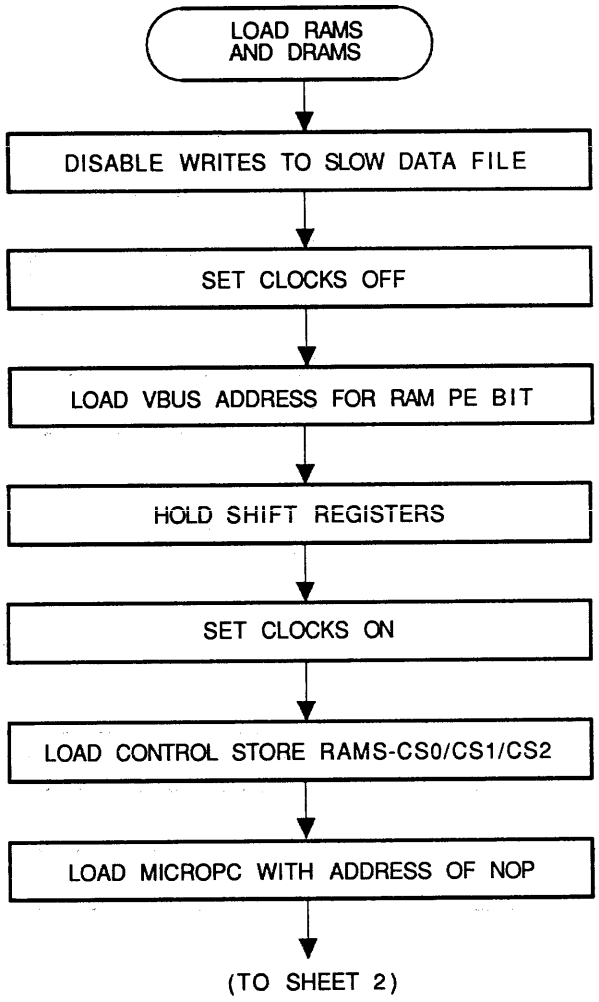


Figure 2-31 RAM/DRAM Loading Events (Sheet 1 of 5)

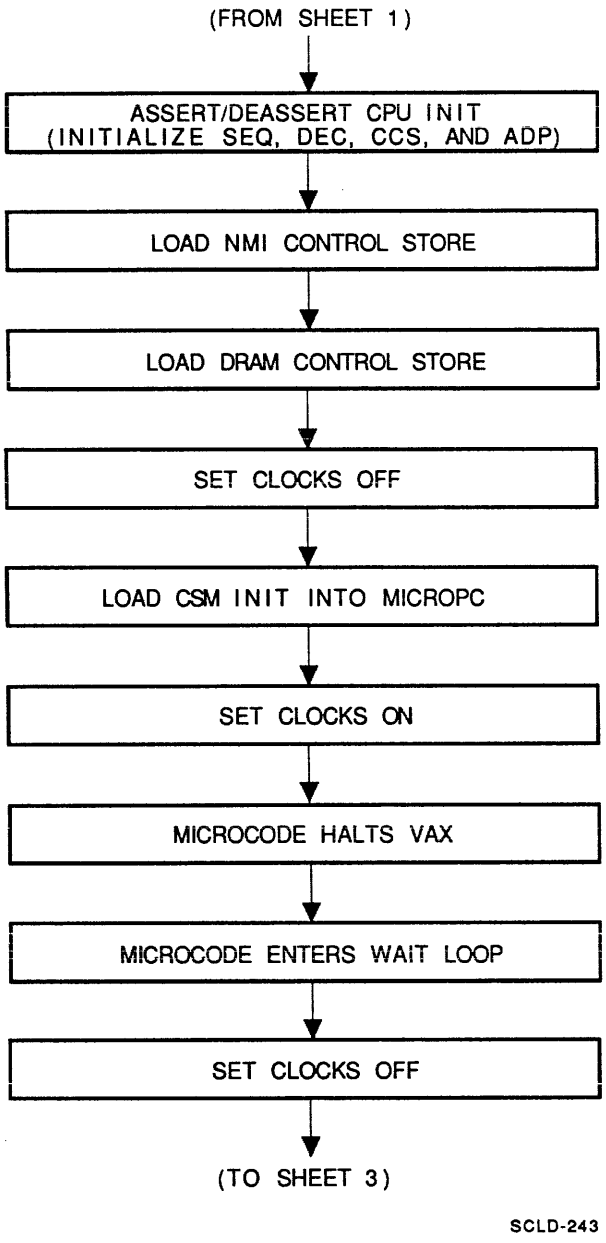


Figure 2-31 RAM/DRAM Loading Events (Sheet 2 of 5)

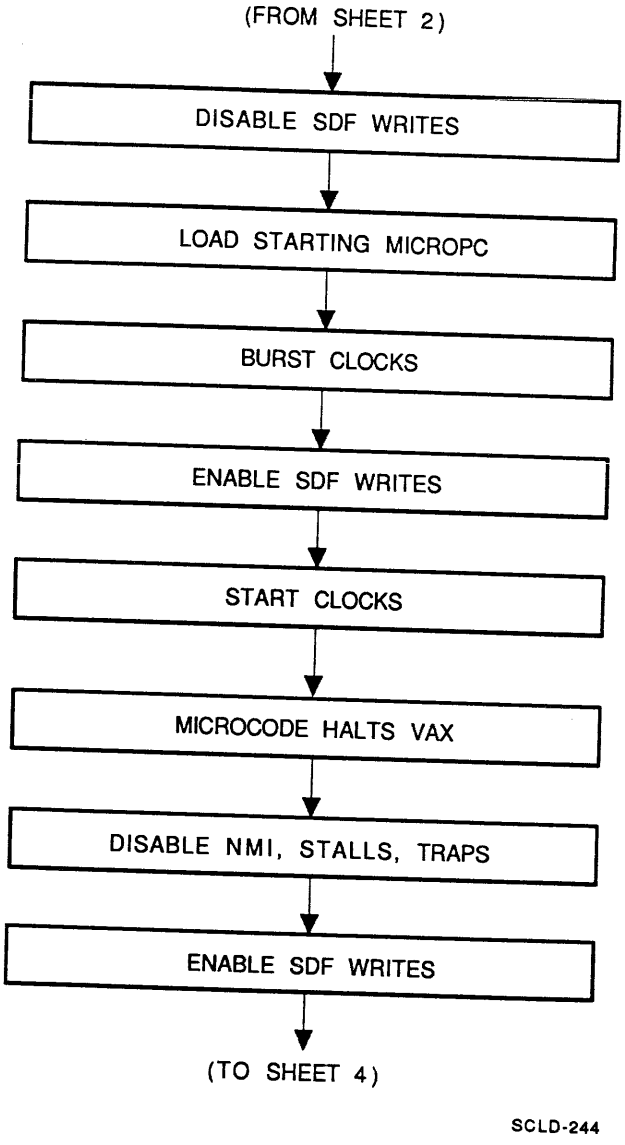


Figure 2-31 RAM/DRAM Loading Events (Sheet 3 of 5)

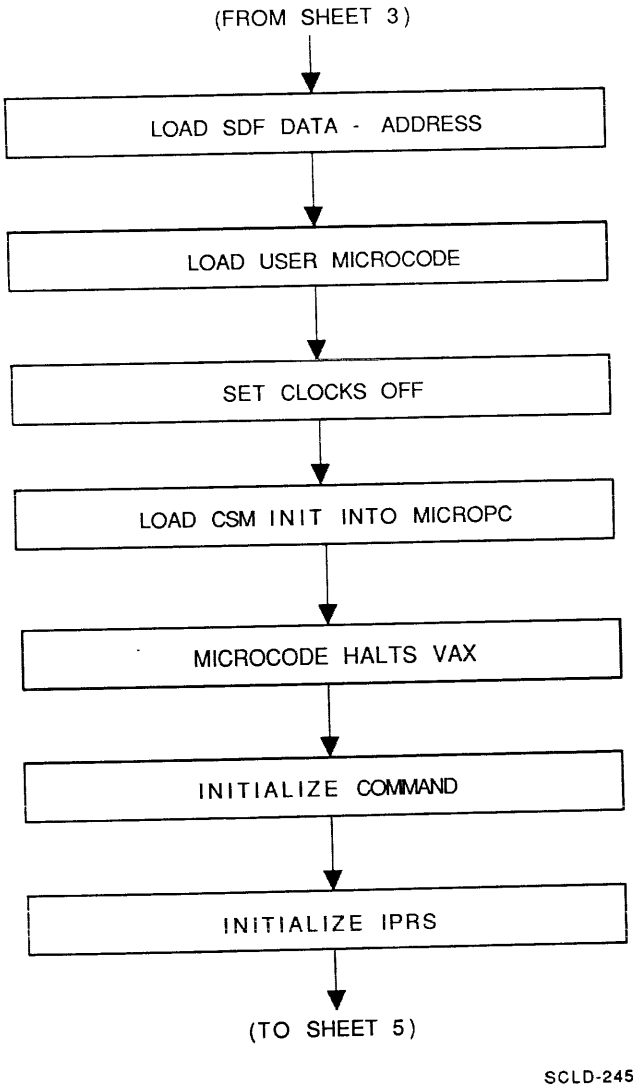


Figure 2-31 RAM/DRAM Loading Events (Sheet 4 of 5)

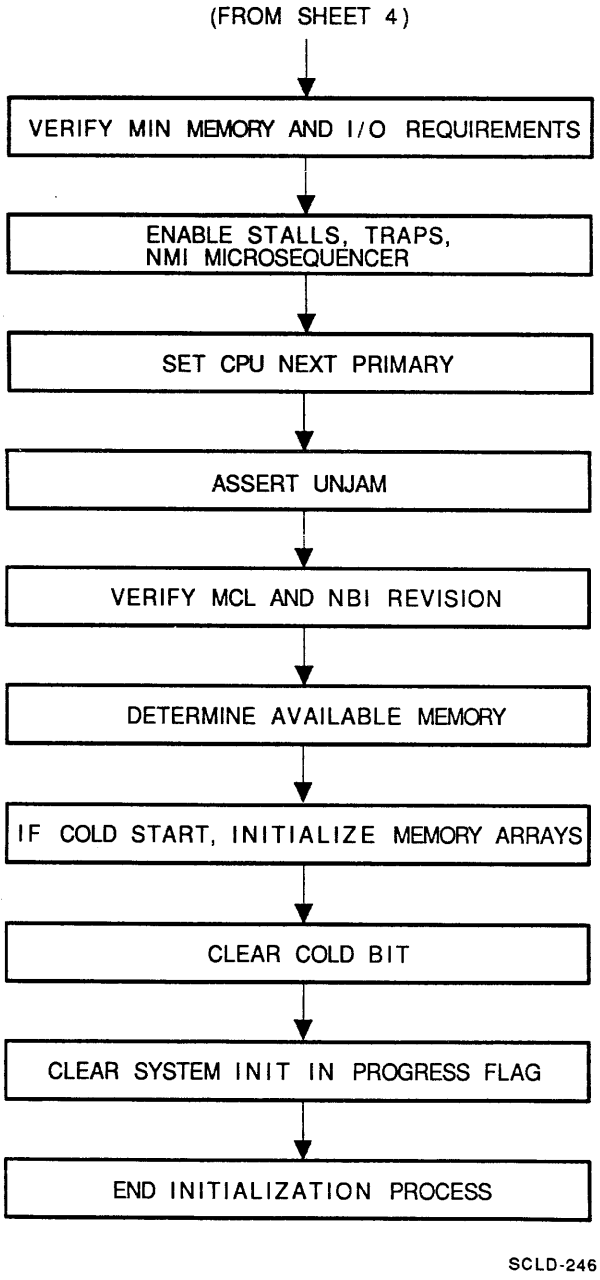


Figure 2-31 RAM/DRAM Loading Events (Sheet 5 of 5)

Table 2-9 lists some of the key hardware signals and functions used during the initialization process, and identifies the console interface logic that performs the process. These signals and functions are described in more detail in the discussion of the individual logic areas identified.

Table 2-9 Key Initialization Signal/Functions

Signal/Function	Console Logic	Comments
ACLO/DCLO	Synchronizer	Received from EMM -- Translated and clock synced. Initializes console interface registers and sends power-fail interrupt to VAX 8800 CPU.
CPU INIT	Control Reg 0	Created in control register 0 from console data (software command) or forced by assertion of DCLO.
UNLOCK CSEQ	Console Sequencer	Generated by serial data sequence from port B bit 5. Enables the console sequencer for communication with the console.
DISABLE SDF	Control Reg 2	Disables write to slow data file in VAX 8800 CPU.
RAM STROBE COMMAND FLAG	Console Sequencer	Strobe signal and command/-data flag to the VAX 8800 CPU decoders.
DISABLE STALLS, TRAPS, NMI	Control Reg 2	Disable signal to minimize undesirable events (stalls, traps, NMI traffic) during initialization.

2.4.2 VAX 8800 CPU Control

Read/write control of data transfers, and setup of the VAX 8800 CPUs is performed by the console sequencer and the three control registers in the console interface.

Console sequencer functions:

- Console read/write sequencing
- Console interrupt generation
- RAM strobe and flag generation
- Console/CPU isolate and disable

Control register CPU control signals:

Control register 0

- Disable traps
- Disable stalls
- Disable NMI
- CPU init

Control register 1

- NMI UNJAM
- Enable receive interrupts to console
- Enable slow mode
- Enable breakpoint trap
- Set CPU halt

Control register 2

- Disable slow data file writes
- Set CPU timeout

2.4.2.1 Console Sequencer -- Communications between the console and the console interface requires handshaking signals and synchronization of the data transfers to the VAX 8800 CPU. Figure 2-32 shows a simplified drawing of the key signals used in the data transfers.

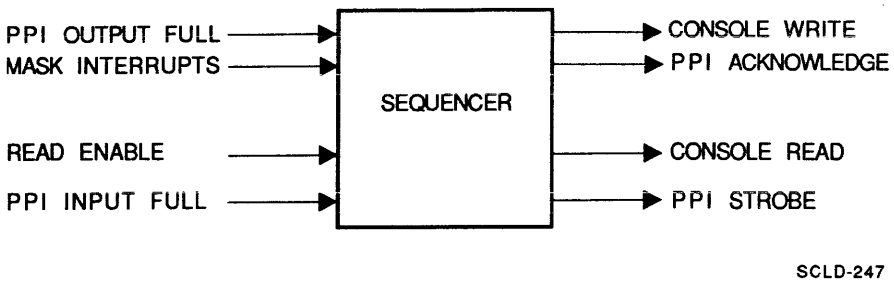


Figure 2-32 Console/Interface Timing Signals

Read/Write Sequencing

The console writes data to the interface by writing the address of the interface or clock location to the port "B" register of the PPI. The handshaking signals associated with this operation are PPI OUTPUT FULL and PPI ACKNOWLEDGE. PPI output full informs the interface that data is now available from the console. PPI acknowledge enables the output buffer of the PPI so that the data can be transferred. Figure 2-33 shows a simplified drawing of the write sequence.

When the interface receives the PPI OUTPUT FULL signal, it responds by asserting PPI ACKNOWLEDGE. The ACKNOWLEDGE signal is held asserted for one to two microseconds. The console sequencer generates the CONSOLE WRITE term for use in the data transfer process. If interrupts are enabled, the console will receive an interrupt from the PPI at the completion of the data transfer.

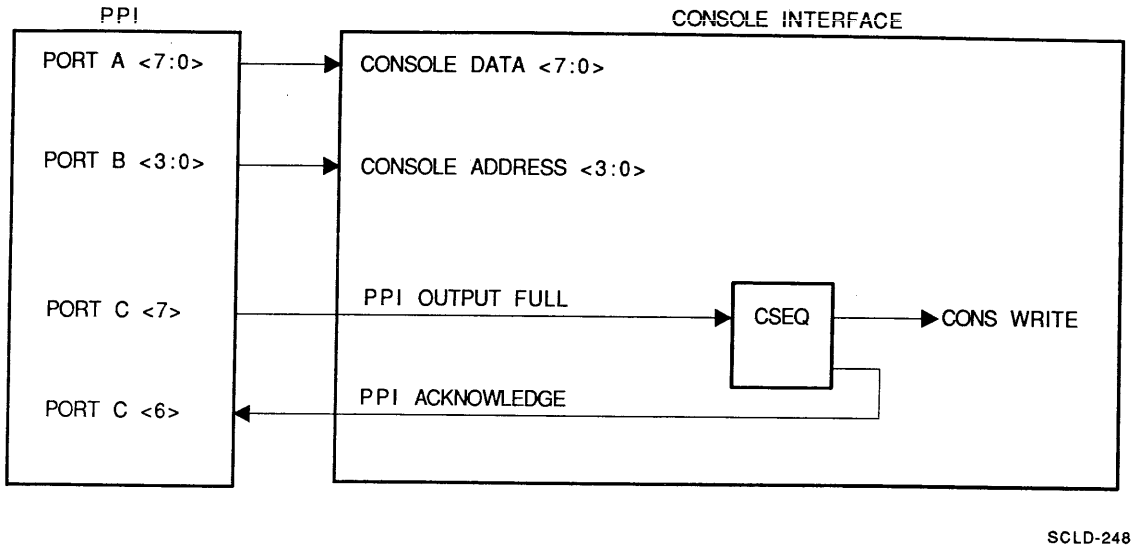


Figure 2-33 Write Sequence

The console sets up the read sequence by writing the address of the location to be read to the interface by means of PPI port B, and asserting READ ENABLE (PPI port B <4>). The address selects the desired mux input to be placed on the data bus to the console.

Read enable initiates a read sequence by sending the expected handshaking signals to the PPI and enabling the bidirectional bus to the console with the appropriate timing. Figure 2-34 shows the read sequence setup.

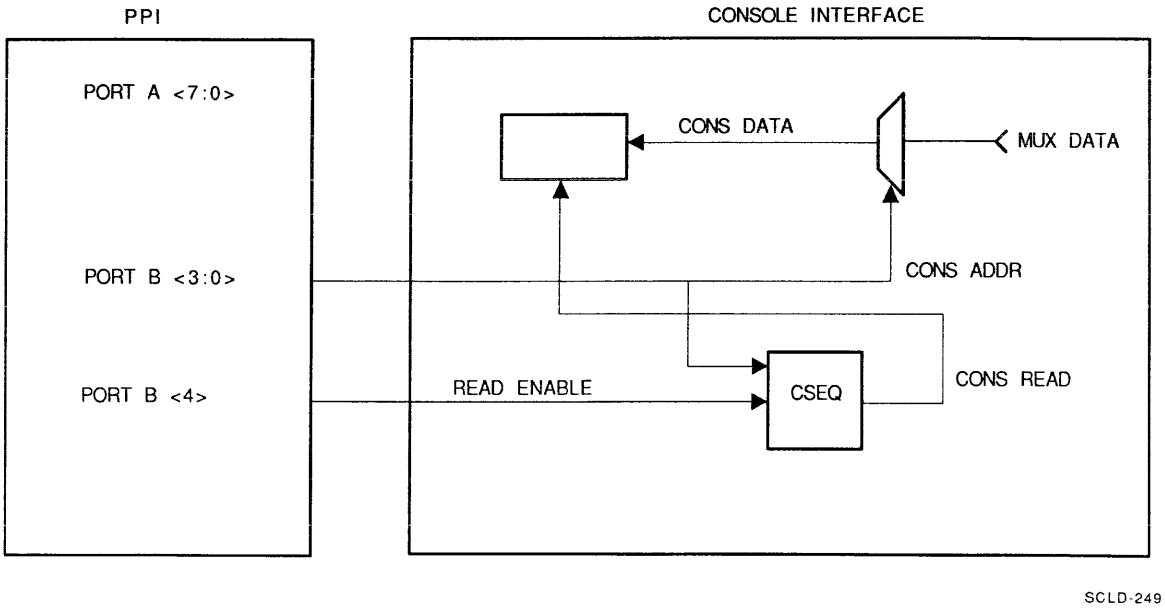


Figure 2-34 Read Sequence (Setup)

The control signals used in the data transfer process are the PPI STROBE and PPI INPUT FULL. The STROBE latches the data into the PPI port A buffer, and the PPI INPUT FULL informs the interface that the buffer is full and will not accept more data until the buffer has been cleared. Figure 2-35 shows the control signals used during the transfer process.

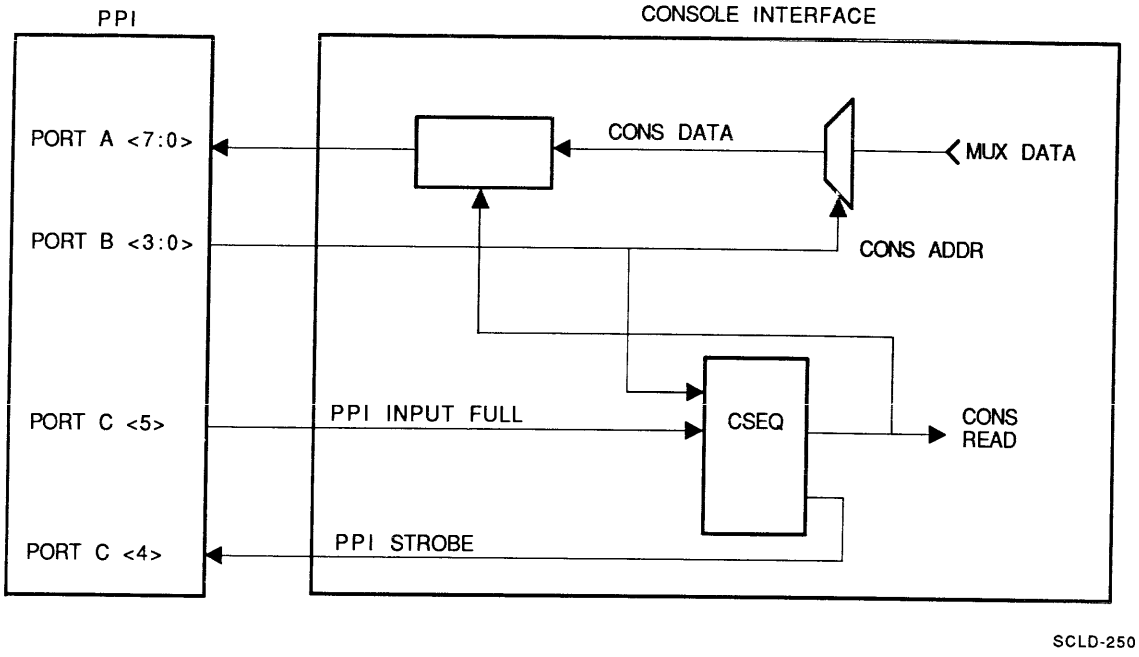


Figure 2-35 Read Sequence (Data Out)

2.4.2.2 Control Registers -- The control registers receive console data from the PPI bus that is used in the control registers to enable or disable the CPU functions. Figures 2-36 through 2-38 show simplified block diagrams of the inputs and outputs of each register, as well as the destination of each output signal.

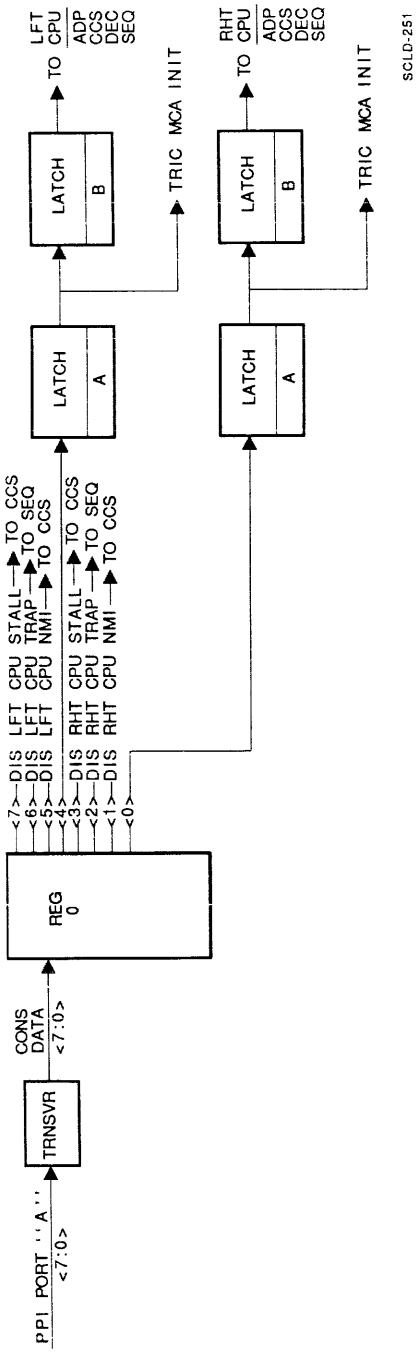


Figure 2-36 Simplified Diagram of Control Register 0

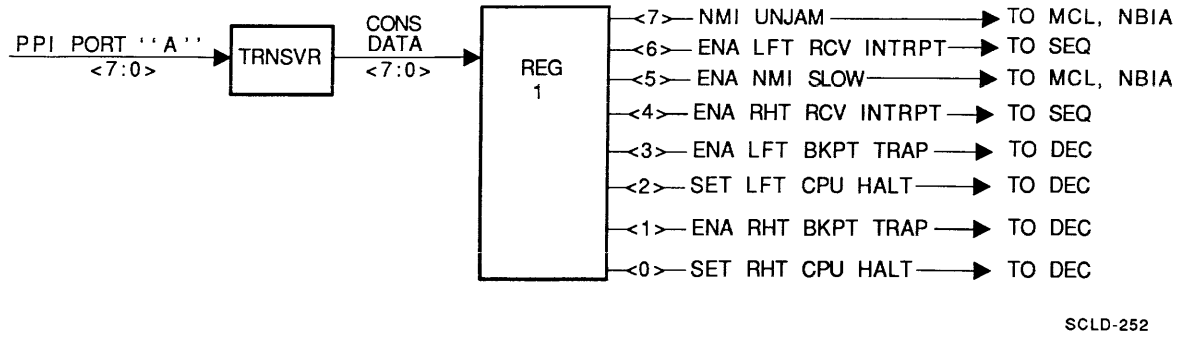


Figure 2-37 Simplified Diagram of Control Register 1

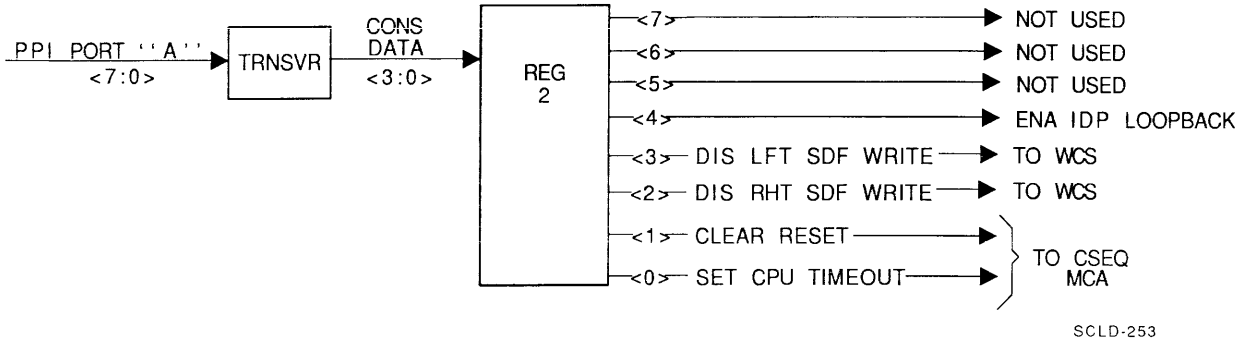


Figure 2-38 Simplified Diagram of Control Register 2

2.4.3 Data Transfers

The receive data buffer (RXDB) and transmit data buffer (TXDB) are used in the transfer of data between the console and the VAX 8800 CPUs. The low byte of the registers contains the information being transferred, and the high byte identifies the use of the data being transferred.

When the console has data to transfer to the CPU, the interrupt status register must be examined in order to verify that the last data transfer has been completed and the RXDB is empty and ready to accept data to be transferred. If the RXDB is empty, the console writes two bytes to the RXDB by means of PPI port A. The hardware in the TRIC MCA generates an interrupt to the CPU if interrupts are enabled. If interrupts are not enabled, the CPU can poll the DONE bit by reading the RXCS.

The console must wait for the deassertion of the DONE signal before writing to the RXDB again. If RXDB interrupts to the console are enabled, the deassertion of DONE will interrupt the console.

The VAX 8800 CPU responds to the interrupt, reads the data from the RXDB, and hardware in the TRIC MCA clears DONE. Parity checks of data transfers between the interface and the CPU are performed and monitored by the CPU. Figure 2-39 shows a simplified drawing of the transfer process.

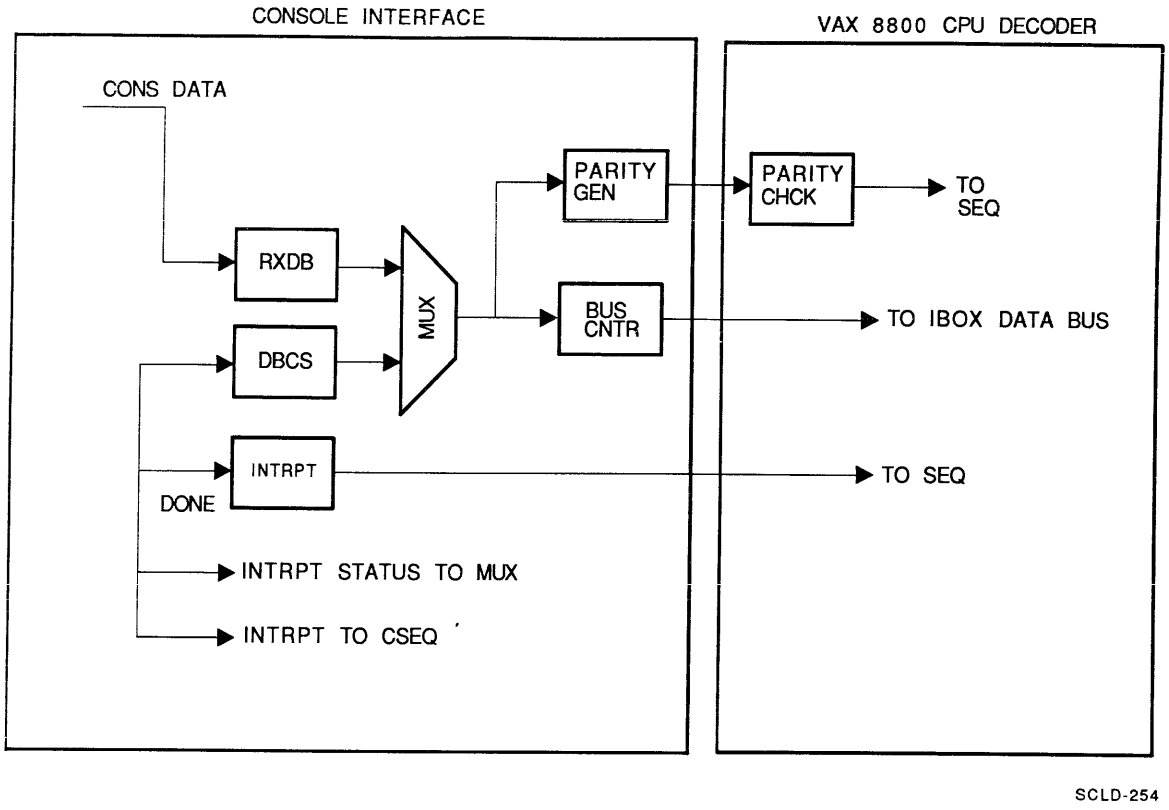


Figure 2-39 Data Transfer Console Interface-to-VAX 8800 CPU

Data transfers from the VAX 8800 CPU to the console interface are performed under the control of the microcode. The microcode monitors the READY signal to determine if the transmit data buffer (TXDB) has been cleared or if it still contains the last data word sent. When the console receives an interrupt (deassertion of READY) informing it that there is data in the TXDB, the console initiates a TXDB read and the TRIC MCA sets READY for another CPU write. Figure 2-40 shows a simplified drawing of the CPU-to-interface data transfer.

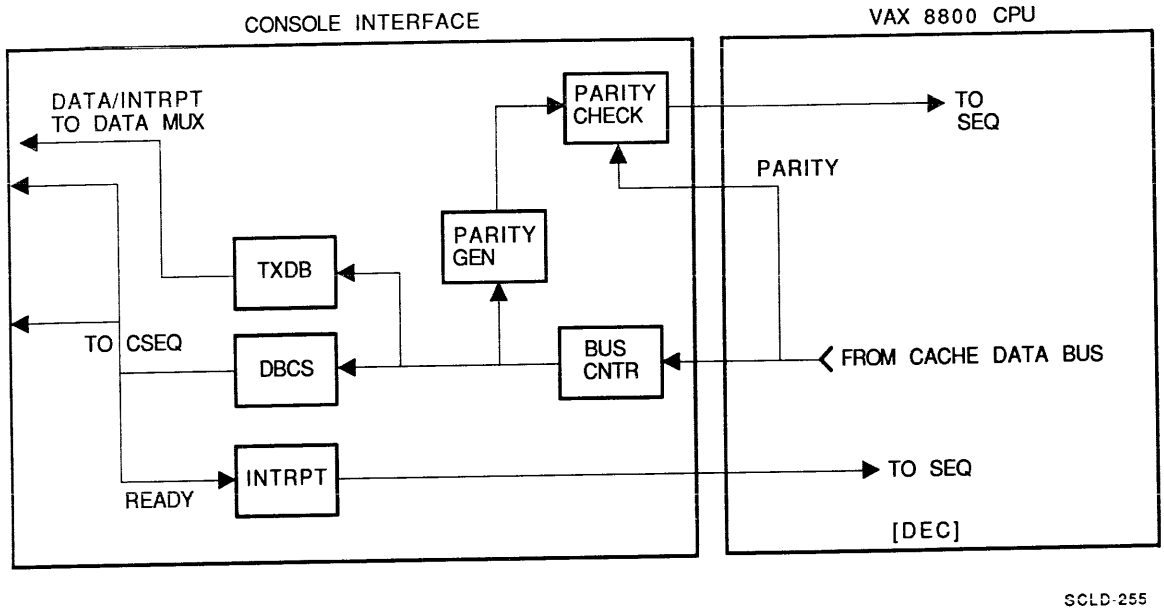


Figure 2-40 Data Transfer VAX 8800 CPU-to-Console Interface

2.5 CONSOLE/VAX 8800 CLOCKS AND TIMING

The Console Interface contains a 1 MHz clock, an interval timer, and a timeout circuit that is used by the VAX 8800 system. The 1-MHz clock is the basic timing source for the interval timers, and is also used during data transfers to the PPI as a timing control signal. The VAX-11 interval clock is a 32-bit timer used by the operating system and the diagnostic software to time events and generate interrupts. Timeout circuits monitor the VAX 8800 CPUs during the program mode.

2.5.1 One-MHz Clock

The function of the 1-MHz clock logic shown in Figure 2-41 is to provide a constant clock frequency to the TRIC MCA interval clocks and a timing source for PPI data transfers. A functional description of the basic VAX 8800 clock is provided in Section V of this manual.

The EN ICLOCK INC signal is asserted once each microsecond and remains asserted from one VAX 8800 CPU clock cycle. EN ICLOCK INC is derived from the 7-bit MOD N counter of the clock module and applied to the terminal register interval clock on the TRIC MCA.

The MSB of the MOD N counter is phase-locked with the constant frequency of the crystal oscillator and produces an output (interval sync) that is equal to (N) times the 250-kHz oscillator. The value of (N) is held constant in the clock period register and is used as a comparator reference in the generation of the 1-MHz clock signal.

The comparator reference logic uses the N clock period to produce three reference values; 3N/4, 2N/4, and N/4. The three reference values and zero are used as a comparison against the changing interval sync value and will create the EN ICLOCK INC whenever the comparison is found. The resultant signal is generated at a rate four times the oscillator frequency (250 kHz) to produce the 1-MHz clock regardless of the value of N.

The latched 1-MHz ENABLE signal is used on the CSEQ MCA in the generation of the following PPI handshaking signals:

- PPI STROBE
- CONSOLE READ
- PPI ACKNOWLEDGE
- CONSOLE WRITE
- RAM STROBE
- COMMAND FLAG

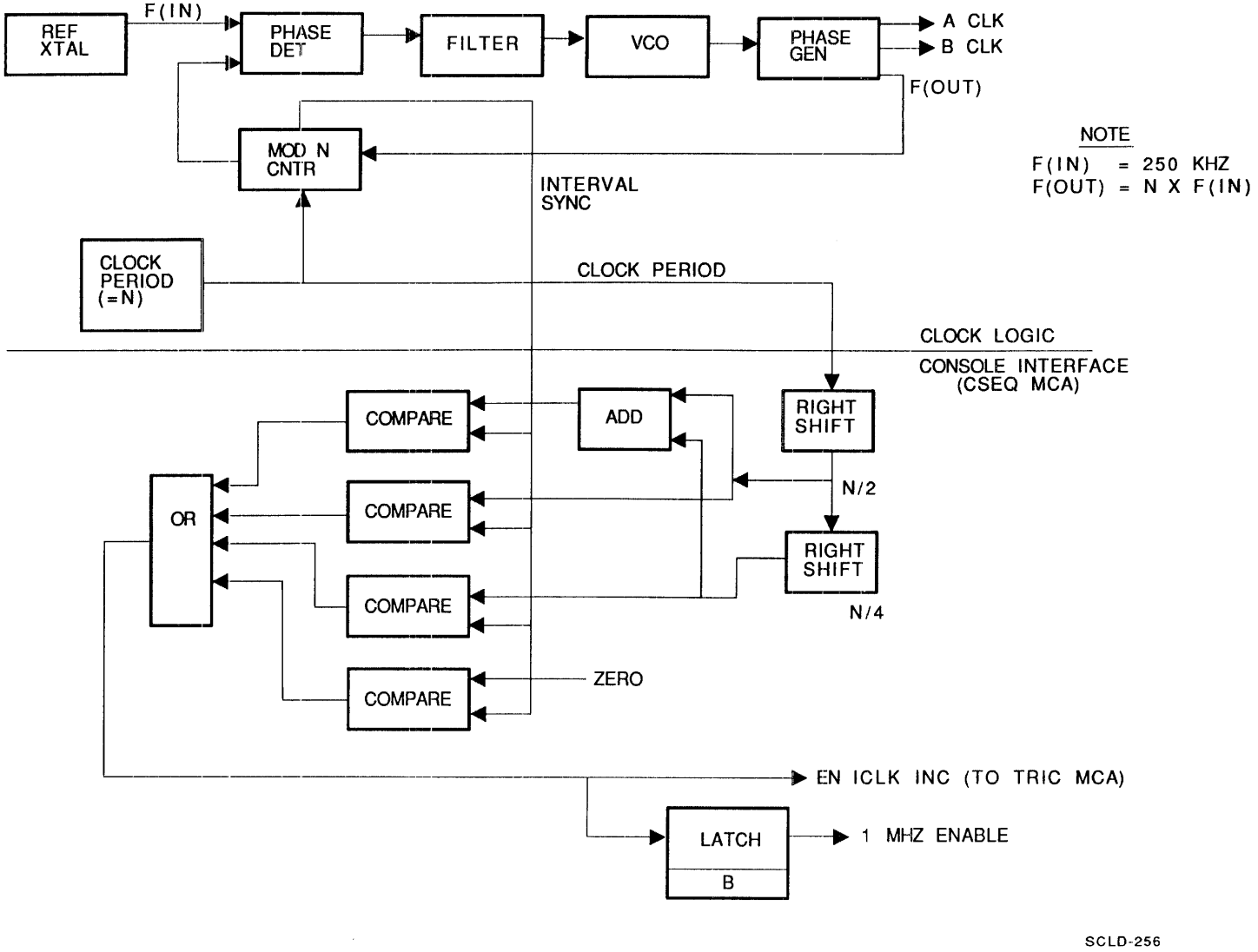


Figure 2-41 1 MHz Clock Generation

2.5.2 Interval Clock

Each VAX 8800 CPU has an interval clock residing on the console interface. The clock logic and the three registers that provide control and data transfer are evenly sliced, and located on two identical MCAs. Figure 2-42 shows the bit configuration of the registers, and Figure 2-43 is a simplified block diagram of the interval clock.

The three registers that are used for the interval clocks are:

1. Interval Count Register (ICR)
2. Next Interval Clock Register (NICR)
3. Interval Count Control/Status (ICCS)

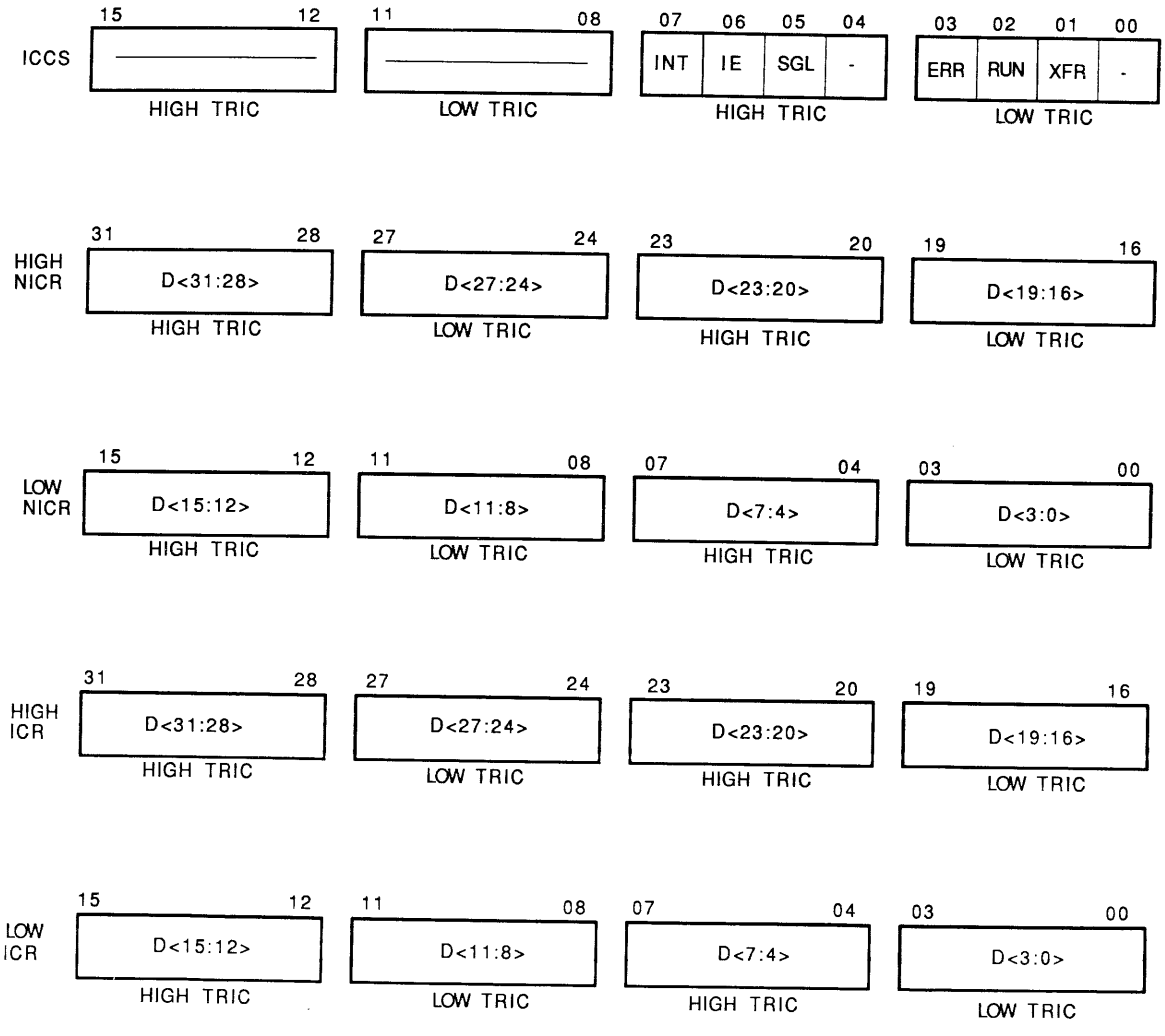


Figure 2-42 Interval Clock Registers Bit Configuration

The interval timer provides a method for accurately measuring variable time intervals and enables the software to perform time-dependent events. The timer informs the CPU of a completed interval by means of an interrupt generated by the overflow of the interval count register being incremented to a full count.

The interval count register (ICR) is a 32-bit counter that increments at the rate of one microsecond per count. The incrementing pulse (EN ICLOCK INC) is supplied by the 1-MHz clock on the CSEQ. The next interval count register (NICR) is a 32-bit register containing the value to be loaded into the ICR each time the ICR overflows. Control and status is provided by the interval clock control and status register (ICCS). Table 2-10 shows the bit configuration of the ICCS.

Table 2-10 ICCS Bit Configuration

Bit	Name	Function
<07>	INT (Interrupt Request)	Set when the ICR overflows. Interrupts the CPU if interrupts are enabled. Cleared by writing 1 to this bit.
<06>	IE (Interrupt Enable)	Allows interrupt to occur at ICR overflow.
<05>	SGL (Single)	Used for maintenance functions. When set, the count register will increment by one. Set for one cycle only.
<03>	ERR (Error)	Indicates that a second overflow interrupt has been requested before the first is serviced. Cleared by writing 1 to this bit.
<02>	RUN	Provides the function of counter enable. Cleared on powerup and initialize.
<01>	XFR (Transfer)	Transfers the contents of the NICR to the ICR. Set for one cycle only.

The procedure for using the timer requires loading the NICR with a value corresponding to the two's complement of the number of microseconds between interrupts and setting the run, interrupt enable, and transfer bits in the ICCS. The NICR will be loaded into the ICR and the ICR will begin counting. When the ICR makes the transition from all ones to zero, an interrupt is generated and the next value is loaded from the NICR. If a transition is again reached before the last interrupt is serviced, the error flag is set and counting continues.

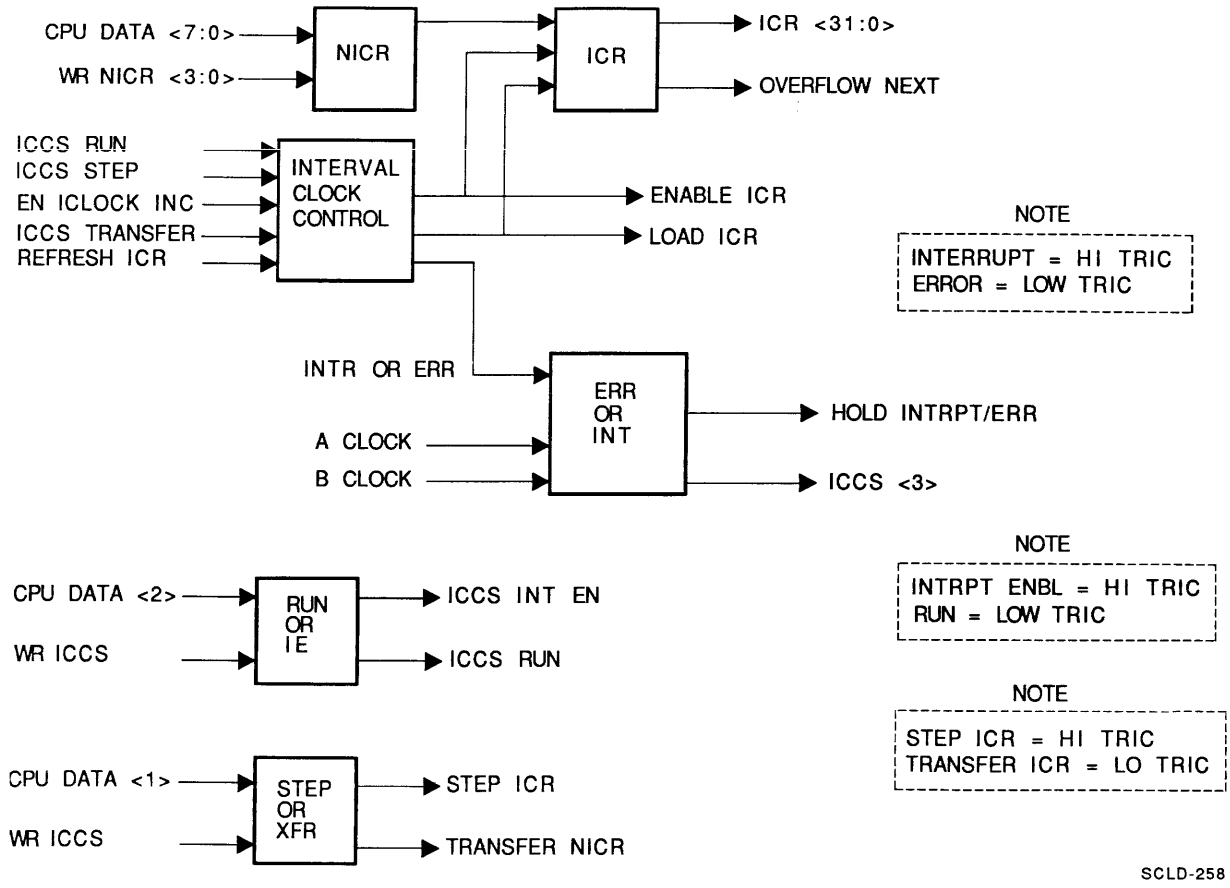


Figure 2-43 Interval Clock Simplified Block Diagram

2.5.3 CPU Timeouts

The console interface monitors the VAX 8800 system for a possible CPU hang. A timeout flag is maintained for each CPU and is checked periodically by the console software. Figure 2-44 shows a simplified block diagram of the timeout logic on the console interface. Figure 2-45 shows the bit configuration of the read-only clock status/timeout register (CST).

The timeout flag is set by a signal from control register 2, which is set by console software. The flag is cleared when the CPU hardware releases a stall. Time detects a CPU infinite stall.

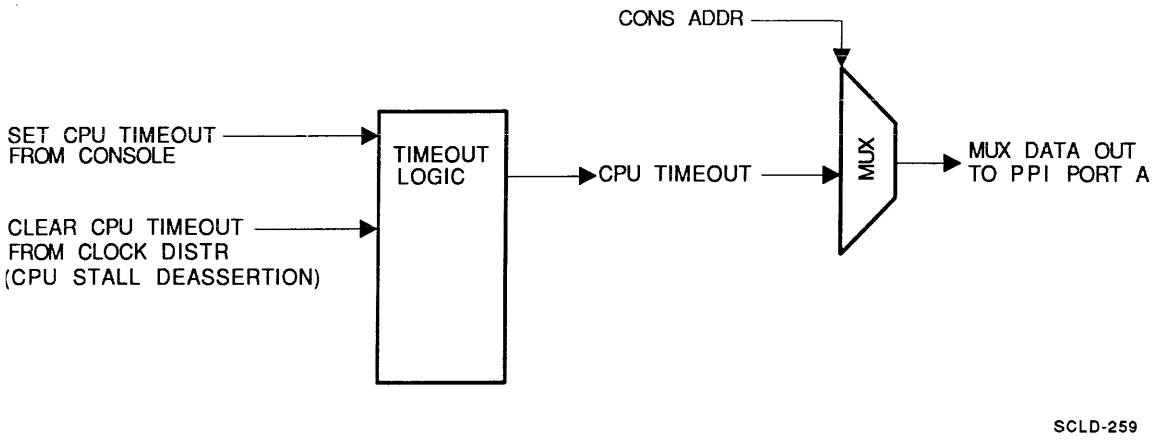


Figure 2-44 Simplified CPU Timeout

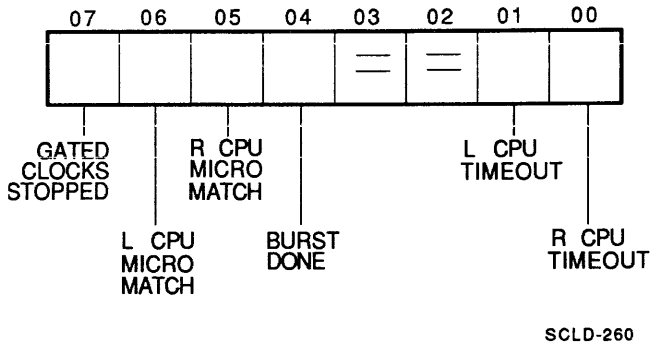


Figure 2-45 Clock Status and Timeout Register (CST)

2.5.4 Visibility Bus

The console uses the VBus control/access logic on the console interface to address and read VBus data in the CPU. Figure 2-46 shows the signals involved in the reading of the VBus information. Figure 2-47 shows a simplified block diagram of the VBus logic.

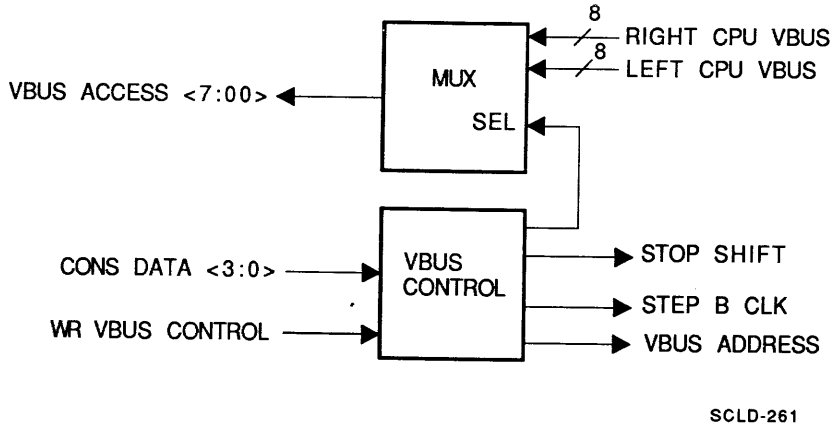


Figure 2-46 VBus Control/Data Signals

Bit 3 of the VBus control register is the mux select bit and selects either the left or right CPU inputs to the console. The stop shift and VBus address are used in the process of selecting which data is applied to the console interface. The address bit supplies the input to the serial shift register and is clocked by the STEP B signal. Stop shift freezes the address to prevent subsequent B clocks from shifting the address.

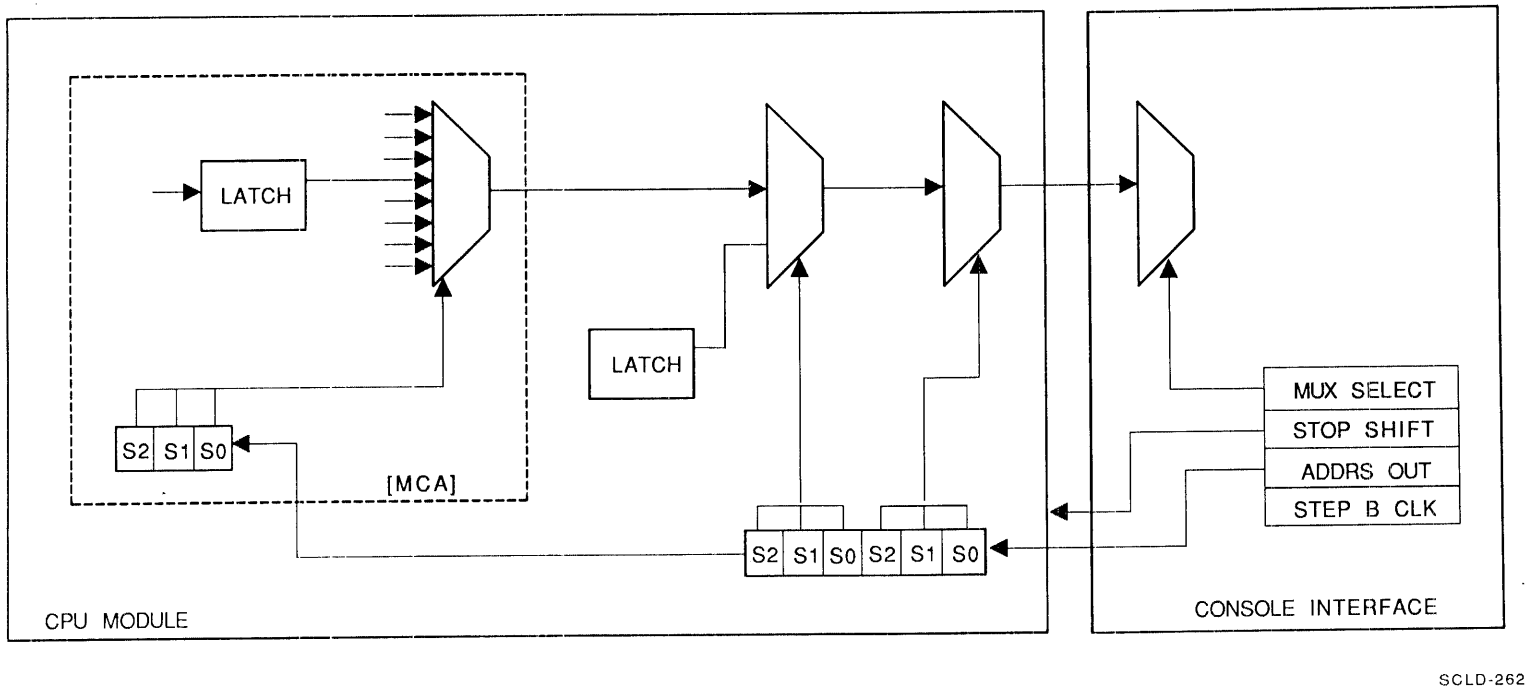


Figure 2-47 VBus Logic Simplified Block Diagram

3.1 GENERAL

This chapter contains additional detailed information concerning specific areas of the console subsystem. Included in this chapter are:

- Diagrams and descriptions of the CSEQ and TRIC MCAs
- Console register summary
- Console subsystem cabling

3.2 TERMINAL REGISTER INTERVAL CLOCK (TRIC)

The console interface contains four 19-0TRIC-00 TRIC MCAs to implement the transfer of data and status between the console and the VAX 8800 system. Despite being identical MCAs, the TRICs have minor variations in wiring in order to facilitate the high and low slicing. Figure 3-1 shows a block diagram of a single TRIC MCA. Figures 3-2 and 3-3 show the pin layout and body drawing of the TRIC MCA. Table 3-1 lists each TRIC MCA pin and identifies the signal assigned. Table 3-2 lists some of the key signals on the TRIC MCA and provides a description of the signal functions.

Data to be transferred from the console to the VAX 8800 CPU is received from the interface TTL-to-ECL translators as CONS DATA. CONS DATA is applied to both the receive data buffer (RXDB) and the bidirectional bus output mux. The unbuffered data path directly to the output mux is used for the transfer of microcode during the initialization process.

The normal path for data transfers is by means of the receive data buffer. The SELECT RXDB and WRITE ENABLE signals enable the loading of the RXDB. The 2-byte output of the receive data buffer (RXDB DATA) is applied to the data mux along with the control/status (DBCS DATA) information and the interval clock outputs (Four bytes of ICR data and one byte of ICCS DATA). The 3-bit RCV ADRS determines which of the eight mux inputs will be applied to the bidirectional bus output mux. EN LOAD REG determines if the output to the VAX 8800 CPU will be buffered (IPR DATA) or unbuffered (CONS DATA).

Data transfers from the VAX 8800 CPU to the console are received as incoming data on the bidirectional bus and latched into the transmit data buffer (TXDB) by WR TXDB. The XMIT DISABLE signal blocks the buffered and unbuffered data output from the bidirectional bus during a CPU-to-console transfer. An interrupt triggered by the deassertion of the READY condition informs the console of received data waiting to be processed. The VAX 8800 CPU cannot transfer additional data until READY has been reasserted.

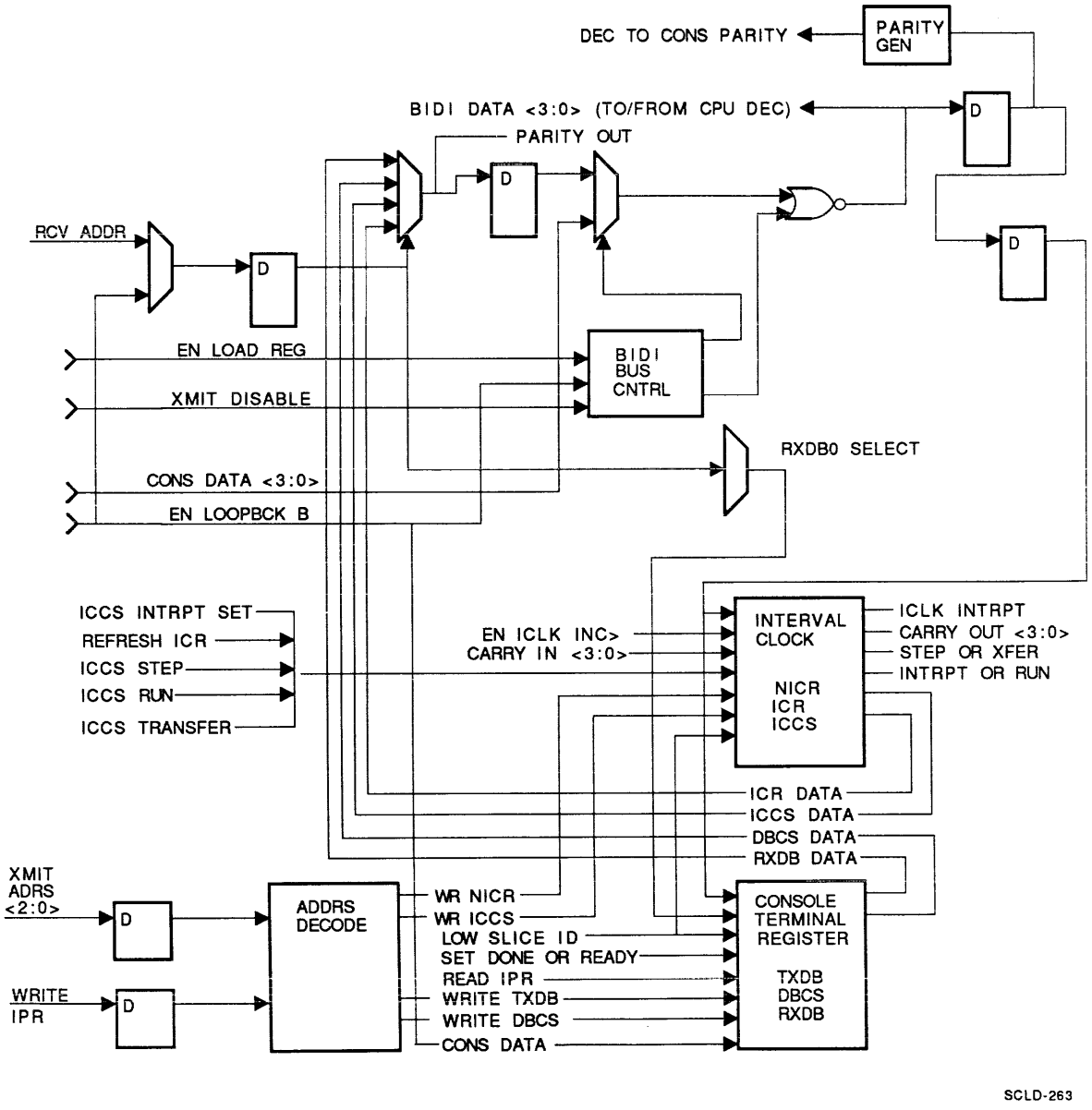


Figure 3-1 TRIC MCA Block Diagram

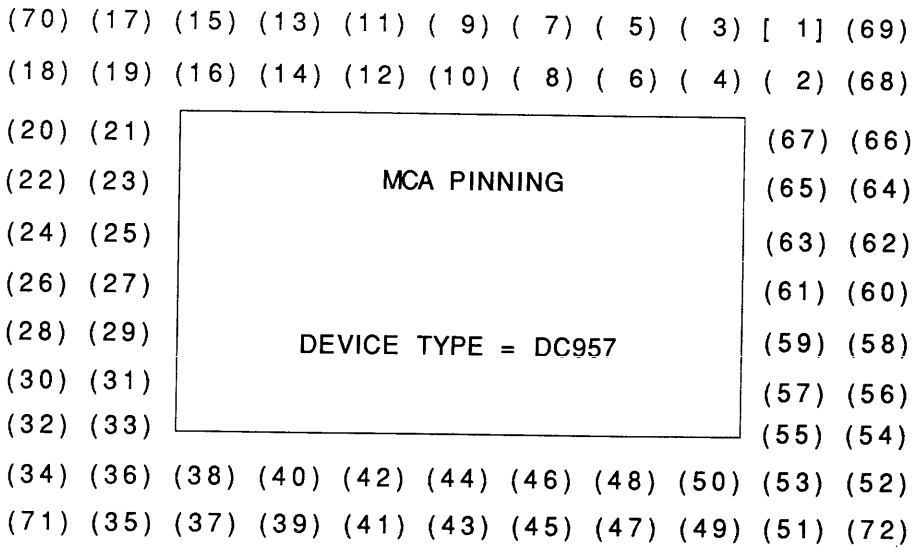


Figure 3-2 TRIC MCA Pin Layout

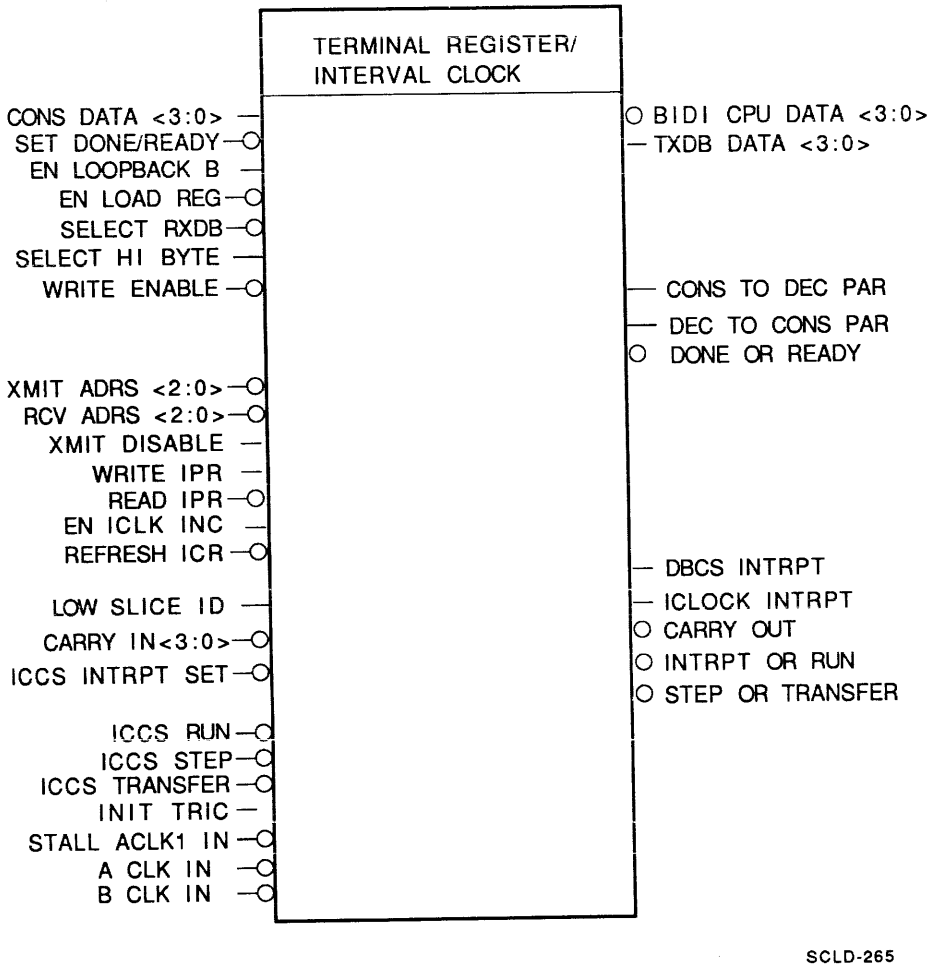


Figure 3-3 TRIC MCA Body Drawing

Table 3-1 TRIC MCA Pin Assignments

Pin	Signal Assignment	Pin	Signal Assignment
1	ICLK INTRPT	37	-XMIT ADRS <0>
2	-BIDI CPU DATA <1>	38	-XMIT ADRS <1>
3	GROUND	39	WRITE IPR
4	-CARRY IN <3>	40	-XMIT ADRS <2>
5	DATA BUFF CNTRL/STATUS INTRPT	41	-A CLCK1 IN
6	-CARRY IN <2>	42	CONS DATA <2>
7	-CARRY OUT <2>	43	NEG 52V
8	CONS TO DEC PARITY	44	CONS DATA <3>
9	NEG 52V	45	CONS DATA <0>
10	-CARRY OUT <1>	46	CONS DATA <1>
11	DEC TO CONS PARITY	47	-ICCS TRANSFER
12	-CARRY IN <1>	48	INIT TRIC
13	TRANSMIT DATA BUFFER <3>	49	-ICCS INTERRUPT SET
14	-DONE OR READY	50	ENABLE LOOPBACK B
15	GROUND	51	-REFRESH ICR
16	-BIDI CPU DATA <3>	52	-ENABLE LOAD REGISTER
17	TRANSMIT DATA BUFFER <2>	53	XMIT DISABLE
18	TRANSMIT DATA BUFFER <1>	54	LOW SLICE ID
19	-BIDI CPU DATA <0>	55	-RECV ADDRESS <1>
20	GROUND	56	-RECV ADDRESS <2>
21	TRANSMIT DATA BUFFER <0>	57	-RECV ADDRESS <0>
22	-CARRY IN <0>	58	-ICCS RUN
23	-CARRY OUT <0>	59	-ICCS STEP
24	NOT ASSIGNED	60	GROUND
25	-STALLED A CLOCK1 IN	61	ENABLE ICLK INC
26	GROUND	62	NOT ASSIGNED
27	SELECT HIGH BYTE	63	-B CLCK1 IN
28	-WRITE ENABLE	64	-CARRY OUT <3>
29	NOT ASSIGNED	65	-INTERRUPT OR RUN
30	-READ IPR	66	GROUND
31	NOT ASSIGNED	67	-BIDI DATA <2>
32	-SET DONE OR READY	68	- STEP OR TRANSFER
33	NOT ASSIGNED	69	GROUND
34	NOT ASSIGNED	70	GROUND
35	NOT ASSIGNED	71	GROUND
36	-SELECT RECEIVE DATA BUFFER	72	GROUND

Table 3-2 TRIC MCA Signal Descriptions

Signal	Comments
BIDI CPU DATA <3:0>	Bidirectional data bus between the VAX 8800 CPU and the TRIC MCAs. Consists of four bits of data from each of the MCA slices.
RECV ADRS <2:0>	Three bits of address data from the VAX 8800 CPU DEC module. Selects the mux input that will be applied to the bidirectional bus for output to the VAX 8800 CPU.
EN LOAD REG	A control signal from the console address decode circuitry. Used to create the SELECT LOAD PATH signal. SELECT LOAD PATH selects either the buffered (RXDB) or unbuffered (RAM load) data to be applied to the bidirectional data bus and forces the deassertion of INT XMIT DISABLE that allows the TRIC to drive the bus.
XMIT DISABLE	Control signal from the DEC module that disables bidirectional bus data output to the VAX 8800 CPU when input data is being sent by the console interface. (The default state of this signal is asserted.) The bidirectional bus between the clock and decoder modules is disabled at both ends when not used.
CONS DATA <3:0>	Eight bits of console data from the PPI port A. Sliced into four bits each on the high and low TRICs; this data is applied to the data buffer for output to the VAX 8800 CPU.
EN LOOPBACK B	Console control signal from the PPI port C. Enables testing of the receive and transmit data buffers, and the load path to the CPU.
XMIT ADRS <2:0>	VAX 8800 CPU control signal from the DEC module. The 3-bit signal is used to if the incoming data word from the CPU is console data or interval timer control. The decoded address generates the write enable signals for the interval timer and TXDB.
WRITE IPR	A latched enable signal derived from the XMIT-to-CONS signal from the CPU DEC. WRITE IPR enables the decoders that select the write signals for the interval clock and transmit data buffers.

Table 3-2 TRIC MCA Signal Descriptions (Cont)

Signal	Comments
RXDB0 <3:0> DATA	Output of the high and low slice receive data buffers. RXDB is the output data from the console to the VAX 8800 CPU.
DBCS <3:2>	Two bits of status information for output to the VAX 8800 CPU from the data buffer control and status register. Informs the VAX 8800 CPU of the state of the ready or done bits, and if interrupts are enabled for the transmit and receive data buffers.
ICCS <3:2>	Two bits of status data for output to the VAX 8800 CPU from the interval clock control/status register. ICCS data informs the CPU of the state of the interval clock. Interrupt status and error and run information is supplied.
ICR <3:0>	Four bytes of interval count register data to the VAX 8800 CPU. Two nibbles are supplied by each TRIC MCA. The interval timer data is applied to the TRIC output mux for transfer to the bidirectional bus.
WR NICR WR ICCS WR DBCS WR TXDB	WR NICR, WR ICCS, WR DBCS, and WR TXDB are write-enable signals that are derived from the XMIT ADRS and WRITE IPR input from the VAX 8800 CPU.
EN ICLOCK INC	Latched output of the 1-MHz clock on the CSEQ MCA. EN ICLOCK INC is the primary enable signal for incrementing the interval timer.
CARRY IN <3:0> CARRY OUT <3:0>	CARRY IN and CARRY OUT are four bits of count control signals that are used in the counting process of the interval timer count register.
ICCS INTRPT SET	TRIC MCA control signal used for setting the interval timer error bit of the ICCS.
REFRESH ICR	Internally generated control signal used in the transfer and loading of NICR and ICR count data. This signal is also used to create the interrupt and error bits of the ICCS.

Table 3-2 TRIC MCA Signal Descriptions (Cont)

Signal	Comments
ICCS STEP ICCS RUN ICCS TRANSFER	ICCS STEP, RUN, and TRANSFER are internally generated signals derived from the CPU data input, and are enabled by the WR ICCS term. They are used internally to enable the step and run of the counter and to transfer the NICR count to the ICR.
LOW SLICE ID	Differentiates high and low slices of the TRIC.
SET DONE or READY SELECT RXDB	The SET DONE or READY, and SELECT RXDB are generated by the console address decoder logic. The DONE and READY signals are used in the interrupt logic to the VAX 8800 CPU to show the state of the receive and transmit data buffers. SELECT RXDB selects the RXDB to receive data from the console.
READ IPR	Derived from the RECV FR CONS signal from the DEC module. Asserted for one cycle when the CPU is reading a TRIC IPR. This signal is also one of the control signals used for the RXDB data overrun error bit and DBCS done bit.
WRITE ENAB	Control signal from the CSEQ that enables latching of data received from the console.
SELECT HIGH BYTE	Control signal from the console address decoder. Determines selection of the low and high slice of data for the TXDB.
DEC TO CONS PAR	Parity bit calculated from eight bits of data from the DEC module to the console interface.
ICLOCK INTRPT	Interval clock interrupt from console to CPU SEQ.
STEP OR XFER INTRPT OR RUN	Internally generated interval clock signals. Created by CPU data and enabled by WR ICCS. Latched to create the ICCS STEP, ICCS XFER, ICCS INTRPT, and ICCS RUN depending on whether the signal is coming from high or low TRIC MCA.
TXDB DATA	Mux selected output data from the VAX 8800 CPU to the console. The TRIC output mux selects either high or low slice data to be applied to the interface data output mux.

Table 3-2 TRIC MCA Signal Descriptions (Cont)

Signal	Comments
DBCS INTRPT	Internally generated console interrupt to the CPU SEQ created from the DONE (High Slice) and READY (Low Slice) bits and enabled by CPU data and WR DBCS.
DONE OR READY	DONE and READY are used to identify the source of interrupts (DONE = RXDB -- READY = TXDB).

3.3 CONSOLE SEQUENCER MCA (CSEQ)

The console interface contains a single 19-0CSEQ-00 console sequencer MCA to implement the sequencing of read/write transfers between the console and the VAX 8800 system. The CSEQ also provides write address timing to the IPRs and a 1-MHz synchronized clock for the interval timer. Figure 3-4 shows a block diagram of the CSEQ MCA. Figures 3-5 and 3-6 show the pin layout and body drawing of the CSEQ MCA. Table 3-3 lists each CSEQ MCA pin and identifies the signal assigned. Table 3-4 lists some of the key signals on the CSEQ MCA and provides a description of the signal functions.

3.3.1 Console Strobe Sequencer

The console strobe sequencer controls the direction of data transfers on the PPI bidirectional data bus between the console and the interface with the assertion of the CONS READ signal. It also controls the transfer of data to the console by providing a latch signal (PPI STROBE) to the PPI port A buffer.

3.3.2 Read Acknowledge

The read acknowledge logic controls the timing of data transfers from the VAX 8800 CPU by informing the CPU (DATA ACCEPTED) when data from the transmit data buffer has been received by the console.

3.3.3 Console Write Sequencer

Control sequencing of data transfers from the console to the console interface is accomplished by the console write sequencer logic. It also generates the control signals that enable writing to the console clock, control registers, VBus control register, and the interface control registers.

3.3.4 Control Store Load Sequencer

The control store load sequencer generates the control signals that enable loading of the control store RAMs during the initialization process.

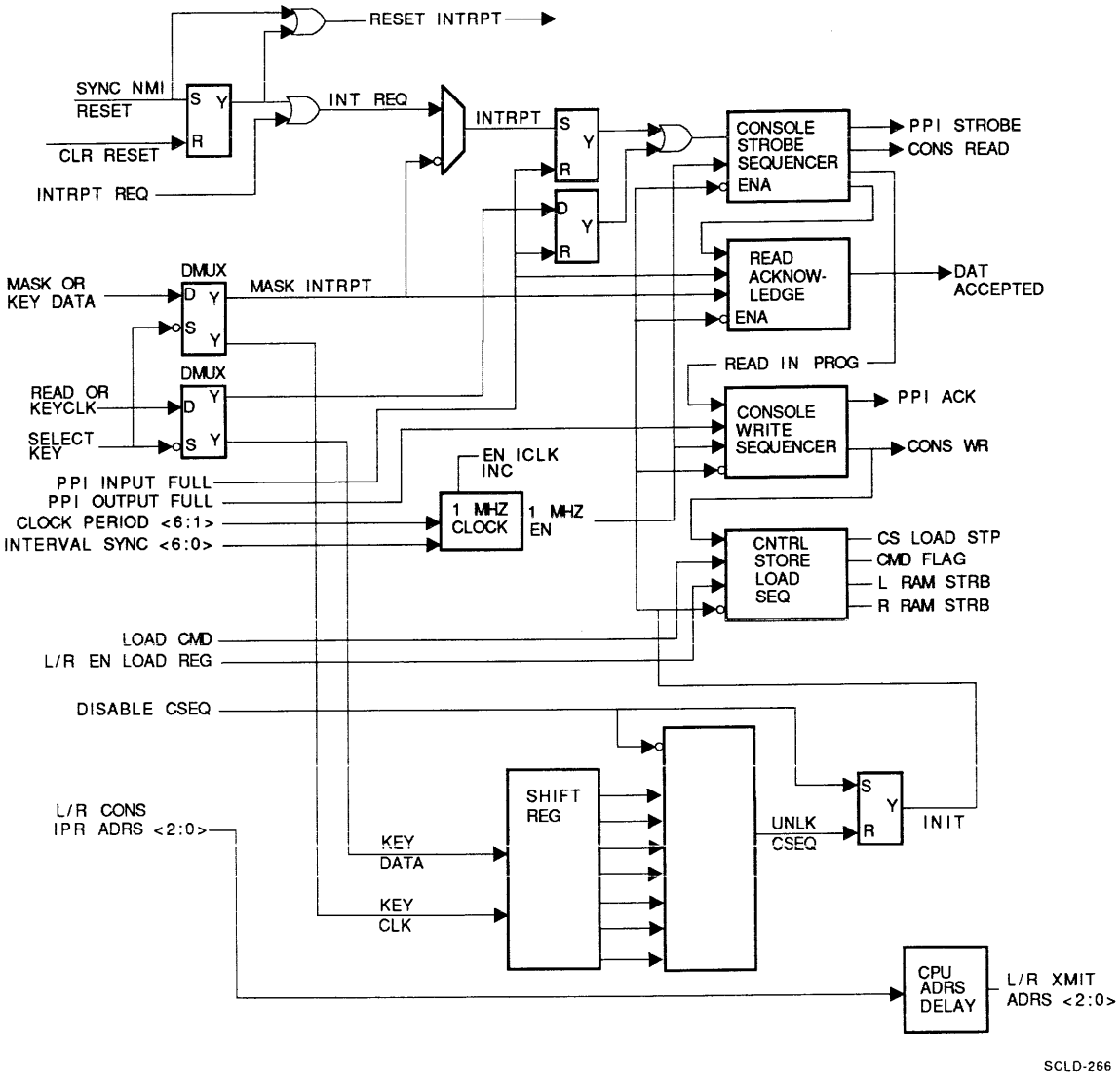


Figure 3-4 CSEQ MCA Block Diagram

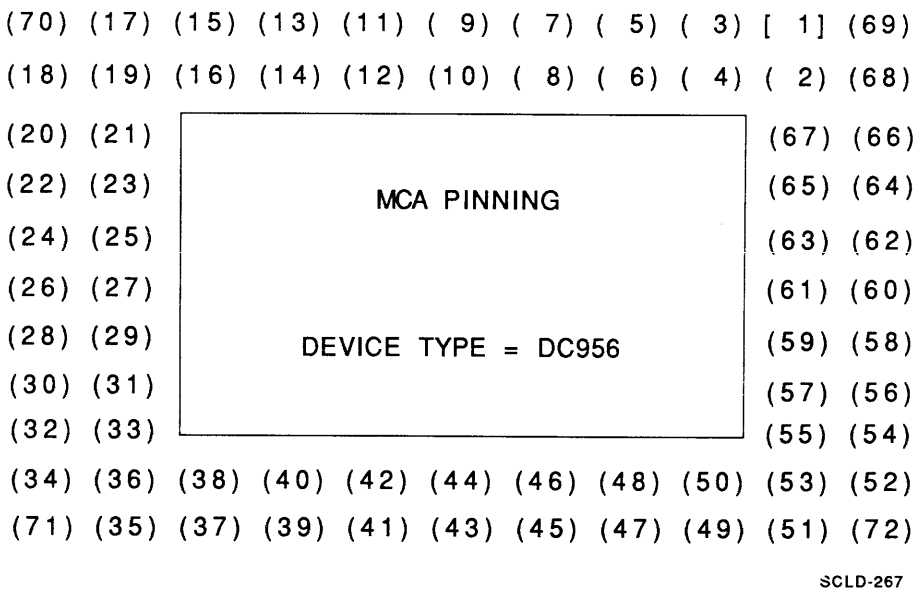


Figure 3-5 CSEQ MCA Pin Layout

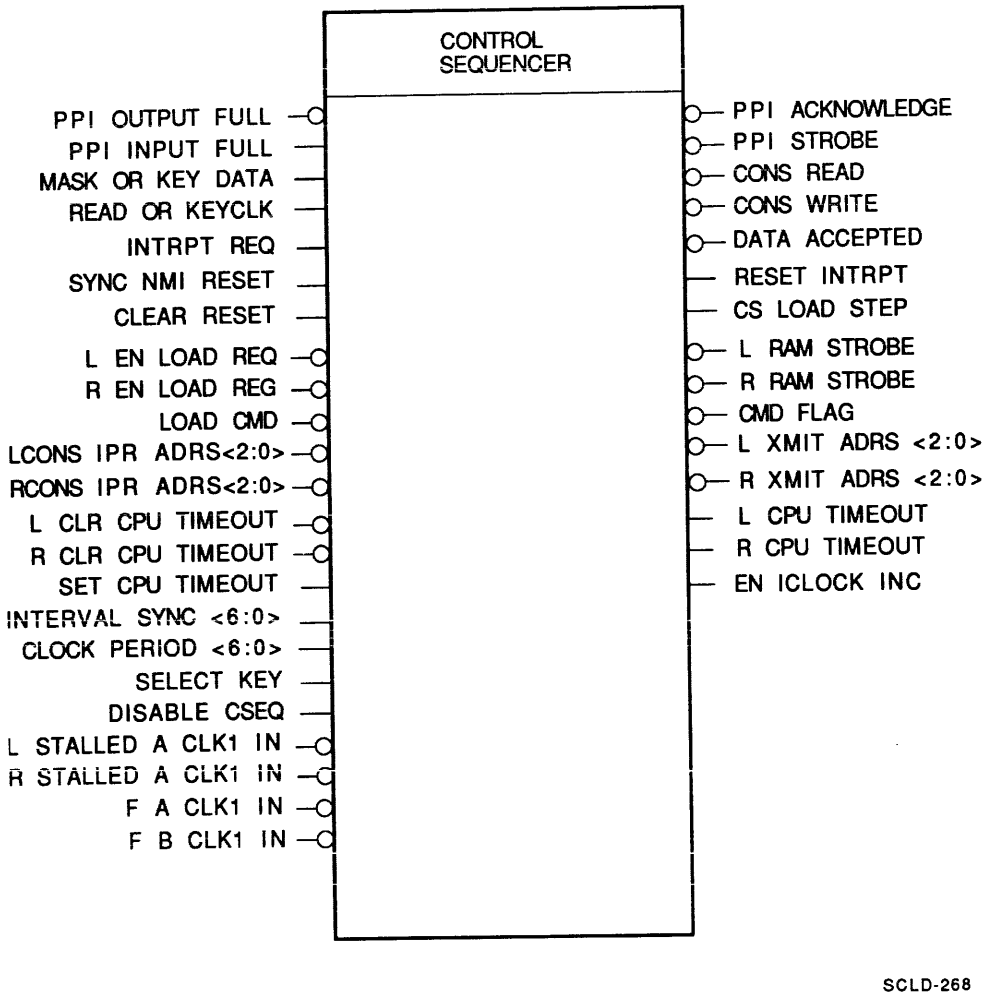


Figure 3-6 CSEQ MCA Body Drawing

Table 3-3 CSEQ MCA Pin Assignments

Pin	Signal Assignment	Pin	Signal Assignment
1	-DATA ACCEPTED	37	-R EN LOAD REG
2	L CPU TIMEOUT	38	-LOAD CMD
3	GROUND	39	INTERVAL SYNC <6>
4	INTERVAL SYNC <1>	40	CLOCK PERIOD <4>
5	-R XMIT ADRS <1>	41	CLOCK PERIOD <2>
6	-R CLR CPU TIMEOUT	42	CLOCK PERIOD <1>
7	-CONS READ	43	NEG 52V
8	R CPU TIMEOUT	44	-L STALLED A CLK1 IN
9	NEG 52V	45	-R CONS IPR ADRS <0>
10	-CMD FLAG	46	-R CONS IPR ADRS <2>
11	-PPI ACKNOWLEDGE	47	-L CONS IPR ADRS <0>
12	INTERVAL SYNC <2>	48	-L CONS IPR ADRS <2>
13	-R RAM STROBE	49	-L CONS IPR ADRS <1>
14	-L XMIT ADRS <1>	50	MASK OR KEY DATA
15	GROUND	51	-R CONS IPR ADRS <1>
16	-PPI STROBE	52	READ OR KEYCLK
17	-L RAM STROBE	53	SELECT KEY
18	-R XMIT ADRS <2>	54	CLEAR RESET
19	EN ICLOCK INC	55	SET CPU TIMEOUT
20	GROUND	56	-R STALLED A CLK1 IN
21	INTERVAL SYNC <5>	57	SYNC NMI RESET
22	CS LOAD STEP	58	CLOCK PERIOD <6>
23	-CONS WRITE	59	-F B CLK1 IN
24	NOT ASSIGNED	60	GROUND
25	INTERVAL SYNC <4>	61	-L CLR CPU TIMEOUT
26	GROUND	62	NOT ASSIGNED
27	-L EN LOAD REG	63	-L XMIT ADRS <0>
28	INTERVAL SYNC <3>	64	-R XMIT ADRS <0>
29	-F A CLK1 IN	65	PPI INPUT FULL
30	CLOCK PERIOD <3>	66	GROUND
31	CLOCK PERIOD <5>	67	RESET INTRPT
32	DISABLE CSEQ	68	-L XMIT ADRS <2>
33	NOT ASSIGNED	69	GROUND
34	INTERVAL SYNC <0>	70	GROUND
35	INTRPT REQ	71	GROUND
36	-PPI OUTPUT FULL	72	GROUND

Table 3-4 CSEQ MCA Signal Descriptions

Signal	Comments
SYNC NMI RESET	Generated by the NMI reset from the NBIA. Latched in the synchronizer logic to become SYNC NMI RESET. The CSEQ MCA detects the rising edge and sets LATCHED RESET, which generates an interrupt to the console. The OR of LATCHED RESET and SYNC NMI RESET are used for bit 5 of the console interrupt status register.
CLEAR RESET	Generated by control register 2 as a result of a console command. Clears LATCHED RESET.
TERM REG INTRPT	Internally generated interrupt enable signal. One of the signals used to create PPI STROBE. Derived from RXDB DONE and TXDB READY of both CPUs.
MASK OR KEYDATA	Control signal received from PPI port B, bit 5. Performs one of two functions depending on the state of the SELECT KEY input. If the SELECT KEY bit is set, it provides the input to the shift register for generating the UNLOCK CSEQ term. If the SELECT KEY is not set, it performs the function of interrupt mask.
READ OR KEYCLK	Control signal received from PPI port B, bit 4. Dual function determined by the SELECT KEY signal. If the SELECT KEY bit is not set, read enable is passed to the strobe sequencer. If SELECT KEY is set, the demux output is a clocking pulse to the CSEQ unlock shift register.
PPI INPUT FULL PPI OUTPUT FULL	Console control signals from PPI port C. INPUT FULL indicates that the PPI input buffer contains data for the console. PPI INPUT FULL is used internally on the CSEQ to terminate a read sequence. PPI OUTPUT FULL indicates that the output buffer has data from the console that is to be transferred to the VAX 8800 CPU. The OUTPUT FULL signal is used to generate the CS LOAD ENABLE and the PPI ACKNOWLEDGE.
CLOCK PERIOD	Control signals from the clock module used to create INTERVAL SYNC the 1-MHz clock signals.

Table 3-4 CSEQ MCA Signal Descriptions (Cont)

Signal	Comments
LOAD CMD	Console control signal derived from the console address decoder logic. Generates the command flag signal that is sent to the VAX 8800 decoder module. COMMMAND FLAG is used by the gateway control to load the RAMs during the initialization process.
L/R EN LOAD REG	Console control signals derived from the console address decoder logic. These are the primary signals used in the generation of the left and right RAM strobe signals to the gateway control on the decoder module. The strobe signals are used for RAM loading during initialization.
DISABLE CSEQ	ENABLE/DISABLE signal that is used to disable the CSEQ during a PRO-38N power failure or console cable disconnect.
L/R CONS IPR ADRS L/R XMIT ADRS	Internally generated address signals from the DEC. Clock delayed signals (L/R XMIT ADRS) used to create the write enable signals for the TRIC MCA. The CSEQ only provides a delaying mechanism.
PPI STROBE	Internally generated signal that is used for latching console interface data into the PPI port A buffer.
CONS READ	Internally generated signal used to enable/disable the bidirectional port A data transceiver.
DATA ACCEPTED	Internally generated signal used to set the TX READY bit. Informs VAX 8800 system that the TXDB has been read by the console and is ready to accept another data transfer.

Table 3-4 CSEQ MCA Signal Descriptions (Cont)

Signal	Comments
PPI ACKNOWLEDGE	Internally generated signal that enables the console to drive the bidirectional bus from PPI port A.
CONS WRITE	Internally generated signal that is used as a control to create the following enable signals: WR CLOCK PERIOD WR BURST COUNT WR CLOCK CONTROL WR CLOCK REG <2:0> WR VBUS CONTROL WR CONTROL REG <2:0> SET L/R RCVR DONE

3.4 CONSOLE/VAX 8800 REGISTER SUMMARY

3.4.1 Console Registers (Refer to Figure 3-7)

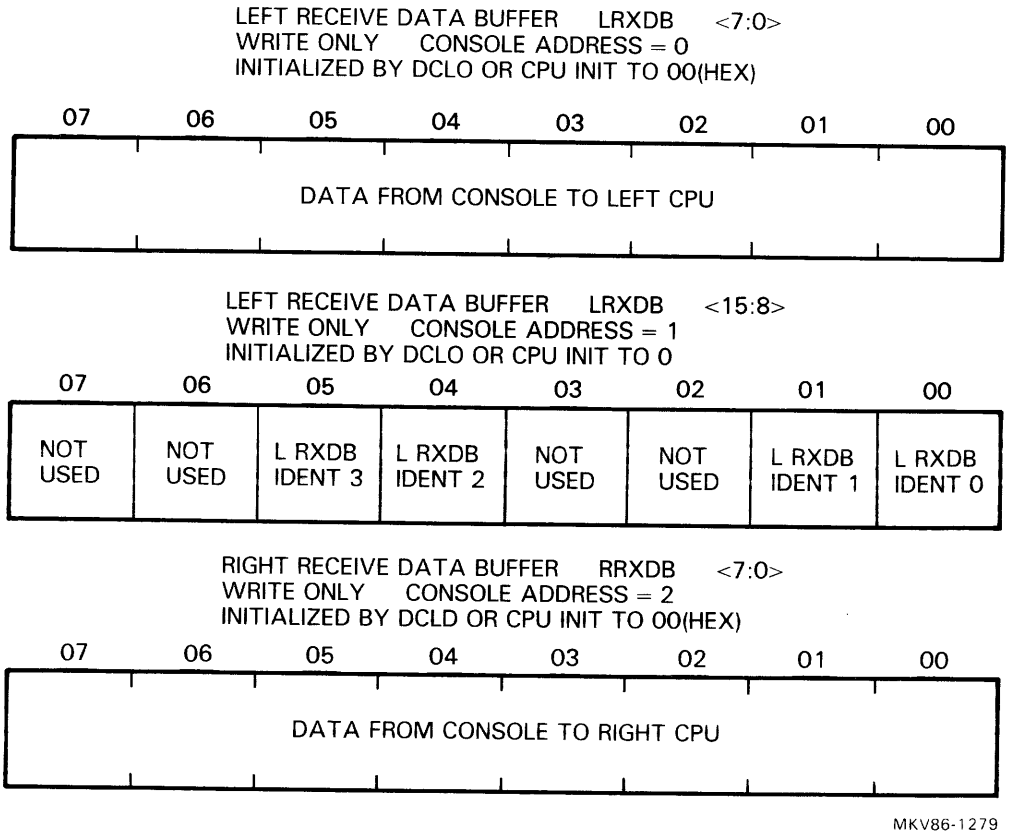


Figure 3-7 Console Registers (Sheet 1 of 7)

RIGHT RECEIVE DATA BUFFER RRXDB <15:8>
WRITE ONLY CONSOLE ADDRESS = 3
INITIALIZED BY DCLO OR CPU INIT TO 0

07	06	05	04	03	02	01	00
NOT USED	NOT USED	R RXDB IDENT 3	R RXDB IDENT 2	NOT USED	NOT USED	R RXDB IDENT 1	R RXDB IDENT 0

LEFT COMMAND LOAD REGISTER LCLR
WRITE ONLY CONSOLE ADDRESS = 4
INITIALIZED BY CPU INIT TO 0

07	06	05	04	03	02	01	00
NOT USED	NOT USED	NOT USED	NOT USED	COMMAND TO GATEWAY CONTROL MCA OF LEFT CPU			

LEFT DATA LOAD REGISTER LDLR
WRITE ONLY CONSOLE ADDRESS = 5
NOT INITIALIZED

07	06	05	04	03	02	01	00
RAM DATA OR ADDRESS TO LEFT CPU							

RIGHT COMMAND LOAD REGISTER RCLR
WRITE ONLY CONSOLE ADDRESS = 6
INITIALIZED BY CPU INIT TO 0

07	06	05	04	03	02	01	00
NOT USED	NOT USED	NOT USED	NOT USED	COMMAND TO GATEWAY CONTROL MCA OF RIGHT CPU			

MKV86-1280

Figure 3-7 Console Registers (Sheet 2 of 7)

RIGHT DATA LOAD REGISTER RDLR
WRITE ONLY CONSOLE ADDRESS = 7
NOT INITIALIZED

07	06	05	04	03	02	01	00
RAM DATA OR ADDRESS TO RIGHT CPU							

CONTROL REGISTER 0 CR0
WRITE ONLY CONSOLE ADDRESS = 8
INITIALIZED BY DCLO TO FF(HEX)

07	06	05	04	03	02	01	00
DISABLE L CPU STALLS	DISABLE L CPU TRAPS	DISABLE L CPU NMI SEQ	LEFT CPU INIT	DISABLE R CPU STALLS	DISABLE R CPU TRAPS	DISABLE R CPU NMI SEQ	RIGHT CPU INIT

CONTROL REGISTER 1 CR1
WRITE ONLY CONSOLE ADDRESS = 9
INITIALIZED BY DCLO TO 00(HEX)

07	06	05	04	03	02	01	00
NMI UNJAM	ENABLE L RXDB INTRPT	NMI SLOW MODE	ENABLE R RXDB INTRPT	ENABLE TRAP ON L MATCH	HALT LEFT CPU	ENABLE TRAP ON R MATCH	HALT RIGHT CPU

CONTROL REGISTER 2 CR2
WRITE ONLY CONSOLE ADDRESS = A
INITIALIZED BY DCLO TO 0

07	06	05	04	03	02	01	00
NOT USED	NOT USED	NOT USED	ENABLE IDP LOOP-BACK	DISABLE L SDF WRITES	DISABLE R SDF WRITES	CLEAR LATCHED RESET	SET CPU TIMEOUT FLAG

MKV86-1281

Figure 3-7 Console Registers (Sheet 3 of 7)

VISIBILITY BUS CONTROL VBC
WRITE ONLY CONSOLE ADDRESS = B
INITIALIZED BY DCLO TO 0

07	06	05	04	03	02	01	00
NOT USED	NOT USED	NOT USED	NOT USED	SELECT R CPU INPUT	STOP ADDRESS SHIFT	STEP B CLOCKS	VBUS ADDRESS OUTPUT

CLOCK CONTROL REGISTER CCR
WRITE ONLY CONSOLE ADDRESS = C
INITIALIZED BY CONSOLE SOFTWARE TO 78(HEX)

07	06	05	04	03	02	01	00
STOP START CLOCKS	DISABLE STOP ON L MATCH	DISABLE STOP ON R MATCH	DISABLE CLOCK BURSTS	ENABLE SLOW CLOCK	NOT USED	NOT USED	NOT USED

BURST COUNT REGISTER BCR
WRITE ONLY CONSOLE ADDRESS = D
INITIALIZED BY CONSOLE SOFTWARE TO 0

07	06	05	04	03	02	01	00
NUMBER OF CYCLES FOR CLOCK TO RUN DURING A BURST OPERATION							

CLOCK PERIOD REGISTER CPR
WRITE ONLY CONSOLE ADDRESS = E
INITIALIZED BY DCLO TO 38(HEX)
INITIALIZED PERIOD = 70.18 NS

07	06	05	04	03	02	01	00
SELECT LO FREQ RANGE	(1/ (CLOCK PERIOD × 250K)) -1						

MKV86-1282

Figure 3-7 Console Registers (Sheet 4 of 7)

LEFT TRANSMIT DATA BUFFER LTXDB <7:0>
READ ONLY CONSOLE ADDRESS = 0

07	06	05	04	03	02	01	00
DATA FROM LEFT CPU TO CONSOLE							

LEFT TRANSMIT DATA BUFFER LTXDB <15:8>
READ ONLY CONSOLE ADDRESS = 1

07	06	05	04	03	02	01	00
NOT USED	NOT USED	L TXDB IDENT <3>	L TXDB IDENT <2>	NOT USED	NOT USED	L TXDB IDENT <1>	L TXDB IDENT <0>

RIGHT TRANSMIT DATA BUFFER RTXDB <7:0>
READ ONLY CONSOLE ADDRESS = 2

07	06	05	04	03	02	01	00
DATA FROM RIGHT CPU TO CONSOLE							

RIGHT TRANSMIT DATA BUFFER RTXDB <15:8>
READ ONLY CONSOLE ADDRESS = 3

07	06	05	04	03	02	01	00
NOT USED	NOT USED	R TXDB IDENT <3>	R TXDB IDENT <2>	NOT USED	NOT USED	R TXDB IDENT <1>	R TXDB IDENT <0>

MKV86-1283

Figure 3-7 Console Registers (Sheet 5 of 7)

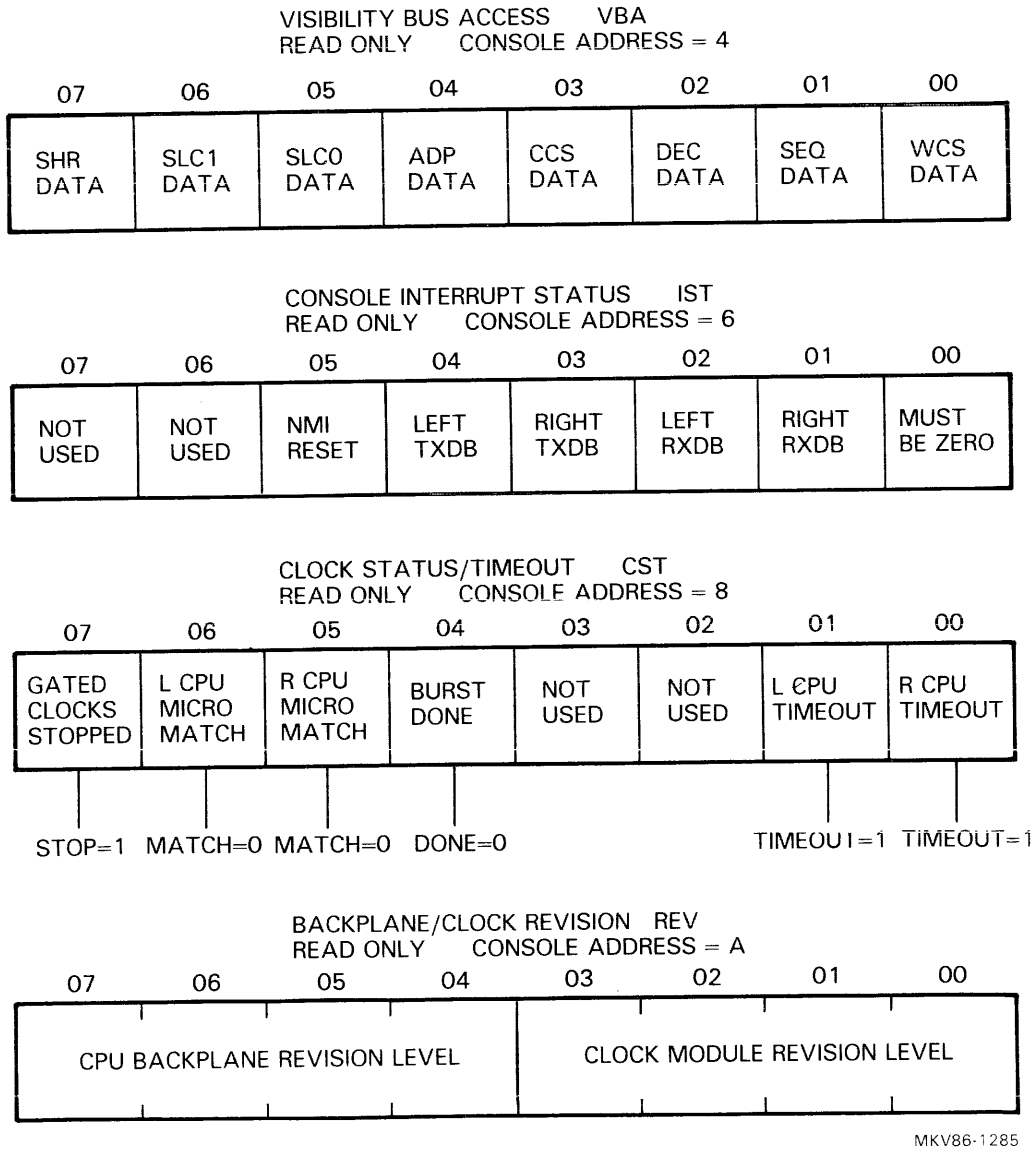


Figure 3-7 Console Registers (Sheet 6 of 7)

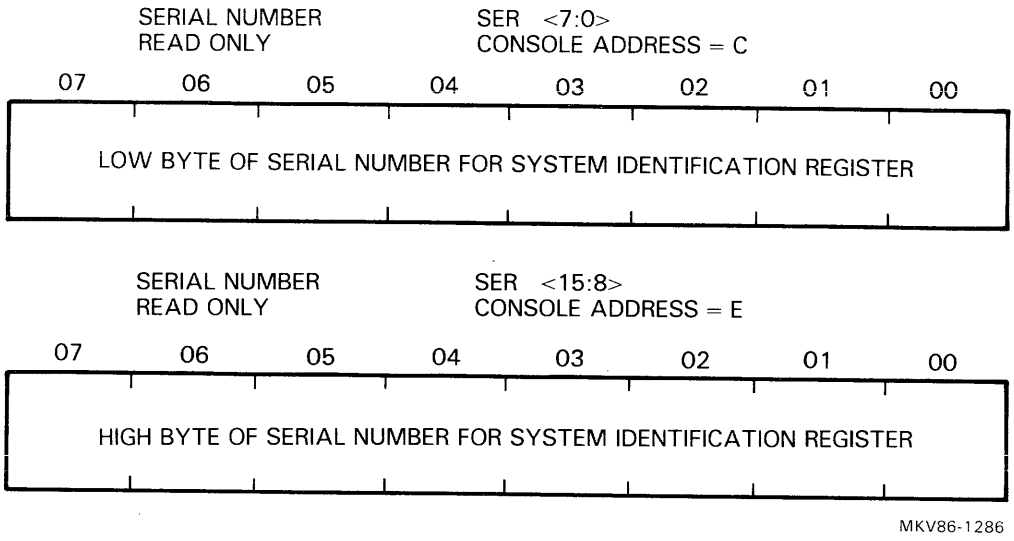


Figure 3-7 Console Registers (Sheet 7 of 7)

3.4.2 VAX 8800 CPU Registers (Refer to Figure 3-8)

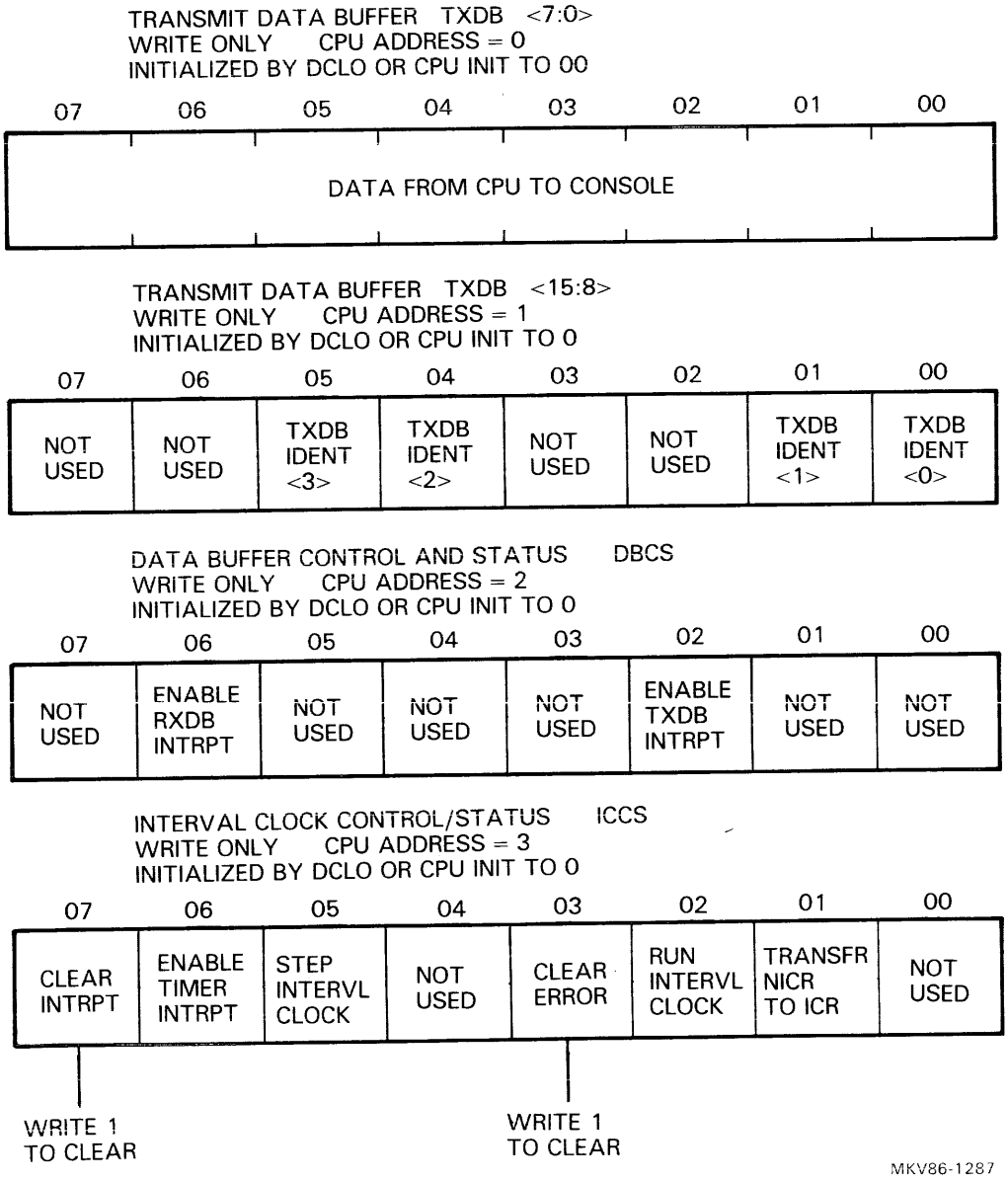


Figure 3-8 VAX 8800 CPU Registers (Sheet 1 of 4)

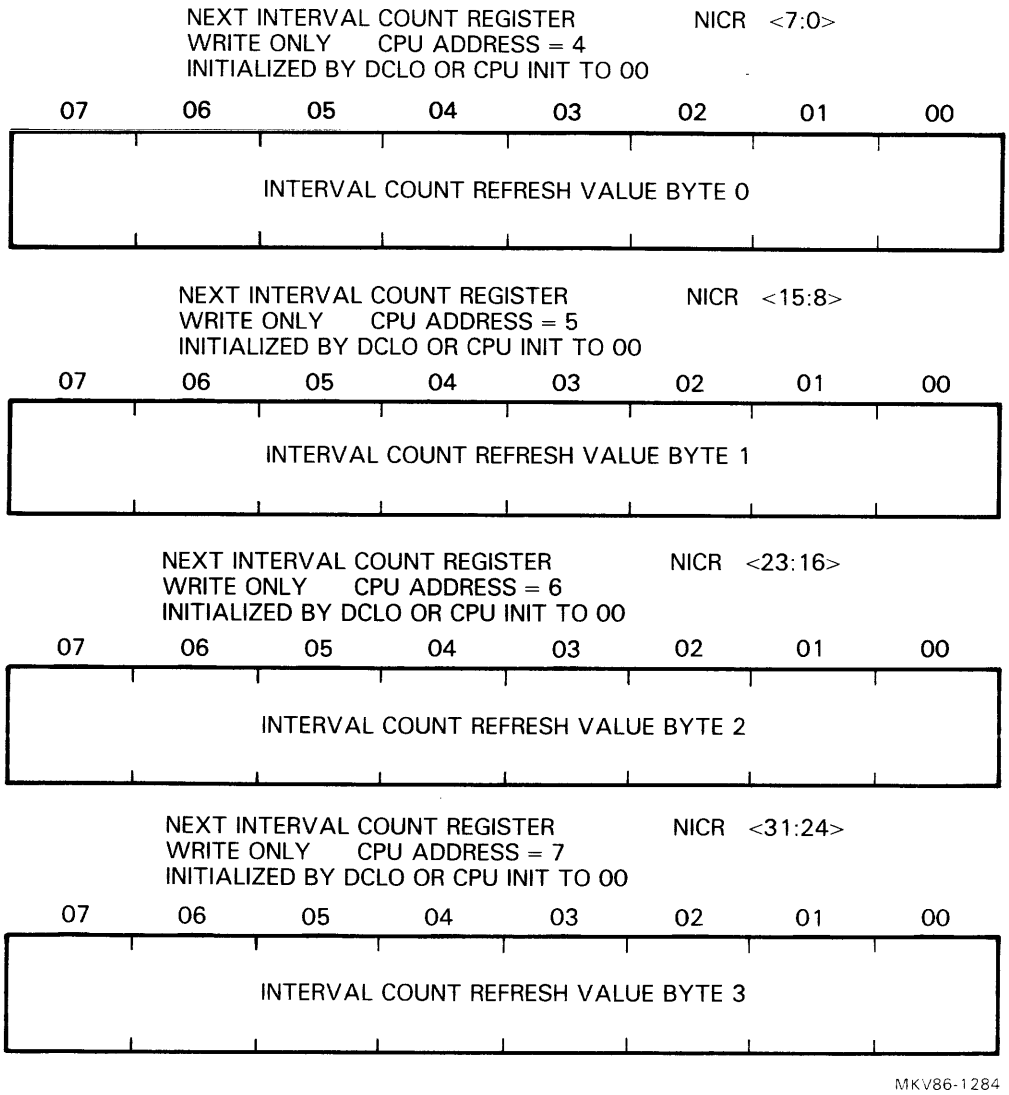


Figure 3-8 VAX 8800 CPU Registers (Sheet 2 of 4)

RECEIVE DATA BUFFER
READ ONLY
INITIALIZED BY DCLO OR CPU INIT TO 00

RXDB <7:0>
CPU ADDRESS = 0

07	06	05	04	03	02	01	00
DATA FROM CONSOLE TO CPU							

RECEIVE DATA BUFFER
READ ONLY
INITIALIZED BY DCLO OR CPU INIT TO 44(HEX)

RXDB <15:8>
CPU ADDRESS = 1

07	06	05	04	03	02	01	00
RXDB ERROR OCCURRED	1	RXDB IDENT <3>	RXDB IDENT <2>	X	1	RXDB IDENT <1>	RXDB IDENT <0>

DATA BUFFER CONTROL AND STATUS
READ ONLY CPU ADDRESS = 2
INITIALIZED BY DCLO OR CPU INIT TO 08(HEX)

DBCS

07	06	05	04	03	02	01	00
RXDB DONE	RXDB INTRPT ENABLED	0	0	TXDB READY	TXDB INTRPT ENABLED	0	0

INTERVAL CLOCK CONTROL/STATUS
READ ONLY CPU ADDRESS = 3
INITIALIZED BY DCLO OR CPU INIT TO 0

ICCS

07	06	05	04	03	02	01	00
INTRPT OCCURRED	TIMER INTRPT ENABLED	0	0	ERROR OCCURRED	INTERVL CLOCK RUNNING	0	0

MKV86-1288

Figure 3-8 VAX 8800 CPU Registers (Sheet 3 of 4)

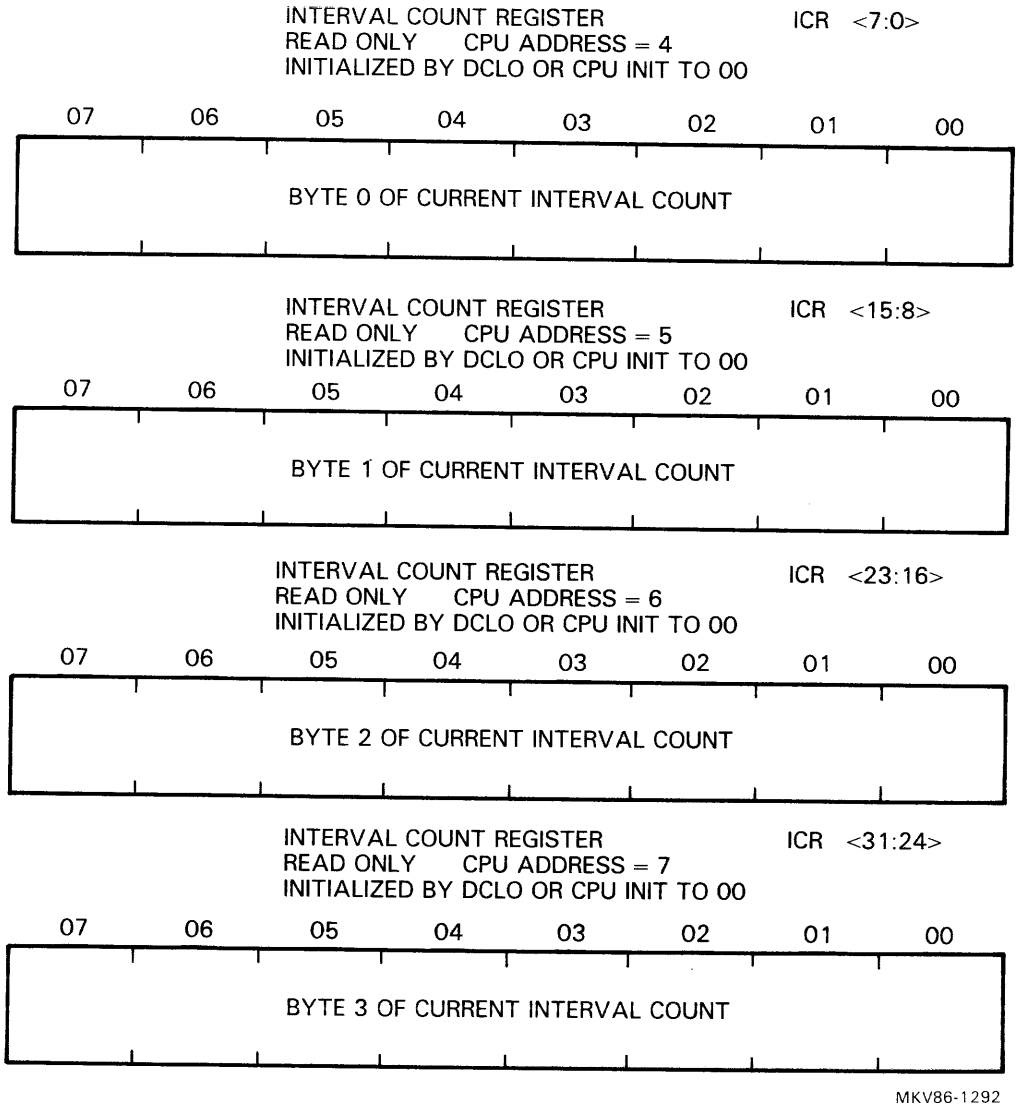


Figure 3-8 VAX 8800 CPU Registers (Sheet 4 of 4)

3.5 CONSOLE CABLING

The cables that connect the console to the VAX 8800 system are listed in Table 3-5 and illustrated in Figure 3-7. The first column of the table (Item) contains a callout reference to Figure 3-9.

Table 3-5 Console Cable List

Item	Description	Part No.	From	To
(A)	Console to EMM cable	17 00655-01	Cons Dist J6	MPS 2 J63
(B)	Console Cable #2	17 00651-01	CPU J22	Cons Dist J4
(C)	Console Cable #1	17 00649-01	CPU J23	Cons Dist J5
(D)	PC 380 BCC26-201 CBL	17 00665-01	Cons Dist J2	PC 380
(E)	PC 380 Power Cable	17 00442-17	88X	PC 380

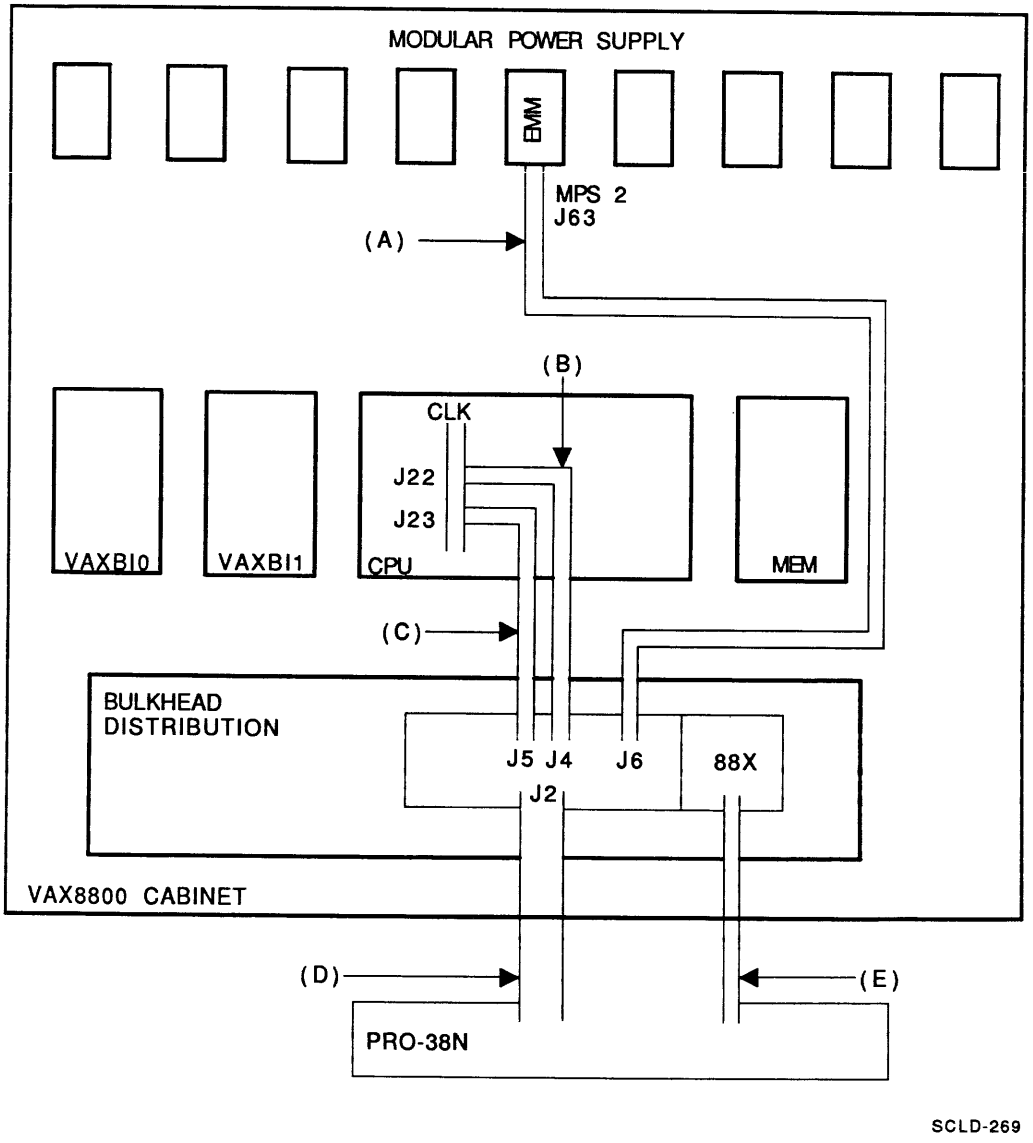


Figure 3-9 Console Subsystem Cabling Diagram

SECTION 4
POWER SYSTEM COMPLEX

CHAPTER 1
GENERAL DESCRIPTION

1.1 INTRODUCTION

The power supply provides the dc voltages necessary to operate the main CPU(s), memory and VAXBI of the VAX 8800 system. Three-phase ac utility power is used to energize the power system. The dc voltages required are developed using the ac power modules and the dc voltage regulators located in the CPU cabinet. In the following paragraphs, the power supply is occasionally referred to as the power system. Both names refer to the power supply.

Chapter 1 provides a general description of the power supply and introduces the components used to generate the dc voltages required to operate the CPU and its related components.

Chapters 2 and 3 contain more comprehensive descriptions of the power supply modules and their functions.

Details of the operating signals and the specifications of the power supply modules are presented in the Appendix.

This chapter introduces the components, configurations, controls, and indicators of the VAX 8800 power system. It includes illustrations of the power system and its basic components, operational descriptions of the components, and general specifications for the power system.

Figure 1-1 is a physical layout of the VAX 8800 system showing the power module layout and the blower subsystem.

Figure 1-2 is a block diagram of the 60 Hz power system.

Figure 1-3 is a block diagram of the 50 Hz power system.

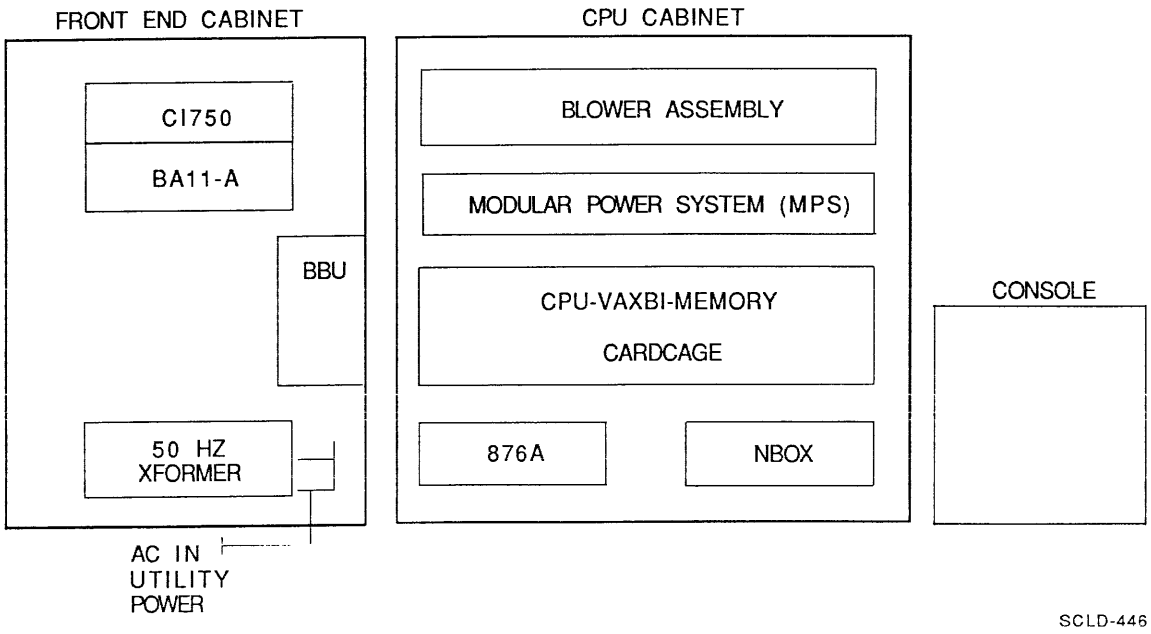


Figure 1-1 VAX 8800 System Physical Layout (Front View)

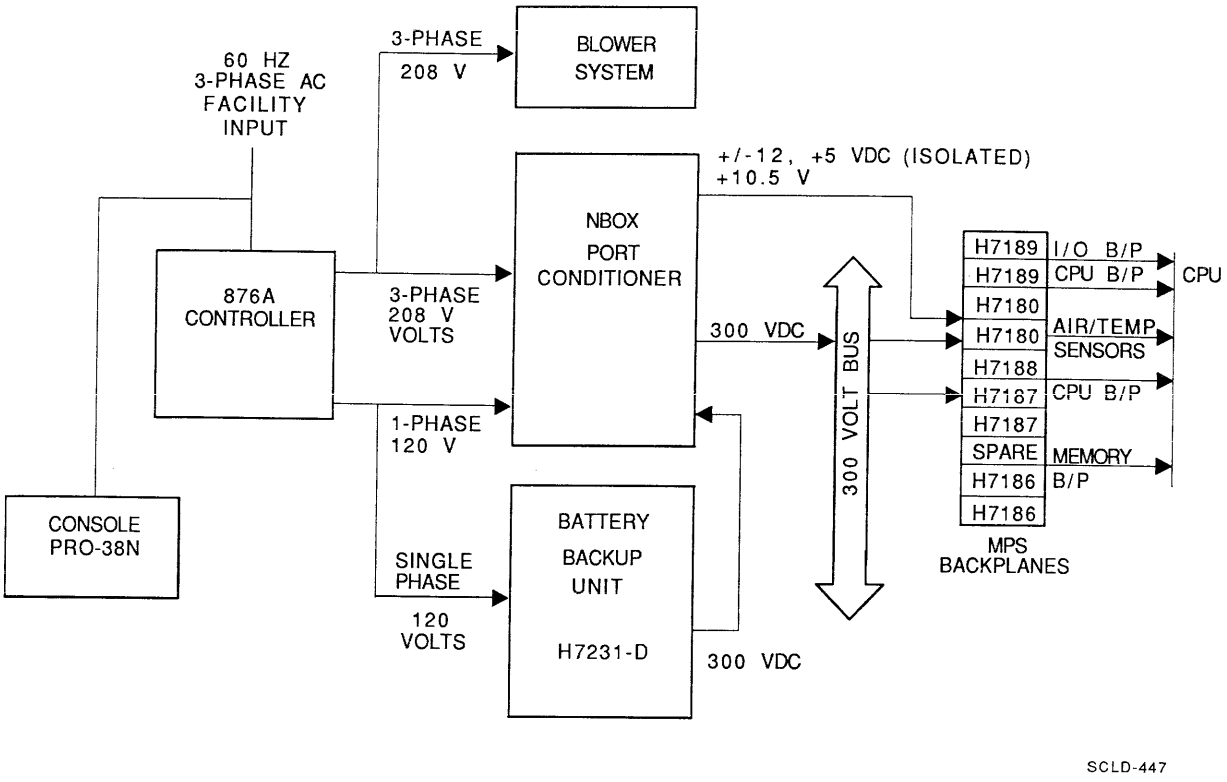


Figure 1-2 VAX 8800 Power System Block Diagram (60 Hz)

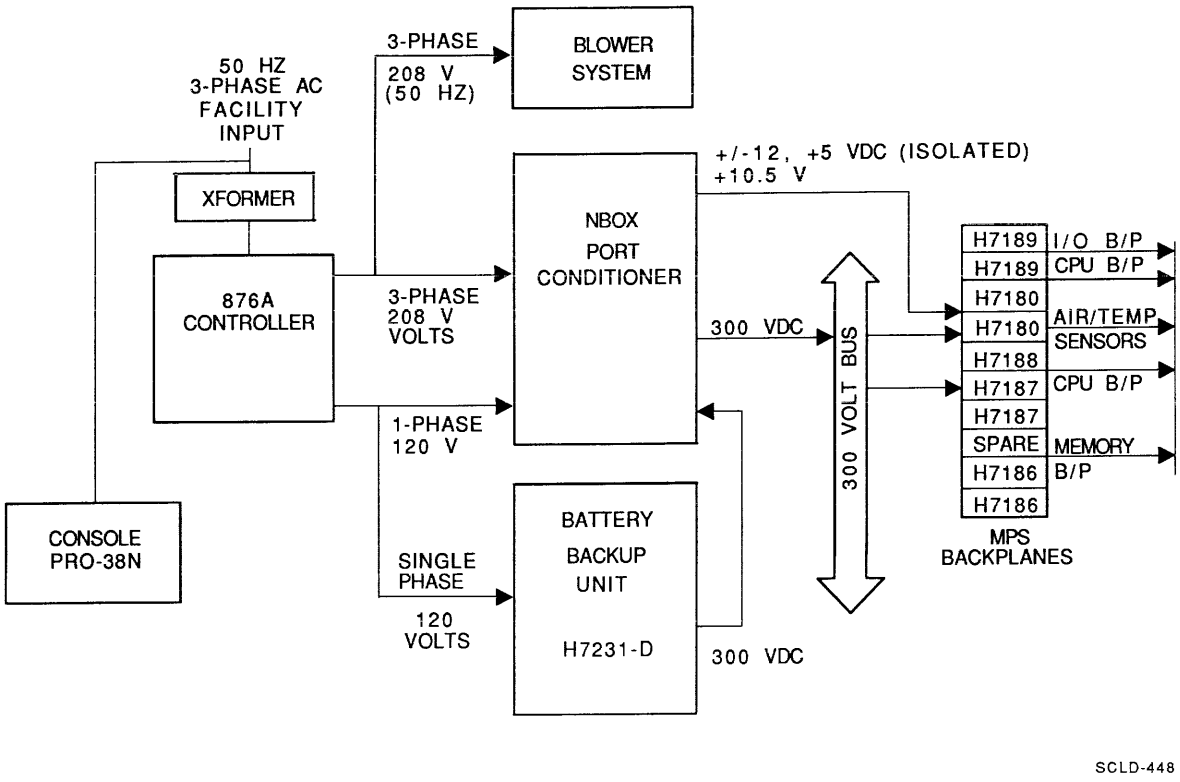


Figure 1-3 VAX 8800 Power System Block Diagram (50 Hz)

1.2 SYSTEM COMPONENTS

The power system shown in Figure 1-1 includes the components listed in Table 1-1.

Table 1-1 Power System Components

Component	Mnemonic	Component Number
876A power controller	876A	876A
NBox power converter	NPC	
Modular power supplies	MPS	H7180, H7186, H7187, H7189
Environment monitor module	EMM	H7188
Battery backup (option)	BBU	H7231-D
Air flow blower system	AFS	

1.2.1 876A Power Controller

The 876A power controller is the main ac input module for the power system. Three-phase ac facility power to the VAX 8800 system is passed through BRKR1, the main system circuit breaker, and on to the 876A for power distribution to the system components.

Power from the 876A feeds switched and unswitched power as follows:

- Switched three-phase and unswitched single-phase power to the NBox.
- Unswitched single-phase power to the BBU.
- Switched three-phase power to the blower system.
- Switched single-phase power to the expander units (CI750).

1.2.2 NBox Power Converter

The NBox PC is a multifunction power assembly that contains the modules listed in Table 1-2.

Table 1-2 NBox Modules

NBox Module	Function
H7170A (X)	Converts 3-phase ac power into 300 Vdc power out for 300 V bus.
H7170A (Y)	Same as above.
ILM	Provides logic signal interface between EMM and other power system components. Also controls BBU operation.
CSP	Converts single-phase ac power to logic level voltages: +5 V, +/-12 V, +10.5 V for EMM, ILM.
NBT	Converts logic signals for start-up and system operation. Also interfaces with DIGITAL power bus.

1.2.3 Modular Power System (MPS)

The Modular Power System (MPS) is a dc power supply module nest located above the CPU cardcage in the main CPU cabinet. The MPS contains the dc power regulators that provide the dc power for operating the main CPU, memory, and extender modules of the VAX 8800 system.

The MPS nest also contains the Environmental Monitoring Module (EMM), described briefly in the following section.

Power is applied to the power supplies, located in the MPS nest, by cabling connected to the 300-Vdc power bus. The 300-Vdc buses are fed from the H7170 ac-to-dc power converters located in the NBox.

There are three 300-Vdc power buses located in the MPS area, one of which connects to power supply module B, the regulator used to provide power to the CPU memory during battery backup operation.

The MPS backplane provides connections for:

- The power control signals (MPS powerup, powerdown, and BBU assignment) to the power regulators
- The EMM, CSP, and ILM modules (via NBox backplane)
- The CPU backplane
- The environmental monitoring signals

1.2.4 Environmental Monitoring Module

The Environmental Monitoring Module (EMM) is a microprocessor-based (8085) module located in the MPS cage. It is used to monitor:

- DC power regulator operation
- Power system status
- Power system start-up and operation
- Environmental conditions within the main CPU cabinet.

The EMM is also the main communications link between the VAX 8800 system console and the power system. Additionally, the EMM provides control during power-up and power-down sequencing and during battery backup operations.

A complete description of the EMM module and a detailed discussion of the functions it performs is presented in Chapter 2.

1.2.5 Cooling System

Cooling for the VAX 8800 system is designed to maintain the internal temperature of the main CPU cabinet within the proper operating range for normal CPU system operation. This is accomplished using an air-moving system powered by a three-phase induction motor. The components of the cooling system include:

- A three-phase 208 Vac, 60 Hz (or 416 Vac, 50 Hz) induction motor with a double-ended shaft
- Quad-inlet, dual-outlet air movers
- A centrifugal blower with two sets of vanes on each end of the motor shaft

The vanes are the forward-curved type and deliver approximately 1800 CFM in free air. As used in the VAX 8800 CPU cabinet, the blower air flow rate is approximately 1100 to 1200 CFM.

The temperature within the CPU cabinet is monitored continuously by the EMM module, which senses the outputs of thermistors strategically placed within the cabinet. The air flow rate within the CPU cabinet is also monitored.

1.2.6 Battery Backup Unit H7231-M

The Battery Backup Unit (BBU) provides the VAX 8800 system with the capability to protect the main CPU memory data in the event that ac utility power is temporarily lost. The BBU consists of:

- A 48-volt rechargeable lead acid battery pack (four 12 V batteries)
- A charging circuit
- A dc-to-dc converter (48 V to 300 Vdc)

The converter converts the 48 volts to 300 volts dc for use in the battery backup mode. The BBU is an option for the VAX 8800 system.

1.3 MECHANICAL CONFIGURATION

The VAX 8800 power system components are housed in two cabinets, the front-end cabinet and the main CPU cabinet. The front-end cabinet contains:

- AC distribution components
- The main circuit breaker (BRKR1)
- The BBU
- I/O expander modules (BA11-A), (CI750)
- Power cables
- 50 Hz transformer (international only)

The components contained in the CPU cabinet are:

- 876A Power Controller
- NBox
- MPS power regulators
- EMM
- Blower system
- 300 V buses
- MPS backplanes
- I/O bulkhead
- Console interface
- Power/Signal distribution cables

Figures 1-4 and 1-5 show the front and rear view of the CPU cabinet, and the location of the power components within the cabinet.

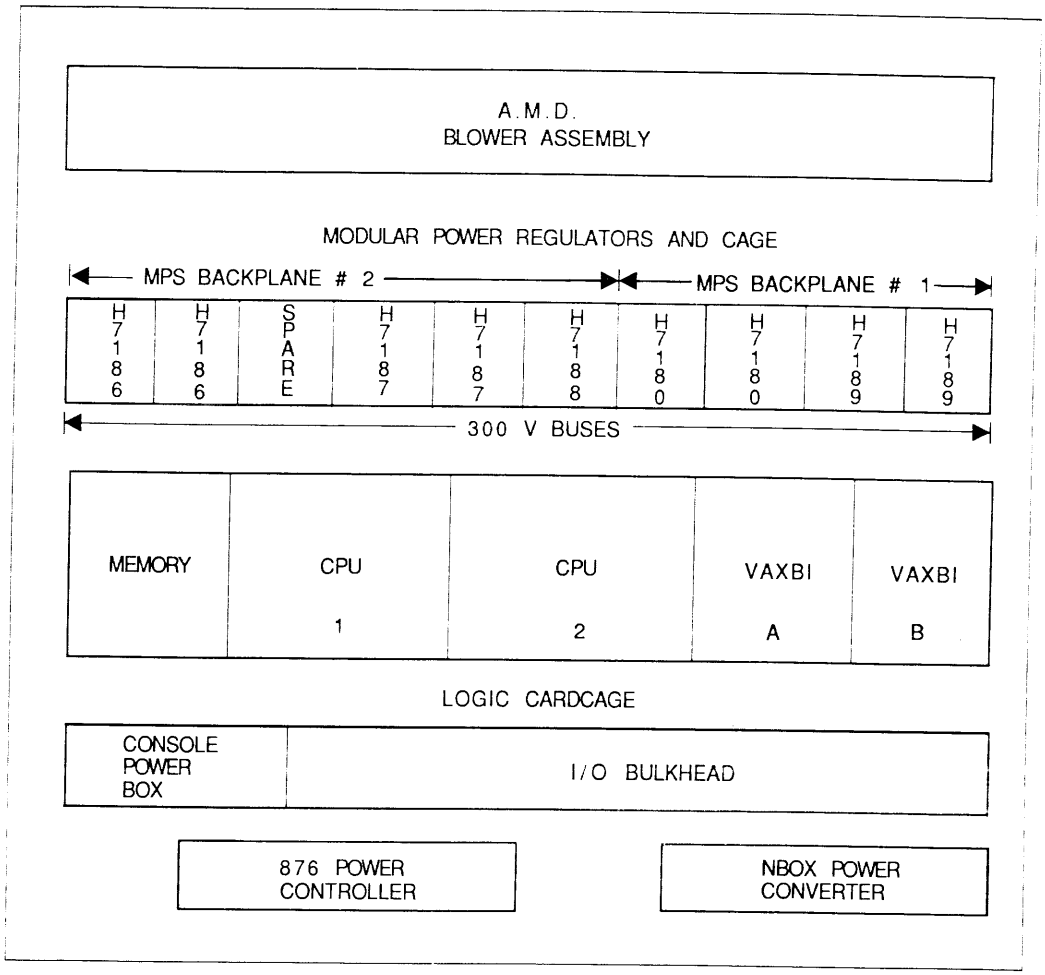
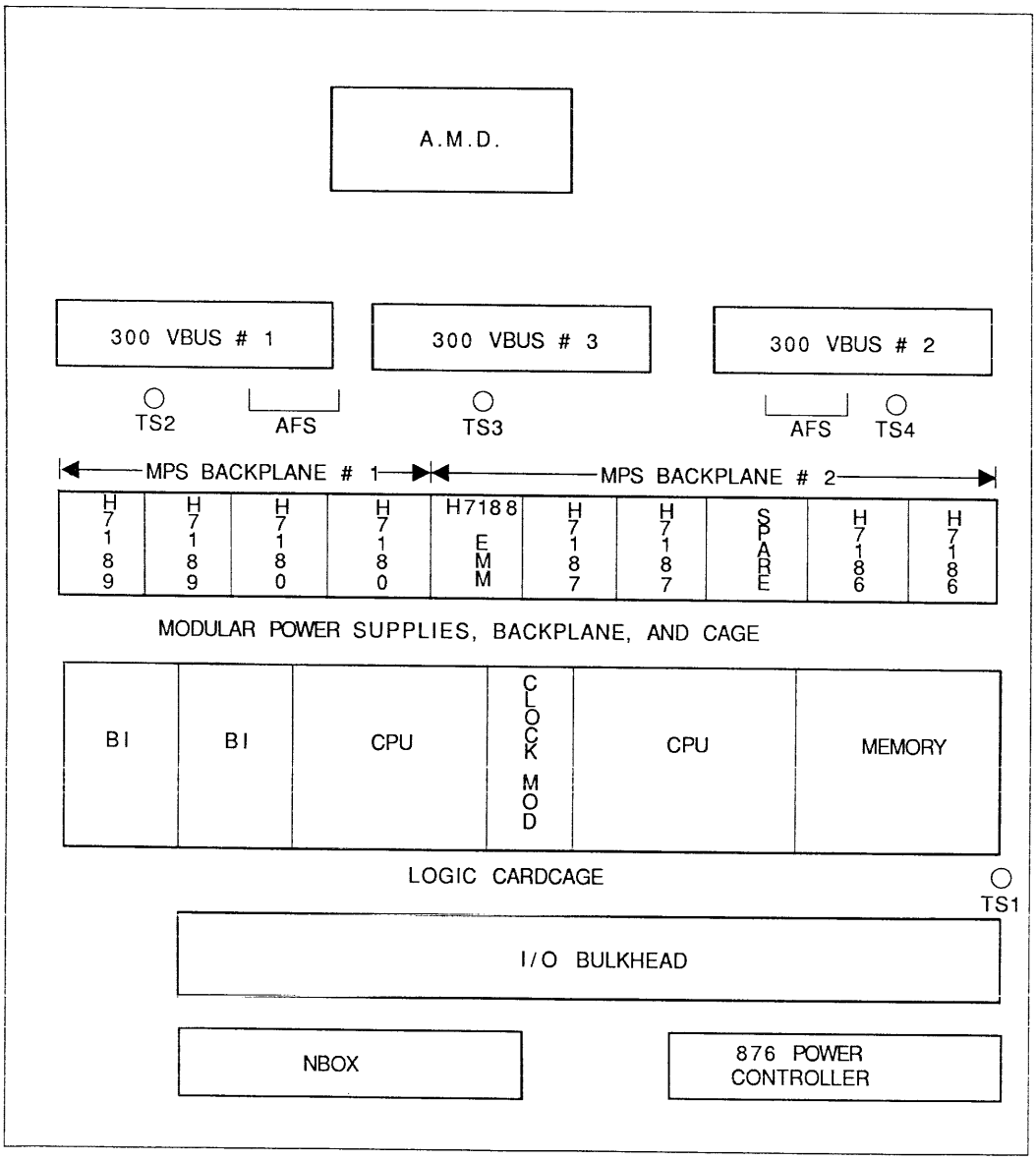


Figure 1-4 VAX 8800 CPU Cabinet - Front View



NOTE: 1. TS = TEMPERATURE SENSE
2. AFS = AIR FLOW SENSE
3. A.M.D = AIR MOVING DEVICE

SCLD-74

Figure 1-5 VAX 8800 CPU Cabinet - Rear View

1.3.1 876A Power Controller

The 876A power controller mounts to the base of the H9650 CPU cabinet at the lower left side, as viewed from the front. The 876A is approximately 10.0 inches wide by 10.0 inches deep by 8.0" high.

1.3.2 NBox Port Conditioner

The NBox mounts to the base of the H9650 cabinet on the lower right side as viewed from the front. The NBox is approximately 19.0 inches wide by 19.0 inches deep by 7.0 inches high and weighs 100 pounds. Connections to the NBox include:

- DIGITAL power control bus
- 300-Vdc power bus
- 300-Vdc storage capacitors
- MPS backplane control logic (EMM)
- Battery backup unit

A minimal amount of cooling is required by the NBox. This is accomplished by placing the NBox in the intake air stream with its heat sinks in the air flow path. Screened openings in the chassis of the NBox allow the moving air to circulate through the chassis.

1.3.3 MPS Modules (Regulators) and Cage

The MPS cage is a sheet metal/tubing structure that, in addition to supporting the MPS modules and the EMM, provides for mounting of the power backplane and the 300-Vdc bus input to the MPS regulators.

The cage is mounted on four corners and bolted to the cabinet directly above the logic cardcage. It has the capacity to contain up to 9 MPS modules and the EMM.

The MPS backplane spans the width of the MPS cage and is fabricated in two sections. The MPS backplanes contain the power control signals and environmental monitoring signals to and from the EMM module.

The MPS regulator modules are inserted into the MPS cage in the designated area. Each module provides regulated dc power that is distributed by means of the MPS backplane or bus bar straps connected to the CPU logic backplanes.

1.3.4 Battery Backup Unit

The optional BBU is mounted to the right side of the front-end cabinet as viewed from the front. The BBU module contains three assemblies:

- The charging circuit
- The dc-to-dc converter circuit
- The 48-volt lead acid battery pack

The BBU module is 7.0 inches high by 15.12 inches deep by 19.0 inches wide and weighs 42 pounds. It requires a minimal amount of cooling.

Therefore, placement of the module in the front-end cabinet satisfies the convection cooling requirements.

1.3.5 Air Flow System

The blower for the air flow system is located at the top of the CPU cabinet, above the MPS cage.

The rotary blower creates a negative pressure within the cabinet, which causes air to be drawn into the cabinet at the bottom and flow through the cabinet components before exiting at the top of the cabinet.

Thermistor sensors (T1-T4) and air flow sensors (AFS) sense the temperature and air flow rate and provide this information to the EMM for monitoring purposes.

1.4 POWER DISTRIBUTION

1.4.1 AC Power

AC utility power is brought into the VAX 8800 system at the rear of the front-end cabinet. Three-phase ac power is cabled to BRKR1, the main system circuit breaker, from which it is connected to terminal blocks for system distribution.

The three-phase ac power is cabled from the terminal blocks to the 876A power controller located in the CPU cabinet. A second three-phase power connection is made to the Dranetz phase monitor port accessed at the front-end cabinet.

A single-phase ac cable is routed to the system console outlet box located in the CPU cabinet.

The 876A distributes three-phase and single-phase ac power to system components by means of power receptacles located on its chassis. Power plugs for the components are inserted into the receptacles provided.

Figure 1-6 shows the receptacles located at the rear of the 876A. Table 1-3 lists the components powered by the 876A and the ac power provided.

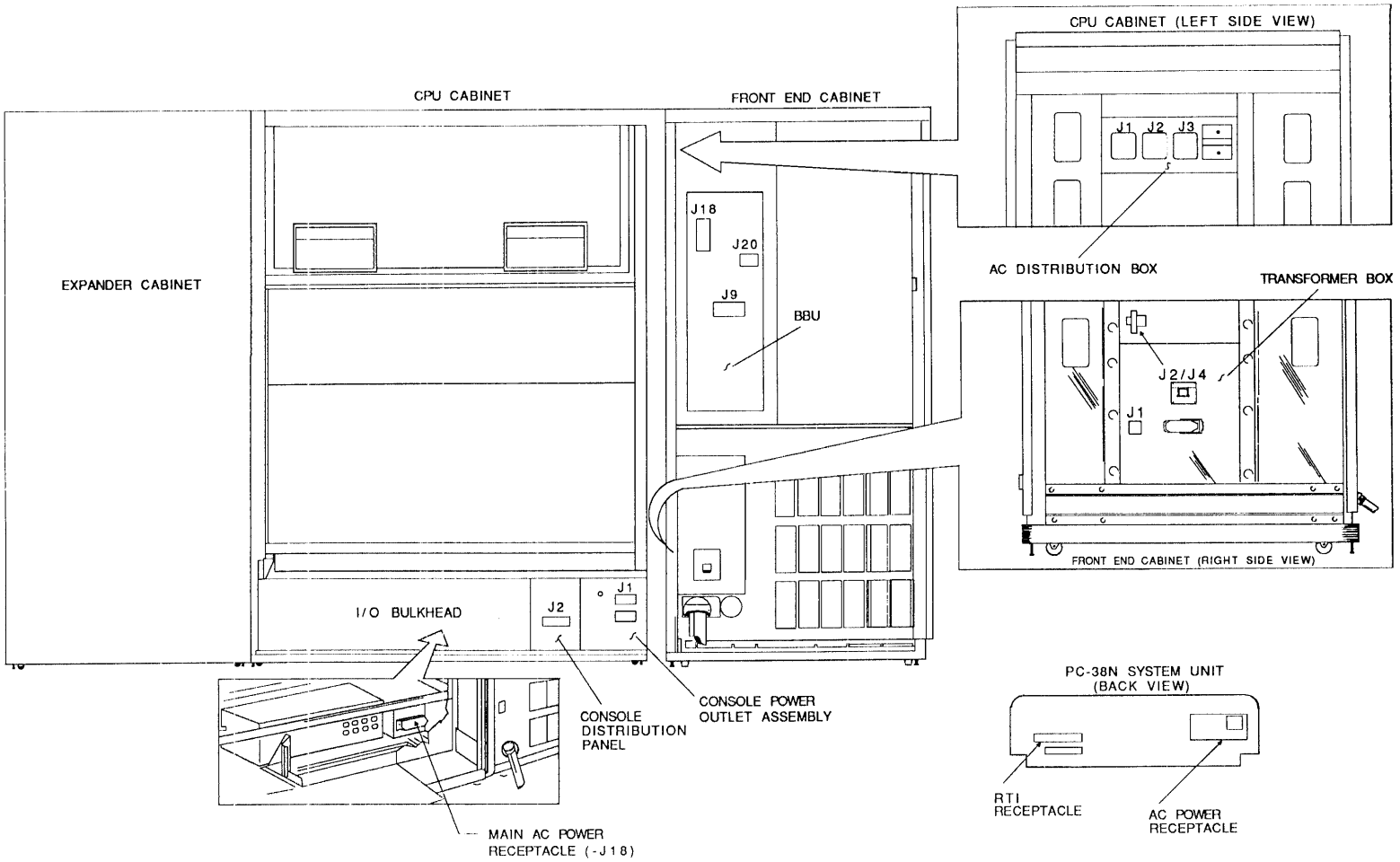


Figure 1-6 876A Rear View Showing Receptacles

Table 1-3 876A Power Distribution

Component	Three-Phase	Single-Phase
NBox	X	X
Blower	X	
BBU		X
CI750	X	

1.4.2 DC Power

The VAX 8800 system uses 300-volt dc power, provided by the NBox (H7170X and Y) or the BBU, to supply the dc regulator modules that produce the dc logic voltages required to operate the system.

During normal system operation, the 300-V power is supplied by the two H7170 power modules located in the NBox. If ac utility power is lost, 300-Vdc power is provided by the BBU controlled through the NBox.

During power dropouts, the 300 Vdc supplied by the BBU is used to support module B, the memory power supply module. A detailed discussion of dc power distribution is presented in Chapter 3.

Figure 1-7 is a block diagram of the components that support the dc power section of the VAX 8800.

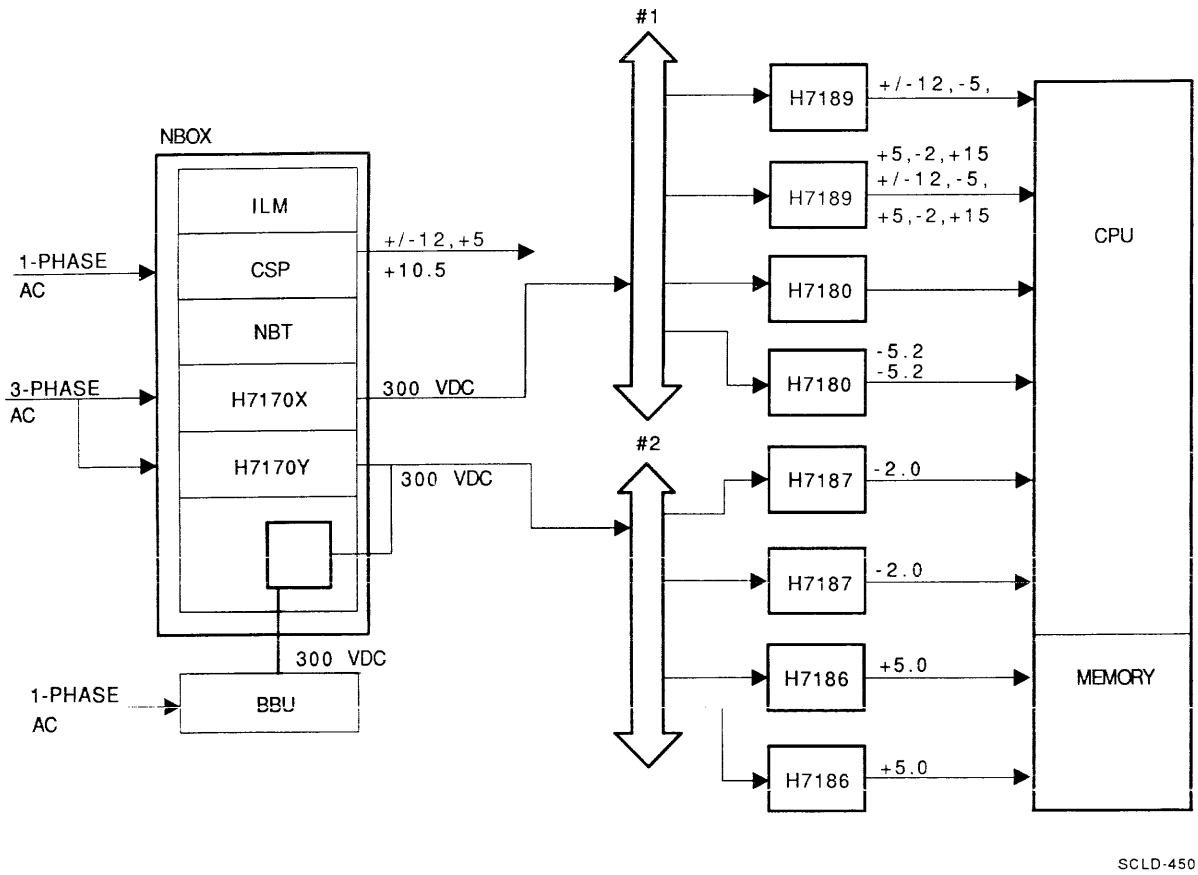


Figure 1-7 DC Power Section Block Diagram

1.4.3 Controls and Breakers

1.4.3.1 Controls -- The primary power controls for the VAX 8800 system are BRKR1, the main circuit breaker, located at the rear of the front-end cabinet, and the ON/OFF (1/0) switch located at the system console.

Placing these two switches in the ON (1) position allows the system to operate in the software-driven mode, enabling the console keyboard to issue commands to the power system by means of the EMM.

There are no front-panel controls for the VAX 8800 system. The PRO-38N power system is the console device for the VAX 8800 that accomplishes all the tasks of a front panel.

1.4.3.2 Circuit Breakers -- There are nine (9) circuit breakers in a fully configured VAX 8800 power system, seven (7) three-phase and two (2) single-phase. Table 1-4 lists the location and characteristics of the circuit breakers.

Table 1-4 VAX 8800 Circuit Breakers

Component	Phases	Breaker ID	Comment
876A PC	3	BRKR2(CB1)	Main Input
"	1	CB 2	BBU
"	3	CB 3	Aux. Out.
"	3	CB 4	H7170X/Y
F.E. Cab.	3	BRKR1(CB1)	Main Sys.CB
"	3	CB 2	Dranetz Ph.Mon.
NBox H7170X	3	CB 1	Module Input
NBox H7170Y	3	CB 1	Module Input
Console	1	CB 1	Console Input

The VAX 8800 power system circuit breaker diagram is shown in Figure 1-8.

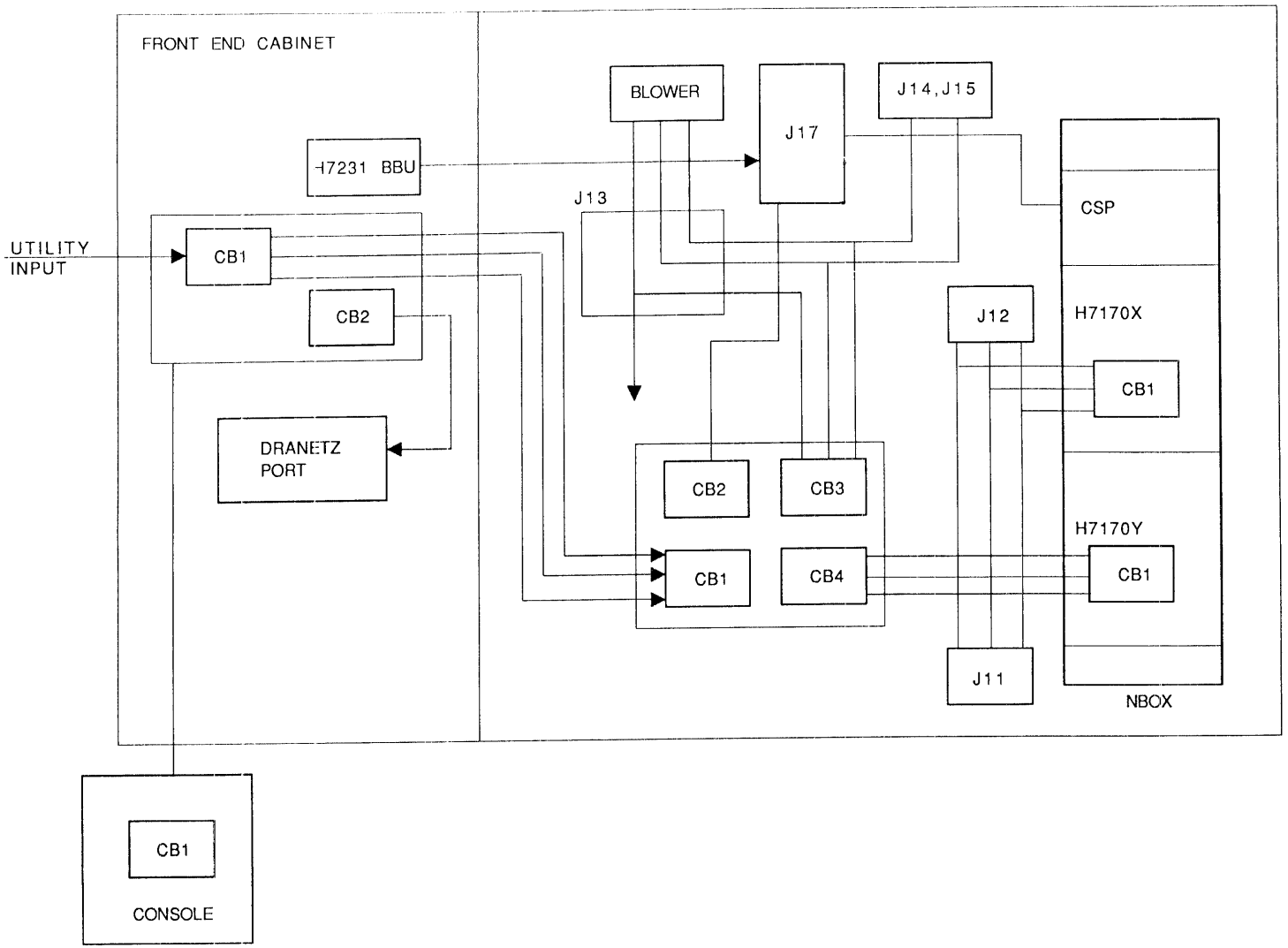


Figure 1-8 VAX 8800 Power System Circuit Location Diagram

1.5 AC POWER SPECIFICATIONS

The following is a summary of the VAX 8800 power requirements.

1.5.1 Electrical Requirements

1.5.1.1 AC Power Sources

Domestic

120/208 Vac, 60 Hz, 60 A, 3-Phase, 5-Wire WYE connected service.

International

240/416 Vac, 50 Hz, 30 A, 3-Phase, 4-Wire Delta connected service.

Taps are available on the 50 Hz transformer to accommodate 380-Vac and 440-Vac inputs. Taps must be selected and wired before power is applied.

Refer to the VAX 8800 System Installation Guide (EK-8800I-IN) for additional details.

Line Voltage

60 Hz equipment - 156-222 V rms line-to-line, 208 V nominal.

50 Hz equipment - 312-444 V rms line-to-line, 416 V nominal.

Line Current

60 Hz equipment - 50 Amps max. @ 156 V rms

50 Hz equipment - 30 Amps max. @ 312 V rms

Power Factor

60 Hz - 0.65 minimum

50 Hz - 0.85 minimum

Power Consumption

6500 Watts maximum; Maximum Heat dissipation 22.2K BTU/hr

Environmental Requirements

DIGITAL STANDARD 102 for Class A environments

Temperature

Ambient Temperature Range - Operating - Centigrade - 15 to 32 degrees

Fahrenheit - 59 to 90 degrees

Storage - Centigrade - -40 to +66 degrees

Fahrenheit - -40 to +151 degrees

Humidity

20 to 80 % relative humidity

Altitude

Sea Level to 8000 feet

Weight

1700 lbs combined weight for both cabinets

Dimensions

30 inches d x 60 inches h x 73.5 inches combined width

1.6 FAULT AND STATUS INDICATORS

The only visible indication of system fault and status is that available to the system operator. The console monitor displays messages to inform the operator when faults occur and the status of the modules in the power system.

Each of the power system components contain front-panel indicators that display fault and status. However, these displays are not visible during normal operation.

The outer front-panel covers of the CPU and front-end cabinets must be removed to view the fault and status indicators.

Fault and status indicators for the power system components are described in the following paragraphs.

1.6.1 876A Power Controller

The 876A power controller has three "POWER" lamps located on the upper right side of the front panel, which light when ac facility power is applied.

Figure 1-9 shows the front panel of the 876A PC.

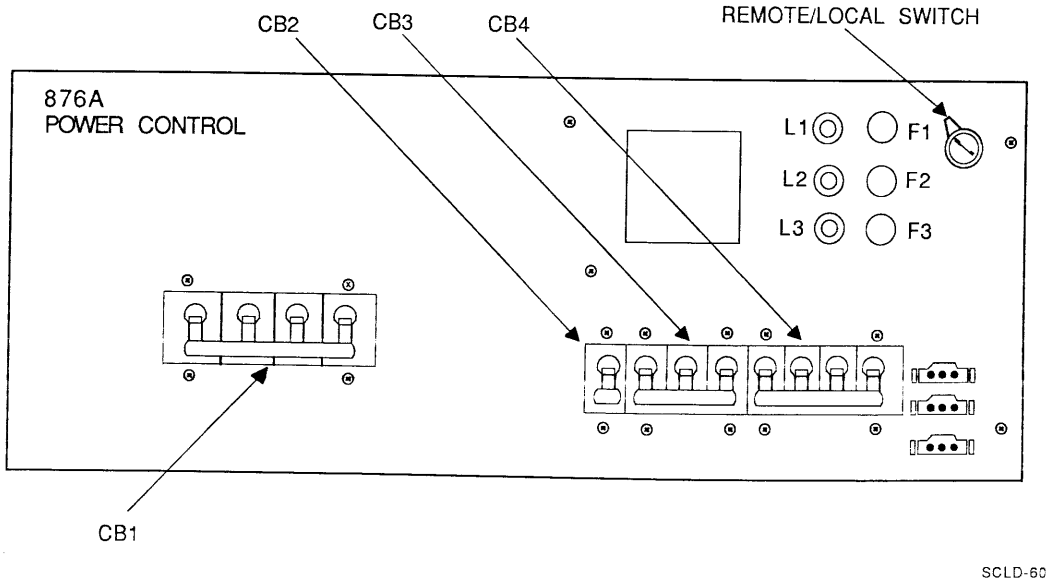


Figure 1-9 876A Power Controller Front Panel

1.6.2 NBox

Figure 1-10 shows the front-panel location of the NBox indicators. Table 1-3 defines the visual indicators provided.

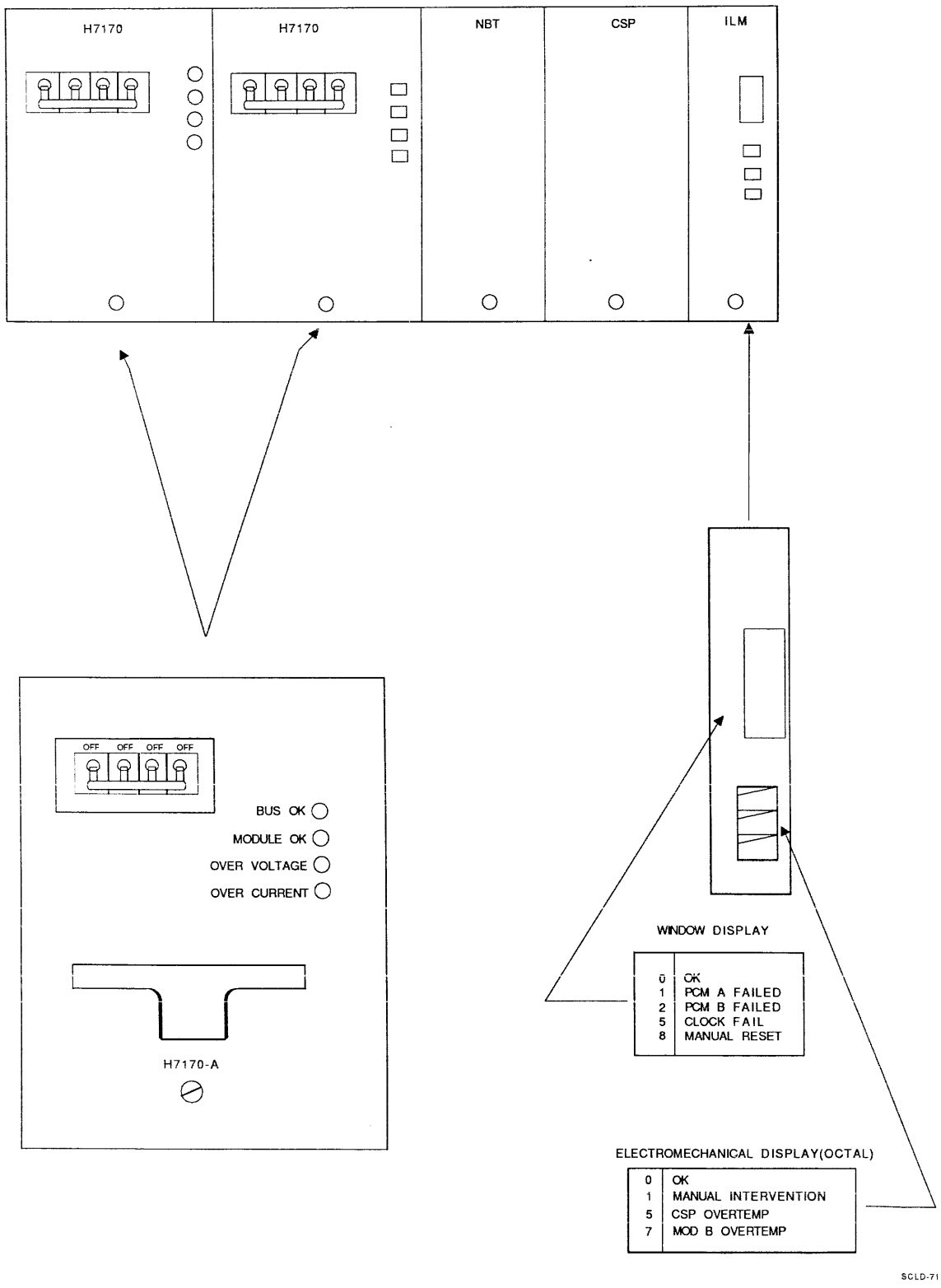


Figure 1-10 NBox Front-Panel Indicators

1.6.2.1 H7170 Built-In Test Equipment -- The front panel of the H7170 has four indicators to display the current operating status of the module. The indicators are colored LEDs that light to indicate status. Table 1-5 lists the indicators and the related indication.

Table 1-5 H7170 Status Indicators

Indicator/Color	Indication
1. GREEN	BUS OK - Bus voltage > 165 Vdc
2. GREEN	MODULE OK - No problems
3. RED	OVERVOLTAGE FAULT - Crowbar has occurred
4. BLINKING YELLOW	OVERCURRENT FAULT

1.6.2.2 ILM Built-In Test Equipment -- The ILM front panel contains two types of fault/status indicators: electro-mechanical magnetic latches and a single-digit LED display.

The electro-mechanical indicators located below the digit display, are defined below:

- Binary 1 = Manual Intervention
- Binary 5 = CSP OT
- Binary 7 = MOD B OT

The single digit LED BITE display on the ILM front panel indicates fault conditions as dfined below:

- 1 = H7170X failed
- 2 = H7170Y failed
- 5 = CLOCK FAIL
- 8 = MANUAL RESET

The only fault indication sent to the EMM for the console is a MOD B OT failure. All other fault indications apply to the NBox.

1.6.3 Modular Power Supply Regulators

Figure 1-11 shows the location of the front-panel indicators for the modular power regulators. All power regulators have the same status indications, as defined in Table 1-5.

Table 1-6 MPS Regulator Indicators

Indicator	Indication	Definition
Green LED	Module OK	The power supply module is operating properly, the output voltage is within regulation range, and no other faults exist.
Red LED	Overvoltage	The module voltages have crowbarred.
Yellow LED (Blinking)	Overcurrent	The module's output current is above its rating.

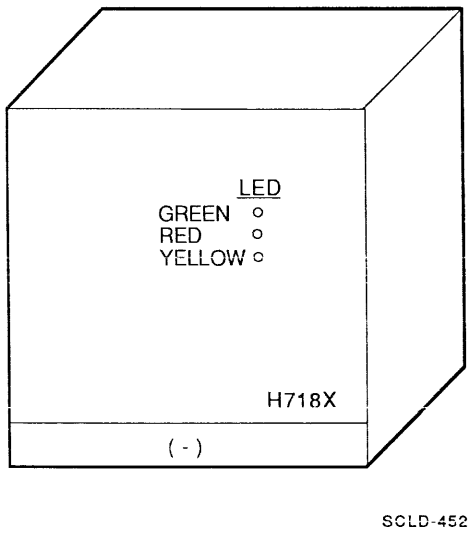


Figure 1-11 Indicators for the Modular Power Regulators

1.6.4 Environmental Monitoring Module

Figure 1-12 is a diagram of the EMM front panel showing the location of the front-panel components. The red LED indicators signal a KEY FAULT.

The four magnetic latches above the LEDs are used to indicate the fault conditions defined in Table 1-8. The octally coded latches are weighted as shown in Figure 1-12.

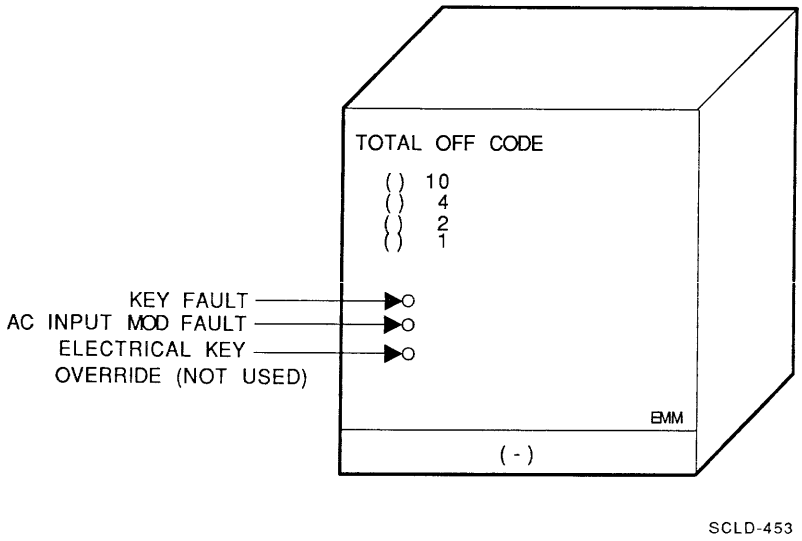


Figure 1-12 EMM Front-Panel Indicators

Table 1-7 EMM Magnetic Status Indicator Codes

Coded Readout (Octal) Black = 0 Yellow = 1	Indication	Device
000	No failure	
001	Not used	
010	Overtemperature	Module B H7186
011	Overtemperature	Module C H7186
100	Overtemperature	Module D H7187 (either one)
101	Overtemperature	Module E H7180 (either one)
110	Overtemperature	Module F H7189
111	Overtemperature	Module H H7189
1000	Not used	
1001	Not used	
1010	Not used	
1011	Not used	
1100	Overtemperature	Cabinet sensors (any one of four)
1101	Not used	
1110	Not used	
1111	Not used	

1.6.5 System Console Device

Operation and control of the VAX 8800 system is performed by the system console, a modified PRO-38N that allows the operator to interact with the system. A complete discussion of the PRO-38N system console, including its specifications and performance characteristics, is provided in Section 3 of this manual.

CHAPTER 2
FUNCTIONAL DESCRIPTION

2.1 INTRODUCTION

Chapter 1 of this document introduced the components that make up the VAX 8800 power system. This chapter describes the power system at a block diagram and flowchart level. More detailed component descriptions are presented in Chapter 3.

2.2 POWER SYSTEM BLOCK DIAGRAM

Figure 2-1 is a functional block diagram of the VAX 8800 power system showing the major components of the system and the interconnections.

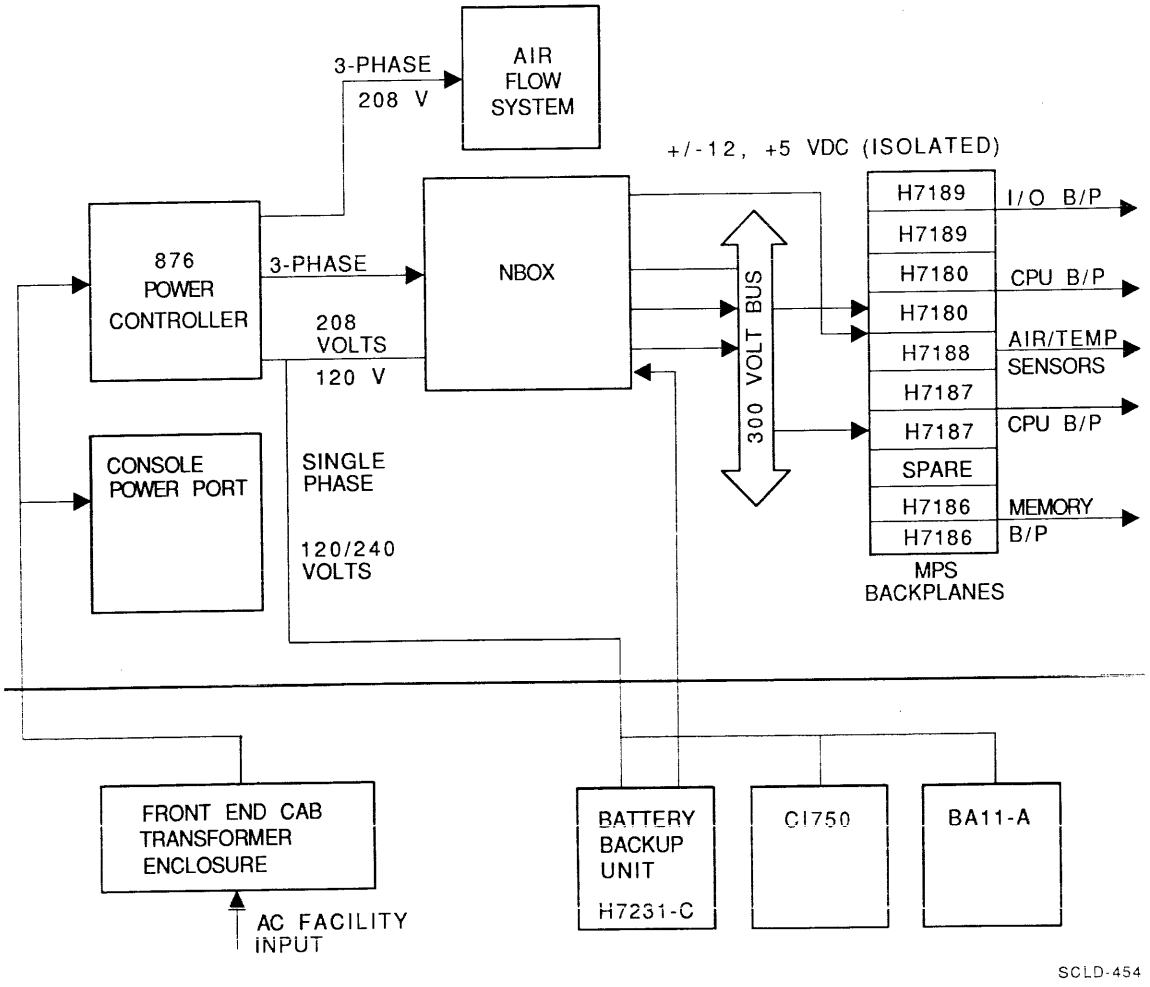


Figure 2-1 VAX 8800 Power System Block Diagram

2.3 SIMPLIFIED OPERATION

Utility ac power is brought into the VAX 8800 system through the front-end cabinet. The ac passes through the main system circuit breaker, BRKR1, and connects to the ac distribution block.

Connections to the distribution block include the following:

- Console power port
- Dranetz phase monitor port (via Circuit Breaker 2)
- 876A power controller

The console/printer power port and the 876A are located in the CPU cabinet.

The console power port is a single-phase ac power receptacle to provide unswitched ac power for the console subsystem and printer (LA50).

The Dranetz power port provides a three-phase ac receptacle into which the Dranetz voltage monitor plug is inserted.

The 876A is described in the following section.

2.3.1 876A Power Controller

The 876A power controller interfaces the three-phase ac utility power input to the VAX 8800 power system. The functions of the 876A are:

1. Distribute 3-phase and 1-phase ac power to system modules
2. Filter ac input power
3. Control application of ac power to modules (K1 power switch)
4. Provide ac reptacles for system modules
5. Protect system modules from power fault conditions (BRKR 2)
6. Interface with DIGITAL power control bus

The 876A, the main VAX 8800 power control module, distributes switched and unswitched ac power to the three-phase and single-phase modules listed in Table 2-1.

Table 2-1 876A Power Distribution

Module	AC Phases
NBox	3, 1
Blower	3
BBU	1
BA11-A	1
Auxiliary port	1

2.3.2 NBox

The NBox is a power converter assembly containing the following modules:

- 2 H7170 power converters
- ILM
- CSP
- NBT

Each of the above modules is described in the following paragraphs.

2.3.2.1 H7170 Power Converter -- The H7170 power converter converts the three-phase ac input power to 300-Vdc output power for use by the MPS dc power regulators.

There are two H7170 PCs in the NBox (NPC), designated H7170X and H7170Y. The outputs of these units supply the 300-Vdc buses connected to the regulators.

When the ac voltage input to either H7170 falls below 156 V rms, both of these units are disabled. Battery backup power from the BBU supplies 300-Vdc power to regulator B for main memory bias support.

All other regulators are disabled when AC LO occurs.

2.3.2.2 Control Start-up Power Module (CSP) -- The CSP receives unswitched single-phase ac power from the 876A power controller and creates the following dc bias voltages:

- +5 V
- +/-12 V
- +15 V
- +10.5 V

The CSP output voltages are used as shown in Table 2-2.

Table 2-2 Modules Using CSP Bias Voltages

Voltage	Modules Used On	Purpose
+5 V	ILM, EMM	Bias
+/- 12 V	ILM, EMM	Bias
+10.5 V(3)	All MPS Regulators	Startup
+15 V	CLK/CNSL pcb	Bias

2.3.2.3 Interface Logic Module (ILM) -- The ILM module performs the following power system functions:

1. Provides interface logic for the power-up sequence
2. Controls logic for the operation of the BBU
3. Generates clocks for MPS regulators
4. Initiates BBU power output to regulator B by means of the NBox
5. Interfaces with the DIGITAL control bus to control utility power

2.3.2.4 New Box Translator Module (NBT) -- The NBT module provides logic signal conversions and combination logic functions on signals used during the power sequences. The NBT also provides the system with "FAIL SAFE," which prevents BBU operation in the event that a system CB is turned off, or tripped.

2.3.3 Modular Power System (MPS)

The modular power system includes the power regulators that provide the dc voltages necessary to operate the following:

- Main CPU
- Memory
- VAXBI modules

The power regulators are powered from the 300-volt dc power bus, which is split into two sections.

The regulators have the following common characteristics:

- Pulse width modulation (PWM)
- 50 kHz clock from ILM
- Line and load regulation
- Remote voltage sensing

The EMM module, which monitors the output of the dc regulators and the environment in the CPU cabinet, is located in the MPS along with the eight (8) regulated power supplies.

The MPS module cage is a mechanical structure internal to the CPU cabinet that supports:

- The modular power supplies
- The EMM module
- The 300-V buses
- The MPS backplanes

The MPS cage is located above the main CPU cardcage.

Table 2-3 lists the regulator modules by part designation, dc output voltage, and the area of the computer system they service.

Table 2-3 Voltage Regulators

Regulator Module	DC Output (volts)	Logic Backplane(s)
H7186 (B)	+ 5.0 Battery	Memory
H7186 (C)	+ 5.0	Memory
H7187 (D)	- 2.0	CPU
H7180 (E)	- 5.2	CPU
H7189 (F, H)	+/- 12.0, - 5.2, + 5.0, - 2.0, +15	VAXBI

The regulator output voltages are connected to the CPU backplanes using the following:

- Flexible cable
- Laminated bus straps
- CPU backplane etches

The laminated bus straps are used to minimize voltage drop when making high current connections with minimum voltage drop.

The MPS backplane connections to the CPU, memory and VAXBI backplanes, viewed from the rear of the cabinet, are shown schematically in Figure 2-2.

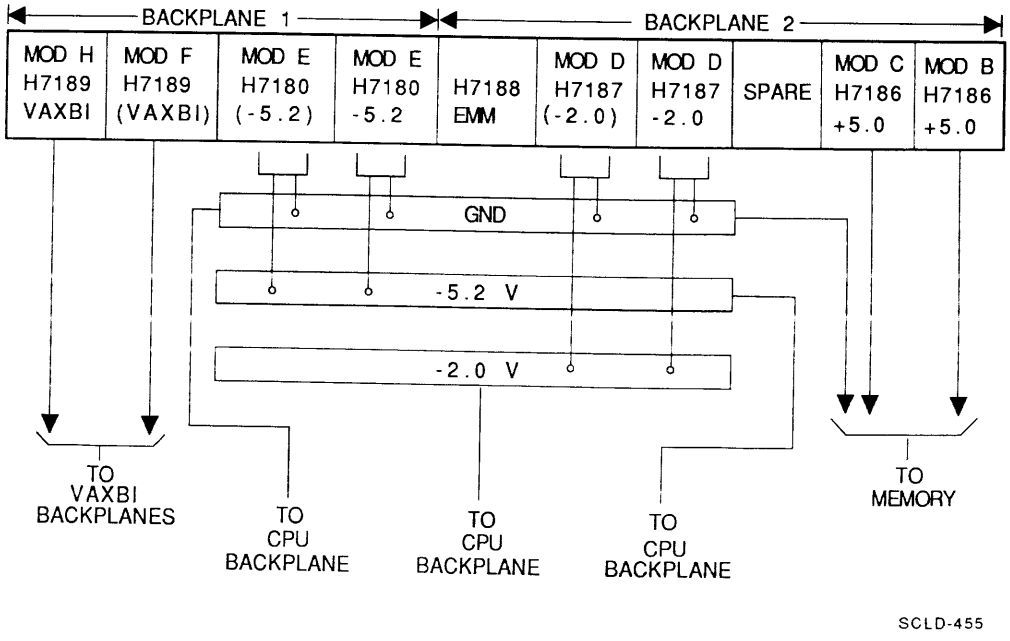


Figure 2-2 MPS Backplane Configuration (Rear View)

2.3.4 Battery Backup Unit (BBU Model H7231-M)

The H7231-M Battery Backup Unit (BBU) for the VAX 8800 power system consists of:

- A battery charger (H7230A)
- A dc-to-dc converter (H7240A)
- A 48-volt lead acid battery pack

The BBU provides battery backup power for the CPU memory modules when a utility power failure occurs during system operation.

The BBU output supports regulator module B, the only active regulator during short-term utility power drop outs.

Charging for the 48-volt, 5 Ah battery pack occurs during normal system operation by means of the unswitched 120-volt, single-phase ac voltage that is fed to the BBU from the 876A power controller.

The ac voltage is applied to the H7230 charging circuit that controls the current to the batteries.

The level of charge on the batteries is monitored to prevent overcharging and to signal the ILM monitor of its charge status.

The BBU is capable of supporting the memory modules for a minimum of 9 minutes from a fully charged battery. It is intended to provide power for the memory modules during short-term power losses that occasionally occur.

The output circuit for the BBU is a dc-to-dc converter module (H7240) that converts the 48-volt dc from the battery pack to 300 volts dc at 700 mA.

The 300-volt output is applied to the NBox where it is interfaced with the 300-volt dc bus used to feed the power regulator modules.

During a power loss period, the 300-volt dc is applied to all regulator modules, but only module B is enabled.

2.3.4.1 BBU Control -- Control of the BBU's power backup operation is performed by the ILM module, which issues commands to the BBU in response to a BBU request from the EMM module.

The AC LO signal is sent to the EMM in response to an AC LO signal received from a power line monitor in the NBox. AC LO occurs when the ac line voltage falls below 156 volts rms. The normal ac voltage level phase-to-phase is 208 volts rms.

Upon receipt of the AC LO signal, the EMM signals the ILM to provide battery backup power (MODULE ENABLE L).

The ILM responds to a BBU request by:

1. Enabling BBU power output
2. Asserting (ILM RQUEST RTN) to the EMM

Module B must indicate that it is functioning correctly by asserting the signal (MODULE OK H).

The TOTAL OFF BUS signal must not be asserted.

If the conditions above are met, the BBU provides 300 volts @ 700 mA to the 300-volt dc bus.

The EMM, in response to a signal (MODULE ENABLE L) from the BBU, initiates a programmed sequence to disable all regulators except Module B.

The BBU continues to supply 300-volt bus power (maximum time 10 minutes) until ac power returns.

The return of ac power is signaled by AC LO deasserted, Mod J reporting OK, and the EMM requesting control of Mod B. If all these are true, then the ILM will turn off the BBU.

If system power is completely shut down by the system operator, AC LO will not be deasserted.

Figure 2-3 is a functional block diagram of the battery backup subsystem showing the control logic signal flow and the power interconnects associated with the BBU module.

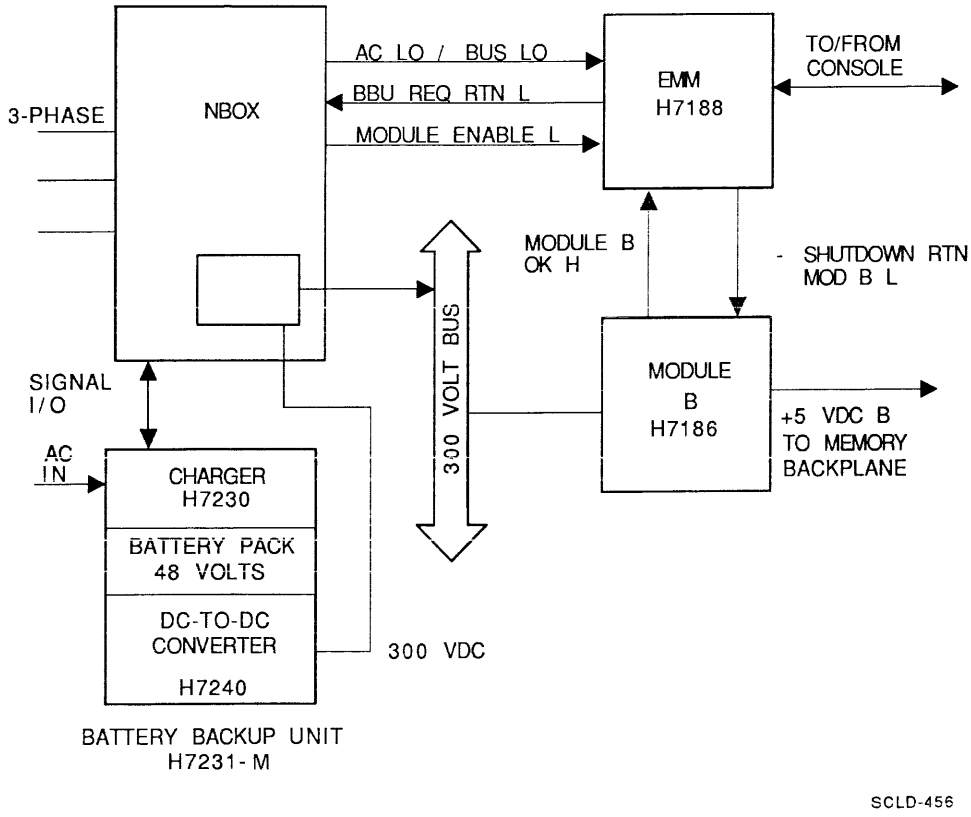


Figure 2-3 Battery Backup Subsystem Functional Block Diagram

2.3.5 Environmental Monitoring Module (EMM)

The environmental monitoring module is a multipurpose logic module that performs the following functions:

- Monitors the status of the modular power supplies
- Monitors the environmental status of the CPU cabinet
- Communicates logically with the system console
- Enables/disables the regulators
- Communicates with the ac power system
- Verifies correct CPU pc board installation

The EMM is an 8-bit microprocessor-based (8085) module that contains the following microprocessor system support chips.

- RAM
- ROM
- I/O
- RS232 communications

The RAM chip provides temporary storage for data and instructions sent to the EMM from the system console over the serial communications line.

The ROM chip contains the preprogrammed instruction set needed to perform routine EMM tasks. These include:

- Regulator on/off control
- AC system monitoring
- Regulator output voltage monitoring
- Regulator output margin testing
- Regulator temperature monitoring
- CPU cabinet environmental monitoring
- BBU on/off control

Figure 2-4 is a functional block diagram of the EMM module showing the communications interface with the system console and the I/O interfaces to the power system monitoring and testing devices.

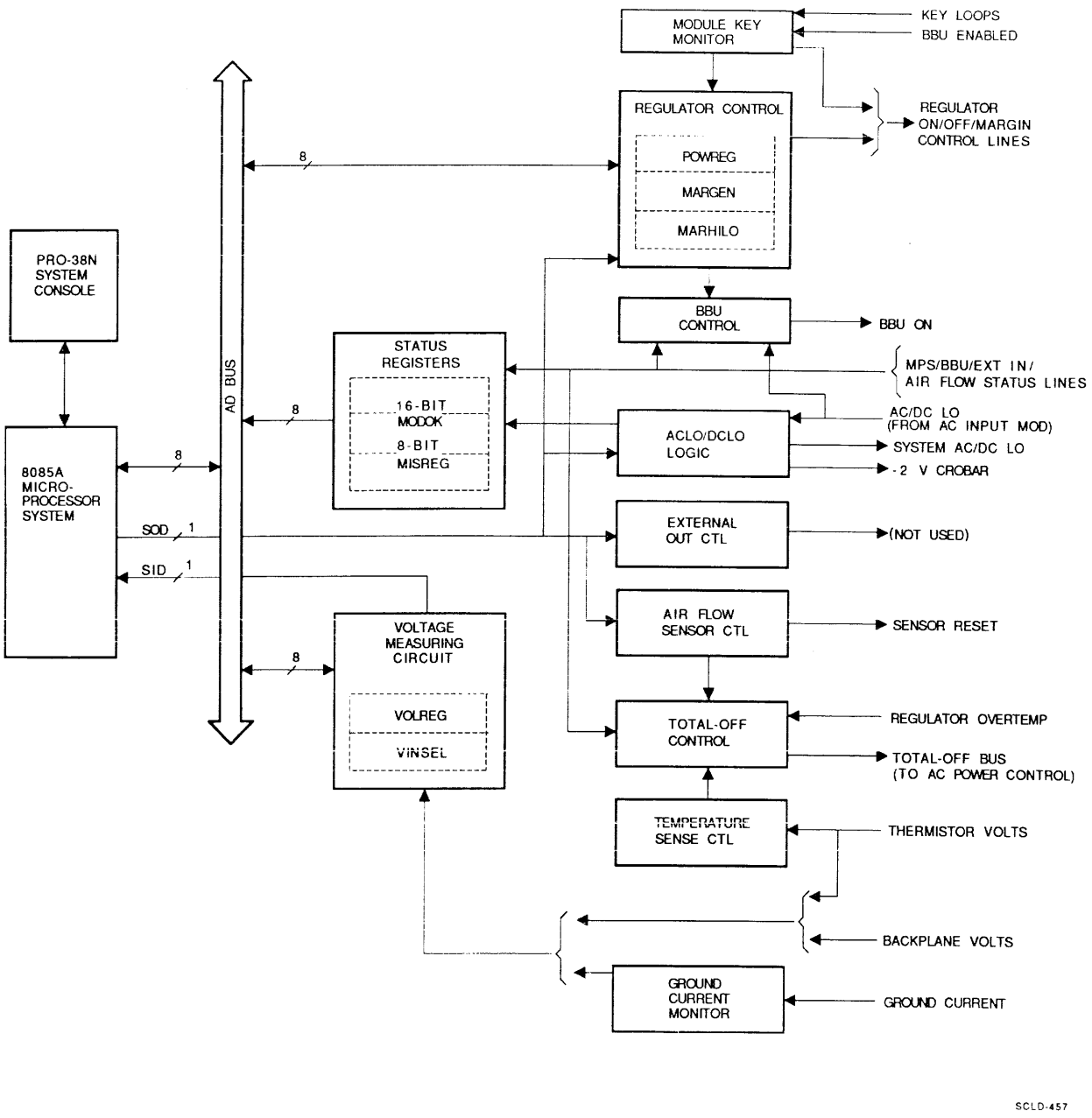


Figure 2-4 EMM Functional Block Diagram

The serial communications link to the system console is vital to the functioning of the VAX 8800 power system.

The system console is the primary user interface to the VAX 8800 system.

The EMM as power system monitor, continually communicates with the system console regarding:

- Function and status of the power system
- Power system fault conditions

The EMM signals the console when a fault occurs, and receives instructions regarding the corrective action to be taken.

2.3.6 Power System Monitoring

The main functions of the EMM are:

1. Monitoring the power system components
2. Notifying the system console of malfunctions

The monitoring is done using devices that provide an I/O interface with the 8085 computer.

As shown in Figure 2-4, the I/O section of the EMM is located on the right side of the block diagram. The monitored I/O signals include the following:

- Key faults
- Regulator control
- BBU control
- MPS/BBU/EXT. IN status
- AC/DC LO
- Regulator overtemp
- CPU temperature

The following paragraphs describe the input and output signals briefly, indicating their purpose and the resulting fault response by the EMM.

2.3.6.1 Key Monitoring -- The PC boards in the CPU cage are keyed to assure correct insertion into the CPU PCB slots. The key loop circuit verifies that the PCBs are correctly inserted. Failure to satisfy this requirement causes:

1. Power-up sequence halted
2. Reinitialization of the microprocessor (8085)

The power-up sequence cannot proceed until the fault is corrected.

2.3.6.2 Regulator Control -- The EMM provides control logic for turning the dc voltage regulators on and off, margin testing, and monitoring the dc output voltage(s) of the regulators.

The internal temperature of the modules is also monitored to prevent overheating and thermal runaway.

All of the regulators are fixed output design.

2.3.6.3 BBU Control -- The EMM communicates with the ILM to monitor BBU status and to request BBU output when an AC/DC LO signal has been received by the EMM.

The EMM, in response to the AC/DC LO signal:

1. Sends a BBU request to the ILM

2. Signals the system console that the AC/DC LO signal has occurred

The ILM enables the BBU (see power-down sequence).

2.3.6.4 Air Flow Status -- Incorporated into the VAX 8800 system design is an air moving system that continually forces air throughout the CPU cabinet for cooling.

The air flow rate is continuously monitored. If the air flow rate falls below the specified minimum, the EMM notifies the system console.

The console responds by initiating a system shutdown procedure.

2.3.6.5 AC/DC LO Signals -- AC LO and DC LO are two fundamental power system fault signals sent to the EMM when a fault condition occurs in the utility power system.

An AC LO indication signals the EMM that a low ac line voltage exists, in either a single-phase or in all three phases of the ac line voltage to the H7170s.

When the EMM receives an AC LO indication, it immediately:

- Notifies the system console that a fault exists
- Requests BBU power through the ILM
- Notifies the CPU, NBI, and memory

The console displays the AC LO condition at the console monitor and initiates a system power-down procedure to disable the system.

2.3.6.6 Regulator Overtemp. and CPU Cabinet Temperature (Thermistor Volts) -- The EMM continuously monitors the thermal environment within the CPU cabinet while the VAX 8800 system is operating.

There are four thermistors in the CPU cabinet:

- T1, located at the forced air input to the CPU cage
- T2, T3, T4, located at the air flow output between the CPU cage and the MPS module cage

The thermistors are temperature-sensitive resistors that vary inversely with temperature. The thermistor volts signal sensed by the EMM is a translation of the value of thermistor resistance, and therefore, an indication of the air temperature in the location of the thermistor.

The temperature values reflected by the thermistors located in the CPU cabinet are continuously monitored by the EMM and compared to the specification limits.

When the EMM senses a temperature outside the upper limits set (red zone), it sends a message to the system console. Then it initiates a power-down (TOTAL OFF) for the system, tripping BRKR2 in the 876A.

The REGULATOR OVERTEMP signal is a logic indication that an individual dc voltage regulator has exceeded its operating temperature limit.

The regulator temperature sensors are bimetallic devices that close when the design temperature is exceeded.

All of the regulators in the VAX 8800 system contain an overtemperature indicator that generates a logic signal to the EMM when the module temperature exceeds its upper limit.

When this occurs, the EMM notifies the system console, initiating a system (TOTAL OFF).

2.4 POWER SEQUENCES

Powering up the VAX 8800 system is a sequential process that must proceed in an orderly manner if successful system operation is to be achieved.

2.4.1 Circuit Breakers

Prior to initiating the power-up sequence for the VAX 8800 system you must verify that all of the circuit breakers (CB) in the system are in the "ON" position.

The main CB, BRKR1, which enables utility power to the system, is the last to be turned on at the start of the power-up sequence.

There are nine (9) CBs in a fully configured VAX 8800 system: seven (7) three-phase ac CBs and two (2) single-phase CBs.

The 876A power controller contains four CBs.

All CBs except BRKR1, located at the rear of the front-end cabinet must be manually set as part of the prepower-up preparations.

Table 2-4 lists the circuit breakers contained in the system and the modules in which they are located. The function of the CBs are also indicated.

Table 2-4 System Circuit Breakers

Part Designation	Phases	Amp. Rating	Comment
CB1 (BRKR1)	3	50	Main system CB (front-end cab) to phase monitor
CB2	3	2	
CB1	1	?	Console input
CB1 (BRKR2)	3	40	876A input
CB2	1	20	876A for H7231-M
CB3	3		876A to blower
CB4	3	30	876A to NBox (H7170s)
CB1	3	6	H7170 input
CB1	3	6	H7170 input

The power sequences presented in the following section are discussed in conjunction with the power flowcharts shown in Figures 2-5 through 2-8.

There are four power sequences for the VAX 8800 system:

1. Routine powerup
2. AC powerup from BBU operation
3. Powerdown from console command
4. Powerdown from power interrupt with BBU

2.4.2 Summary of Power-Up Sequence

The first step in the power-up sequence is to manually put all CBs in the "ON" position. BRKR1, the main circuit breaker for the system, is put on last.

The three-phase ac utility power enters the front-end cabinet and is distributed to the power controlling components of the power system.

Three-phase ac power is sent to:

- Dranetz phase monitor inside the front-end cabinet
- 876A inside the CPU cabinet

Single-phase unswitched ac is also sent to the system console ac port to power up the console/printer.

NOTE

When power is applied to the PRO-38N system console unit, the console proceeds to execute its own powerup and initialization routine. See the following publications for details of the console start up procedure.

1. Section 3 of this manual
2. VAX 8800 System User's Guide

Inside the CPU cabinet, unswitched single-phase ac is sent from the 876A to:

- The BBU
- The CSP module (located in the NBox)

After a delay of less than a minute, the CSP module generates +5 Vdc, and +/-12 Vdc bias voltage for the following modules:

- NBT
- ILM
- EMM

A +10.5 Vdc bias for the MPS regulator start-up power is also generated by the CSP.

The NBT module, also located in the NBox, sends a FAIL SAFE ENABLE signal to the BBU. In response, the BBU sends a BBU AVAIL L signal to the EMM.

Upon receiving the +5 V and +/-12 Vdc bias voltages from the CSP module, the ILM module powers on, performs a selftest, and displays the number "0" on its numerical display, if "OK."

Similarly, the EMM module powers on after receiving the +5 V and +/-12 Vdc bias voltages from the CSP.

The key circuit, which checks the correct placement of CPU PCBs, signals the EMM if no faults are detected, indicating that the power-up sequence may proceed.

If a fault is detected, the key fault LED lights and the power-up sequence halts until the problem is resolved. During a fault, the console cannot communicate with the system.

The control console issues a POWER ON command to the EMM, which is acknowledged by the EMM if no key faults exist.

NOTE

If the console has been set to the AUTOPOWERON mode, it will power up automatically when utility power is applied. Otherwise, the console operator must type the POWER ON command to enable the power system.

The CSP provides +15 V to:

- AIR FLOW SENSOR PCB
- CLK/CNSL PCB

A POWER REQUEST signal is also sent to the 876A power controller power switch, S1, from the ILM to enable ac power to:

1. Blower motor
2. NBox (H7170s)
3. BA11-As

The H7170s generate 300 Vdc to supply the 300-Vdc buses feeding the regulators in the MPS rack. The regulators are enabled sequentially by the EMM, with module D being the last to be enabled.

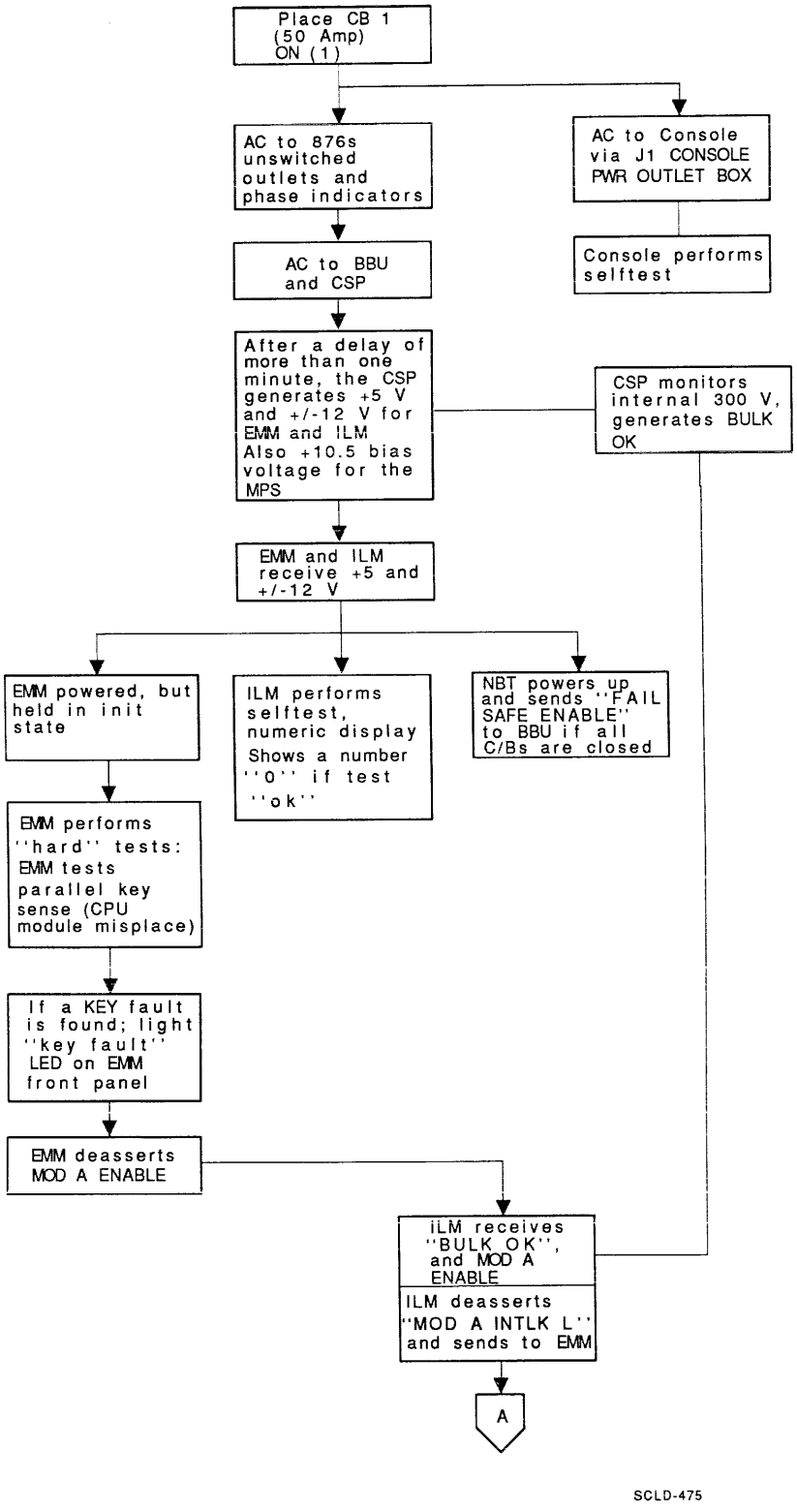


Figure 2-5 System Power-Up Flowchart (Sheet 1 of 4)

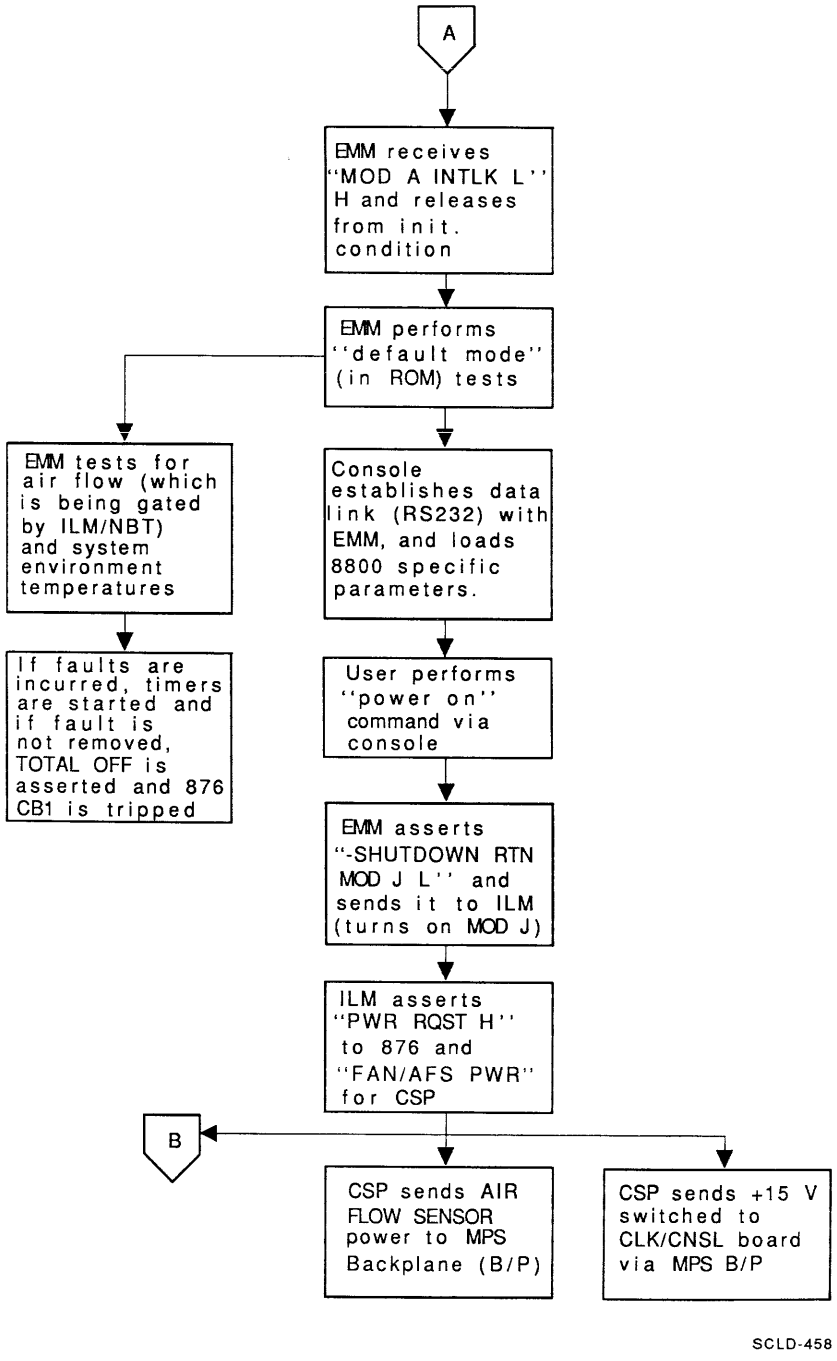


Figure 2-5 System Power-Up Flowchart (Sheet 2 of 4)

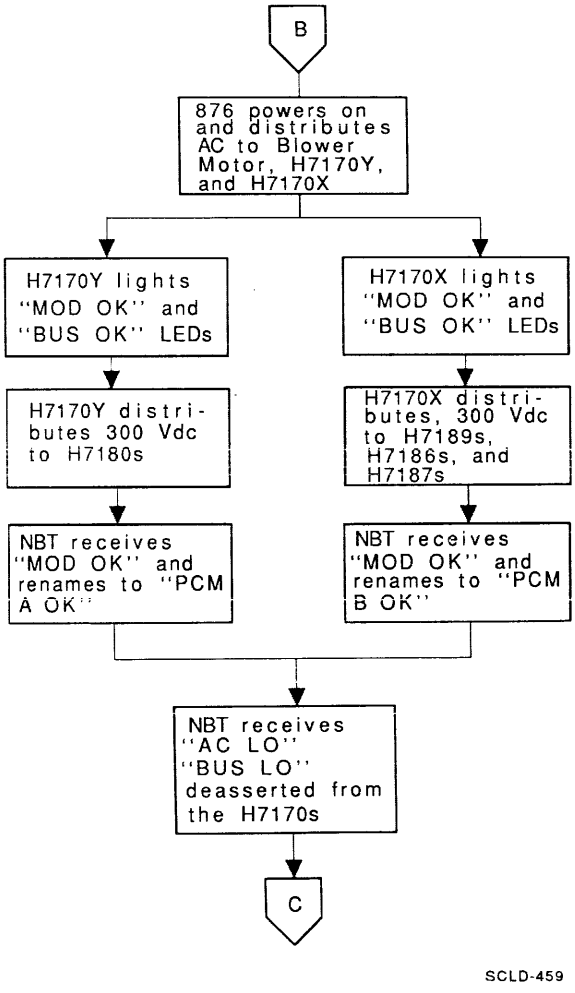


Figure 2-5 System Power-Up Flowchart (Sheet 3 of 4)

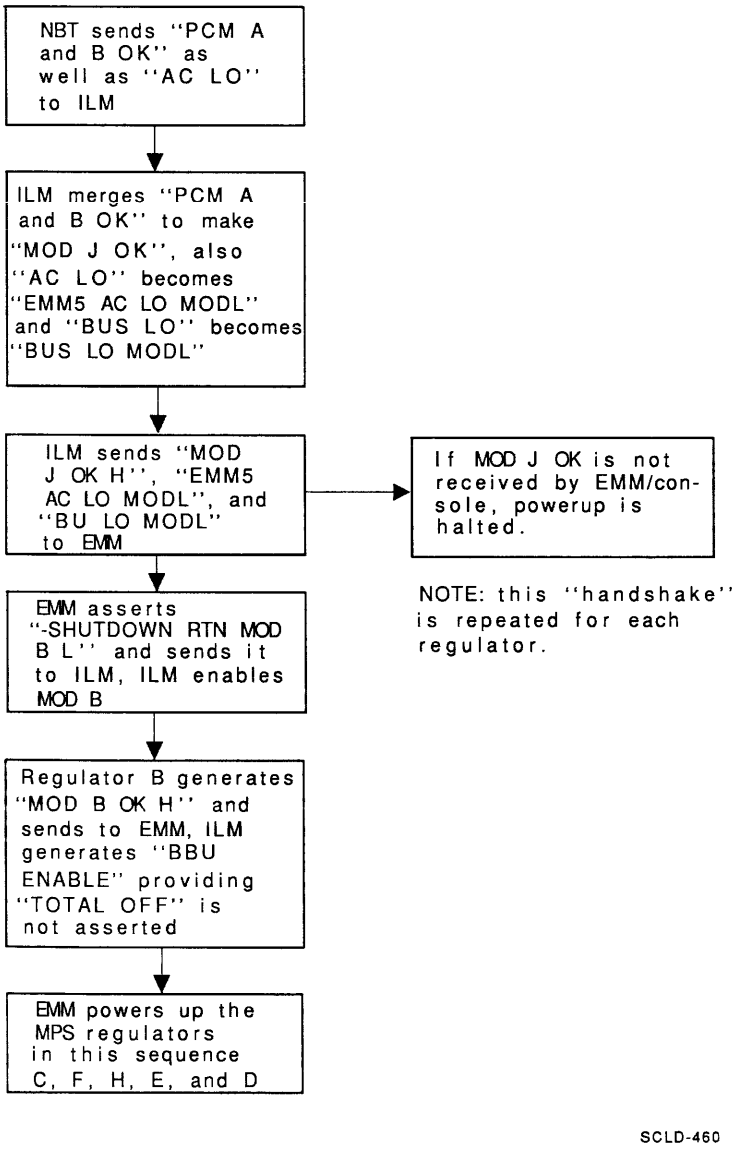


Figure 2-5 System Power-Up Flowchart (Sheet 4 of 4)

2.5 SYSTEM POWER-UP FLOWCHART (FIGURE 2-5)

1. The main circuit breaker (CB 1), located in the rear of the front-end cabinet, must be placed in the "ON" position.

2. AC is distributed to the console (BRO-38N and LA50 printer), via J1/J2 of the console power outlet box, and to the unswitched outlets and phase indicators (L1, L2, L3) of the 876 Power Controller. Upon receipt of ac power, the PRO-38N performs a selftest, and loads/runs the console software (this takes approximately 1 minute).

3. The unswitched outlets (J17) of the 876 distribute ac to the battery backup unit connector J22, and to the NBox J5 (the control start-up power (CSP) module connector J69). The CSP is located in the NBox. The charging circuits within the BBU (H7231) become operative at this point, and need no intervention/control from ILM/console.

4. After a delay of not more than one minute, the CSP generates +5 V and +/- 12 V, which is distributed to the EMM (via J4 CSP and J15, J9 connectors on NBox, to J64 MPS (modular power supply) connector, to J50 of the EMM), and to the interface logic module (ILM) (via J4 of the CSP to J5 of the ILM). The ILM is also located in the NBox. The CSP also distributes +10.5 V to the MPS backplane, which is used as bias (turn-on) voltage by each regulator inserted in the MPS B/P. The CSP also monitors its input ac voltage as reflected by a rectified, and filtered (300-Vdc NOM) voltage. The resultant signal is CSP BULK OK which is sent to the ILM.

5. The new box translator (NBT) receives power if the circuit breakers for the H7170s and the main circuit breaker in the front-end cabinet are in the "ON" position. This powers up the NBT's fail safe enable circuitry, which monitors the following CBs: front-end cab CB1 (main CB), 876A CB1. If all the above CBs (4) are in the 1 position (ON), the NBT sends "FAIL SAFE ENABLE" to the BBU (via J320 of the NBT, to J378 of the NBox, to J20 of the BBU). The BBU sends "BBU AVAIL L" to the EMM to indicate it is ready to supply power when needed. The BBU also sends its status to the ILM, which displays this information via the "batt stat" led, which is located on the front panel of the ILM.

6. When the ILM receives +5 V and +/-12 V, it powers up and performs a selftest. If the results of this test are "OK", the ILM displays the number "0" in the numeric display element. The numeric display element is located on the front panel of the ILM.

7. When the EMM receives +5 V and +/-12 V, its internal circuitry is active, but the uP 8085 is held in the init state awaiting "MOD A INLK L" H. In this init state, the EMM performs "hardware tests" such as MPS overtemp, and parallel key sense (CPU module misplace) tests. If a key fault is detected, the EMM illuminates the "key fault" LED on its front panel and deasserts "MOD A ENABLE".

   When the EMM is in the INIT state, the communications bus is deactivated, and the system is held incommunicado.

8. When the ILM receives both "CSP BULK OK H" and "MOD A ENABLE", the ILM deasserts "MOD A INTLK L" and sends it to the EMM (via J5 ILM and J9 NBox connectors to J58 MPS connector, to J50 of the EMM). The deassertion of "MOD A INTLK L" allows the EMM to begin functioning from an initialized state. At this point, the EMM begins to perform "default mode" tests that are located within H7188AB ROM. These tests are: system temperature, system air flow (which is being forced, or defaulted by the ILM/NBT to the "LOW" or "OK" state until "MOD J ENABLE" is asserted from the EMM). This allows the system to "IDLE" with no power on and no air mover without faulting.

9. The console can now establish a data link (RS232) with the EMM, and tries to access the EMM by sending the EMM's ID number/console code. The code is system-specific (i.e., VAX 8800 vs VAX 8750 and matches an ID code in the MPS II backplane). Refer to AD drawings for specific codes. If no codes are matched, the communications line is defaulted to an error message. If the EMM's ID code is matched, then the system-specific limits are loaded. The EMM now begins testing the system to the specific limits set in the console code.

10. Assuming this is an initial power-up situation, the "POWER ON" command is sent by the console operator.

11. Providing there are no faults within or detected by the EMM, power begins to sequence up. This sequence is initiated when the EMM asserts "-SHUTDOWN RTN MOD J L" and sends this signal to the ILM (via J50 connector for the EMM, to J58 MPS connector, to J9 of NBox, to J5 pin 49 of the ILM).

12. When the ILM receives "-SHUTDOWN RTN MOD J L" it asserts "PWR ROST H" for the 876 (via J5 of the ILM, to J8 NBox and to J4 of the 876).

13. The CSP sends air flow sensor power to the AIR FLOW SENSORS, located in the main CPU cabinet. The AFS power is sent via J9 of the CSP, to J58 and J60 pin 5 of the MPS (modular power supply) backplane connectors. The ILM releases the outputs of the air flow sensors to allow them to report a fault (until this time they were held low "OK" by pins 74, 42 of the ILM).

14. The CSP sends +5 V switched to the CLK/CNSL board (via J4 of the CSP to J15 pin 14 the NBox connector), to J64 (p14) then to J56 P11, then to J59 pin 11, then to J65 P17. A cable then routes this to the CPU backplane.

15. The 876 completes its power-on sequencing and distributes ac to the blower motor (via J13 of the 876, to the blower motor connector J1), to H7170Y (via J12 of the 876, to J380 NBox connector for the H7170Y), and to H7170X (via J11 of the 876, to J382 NBox connector for the H7170X).

16. Providing H7170Y and H7170X have no internal faults reflected by the "MOD OK" and "BUS OK" LEDs (located on the front panel) being lit, each will begin producing 300 Vdc.

17. The H7170Y distributes 300 Vdc directly to the H7180 regulators in the MPS II backplane (via the 300-Volt bus).

18. The H7170X distributes 300 Vdc to the H7189, H7187, and H7186 regulators in the MPS backplane (via the 300-Volt distribution module). The distribution module has four connectors, J100 through J103. Each connector interfaces some power system component to the 300-Volt bus. J100 interfaces to the H7170X; J101 interfaces to the H7189 regulators. J102 interfaces to the H7187 and H7186 regulators, and J103 interfaces the BBU connector J9 to the distribution module.

19. Both H7170s send control signals to the EMM, via the new box translator (NBT) and the interface logic module (ILM). The H7170Y sends "MOD OK" H to the NBT (via J1 and J381 NBox connectors for the H7170Y to P320/J320 connector for the NBT), which is renamed "PCM A OK" H.

20. H7170X sends "MOD OK" H to the NBT (via J1 and J379 NBox connectors for the H7170X to P320/J320 connector for the NBT), which is renamed "PCM B OK H".

21. Each H7170 sends an "AC LO L" and a "BUS LO L" in the deasserted state to the NBox connector P320/J320 for the NBT. At the backplane, the "AC LO" signals are combined to generate one input to the NBT, and so are the "BUS LO" signals.

22. The NBT sends "PCM A OK H" and "PCM B OK H" as well as the "AC LO" and "BUS LO" signals to the ILM (via P320/J320 of the NBT, to J5 of the ILM).

23. The "PCM A OK" and "PCM B OK" signals, within the ILM, are merged to become "MOD J OK H". The signal "AC LO" becomes "EMM5 AC LO MODL" and "BUS LO" becomes "BUS LO MODL".

24. The ILM sends "MOD J OK H", "EMM5 AC LO MODL L", and "BUS LO MODL L" to the EMM (via J5 and J9 of the ILM, to J58 MPS connector, to J50 of the EMM).

25. The EMM powers up each regulator in the MPS B/P in a particular sequence. Starting with MOD B (+5), MOD C, MOD F, and MOD H (BIP), MOD E (-5), and MOD D (-2). The EMM verifies that each regulator has powered up by sensing each regulator's "MOD OK" signal.

26. MOD B is powered on directly by the ILM before MOD C. First the EMM asserts "-SHUTDOWN RTN MOD B L" and sends it to the ILM, then the ILM asserts "ILM MOD B SHUTDOWN RTN L" and sends it directly to MOD B regulator. Regulator MOD B is the +5-V battery power regulator for the memory array modules.

27. When the ILM turns on MOD B regulator, the EMM verifies that MOD B is on by sensing the "MOD B OK H" signal from the regulator. Once the ILM senses that "MOD B OK H" is asserted, and providing the "TOTAL OFF" signal is not asserted, it asserts the "BBU ENABLE" signal. The "BBU ENABLE" signal enables the EMM/ILM to assert "BBU REQ RTN L", and PHASE OVERRIDE A when it detects an AC LO condition, which turns on the BBU.

The power system power-down flowchart is shown in Figure 2-6.

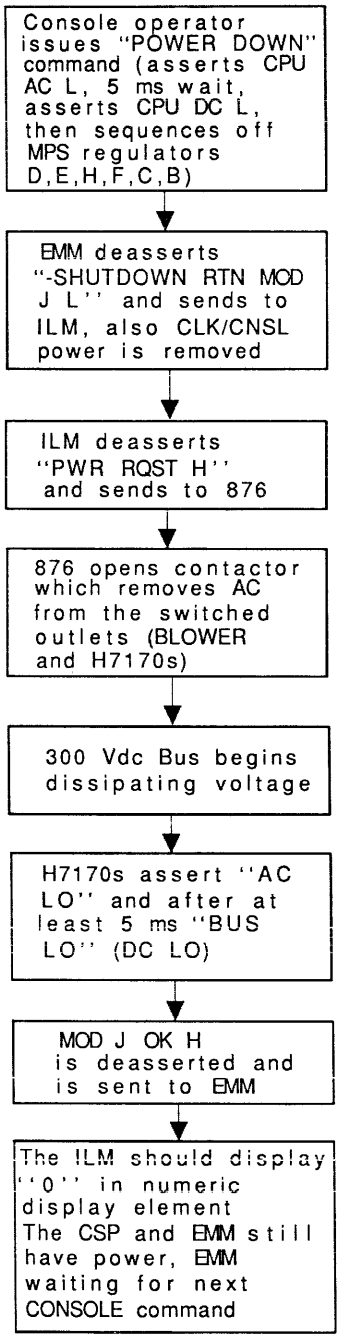


Figure 2-6 System Power-Down Flowchart

2.6 CONSOLE POWER-DOWN FLOWCHART (FIGURE 2-6)

1. The console operator issues the "POWER DOWN" command. This asserts both CPU AC L, and 5 ms later, CPU DC L. All of the MPS regulators are disabled in this sequence D,E,H,C,B,J.

2. The EMM receives the "POWER DOWN" command and begins the power-down sequence by deasserting "-SHUTDOWN RTN MOD J", which is sent to the ILM (via J50 of the EMM, to J58 of the MPS, to J5 of the ILM). Also the power (switched 15 V) to the CLK/CNSL board is removed.

3. When the ILM receives "-SHUTDOWN RTN MOD J L" negated, it deasserts "PWR RQST H", which is sent to the 876 power controller (via J5 of the ILM, to J4 of the 876).

4. When the 876 receives "PWR RQST H" negated, it opens the contactor that removes ac from the switched ac outlets. These outlets supply ac to the blower motor and to the H7170s.

5. The 300-V bus begins dissipating its voltage. The 300-V bus should reach 0 volts in 5 minutes.

6. By this time the H7170s will have asserted "AC LO". After at least 5 milliseconds (holdup) from the assertion of "AC LO", the H7170s assert "BUS LO" (DC LO). In this case, these signals are ignored because the console had previously asserted CPU AC L and CPU DC L.

7. The ILM should display a "0" in its numeric display element, which indicates it still has +5 V and +/- 12 V from the CSP. The CSP and EMM also still have power, and the EMM is just waiting for the next console command to be issued to it.

Figure 2-7 is a flow diagram of the signal events that occur during short-term power interruptions.

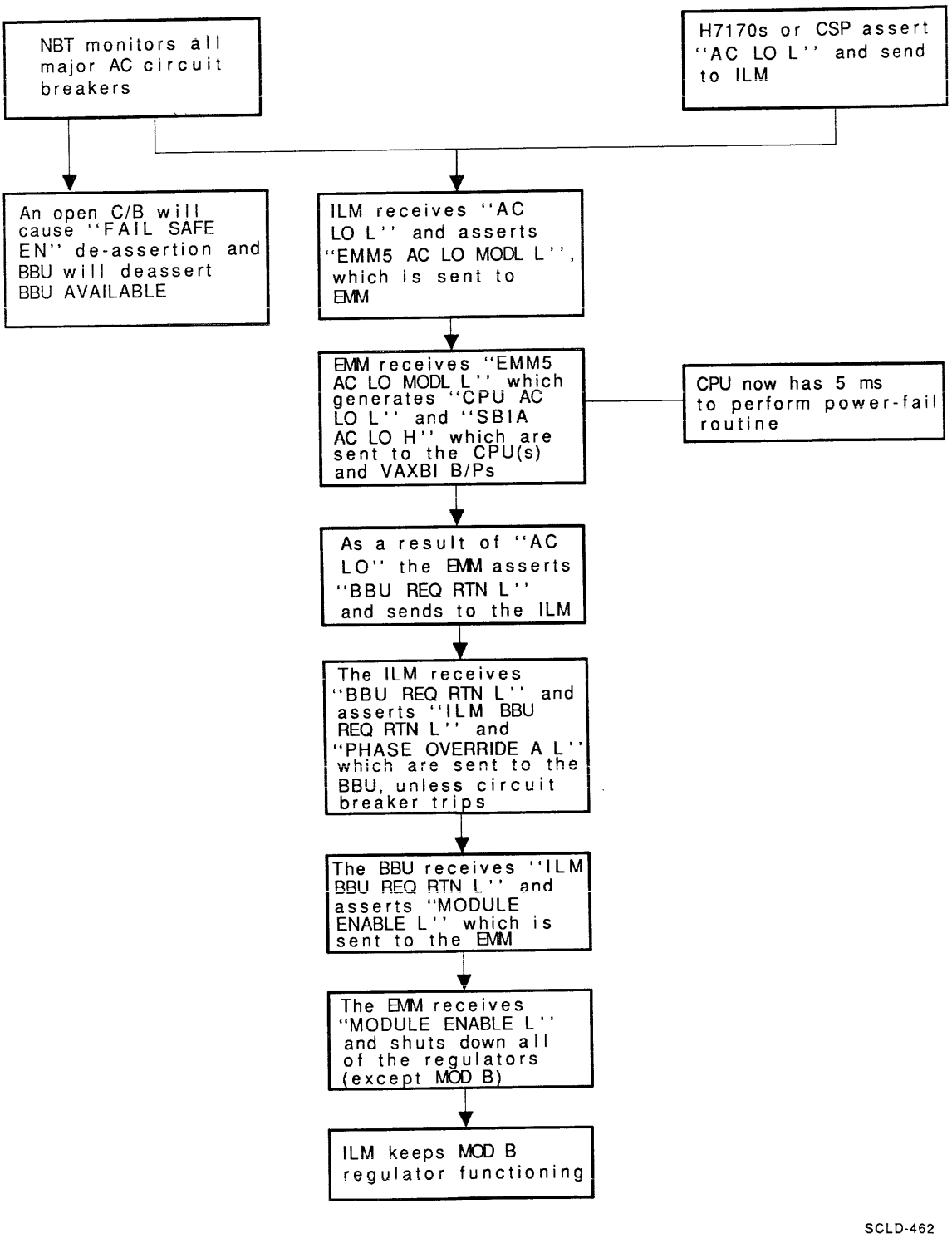


Figure 2-7 Powerdown/Power Interrupt with BBU Flowchart

2.7 POWER-DOWN/POWER INTERRUPT WITH BBU FLOWCHART (FIGURE 2-7)

1. The VAX 8800 backup translator contains a circuit that monitors the main circuit breakers in the backup system.

   The four circuit breakers are: the main breaker (CB1) located in the rear of the front-end cabinet, the main breaker (CB1) for the 876, and the breakers for the H7170s. If any of the above breakers should trip, the battery backup unit will not operate, due to the assertion of "FAIL SAFE ENABLE L".

2. Either of the H7170s or the control start-up power (CSP) module can assert "AC LO L". Once "AC LO L" is asserted, it is sent to the interface logic module (ILM) (via NBT connector P320/J320, to J5 of the ILM).

3. When the ILM recognizes that "AC LO L" is asserted, it asserts "EMM5 AC LO MODL L" and sends it to the environmental monitoring module (EMM) (via J5 and J9 of the ILM, to MPS connector J58, to J50 of the EMM).

4. The EMM receives "EMM5 AC LO MODL L", which generates "CPU AC LO L" and SBIA AC LO H". These signals are sent to the CPU(s) backplane (via J50 of the EMM, to J65 MPS backplane connector). The VAXBI "AC LO" signal is jumpered between the CPU(s) backplane and the VAXBI backplane.

5. When the EMM recognized that "EMM5 AC LO MODL L" was asserted, it began the process that is defined in the BATTERY BACKUP UNIT. This is accomplished by asserting "BBU REQ RTN L", which is sent to the ILM (via J50 of the EMM, to J58 MPS connector, to J5 of the ILM).

6. When the ILM receives "BBU REQ RTN L", it asserts two signals ("ILM BBU REQ RTN L" and "PHASE OVERRIDE A L") if Mod J is requested, and Mod B is "OK". These signals are sent to the BBU (via J5 of the ILM, to J18 of the BBU). "PHASE OVERRIDE A L" disables an internal ac monitoring circuit in the BBU, which allows any BBU request to be honored. The ILM has also taken control of MOD J EN, which it keeps enabled throughout the power outage, awaiting return of ac.

   If any of the major ac circuit breakers should trip, the NBT deasserts "FAIL SAFE ENABLE L", which causes the BBU to deassert "BBU AVAIL L" and inhibits the BBU from recognizing any request for battery backup operation.

   At this time, the ILM has a circuit that monitors: "BBU REQ RTN L" from the EMM, "BUS LO L" from the H7170s, and "BUFFERED CSP BULK OK L" from the control start-up power (CSP) module. When any of these signals are asserted, the control of MOD B regulator, by the EMM, is taken over by the ILM (see note).

7. The BBU receives "ILM BBU REQ RTN L" and asserts "MODULE ENABLE L", which is sent to the ILM, which then gates it by AC LO L or H7170's MOD OK deassertion. It then sends "MODULE ENABLE" to the EMM (via J18 of the BBU, to J58 of the MPS, to J50 of the EMM). The BBU also begins supplying 300 Vdc to the 300-V bus (via J9 of the BBU to J103 of the 300-V distribution module).

8. When the EMM recognizes that "MODULE ENABLE L" is asserted, it begins to shut down all of the MPS regulators, except MOD B. This is done by deasserting all of the "-SHUTDOWN.RTN MMOD X L" signals. This shutdown occurs within 6ms of assertion of MODULE ENABLE L. These shutdown signals are interfaced to each regulator plugged into the MPS backplane.

9. The ILM recognizes the need to keep MOD B regulator on and does so by turning on the BBU, which provides 300 Vdc to MOD B. MOD B regulator is now operating (providing +5 Vdc to the memory arrays).

NOTE

If the loss of ac affected the CSP, the CSP would power down and not be able to provide the +5 V and +/-12 V to the EMM and ILM. This means that MOD B regulator provides +5 V for the ILM and the ILM uses +5 to generate +/-12 for MOD B (MOD B Overtemp and MOD B OK signals). The only components in the power system still working are the: MOD B regulator, ILM, NBT, and BBU.

If the H7170s lose a phase, the CSP remains on and provides the EMM and ILM with +5 Vdc and +/-12 Vdc.

Figure 2-8 is a flowchart of the ac power-up from the BBU operation sequence.

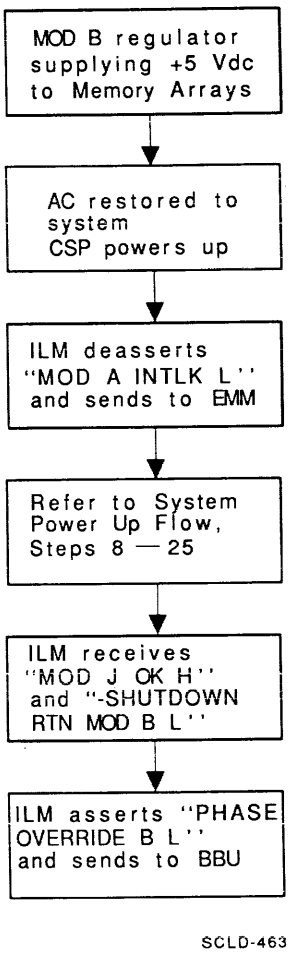


Figure 2-8 AC Powerup from BBU Operation Flowchart

2.8 POWERUP FROM BBU FLOWCHART (FIGURE 2-8)

1. MOD B regulator is under BBU control and providing +5 Vdc to the system's memory arrays.

2. AC is restored to the system and the CSP begins to power up. The CSP generates the signal "DELAYED CSP BULK OK H" and sends it to the ILM (via J4 of the CSP, to J5 of the ILM).

3. The ILM deasserts "MOD A INTLK L" and the power-up sequence is the same as a console powerup. The "POWER ON" command is executed. (Refer to the system power-up flow, steps 8 through 25.)

4. The ILM receives "MOD J OK H" from the H7170s via the NBT (via P320/J320 NBT connector, to J5 of the ILM) and "-SHUTDOWN RTN MOD B L" from the EMM (via J50 of the EMM, to J58 MPS connector, to J5 of the ILM).

5. As a result of "MOD J OK H" and "-SHUTDOWN RTN MOD B L", the ILM asserts "PHASE OVERRIDE B L" and sends it to the BBU (via J5 of the ILM, to J18 of the BBU). When the BBU receives "PHASE OVERRIDE B L", it stops supplying 300 Volts and is put in the charge mode.

3.1 INTRODUCTION

This chapter focuses on those parts of the power system that were described briefly in the preceding chapters, but, because of their importance in the power system, require additional information. Among the components to be described in this chapter are:

1. 876A power controller
2. NBOX power converter
3. Power supply regulators
4. Buses and backplanes
5. EMM module
6. BBU
7. Air flow and cooling

3.2 BLOCK DIAGRAM OF THE VAX 8800 POWER SYSTEM

A detailed block diagram of the VAX 8800 power system, shown in Figure 3-1, identifies the key components of the power system and the interconnections.

The 876A power controller, which interfaces utility input power to the operating modules of the power system, is discussed in the following paragraphs.

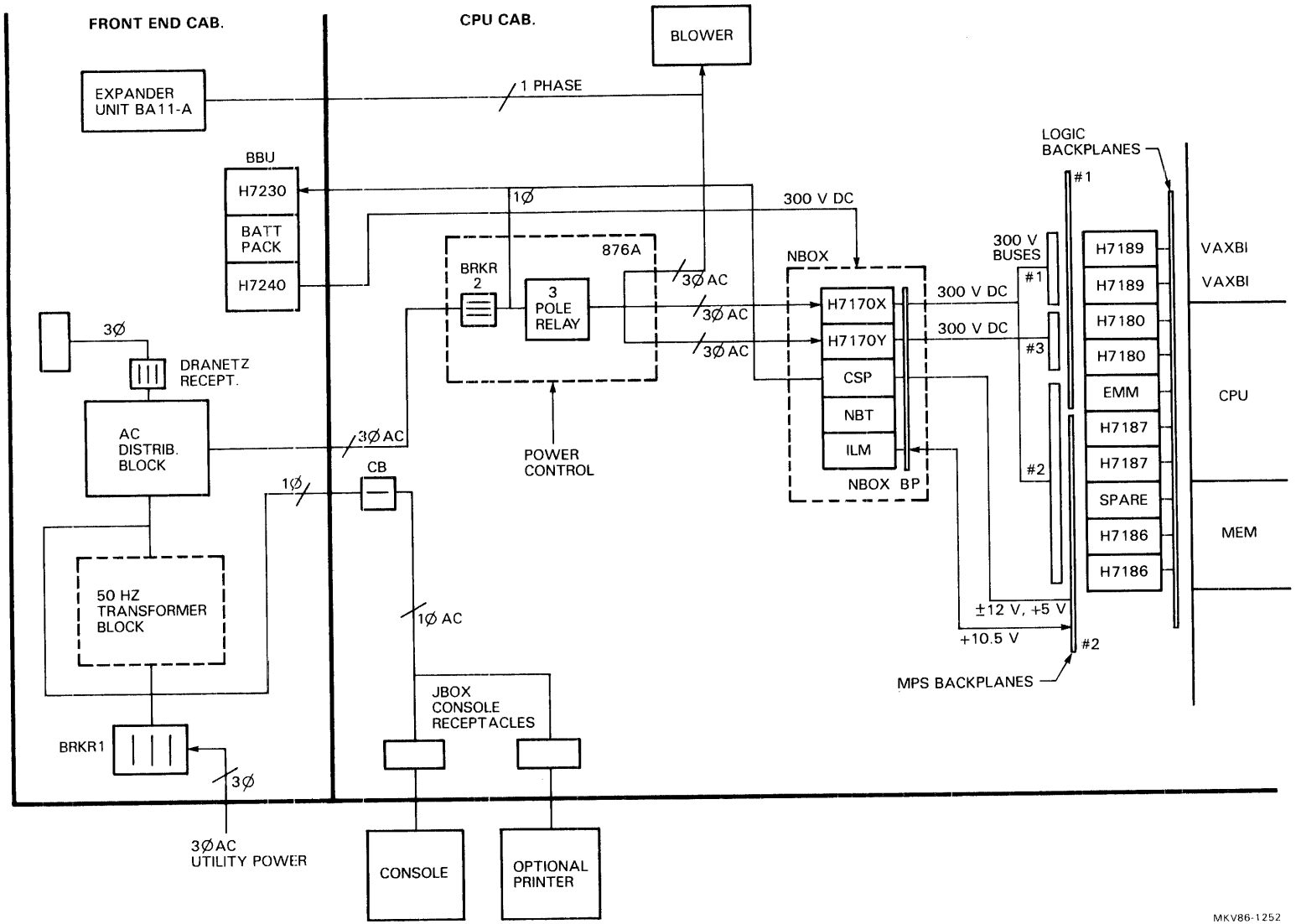


Figure 3-1 Power System Block Diagram

3.3 876A POWER CONTROLLER

The 876A PC receives utility power from BRKR1, the main system circuit breaker, and distributes switched and unswitched ac power to the system components, as shown in Table 3-1.

Table 3-1 876A AC Power Distribution

Component	AC 60/50 Hz, 120/240 power Supplied
BBU	1-phase, unswitched
CSP module (NBox)	1-phase, unswitched
Blower motor	3-phase, switched
H7170X&Y (NBox)	3-phase, switched
BA11-A (expander)	1-phase, switched
Aux. port	1-phase, switched

AC is distributed to the system components using receptacles located at the rear of the 876A. The system components are mated with the designated receptacles to receive ac power when the system is energized.

Figure 3-2 shows the front and rear panels of the 876A, illustrating the circuit breakers, lamps, and receptacles and their locations.

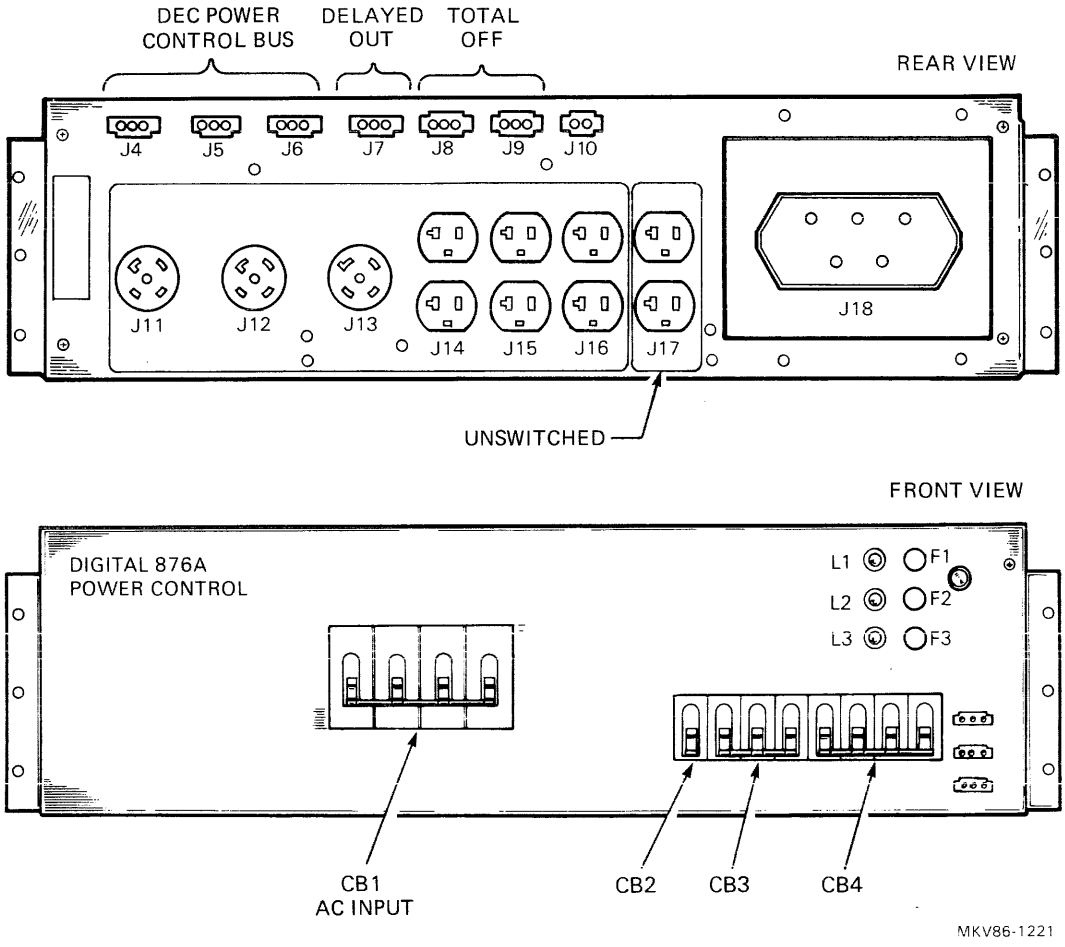


Figure 3-2 876A Front and Rear Panels

Figure 3-3 is a block diagram of the 876A showing the internal components and their interconnections.

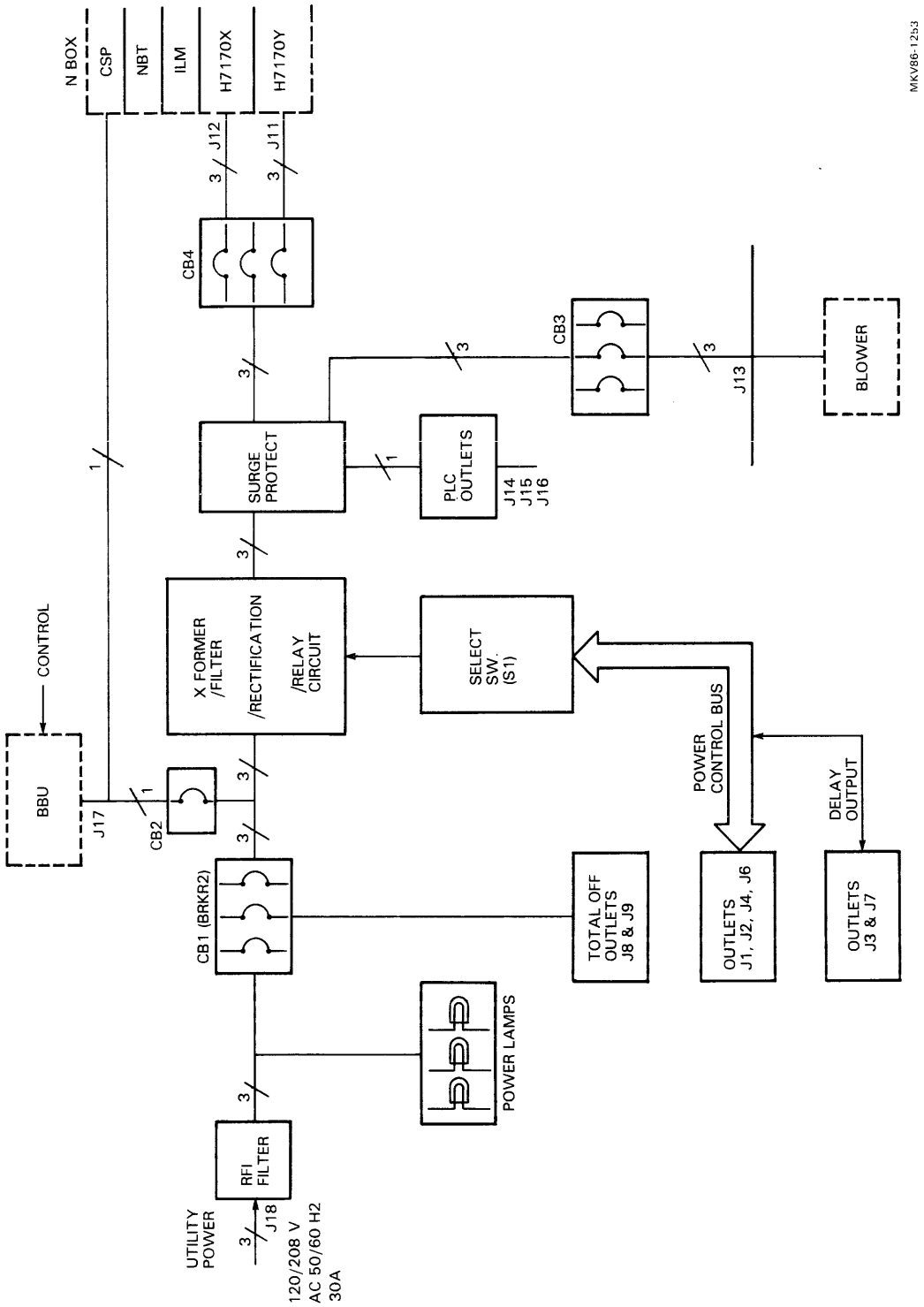


Figure 3-3 876A Power Controller Block Diagram

Referring to Figure 3-3, three-phase ac utility power is immediately sent to an RFI filter upon entering the 876A enclosure. The ac is then applied to:

1. AC indicator lamps, L1-L3
2. CB1(BRKR2), 876A main circuit breaker

From CB1, single-phase ac power is routed to the BBU-CSP branch through a single-phase circuit breaker (CB2). Three-phase power is applied to the transformer/filter/rectification/relay circuits.

The output of this section of the 876A is referred to as switched ac. It is controlled by the power control bus and S1, a three pole power relay.

The switched three-phase ac power is sent through surge protectors to:

- CB3 for 3-phase blower
  1-phase BA11
  1-phase aux. port

- CB4 for 3-phase H7170 power converter (2) located in NBox

3.4 NBOX POWER CONVERTER ASSEMBLY

The NBox is a multifunction assembly that performs the following functions in the VAX 8800 power system:

1. Converts three-phase ac input power to 300 Vdc for distribution to the MPS regulators by means of the 300-V Bus.

2. Converts single-phase ac input power to logic level voltages for the ILM and EMM modules and the MPS regulators.

3. Controls the operation of the battery backup unit (BBU).

4. Communicates with the EMM during power-up and power-down sequencing and battery backup operations.

5. Distributes 300-Vdc power from the H7170s and the BBU to the 300-Vdc buses.

6. Monitors circuit breaker status.

7. Monitors AC LO and DC LO signals.

A detailed block diagram of the NBox with system interconnections is shown in Figure 3-4.

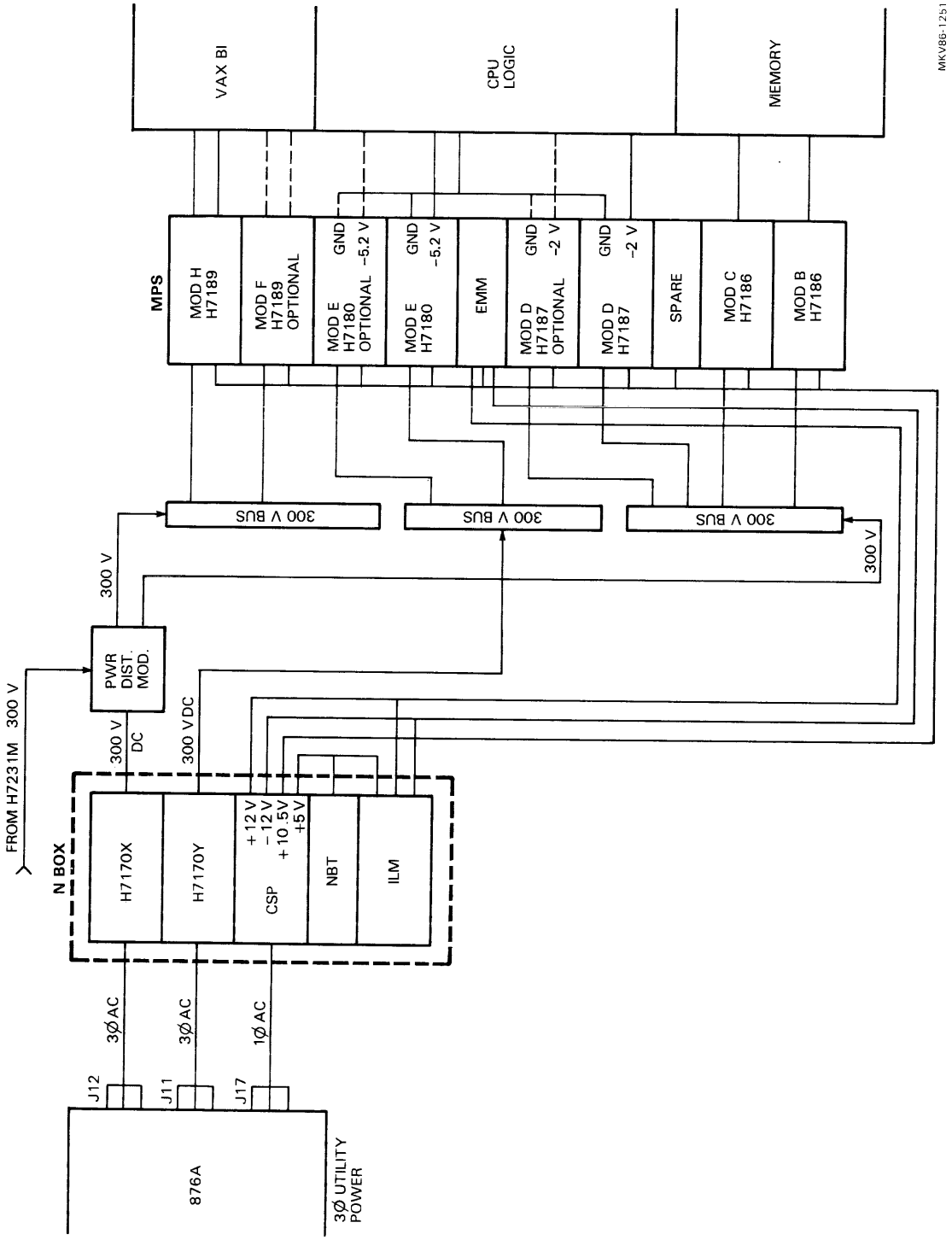


Figure 3-4 NBox System Interconnect

3.4.1 NBox Modules

The function or functions performed by each of the modules contained in the NBox are listed in Table 3-2. A detailed description of each module is presented in the following paragraphs.

Table 3-2 NBox Modules

Module	Function(s)
H7170 (2)	Convert 3-phase ac to 300 Vdc for MPS regulators.
CSP	Converts 1-phase ac power to logic level voltages (+5 V, +/- 12 V for EMM and ILM), (+ 10.5 V for MPS modules), (+15 V for CLK/CNTL module).
ILM	Provides interface logic between MPS and EMM. Controls operation of BBU.
NBT	Converts logic signals during powerup. Monitors circuit breaker status. Monitors AC LO and DC LO signals.

3.4.1.1 H7170 -- There are two H7170 power conversion modules located in the NBox that perform identical functions, that is, converting three-phase ac input power to 300-Vdc power.

The 300-V unregulated dc output from the two converters, H7170X and H7170Y, are connected to separate sections of the 300-Vdc power bus located within the MPS cage.

The 300-V output from the BBU is brought into the NBox to interface with the outputs of the H7170s and be distributed to the 300-V power bus during battery backup operation.

3.4.2 CSP

The control start-up power (CSP) module converts single-phase ac input power to logic level dc voltages for use by the modules indicated in Table 3-3.

Table 3-3 CSP Voltages

DC Logic Voltage	Modules Supplied
+/-12 V	EMM, ILM
+5 V	EMM, ILM
+10.5 V	MPS Regulators

3.4.2.1 ILM -- The interface logic module (ILM) is a single PC board assembly that provides logic signal interfacing and control functions to the following power system modules:

- CSP
- EMM
- BBU
- H7170 (X and Y)
- System clock
- Module B
- NBT module

The ILM also contains BITE (built in test equipment) circuits to drive the front-panel indicators described in Chapter 1.

Figure 3-5 shows the PC board connector for the ILM module and the signals that interface to the ILM.

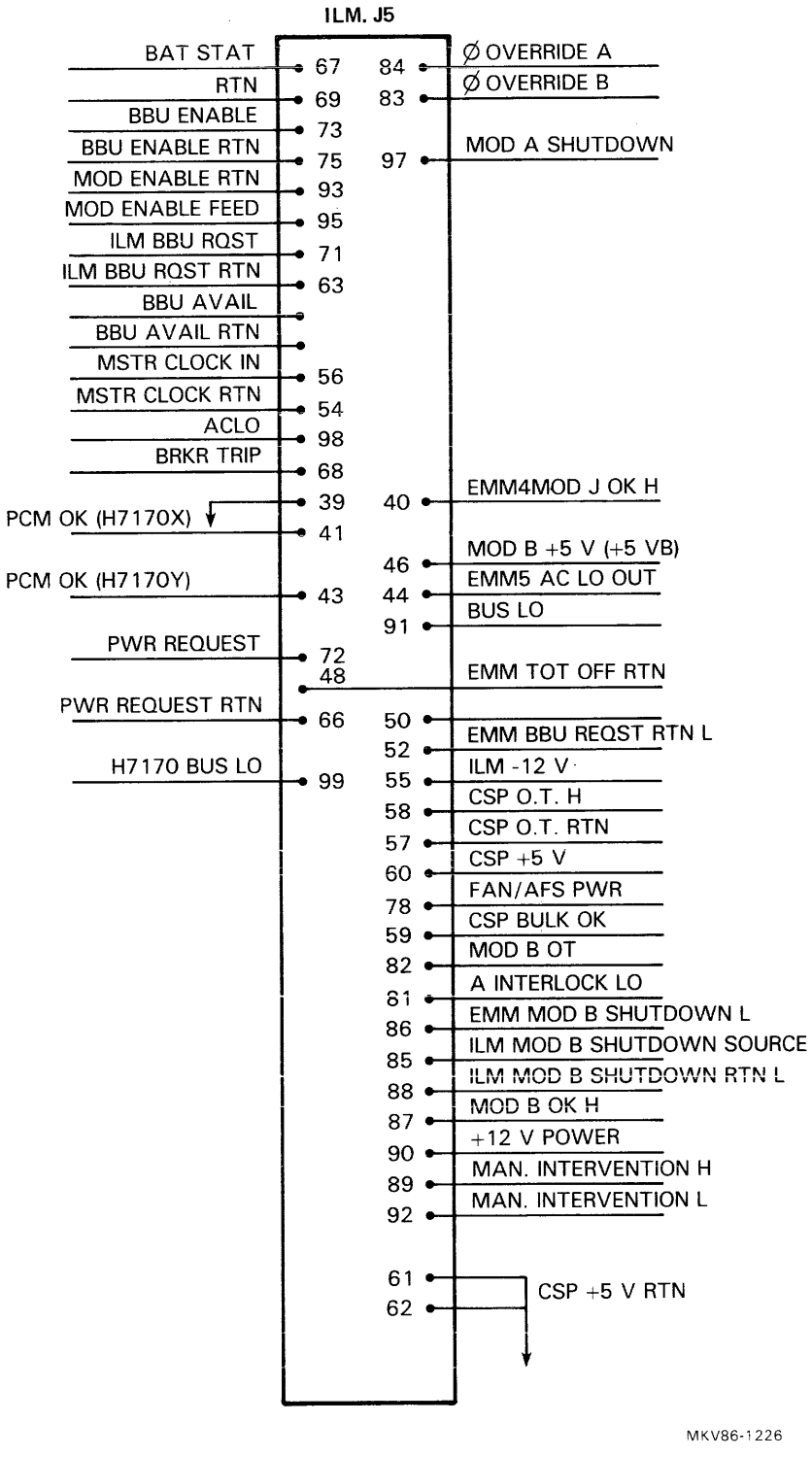


Figure 3-5 ILM PC Board Signals

The ILM receives +5 V and +/- 12 V during the power-up sequence from the CSP module, as explained in Chapter 2. After powering up, the ILM receives CSP BULK OK HIGH and KEY SENSE HIGH, which causes three signals, internal to the ILM, to be activated:

- DELAYED CSP BULK OK
- A INTERLOCK L
- AC POWER UP RESET L

The resulting ILM output, A INTERLOCK LO, is sent to the EMM to release the 8085 microprocessor, allowing the EMM to communicate with the system console.

The EMM, upon receiving a POWER ON command from the console, asserts SHUTDOWN RTN MOD J L, causing the ILM to assert power request, which energizes the power relay, K1, located in the 876A.

When K1 is energized, ac power is delivered to the power system components provided:

1. The ILM asserts PCM ENA L to request power.
2. BUFFERED CSP BULK OK L is asserted.
3. AC POWER UP RESET H is not asserted.
4. Timeout has not occurred.

Upon receipt of PCM A OK and PCM B OK from the H7170 power converters, the ILM asserts MOD J OK H to indicate proper operation of the H7170s.

The ILM is able to request battery backup power if:

- Module B is operating properly (MOD B OK H asserted), and

- AC power has dropped below 156 V rms (DELAYED CSP BULK OK or MOD J OK H is not asserted)

The ILM asserts ILM BBU ENABLE RETURN L (PIN 75) to enable the BBU to respond to a BBU REQUEST signal from the ILM. The BBU does not respond to a BBU REQUEST signal if:

1. TOTAL OFF signal is asserted, or
2. MOD B OK is not asserted

BATT STAT is a three-state signal sent to the ILM from the BBU to indicate battery status. The LED is located on the front panel of the ILM (H7061). The three indications are:

- OFF
  - No current or signal. BBU not available.
- READY
  - Continuous current flow or signal. LED "ON." BBU available and batteries fully charged.
  - Oscillating current flow at 1 Hz. LED blinking "slowly." BBU available but batteries not fully charged. Charging in process.
- ON
  - Oscillating current flow at 10 Hz. LED blinking "fast." BBU is on and providing power.

Phase Override

This signal is used to override the ac level detect circuitry located in the BBU when using a three-phase system. The BBU responds to a BBUR signal from the ILM gated by the following signals:

- PHASE OVERRIDE A -- Assures that the BBU will turn on.
- PHASE OVERRIDE B -- Assures that the BBU will turn off.

Power Control

System power is requested by the EMM in response to a power request issued by the system console. In response, the EMM sends MOD J SHUTDOWN RTN L to the ILM.

The ILM responds by closing relay contacts K1, (POWER REQUEST and POWER REQUEST RTN), cabled to the 876A. The POWER REQUEST signals activate the relay within the 876A and provide switched ac power to the system.

System Clock

The system clock is a 100-kHz signal generated within the ILM by means of MSTR CLOCK IN and MSTR CLOCK RTN. The master clock synchronizes the pulse width modulated power regulators to the rest of the system.

Bite Indicators

The built in test equipment (BITE) located on the ILM module front panel are magnetically latching indicators that continue to display after power has been removed. The indicators automatically reset at powerup or when a fault condition has been corrected.

The monitored faults are:

- CSP OT (overtemperature)
- MANUAL INTERVENTION
- MOD B OT

The ILM also contains a seven-segment LED digital display that indicates the following:

- CLOCK FAILURE
- H7170X, H7170Y MOD OK (PLMA, B OK)
- MANUAL RESET

Power Status Indications to ILM

The H7170s located in the NBox signal the ILM when the power input to the H7170s starts to deteriorate. The signals are:

- AC LO -- This signal performs a power-fail routine, and is sent to the CPU via the EMM.

- BUS LO -- This signal is sent to the CPU via the EMM. The signal indicates that the system voltages are no longer within acceptable parameters, and the system must shut down immediately.

Bias Circuit

The dc bias voltages for ILM logic are derived from the CSP inputs (+12 V POWER, ILM -12 V and CSP +5 V) and/or MOD B during normal operation. During battery backup, +5 V is fed from Mod B (MOD B +5 V) to develop the bias voltages.

The +12 V dc from the CSP is wire-ORed with the ILM +12 V BIAS to provide the source voltage for the ILM MOD B SHUTDOWN SOURCE signal during powerup.

3.4.2.2 NBT Module -- The New Box Translator (NBT) module is used to convert logic signals during the power-up sequence to perform operating system functions required during and after powerup.

The signals that are translated are:

Input		Output
MOD OK X	to	H7170 X OK
MOD OK Y	to	H7170 Y OK
AC LO IN	to	AC LO
BUS LO	to	BUS LO L
BRKR TRIP	to	TOT OFF

Power for the NBT module is provided by CSP +5 V during normal operation and MOD B +5 V during BBU operation. Both supplies are ORed together on the PC board.

The NBT monitors circuit breaker status (BRKRs 1 and 2) and CB1 on both H7170s and sends out a signal [FAIL SAFE ENABLE (FSE FEED)] when any of the above breakers are tripped.

It also sends out a TOT OFF command to shut down the power system in response to a BRKR TRIP signal from the ILM.

Figure 3-6 is a PC board connector diagram showing the signals in and out of the NBT.

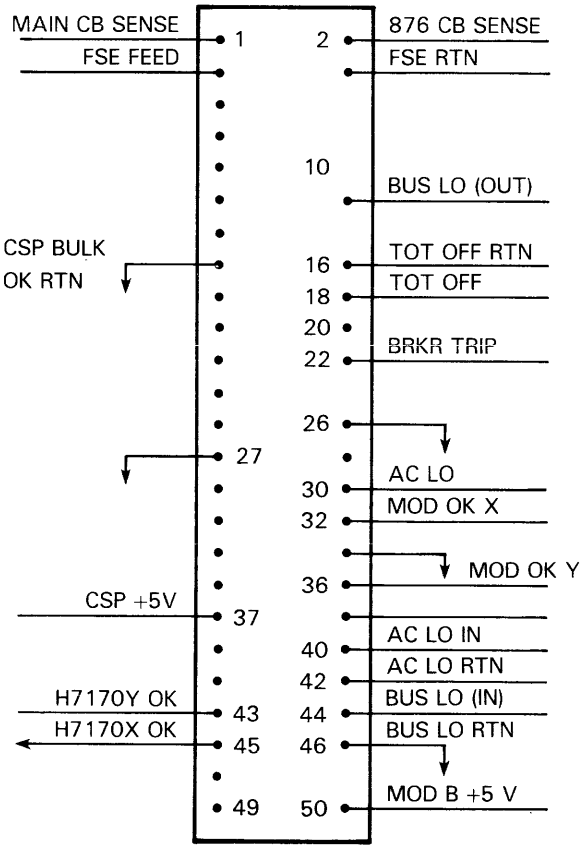


Figure 3-6 NBT PC Board Signals

3.4.3 Modular Power System -- MPS

The modular power supplies in the VAX 8800 power system complex are pulse width modulated (PWM) dc power regulators. The regulators provide the dc logic voltages necessary to operate the VAX 8800 computer system.

Figure 3-7 is a front view of a fully configured MPS assembly for a dual-CPU VAX 8800 system. The regulators, their indicators, and the MPS backplanes are described in the following paragraphs.

Regulators not used in a single-CPU VAX 8800 system are indicated by an asterisk (*).

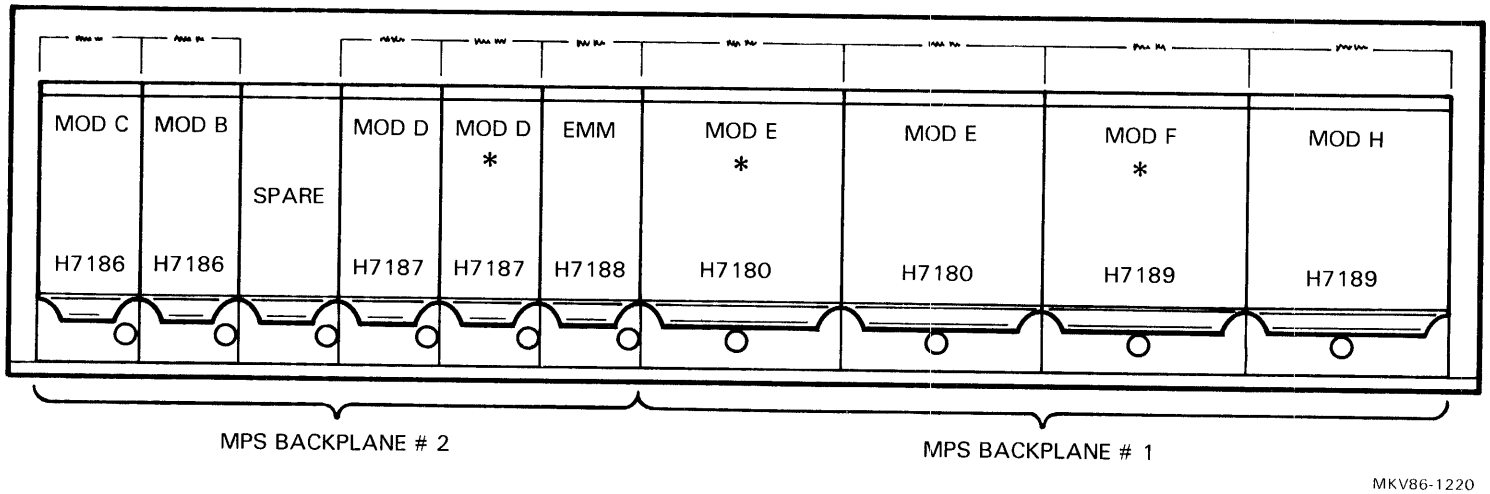


Figure 3-7 MPS Regulator Configuration

The dc logic voltages supplied by the MPS are listed in Table 3-4.

Table 3-4 VAX 8800 MPS Regulators

Quantity	Regulator	Volts (DC)
1	B (H7186)	+5 V
1	C (H7186)	+5 V
1 or 2	D (H7187)	-2 V
1 or 2	E (H7180)	-5.2 V
1	F (H7189)	+/-12 V, +5 V, -5.2 V, -2 V, +15 V
1	H (H7189)	+/-12 V, +5 V, -5.2 V, -2 V, +15 V

3.4.3.1 H7186 +5.0-Volt Regulator -- The H7186 regulators, modules B and C, generate an output of +5 Vdc at up to 85 A, to provide power to the CPU memory modules.

Module B is also used to provide power to the memory modules during battery backup operation, when ac facility power is lost. Figure 3-8 is a block diagram of the H7186.

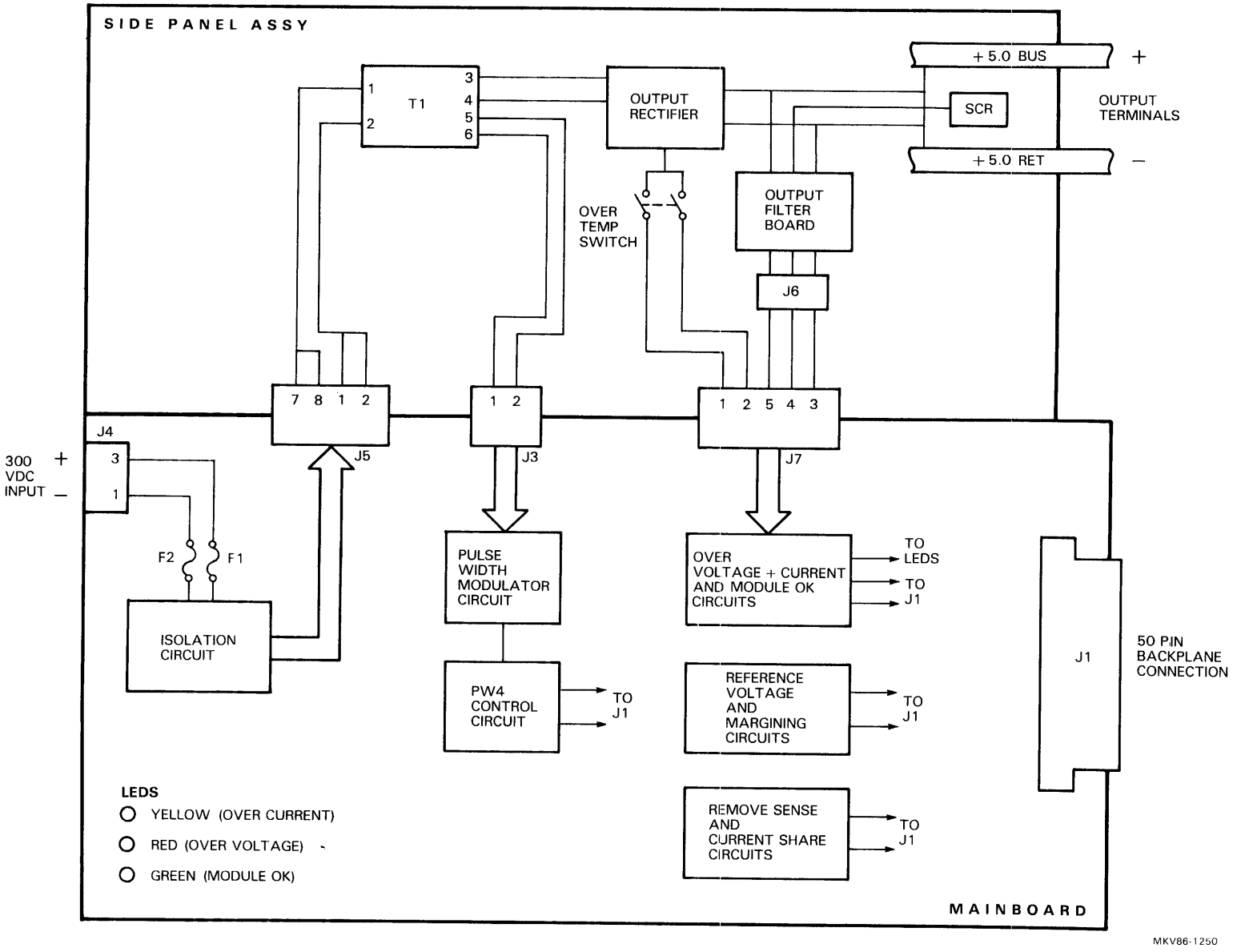


Figure 3-8 H7186 Block Diagram

The H7186 is made up of two major subassemblies:

1. Side Panel
2. Main Board

3.4.3.2 Side Panel -- The functions and interconnects for the H7186 side-panel assembly are defined in Table 3-5.

Table 3-5 H7186 Side-Panel Components and Interconnects

Component/Interconnect	Description/Function
J6	Connects output filter board to main board.
T1	Power transformer - steps down and isolates the dc input voltage.
Overtemperature Switch	Senses temperature on output rectifier heatsink.
	Indicates overtemperature condition at 100 degrees C (212 degrees F)
	Generates shutdown signal to EMM.
SCR (Silicon Rectifier)	Overvoltage protection device.

3.4.3.3 H7186 Main Board -- The functions of the logic circuits and interconnects on the main board are defined in Table 3-6.

Table 3-6 H7186 Main Board Circuits and Interconnects

Circuit/Interconnect	Description/Function
J1	Connects signals from the main board to the MPS backplane.
J2	Connects overtemperature switch and J6 to the main board.
J3	Connects T1 output to the main board.
J4	Connects 300-Vdc input to the main board.
J5	Connects output of isolation circuits to step-down transformer.
F1 and F2	5-A 600-V fuses used for input overload protection.
Isolation Circuit	Isolates high voltages on the main board.
Pulse Width Modulator and Control Circuit	Provides a stable voltage output using variable pulse width to compensate for input voltage and load fluctuations.
Overvoltage Circuit	Crowbar protection circuit designed to disable output voltage.
	Lights red LED for overvoltage.
	Overvoltage -- (7.3 Vdc max. to 6.3 Vdc min.).
Overcurrent Circuit	Disables output voltage.
	Blinks yellow LED on/off to signal overcurrent.

Table 3-6 H7186 Main Board Circuits and Interconnects (Cont)

Circuit/Interconnect	Description/Function
Ref Voltage and Margin Circuit	Provides 1 of 3 jumper selected voltage outputs. Output selected can be margined +/-5% through the EMM.
Module OK Circuit	Lights green LED to signal MOD OK.
Remote Sense Circuit	Enables sensing of output voltage at remote load.
Current Sharing Circuit	Allows sharing of load current when cross-connected with another H7186.

3.4.4 H7187 -2.0-Volt Regulator

The H7187 regulator, Module D, provides an output of -2.0 Vdc at up to 100 A. The output is connected by bus bar straps to the CPU backplane area.

The block diagram for the H7187 is identical to that shown for the H7186 in Figure 3-8 due to the similarity of components.

The mechanical configuration of the H7187 is virtually identical to the H7186 with minor exceptions. It is made up of two subassemblies, the side panel and the main board.

The components, circuits, and interconnects for each subassembly are described in Tables 3-5 and 3-6.

Two module D regulators are used in the dual-CPU configuration.

3.4.5 H7180 -5.2-Volt Regulator

The H7180 regulator, module E, located in MPS Backplane 1, provides an output of -5.2 Vdc, at up to 200 A. Bus bar straps connect the dc voltage from MOD E to the CPU backplane area.

Figure 3-9 is a block diagram of the H7180.

Two module E regulators are used in the dual-CPU system configuration.

The H7180 consists of two major subassemblies, the side panel and the main printed circuit board. The components, circuits, and interconnects on each subassembly are discussed in the following paragraphs.

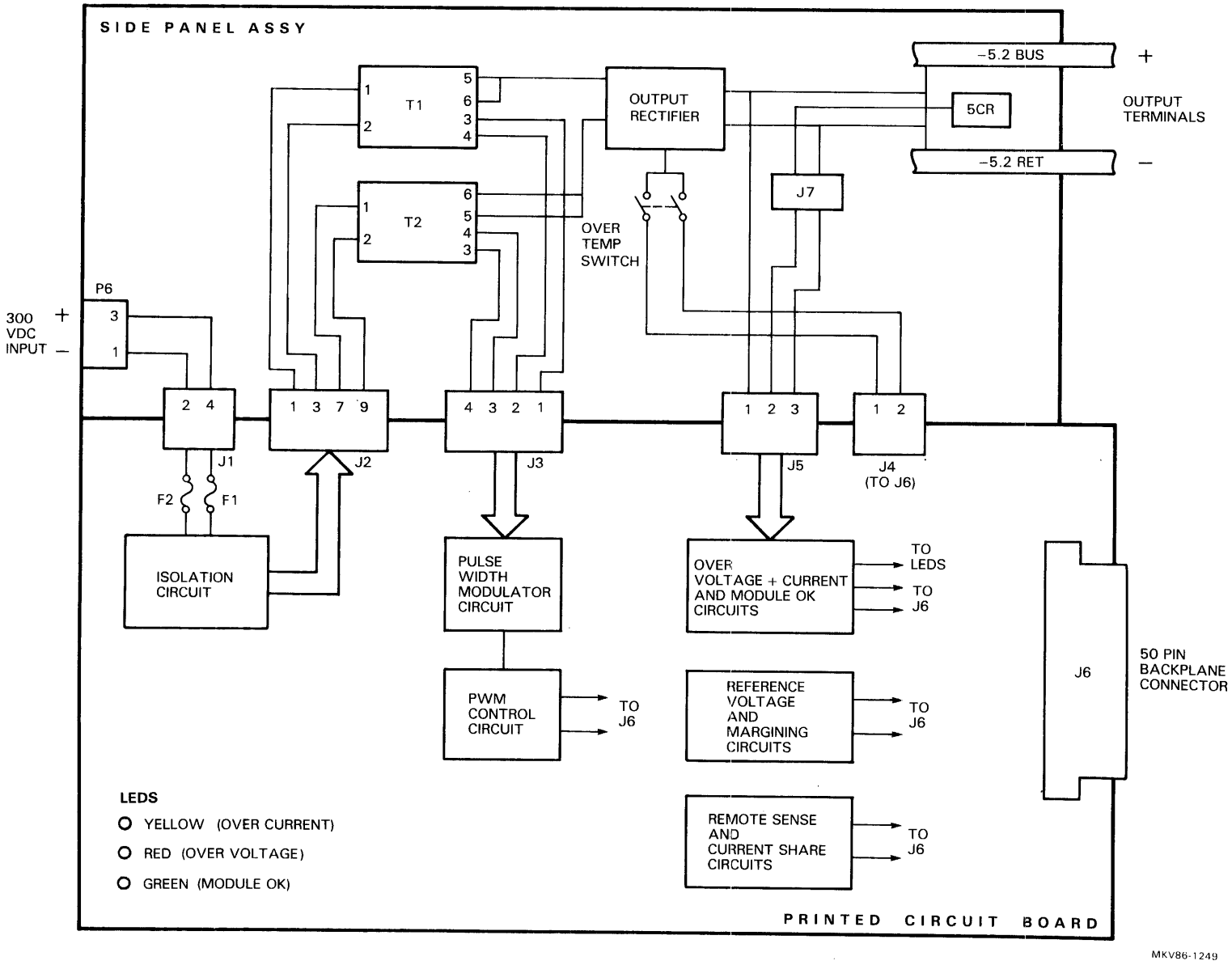


Figure 3-9 H7180 Block Diagram

3.4.5.1 H7180 Side Panel -- The functions of the components and interconnects on the side panel assembly are defined in Table 3-7.

Table 3-7 H7180 Side Panel Components and Interconnects

Component/Interconnect	Description/Function
P6	Connects 300 Vdc to the regulator.
P7/J7	Connects crowbar SCR and power output return terminal to PC board.
Transformer 1 and 2	Power transformers used to step down and isolate dc input voltage.
S ---	Switch/Sensor mounted on output rectifier heatsink to indicate overtemperature condition (100 deg. C 212 deg. F). Generates shutdown signal to EMM.
SCR (Silicon Rectifier)	Overvoltage protection device used at the power output terminals

3.4.5.2 H7180 Main PC Board -- The functions of the logic circuits and interconnects on the H7180 main printed circuit board are defined in Table 3-8.

Table 3-8 H7180 Main PCB Circuits and Interconnects

Circuit/Interconnect	Description/Function
J1	Connects 300 Vdc to PC board. Connects isolation circuits to the step-down transformers and PWM circuitry.
J4	Connects overtemperature switch to PC board.
J5	Connector that interfaces power output terminal and J7 to PC board.
J6	Connects PC board to MPS backplane.
F1 and F2	Input overload protection fuses, 10-A 600-V type.

Table 3-8 H7180 Main PCB Circuits and Interconnects (Cont)

Circuit/Interconnect	Description/Function
Isolation Circuit	Isolates high voltages from PC board.
Pulse Width Modulator	Provides a stable output voltage using variable pulse width control (PWM) to compensate for input voltage and load fluctuations.
Overvoltage Circuit	SCR Crowbar protection circuit designed to disable the output voltage.
	Lights red LED on output fault.
	Overvoltage condition (7.0 Vdc max. 6.0 Vdc min.).
Overcurrent Circuit	Disables output voltage.
	Blinks yellow LED on/off on overcurrent condition.
Ref Voltage and Margin Circuit	Connects one of three voltage outputs using jumpers.
	Selected output margined +/-5%.
Module OK Circuit	Verifies fault-free status.
	Lights a green LED.
Remote Sense Circuit	Enables output voltage sensing at remote load.
Current Sharing Circuit	Allows sharing of load current when cross-connected with another H7186 to common load.

3.4.6 H7189 BIP Regulator

The H7189 power regulator is a multiple voltage output power module that provides operating voltages for the Bus Interconnect (VAXBI) module exclusively.

A VAXBI module can contain up to six VAXBI PC boards depending on the number of peripherals used. It provides the interface between peripheral equipment and the VAX 8800 system.

Two H7189 power regulators can be configured into backplane I of the MPS to accommodate two VAXBI modules, as required.

Figure 3-10 is a block diagram of the H7189 power regulator, illustrating its major components.

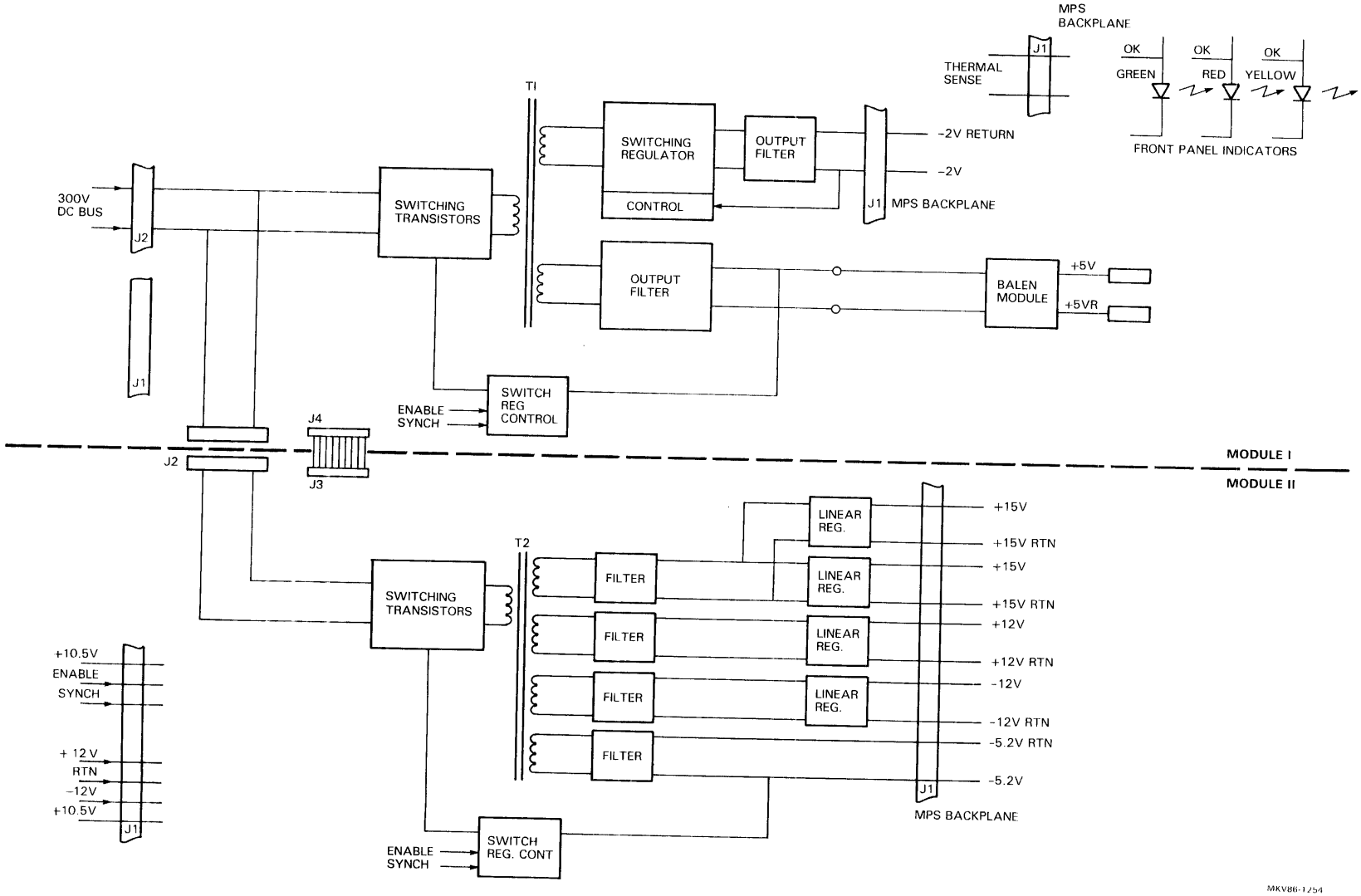


Figure 3-10 H7189 Block Diagram

There are three PC boards in the H7189 regulator:

- Module I
- Module II
- BALUN Module

Modules I and II interface with the MPS backplane through 80-pin PC board connectors. The functions performed by Modules I and II and the BALUN module are shown in Table 3-9.

Table 3-9 H7189 Module Functions

Module	Functions Provided
I	-2 Vdc switched regulator output
	+5-Vdc switched regulator output
	LED drivers -- green red yellow
	Overtemperature sense circuit
II	+15-Vdc linear regulated output (2)
	+12-Vdc linear regulated output
	-12-Vdc linear regulated output -5.2-Vdc switched regulator output
BALUN	Storage capacitors
	Power output (5 V) connectors

The H7189 power module uses a combination of linear regulation and switching regulation to generate the dc output voltages required.

A brief functional description of the H7189 is presented in the following section.

3.4.6.1 H7189 Functional Description -- 300 Vdc is brought into the H7189 by the bus connector and applied to the two switching transistor modules connected to transformers T1 and T2 (Figure 3-10). T1 and T2 provide:

- High voltage isolation between the bus and regulators
- Stepped down voltage to the regulators
- Stepped up current to the regulators

Transformer T1, located on module I, applies secondary output voltage to:

- -2-V switching regulator and output filter
- +5-V output filter. The +5-V supply uses a switching regulator at the primary of T1

Transformer T2, located on Module II, applies secondary output voltage to:

- +15-V filter feeding two +15 V linear regulators
- +12-V filter and linear regulator
- -12-V filter and linear regulator
- -5.2-V filter (the -5.2-V supply uses a switching regulator at the primary of T2.

Three types of regulation are used on the H7189 module:

- High-powered switching regulation performed at the primary of T1 and T2
- Low-powered switching regulation performed at the secondary of T1 (-2 V)
- Linear regulation performed by Module II regulators

High-powered switching regulation is used for those outputs that provide high-current power to the VAXBI. These outputs are:

- +5 V
- -5.2 V

The high-powered switching regulation provided for the +5-V and -5.2-V supplies also provides a quasi-regulated voltage to the inputs of the following regulators:

- -2 V
- +15 V (2)
- +12 V
- -12 V

The -2-V supply uses a switching regulator for higher power handling capacity.

Linear regulators are used for the Module II voltage outputs because the current required is lower.

The switching regulator control circuits located on Module I and II PC boards provide interfacing and control functions for the high-power switching regulators. These functions include:

- POWER ENABLE -- Enables operation of the regulators
- SYNCH -- Synchronizes switching regulators to system clock

These signals are brought into the H7189 module through the MPS backplane connectors for Modules I and II.

Other signals brought into the H7189 are +/-12 V and +/-12 V RTN, the regulator start-up bias voltages.

The front panel contains three LED indicators that display regulator operating status.

Indicator (LED)	Indication
GREEN	No faults -- Normal operation
RED	Overvoltage
YELLOW	Overcurrent

The GREEN LED is driven by an AND circuit sensor that lights when all regulator outputs are operating correctly.

The RED LED will light when any of the regulators have an overvoltage fault condition.

The yellow LED will light when any of the regulators have an overcurrent fault.

A THERMAL SENSE circuit, located on Module I, signals the EMM when an overtemperature condition has been sensed.

Table 3-10 lists the output voltage, current, and power provided by the H7189 regulator to the VAXBI module.

Table 3-10 H7189 Outputs

Output Voltage (Vdc)	Current (Amps)	Power (Watts)
+ 5.0	55	275
- 2.0	30	30
- 5.2	25	130
+ 12.0	2	24
- 12.0	2	24
+ 15.0	.5	7.5
+ 15.0	.5	7.5

Tables 3-11 and 3-12 describe the components and functions of the two PC board subassemblies, Module I and Module II, of the H7189 regulator.

Table 3-11 H7189 Module I Circuits and Interconnects

Component/Interconnect	Description/Function
J 2	80-pin PC board connector to MPS backplane
J 2	300-Vdc bus power connector
J 4	Intermodule connector
T 1	Isolation transformer with multiple outputs
Q1,Q2	Power switching transistors for 5-V regulator
	Switching regulator for +5-V supply

Table 3-12 H7189 Module II Circuits and Interconnects

Circuit/Interconnect	Description/Function
J 1	MPS backplane connector
J 2	300-Vdc bus connector
J 3	Intermodule connector
	Regulates +15-V supply (2)
	+12-V linear regulator
	-12-V linear regulator
	Switching regulator for -5.2 V

3.4.7 MPS Regulator BITE Indicators

The regulators contained in the MPS cage have built in test equipment (BITE) indicators to provide a visual indication of the operating status of each regulator, as described in Chapter 1. Table 1-3 is repeated here for convenience, as is Figure 1-7.

Indicator	Indication	Definition
Green LED	Module ok	The power regulator is operating properly. The output voltage is within regulation range. No faults are evident.
Red LED	Overvoltage	The regulator voltage has crowbarred.
Yellow LED	Overcurrent	The regulator output current is above its rating.

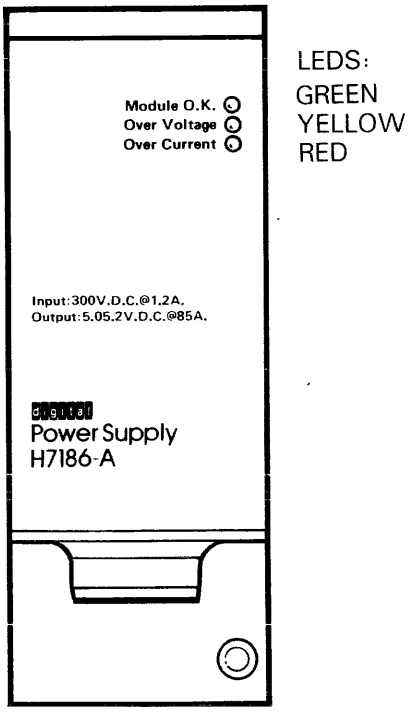


Figure 3-11 BITE Indicators

3.4.8 Buses and Backplanes

Signal and power connections between functional sections of the VAX 8800 system are made using the following:

- Multilayer, multiconnection printed wiring boards
- Flexible, multiconductor printed wiring cables
- Flexible, insulated, single- and multiconductor cable
- Laminated, high-current, rigid conductor straps
- Rigid, high-current conductor buses

The major functional sections of the VAX 8800 system are:

- MPS
- 876A
- NBox
- CPU
- VAXBI

The power and signal connections in and between these sections are discussed in the following paragraphs.

3.4.9 MPS Backplane

The MPS backplane consists of two hardwire-etched conductor sections that span the width of the MPS cage. The backplane provides electrical contacts between the regulators and the following:

- EMM
- CSP
- CPU backplane
- VAXBI backplane
- System clock

The EMM interacts with the regulators <B:H> to:

1. Monitor regulator output voltage
2. Monitor regulator temperature faults
3. Margin test regulator output
4. Start up/shut down regulators
5. Monitor fault status

The CSP provides start-up power to the regulators (+10.5 V) and operating voltages (+/-12 V and +5 V) to the EMM and ILM modules by means of the MPS backplane.

The system clock provides synchronizing signals to the pulse width modulator circuits of the regulators.

The regulator dc outputs are connected to the CPU backplane from the MPS backplane. High-current outputs are connected to the CPU backplane using laminated copper straps to reduce voltage drops.

The voltage outputs of the H7189 regulators are connected to the VAXBI backplane from the MPS backplane using flexible cable.

Each regulator interfaces to the MPS backplane through multipin PC board connectors that mate with the regulator PCBs.

Table 3-13 identifies the connector(s) for each regulator type.

Table 3-13 MPS Regulator Connectors

Regulator	Connector	No. of Pins
H7180	J6	50
H7186	J1	50
H7187	J1	50
H7189	J1 and J2	80

Figure 3-12 illustrates the backplane and bus organization of the power system. Figures 3-13 and 3-14 show details of the backplanes.

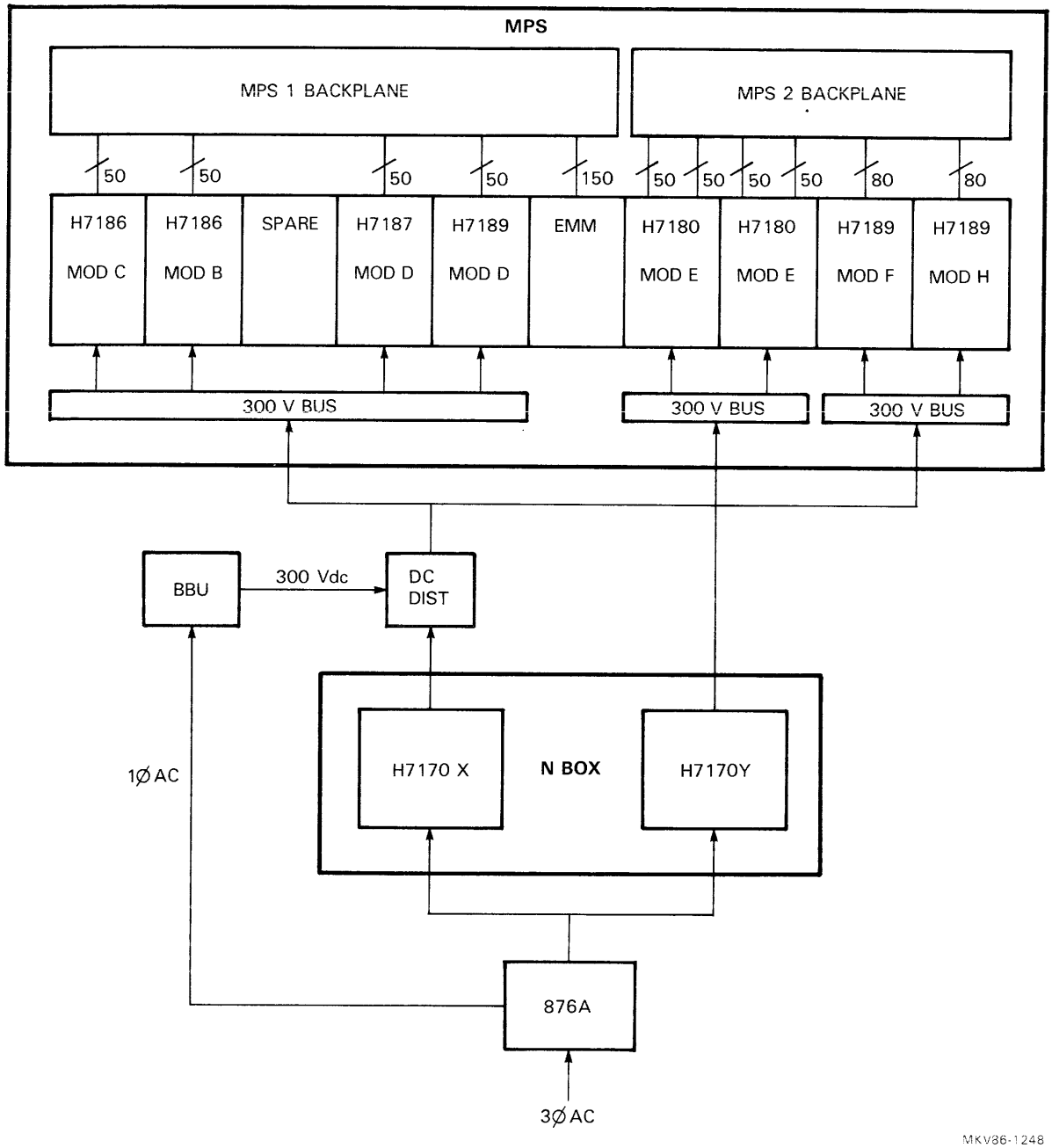


Figure 3-12 Organization of the Power System

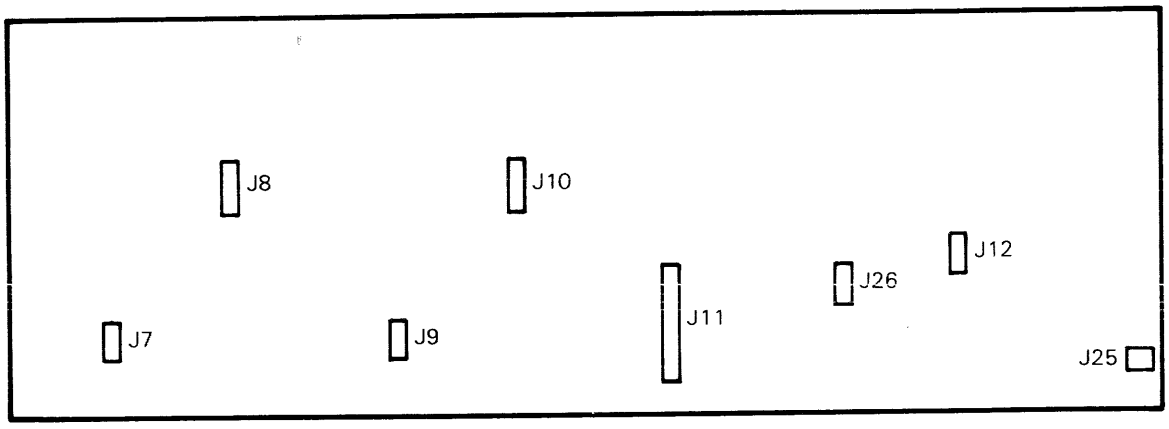


Figure 3-13 MPS I Backplane

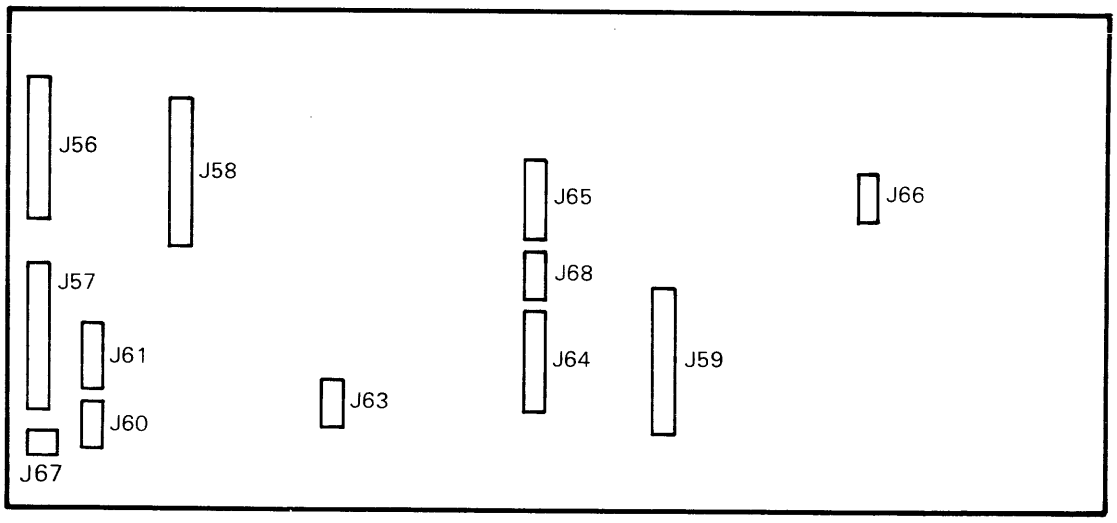


Figure 3-14 MPS II Backplane

3.4.9.1 300-Vdc Buses -- The 300-Vdc buses carry the high-power dc voltages developed by the H7170X and Y ac-to-dc power converters to the MPS regulators.

There are two 300-Vdc bus sections supplying the regulators, as indicated in Table 3-14.

Table 3-14 300-V Buses, Power Sources, and Loads

Bus No.	Power Source	Regulator Load
1 SECA	H7170X	MOD F (H7189)
	BBU	MOD H (H7189)
2	H7170X	MOD B (H7186)
	BBU	MOD C (H7186)
		MOD D (H7187) 2*
1 SECB	H7170Y	MOD E (H7180) 2*

* 2 required for dual-CPU configuration.

The 300 Vdc is cabled from the NBox to the buses located in the MPS cage. The buses are rails located below the regulators.

Connections to the regulators are made through high-current, high-voltage leaf contacts located on the front panel of the regulators.

The connectors mate electrically and mechanically with the buses when the regulator modules are inserted into the designated slots.

Each regulator must make at least two mechanical and electrical connections in the MPS:

- Multipin PC board to the MPS backplane
- 300-Vdc power to the bus

3.4.10 Environmental Monitoring Module

The environmental monitoring module (EMM) performs a variety of power system functions, as described in Chapter 2. In this chapter, the major logic elements contained in the EMM module are described in more detail.

The major elements included in the EMM are shown in Figure 3-15. These elements are:

- 8085A Microprocessor System
- Electric Key Monitor
- Regulator Control
- Status Registers
- AC LO/DC LO Circuits
- Total-Off Control
- Voltage Measuring Circuit
- BBU (Battery Backup Unit) Control
- Air Flow Sensor Control
- Temperature Sensing Circuits
- System Console Communication/Interface

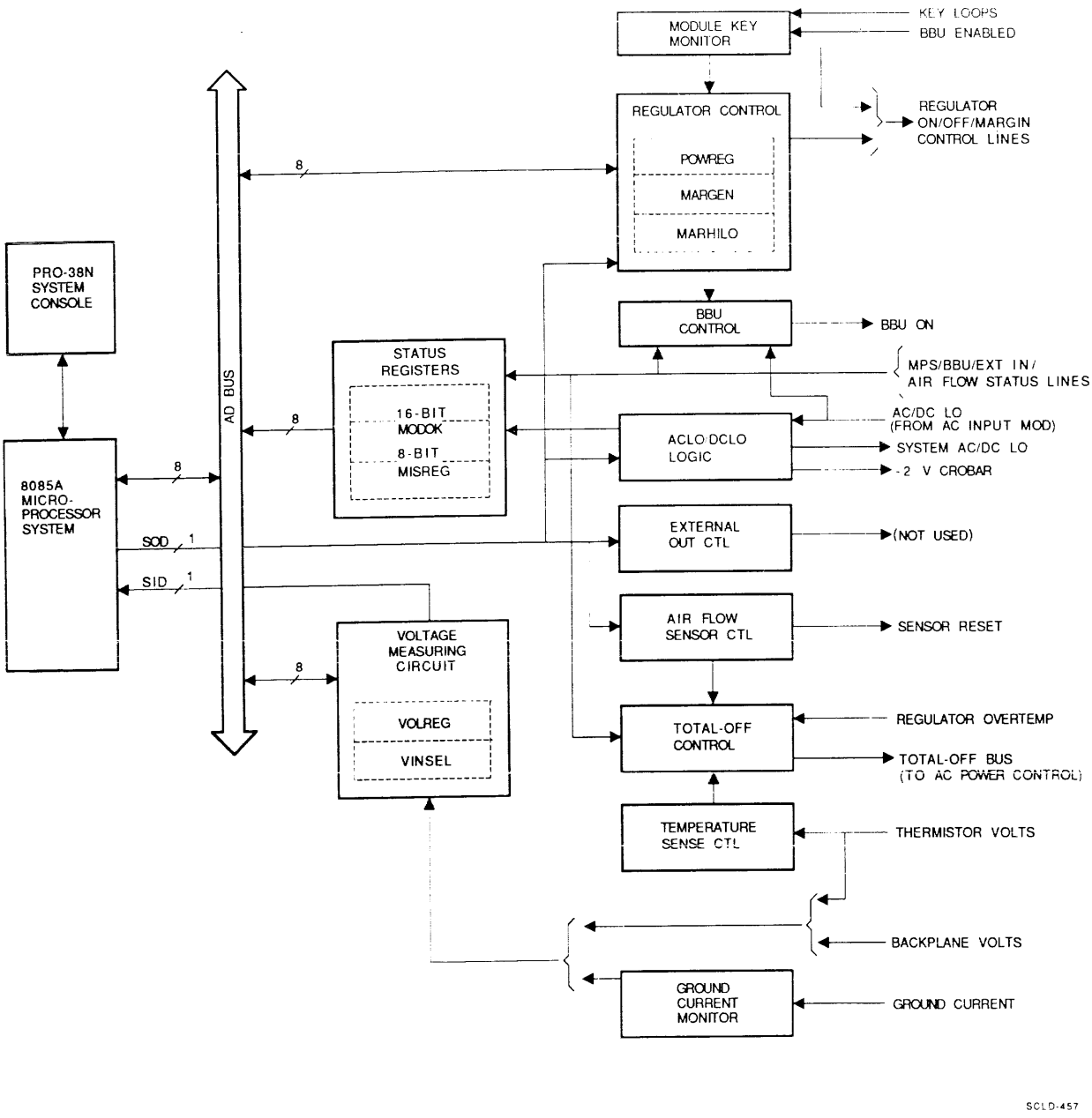


Figure 3-15 EMM Block Diagram

3.4.10.1 8085A Microprocessor System -- The main components in the 8085A microprocessor system are:

- 8085A 8-bit, single-chip microprocessor
- 8x8K byte PROM chip
- 8x1K byte RAM chip
- PCI (Program Communications Interface)

The 8K byte PROM stores the main body of the EMM program executed by the 8085A. The 1K byte RAM provides a working storage area and contains the software registers addressable by the console. The PCI connects the EMM to the console over the EIA RS423 or RS232 serial line.

NOTE

The interface to the serial line is a nonstandard interface that allows the data transmitted by the PCI to also be received by the PCI. This is a requirement of the EMM/console communications protocol.

The 8085A, PROM, RAM, and PCI all connect between an 8-bit address (ADR) bus and a multiplexed 8-bit address/data (AD) bus. The 8085A asserts I/O register addresses and the high-order byte of memory addresses on the ADR bus.

I/O register addresses and the low-order part of memory addresses are asserted on the AD bus, but only at the beginning of a read/write operation. During the rest of the read/write operation, the AD bus is used to transfer data between the 8085A and the addressed I/O register or memory (RAM) location.

The I/O registers in the EMM are the hardware registers (POWREG, MISREG, etc.) used by the EMM to control and monitor power system operation.

Other logic elements in the 8085A microprocessor system include:

- 5.0688-MHz oscillator for 8085A and PCI clock
- synch counter
- parity generator/checker

When a RAM parity error is detected, the 8085A PC (Program Counter) is reset to 0 and a selftest (including a RAM test) is done similar to the power-up sequence.

A selftest error (including a RAM failure) causes an error message to be sent to the console.

To prevent a parity error from resetting the 8085A during selftest operations, the EMM program clears a normally set control bit (the parity check enable bit) in the EMM's MISREG. The status of this control bit can be read at any time by the console. The execution of instructions by the 8085A consists almost entirely of a series of read/write data transfer operations between the 8085A and the memory and I/O register addresses.

The addresses, the data (as processed by the 8085A), and the sequence of read/write operations differ, depending on the instructions being executed. Each read/write operation is called a machine cycle, and one to five of these machine cycles are needed to execute a single 8085A instruction.

In addition to making 8-bit parallel transfers of data over the AD bus, the 8085A can send or receive single bits of serial data. The 8085A's serial output data line (SOD) is used to set or clear single control bits in some of the EMM's control circuits. The 8085A's serial input data line (SID) is used to read the output of the EMM's voltage measuring circuit.

3.4.10.2 Electric Key Monitor -- The electric key monitor circuit stops the power-up sequence by negating a logic level (KEY ENABLE) when a module keying fault is detected.

An LED on the EMM module (MODULE KEY FAULT) signals the fault.

In the VAX 8800 system, a mislocated circuit board in the following groups closes the parallel key.

1. CPU
2. Memory controller
3. Clock
4. I/O adapter module

The parallel key CKT, which is normally open, is closed if faulted to prevent module damage. This can occur when a module is plugged into a backplane slot intended for a different function group PC board.

For example, the parallel key loop is closed (by being grounded) when a CPU module is plugged into a memory controller module slot.

A series key loop is not being used in the VAX 8800 configuration.

In addition to detecting module keying faults, the electric key monitor disables the EMM from receiving console commands if the (module key fault) is asserted.

MODULE ENABLE, which indicates that battery backup is currently supplying power to the cabinet:

1. Holds the EMM microprocessor in an initialized state
2. Causes all regulator modules (except mod B) to shut down

When ac power returns, the 8085A's initialize signal is deasserted, allowing a normal power-up sequence to start.

3.4.10.3 Regulator Control Circuits -- The regulators for the VAX 8800 system, located in the CPU cabinet, are listed in Table 3-4. Figure 3-7 shows the MPS regulator configuration. The EMM's regulator control logic, shown in Figure 3-15, contains:

- POWREG -- allows each MPS regulator to be turned on or off by the EMM in response to console commands
- MARGEN
- MARHILO

The MARGEN and MARHILO registers allow each regulator to be margin-tested high and low, (+/-5%).

3.4.10.4 Regulator On/Off Control Circuits -- The power regulators in the MPS are turned on and off by command from the EMM. The regulator control lines (SHUT DOWN RTN MOD <B:H>) attached to the POWREG register, receive ON/OFF commands under EMM program control.

One POWREG bit, BBU REQ EN, when cleared, enables the automatic activation of battery backup power when AC LO occurs.

Interrupted ac power (AC LO) causes:

- Battery backup power to be activated
- All regulators (except regulator B) to be turned off
- Backup power to be applied through regulator B
- The EMM's microprocessor to be held initialized
- MODULE ENABLE from the BBU to be asserted

MODULE ENABLE negates DC REGL EN, which turns off all the regulators (except regulator B).

Refer to EMM Engineering drawing (D-CS-5415304-0-EMM5) for details of the circuit that allows regulator B to apply +5.0 Vdc to the memory backplane.

3.4.10.5 Regulator Margin Control Circuits -- A regulator is margin-tested by grounding either its high or low margin line. Eight MPS regulators <B:H> can be margined using:

- EMM microprocessor
- MARGEN register
- MARHILO register

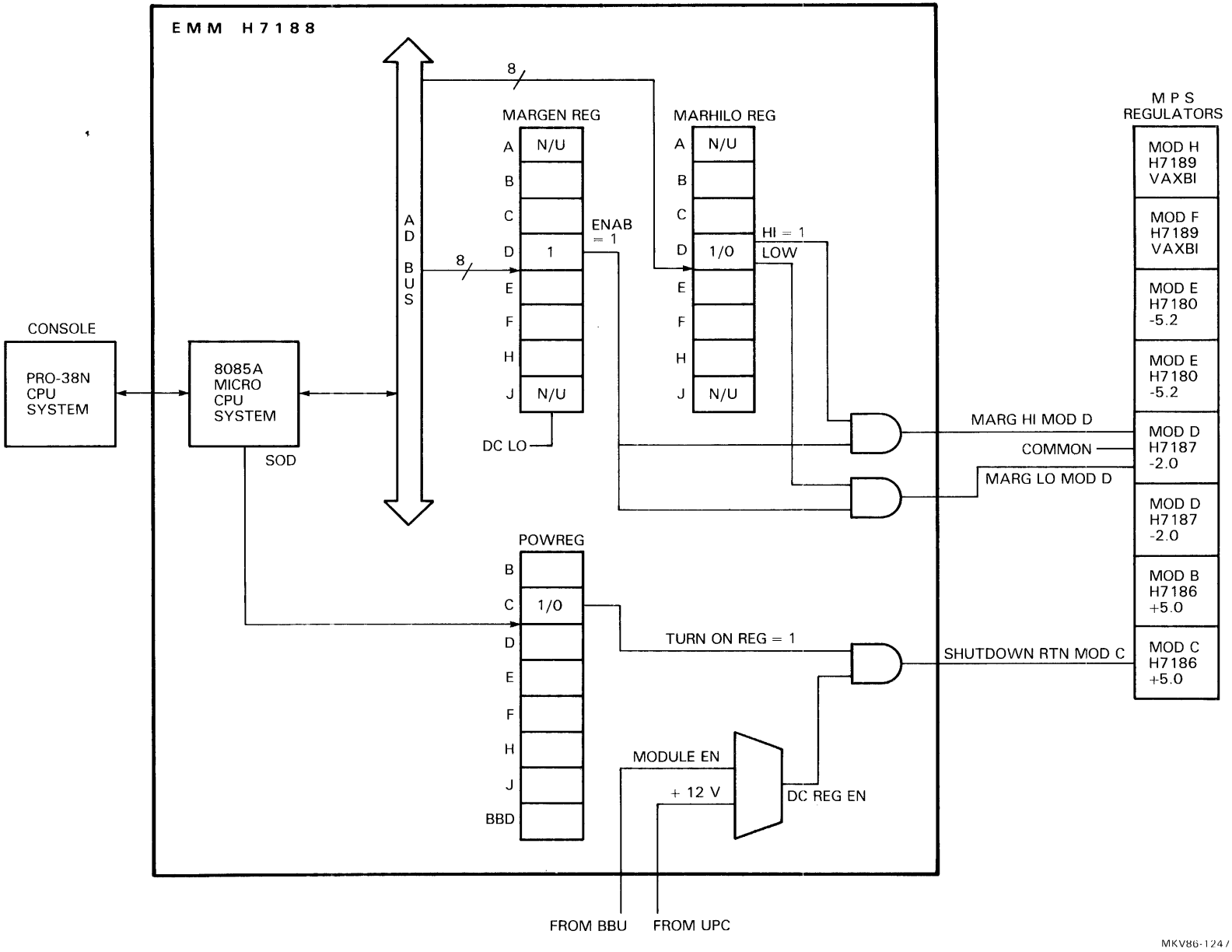


Figure 3-16 Voltage Margining Circuit

Figure 3-16 is a diagram of the EMM margining control circuits.

3.4.11 Status Registers

There are two status registers located on the EMM PC board:

1. MODOK -- 16 bits
2. MISREG -- 8 bits

The MODOK register indicates:

- Regulator module (MODULE <B:H> OK) status
- The EMM module ID (ID BIT <2:0>)
- AC LO status

Each regulator in the MPS cage asserts its MODULE OK status line once power has been applied, if its output voltage is within the regulation range and it has detected no other faults.

The ID number is determined by the backplane etch. It is the EMM's physical address used during EMM/console communications over the serial line. The AC LO signal is discussed later.

The MISREG register status bits indicate:

1. Air flow faults
2. RAM parity error
3. RAM parity check enable
4. -2-V crowbar
5. AC LO (latched)
6. DC LO (latched)
7. BBU AVAIL (fully charged and ready for use)

3.4.11.1 AC/DC LO Circuits -- The main power fault signals for the power system complex are:

1. AC LO
2. BUS LO

AC LO is asserted when the ac-input level to the power converters (H7170s) in the NBox falls below 165 V rms. The ILM responds to the AC LO signal by signaling the EMM (EMM5 AC LO MODL).

BUS LO is asserted when the 300-Vdc output voltage from the H7170s falls below 205 Vdc. It indicates that the input voltage to the MPS regulators is below that needed to maintain regulator specifications. BUS LO, which occurs after AC LO, indicates that:

1. System operator requests POWER DOWN, or
2. AC power is lost

Refer to power-down sequences discussed in Chapter 2 for details of power-down signal flows.

When ac power is lost, The EMM initiates a BBU REQUEST and signals the ILM to start providing BBU power to the memories by means of regulator B. The CPU, SBIA, and the system operator are notified.

The AC LO signal from the NBox causes AC LO IN L to be asserted in the EMM. The BUS LO signal from the NBox causes BUS LO IN L to be asserted. A timing diagram for these signals is shown in Figure 3-15.

BUS LO IN L is also asserted in the EMM when any MODULE OK signal, from a regulator or the NBox, is negated. At powerup, this signal holds BUS LO IN L asserted until the dc regulators are turned on and operating normally.

The EMM AC LO IN and BUS LO IN signals generated, are used to signal power loss warning signals for:

- The main CPU -- CPU AC LO
  CPU DC LO
- The SBIA -- SBIA AC LO
  SBIA DC LO

The AC LO IN and BUS LO IN output signals are latched in the EMM and can only be cleared by a console POWER UP command after the fault condition has been cleared. Figure 3-17 shows the AC/DC LO timing signals.

Figure 3-18 shows a detailed view of the AC LO/DC LO circuit.

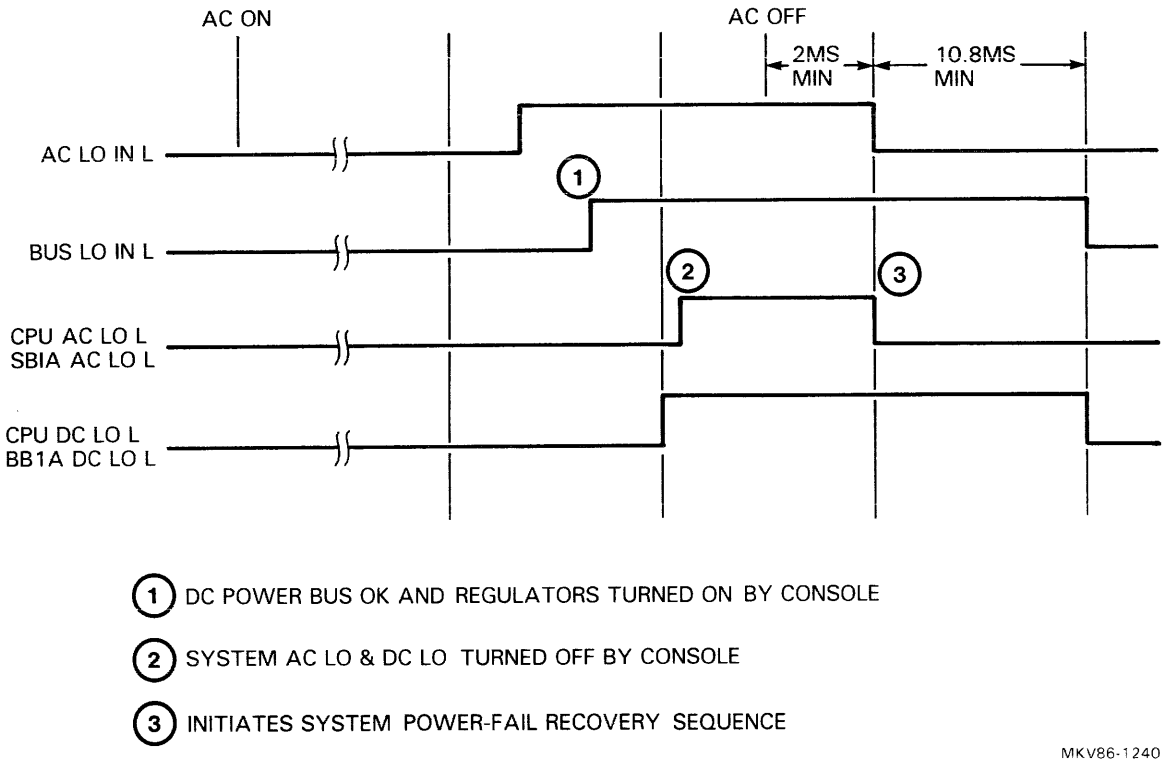


Figure 3-17 AC/DC LO Timing Diagram

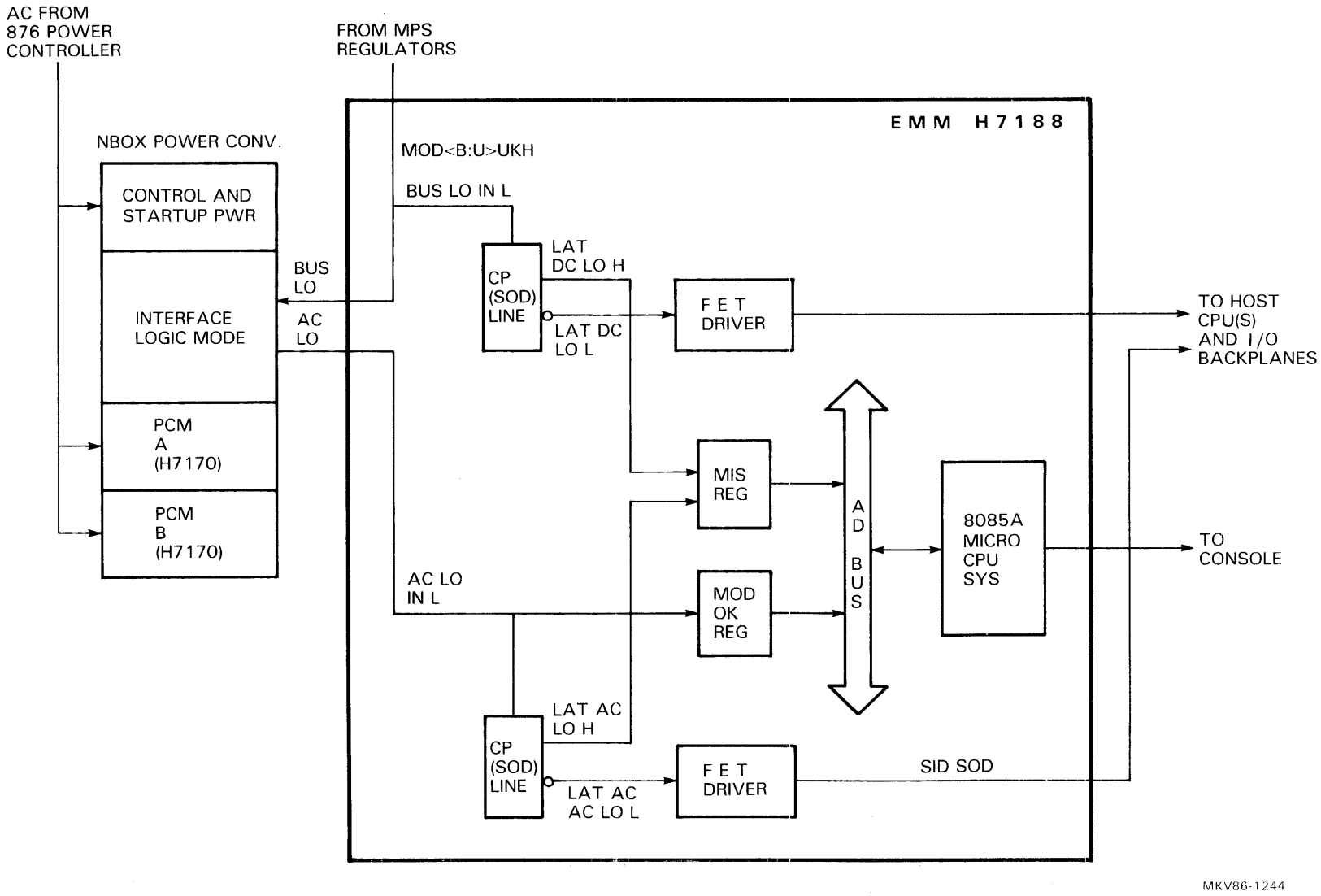


Figure 3-18 AC/DC LO Circuit

3.4.11.2 Total-Off Control and Indicator Circuits -- The total-off circuit initiates an emergency shutdown of ac power to all modules powered by the 876A by tripping BRKR2, the main 876A CB. A TOTAL POWER OFF command can occur when:

1. The CPU cabinet overheats
2. A regulator overheats
3. Air flow falls below specified limit
4. The air filter interlock is open

When BRKR2 trips, ac power is removed from:

- The air mover (blower)
- The BBU
- The NBox (three-phase and single-phase)
- The BAll-A

The TOTAL OFF command originates from two sources:

1. EMM
2. ILM

The EMM monitors The following TOTAL OFF sensors (see Figure 3-15):

- Air flow
- CPU cabinet temperature
- Regulator (C:H) overtemperature
- CPU cabinet and air filter interlocks

During normal system operation the ILM monitors:

1. Regulator B overtemperature (MOD B OT)
2. Air filter screens in bottom of cabinet

During BBU operation, the ILM also monitors MOD B OT.

When any of the conditions cited above default, a TOTAL OFF command is sent by either the EMM or the ILM to disable BRKR2 in the 876A.

AC power can be restored by manually resetting BRKR2 in the 876A.

If the EMM initiates the TOTAL OFF command, the magnetic latches on the front panel of the EMM module display a 4-bit code to indicate the cause of the fault. The latches retain the code when power is removed and reset on powerup.

When a TOTAL OFF command is initiated by the control console, the EMM does not cause the magnetic latch display to indicate the cause of the fault.

The interface logic module (ILM) asserts a TOTAL OFF RTN when SW2 (air filter interlock) is grounded, indicating a fault.

3.4.11.3 Temperature Sensing Circuits -- The CPU temperature sensing circuits monitored by the EMM are voltage divider series resistor strings in which one component is a temperature-sensitive thermistor resistor. The voltage sensing connection to the EMM is at the junction of the fixed resistor and the thermistor.

Measurements made by the temperature-sensing circuits are:

- CPU cabinet temperature -- (THERM SENSE 1)
- CPU/MPS amb. temperature -- (THERM SENSE <2:4>)
- CPU delta temperature -- (THERM SENSE <2:4> - 1)

The EMM's internal program routinely monitors the CPU cabinet temperatures and regulator overtemperatures. Temperature measurements are also made in response to a MEASURE command from the system console. Console requests usually follow temperature default indications from the EMM. Thermistor (T3) is used to initiate an emergency power shutdown when:

1. The EMM fails
2. The EMM program is faulty
3. T3 indicates excessive temperature

The temperature sensing circuitry contained on the EMM module is shown in Figure 3-19.

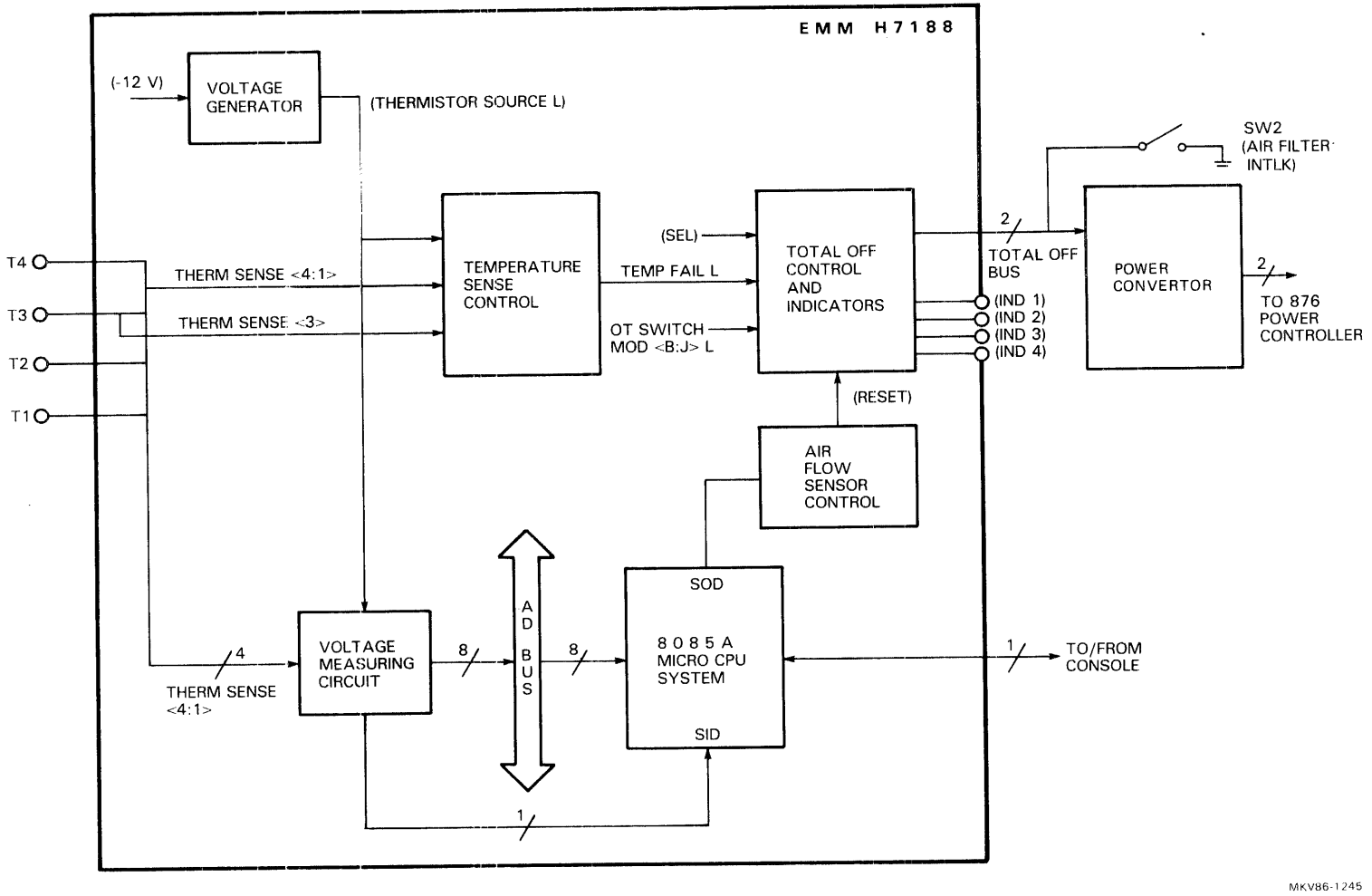


Figure 3-19 Temperature Sensing Circuit

3.4.11.4 Voltage Measuring Circuit -- The EMM performs dc voltage measurements under program control on the following:

1. All MPS regulators <B:H>
2. All thermistors <1:4>

Voltage measurements are performed using the circuits shown in Figure 3-20. The unknown voltage is determined using the following steps:

1. EMM program specifies voltage to be tested.

2. Unknown voltage is selected using VINSEL command and fed to voltage comparator.

3. Reference voltage (8-bit D/A) is also connected to comparator ramps up from 0 V at 27 MV/step.

4. When reference volts = unknown volts, comparator outputs a signal.

5. Digital value in D/A = value of unknown voltage.

6. Analog voltage = 27.6 MV X decimal equivalent of 8-bit digital code for regulator voltage.

7. Analog voltage = 333.3 MV X decimal equivalent (compliment) of 8-bit digital code for thermistor voltage.

NOTE

To maintain positive slope for converting temperature measurements (thermistors have a negative temperature coefficient), the A/D magnitude is the complement of the voltage drop across the thermistor. That is, it is the digital complement of the value measured by the EMM's voltage measuring circuit and contained in the VOLREG after the measurement.

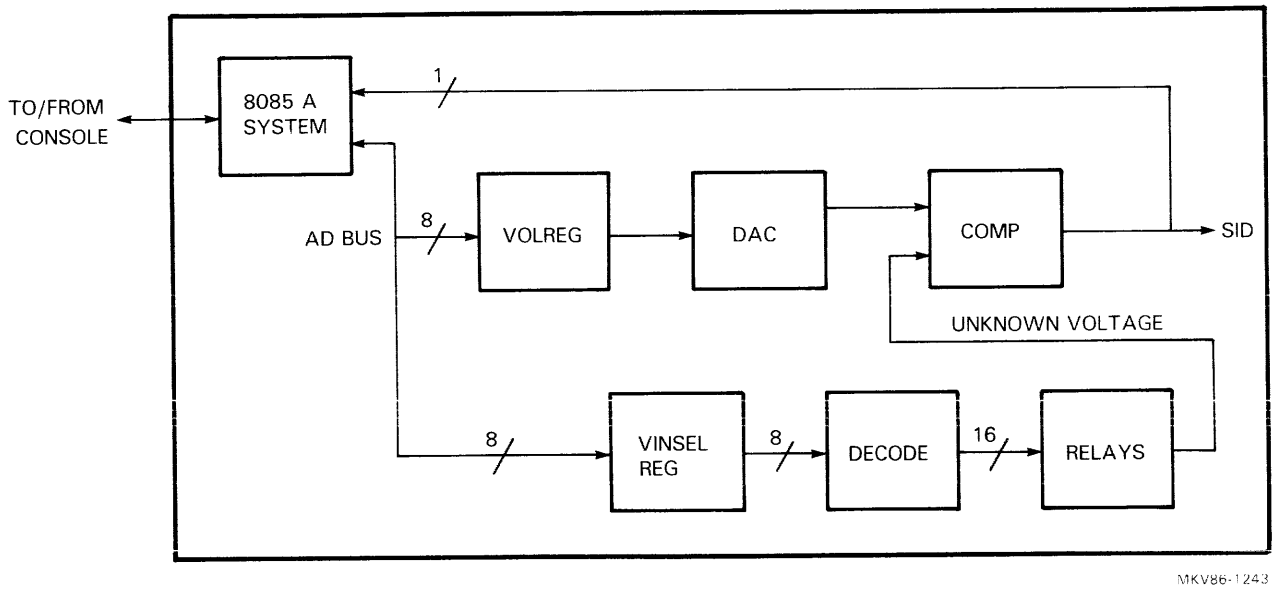


Figure 3-20 Voltage Measuring Circuit

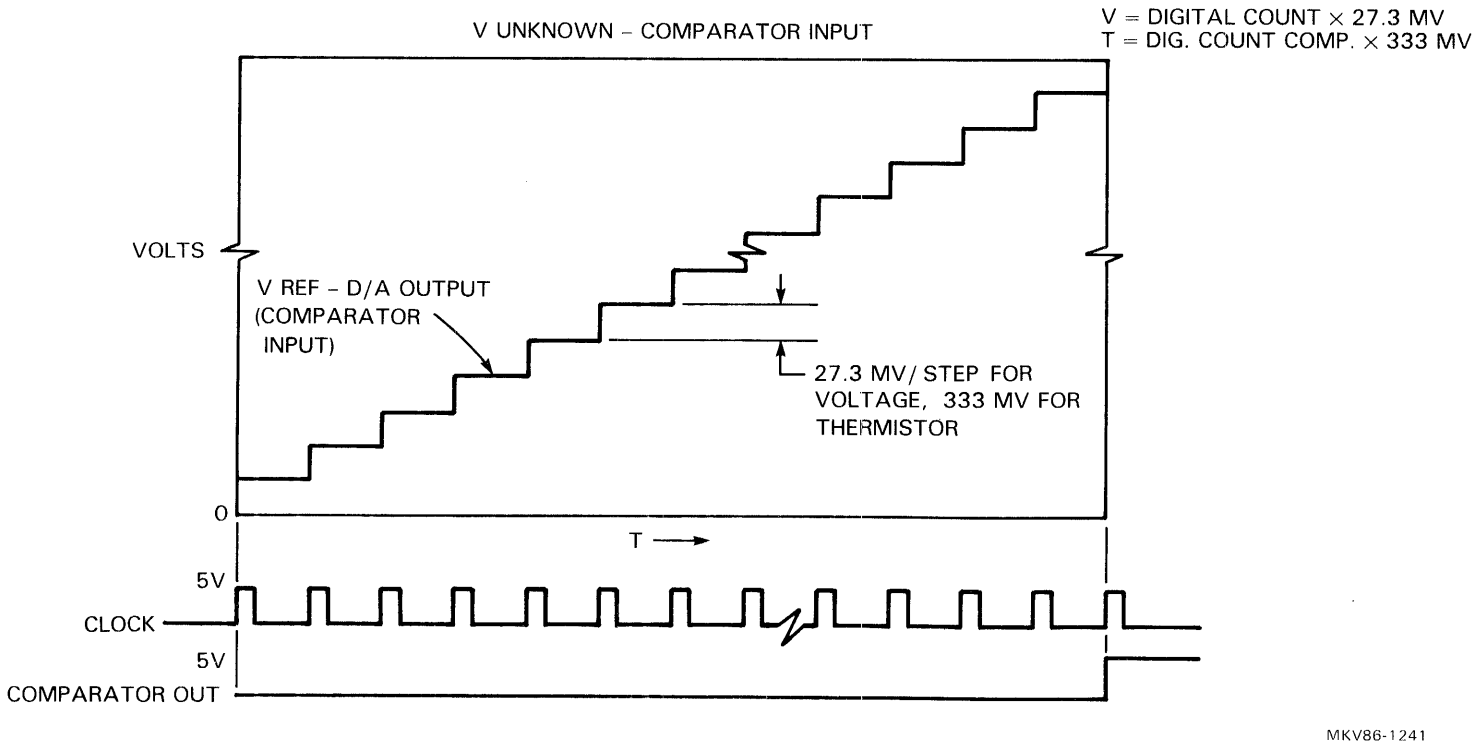


Figure 3-21 Voltage Measuring Technique

3.4.11.5 EMM/Console Voltage Tests -- Voltage measurements of regulator output and thermistor temperature can be requested by the console using the MEASURE command.

Voltage readings returned to the console by the EMM in response to a MEASURE command, consist of:

1. An 8-bit binary value
2. A sign bit
3. An input select code

3.4.11.6 Battery Backup Unit (BBU) Control -- The EMM's BBU control logic activates battery backup power when the following signals are asserted:

- AC LO (from 876A in NBox)
- MODULE OK H (from Regulator B)

The EMM responds to the above conditions by asserting BBU REQ EN to the ILM. The ILM asserts BBU RQST RTN and provides backup power if the BBU batteries are fully charged.

The BBU provides 300 Vdc to the NBox, which is connected to the 300-Vdc bus, as is regulator b. Regulator b, the only regulator active during battery backup operation, provides +5 V to the memory array and the CSP. The CSP module uses MOD B +5 V to create +/-12 V to bias the EMM and ILM modules during battery backup.

When ac power returns to the 876A, the 300-V output from the BBU is disabled and power is supplied by the NBox H7170 modules.

When power is applied to regulator B, the EMM asserts ARRAY +5 V OK, enabling refresh operations in the memory array. The signal is asserted as long as the MODULE OK signal for regulator B is true.

Refer to Figure 3-22 for a detailed diagram of the BBU system.

3.4.12 Battery Backup Unit (H7231-M)

The battery backup unit (BBU) for the VAX 8800 system delivers 300-Vdc backup power to 300-V bus 2 when utility power is momentarily interrupted.

The H7231-M consists of:

- A charger (H7230)
- A dc-to-dc converter (H7240)
- A 48-V rechargeable, lead acid battery pack

The charger receives 120 Vac at its input from the 876A power controller and provides two levels of charge to the battery pack:

1. Charge - 400 milliamp to 59 volts dc battery output
2. Maintenance - 10 milliamp above 59 volts dc

The 400-milliamp charge level is reinitiated when the dc battery output falls below 52 volts.

The dc-to-dc converter (H7240) steps up the 48-V battery voltage to 300 Vdc using a flyback transformer converter. The 300 V generated by the converter is provided on demand to the NBox, which is connected to the 300-Vdc bus feeding the regulators.

Regulation of the H7240 output is accomplished using pulse width modulation (pwm). The ac input line voltage and the dc battery voltage are continuously monitored.

The H7240 has the following protection circuits:

1. An overvoltage protection circuit to inhibit converter operation if the output voltage exceeds 330 Vdc nominal.

2. A primary overcurrent sense circuit to protect circuit components from output overload condition (3 A surge limit, fuse protected).

3. A power transistor overtemperature circuit to inhibit operation above 90 degrees C (194 degrees F).

The battery backup system has two special features:

- FAIL SAFE ENABLE
- DIGITAL power bus monitor

Both signals open the power relay terminating battery backup power. The FAIL SAFE ENABLE signal allows the H7240 to respond to control signal inputs.

The DIGITAL power bus interface monitors the emergency shutdown signal (TOTAL OFF RTN), which, when asserted, disables the output of the H7240.

The BBU interfaces to the power system by means of a 15-pin D type connector. The interface signals are defined in Table 3-15.

Figure 3-22 shows the BBU subsystem block diagram.

Table 3-15 Battery Backup Interface Signals

Signal	Definition
BBU Enable	Activates the series power relay. Must be asserted before battery backup is requested.
BBU ENABLE RTN	BBU ENABLE signal return.
BBU AVAIL	When true, indicates BBU available to provide battery backup power.
BBU AVAIL RTN	BBU AVAIL signal return.
ILM BBU RQST	Requests battery backup power to the load when the loss of the main ac has occurred.
ILM BBU RQST RTN	ILM BBU RQST signal return.
MOD ENABLE L	Enables "Regulator B" to accept battery backup power.
MOD ENABLE RTN	MOD ENABLE signal return.
BAT STAT	Indicates battery status as follows:

State	Current Flow	Batteries
OFF	no current	charging
READY	constant	full charge
	or	
	1 Hz rate (slow)	charging
ON	10 Hz rate (fast)	discharging

NOTE

The signal pairs, BBU ENABLE/BBU and ENABLE RTN ETC, are obtained from opto-couplers that provide ground isolation and high-voltage protection to the ILM.

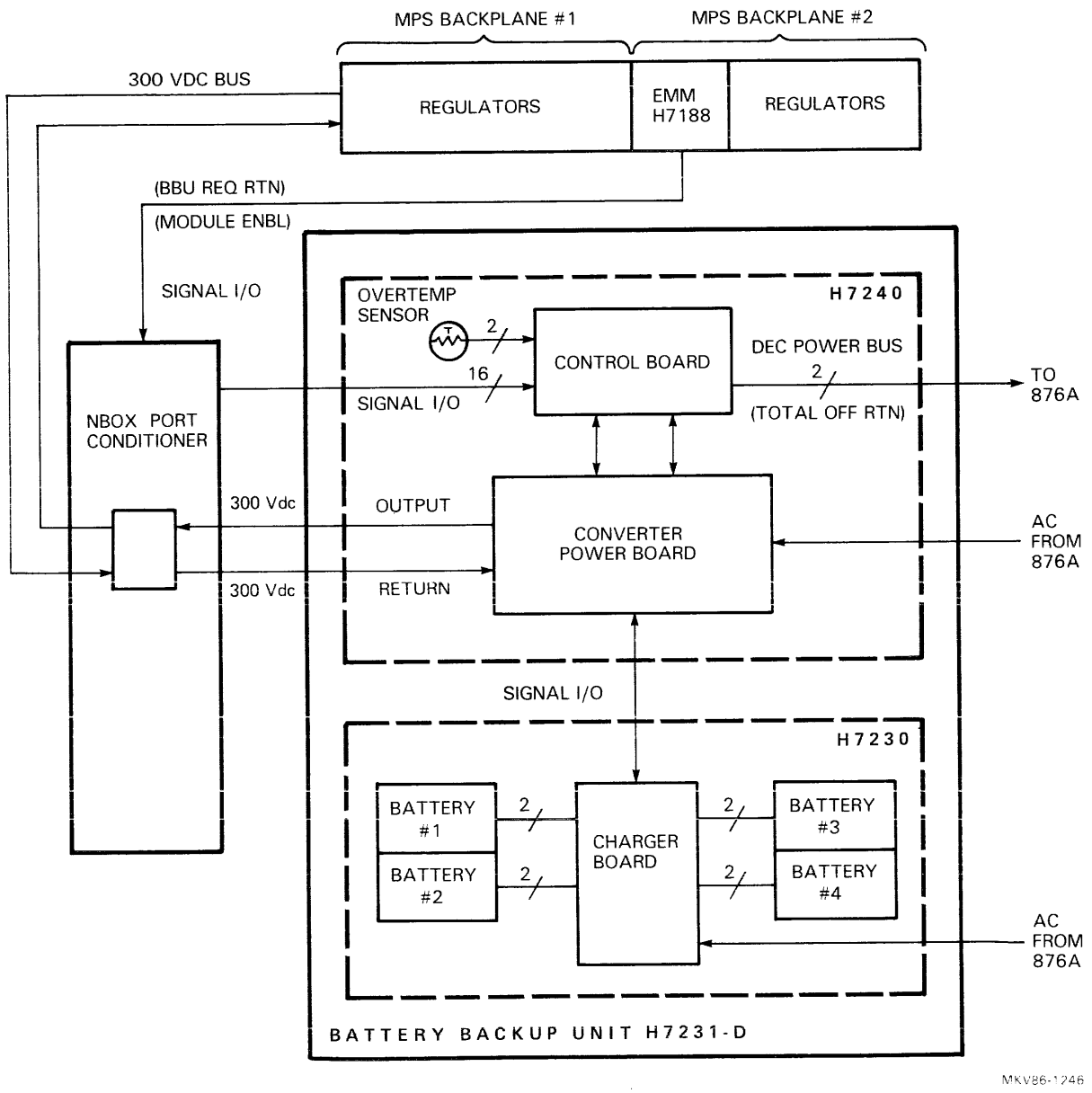


Figure 3-22 BBU Block Diagram

3.4.13 Air Flow Sensing Circuit

The EMM monitors two air flow sensors located in the CPU cabinet, AIR FLOW 1 and AIR FLOW 2. The EMM program checks the state of each sensor by reading the MISREG.

Wnen a single air flow fault is detected, a five-minute shutdown notice is sent to the system's console.

When both detectors indicate an air flow fault, a two-minute shutdown notice is sent to the system's console.

The console can also read the air flow fault sensors at any time.

The air flow sensors are reset by the EMM during a power-up sequence, and at any time thereafter in response to a WRITE command.

Figure 3-23 shows the air flow sensing circuit.

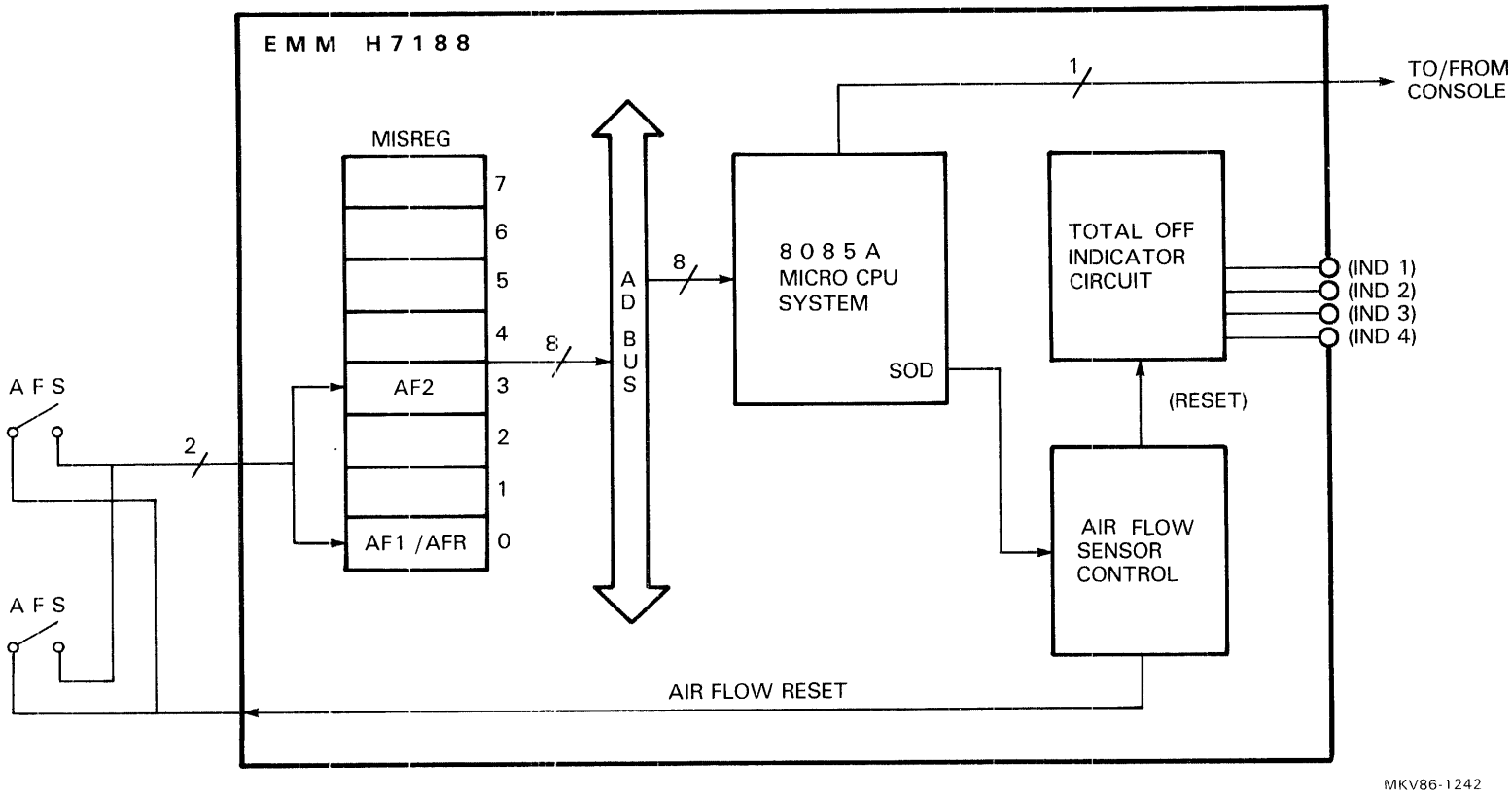


Figure 3-23 Air Flow Sensing Circuit

3.5 COOLING SUBSYSTEM

The cooling air moving through the CPU cabinet is drawn up through the bottom of the CPU cabinet with negative pressure created by the three-phase motor.

The moving air passes through a washable aluminum mesh filter just below the NBox and 876A, located on the bottom shelf, before entering the CPU cardcage. The air filter is interlocked to assure that it is:

1. In place
2. Correctly seated

Upon leaving the cardcage area, the moving air enters the MPS area and exits at the back of the CPU cabinet.

The maximum VAX 8800 system (a dual-CPU configuration) is expected to dissipate a 6500-watt heat load as follows:

1. 4100 watts in the CPU logic module area
2. 2000 watts in the MPS cage
3. The balance of 400 watts in the NBox

An additional 400 watts is dissipated by the three-phase motor.

The power dissipated in the front-end cabinet is not sufficient to require forced air cooling.

The temperature rise in the CPU cabinet is 17 degrees C (62.6 degrees F), with a 10 degree C rise (50 deg. F) contributed by the CPU logic and 7 degrees C (44.6 degrees F) rise allowed for the MPS power regulators.

An air flow rate of 1100 to 1200 CFM has been determined to be sufficient to satisfy the above requirements. The motor/blower assembly is mounted within a steel plenum/housing secured to the four corners of the CPU cabinet above the MPS cage.

Four air outlets at the rear of the cabinet exhaust the heated air. The outlets direct the air into the rear door mounted exhaust housing and plenum, which further guide the air downward and to the rear.

At the exhaust outlets, there is a damper mechanism that deflects the air flow during maintenance. The plastic/sheet metal plenum is lined with acoustically absorbent material that allows it to function as a noise-reduction muffler.

Figure 3-24 is an illustration of the air flow in the CPU cabinet.

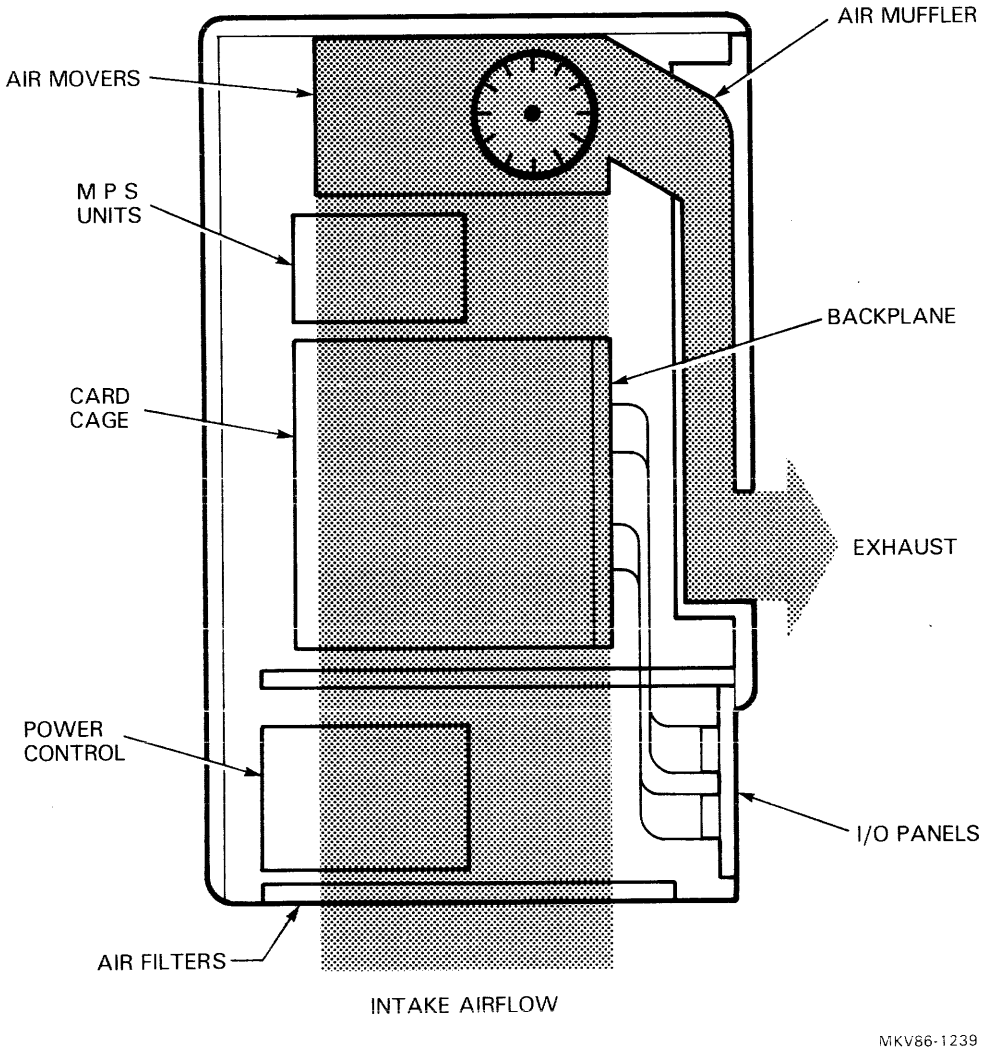


Figure 3-24 Air Flow Path

SECTION 5
CLOCK MODULE

CHAPTER 1
INTRODUCTION

1.1 BASIC OPERATION

The clock generator generates and distributes the system clocks to the left and right CPUs, the memory subsystem, and the I/O adapters. It is located on the clock (CLK) module and controlled by the PRO-38N console. A basic block diagram is shown in Figure 1-1.

The PRO-38N controls the clock generator using three registers. As shown in the block diagram, these registers (the clock control, clock period, and clock burst count registers) are in the clock generator, but are written via the console interface, which is also located on the CLK module. A fourth register (the clock/timeout status register) allows the PC350 to examine clock status bits; these are also read via the console interface. (The status register is actually the output of a data multiplexer within the console interface.) The console interface and the functions it performs in the system are described in the Console Subsystem Technical Description.

The system clocks generated by the clock generator are listed in Table 1-1. Several copies of each clock are transmitted on the system backplane, with each copy having only one destination module. (A module can have more than one copy of the same clock.) The clock generator also generates a slow clock used to increment timeout counters in the CPUs, memory controller, and I/O adapters.

Table 1-1 System Clocks

Clocks	Function
A CLK, B CLK	Main system clocks used by CPU modules and I/O adapters to sequence and synchronize operation.
F A CLK, F B CLK	Free-running system clocks used by memory controller to sequence and synchronize operation. Also connect to I/O adapters (not used by NBI).
W CLK	Free-running system clock used to write RAMs in two CPU module types (SLC0 and SLC1).

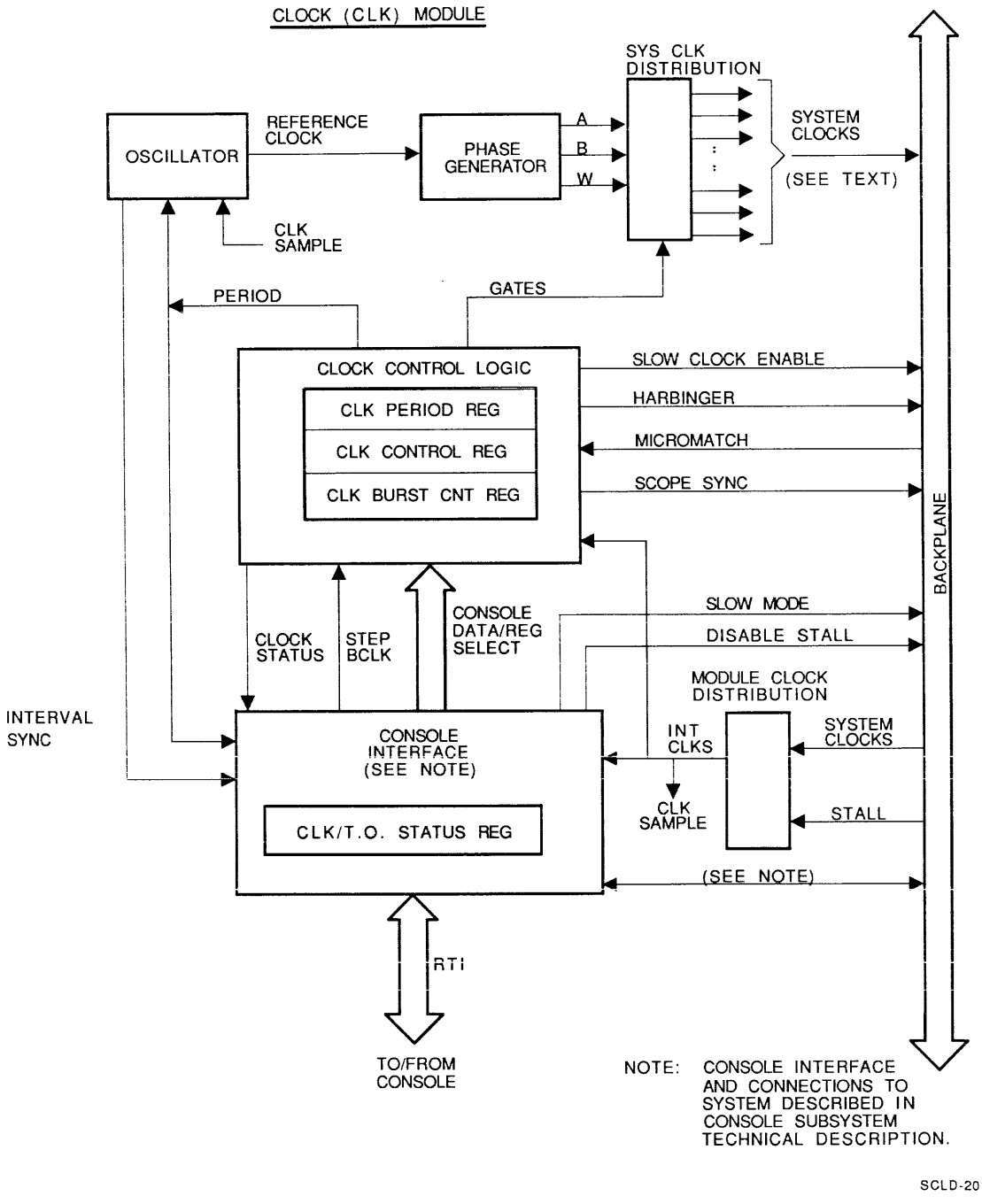


Figure 1-1 Clock Generator (and Console Interface) on Clock Module

1.2 BASIC COMPONENTS AND TIMING

The clock generator consists of the following basic components.

1. Oscillator
2. Phase generator
3. Clock distribution circuits
4. Clock control logic

The oscillator generates a reference clock that is used by the second component, the phase generator, to produce two nonoverlapping clock phases, PHASE A and PHASE B, plus a W PHASE that is similar to PHASE A but longer in duration. The clock distribution circuits, in turn, gate PHASE A and PHASE B with signals from the clock control logic to produce several copies of the main (gated) system clocks, A CLK and B CLK. These clocks may be started, stopped, and bursted by the clock control logic. The clock distribution circuits also produce the free-running (ungated) system clocks, F A CLK, F B CLK (from the A and B phases) and the free-running W CLK (from the W phase).

The slow clock produced by the clock generator, called SLOW CLOCK ENAB, is generated in the clock control logic. It is derived from B CLK (asserted once for every 65,536 B CLKs that are generated) and, thus, will be synchronized with the main system clocks when they are started, stopped, and bursted.

System clock timing is shown in Figure 1-2. The phase generator produces the clocks by dividing the reference clock frequency by six. Because the reference clock is normally 120 MHz, this results in a normal system clock frequency of 20 MHz and, thus, a normal system clock period of 50 ns. However, the system's console can change the reference clock frequency to produce system clock periods ranging from 140 ns down to 40 ns.

The period of SLOW CLOCK ENAB, derived from B CLK, also depends on the reference clock frequency. This signal is asserted once every 3.28 milliseconds at the normal system clock rate.

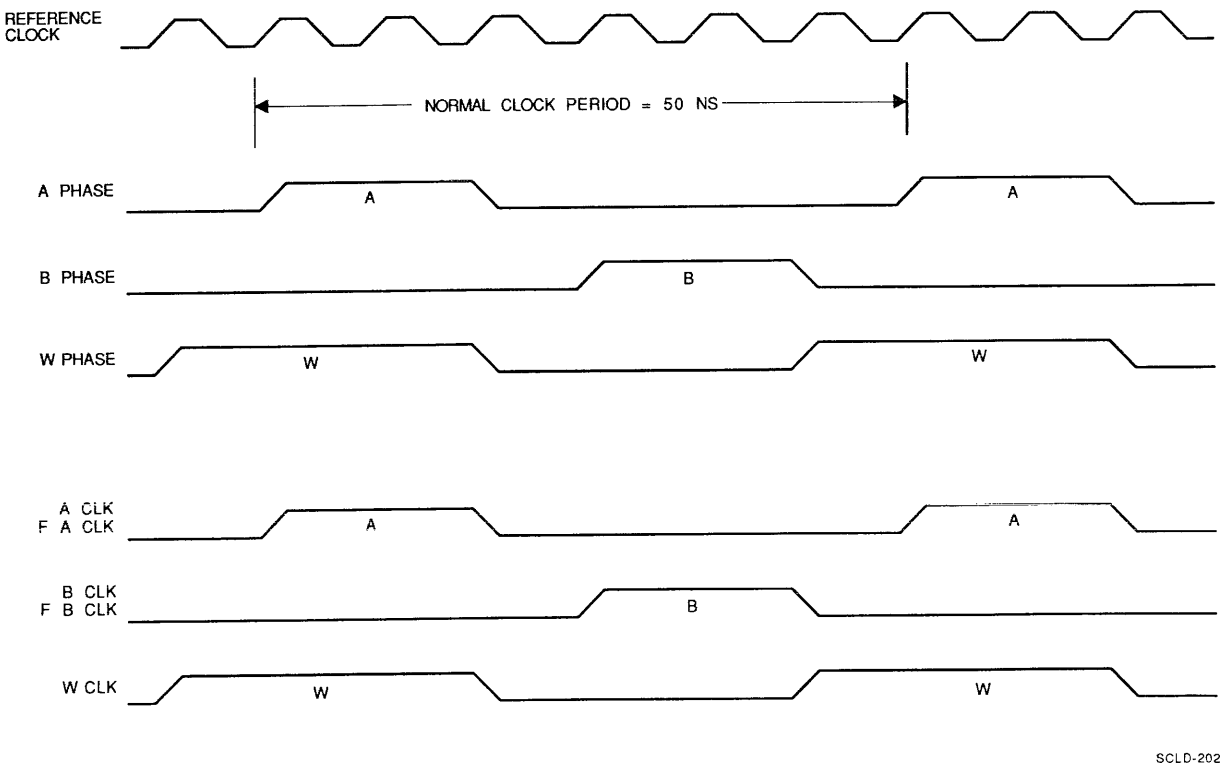


Figure 1-2 System Clock Timing Diagram

1.3 CLOCK CONTROL (BY CONSOLE)

The PC350 console can control the main (gated) system clocks in the following ways:

1. Stop the clocks (unconditionally)
2. Start the clocks
3. Burst the clocks (burst up to 255 pairs of A CLK and B CLK)
4. Single-step the clocks (a burst of one A CLK and B CLK)
5. Enable the clocks to stop on micromatch
6. Change the clock period (also changes free-running clock period)
7. Single-step the B CLK only (when loading VBus address)
8. Disable A CLK stalls (by left or right CPU)

The console can also continuously assert the clock generator's SLOW CLOCK ENABLE output that is used to generate timeout errors in the CPUs, memory controller, and I/O adapters. This is done during diagnostics to set the timeout flags without having to force the error condition.

The clocks are controlled mainly by the three registers in the clock generator. Control bits in the clock control register allow the console to start the main system clocks and to stop them either unconditionally (when a micromatch occurs) or after a specified number of clocks are generated (a clock burst). A control bit in this register also allows the slow clock to be continuously asserted. The number of clocks in a burst of system clocks is specified by the burst count register. (A burst count of one single-steps the clocks.) Clock period is determined by the value in the clock period register. Bit format for the three registers is given in Figures 1-3 through 1-5. Clock control register bit definitions are given in Table 1-2.

Most console control of the system clocks affects both CPUs. That is, the clock outputs to each CPU from the clock generator are synchronized with each other and are not independently controlled. For example, when clocks are unconditionally stopped and then restarted at a different frequency (the clock period changed), sequencing in both CPUs stops and then starts again at the different rate. Similarly, clocks in both CPUs single-step and burst together. This is also true when only the B CLK is single-stepped during the loading of a VBus address. In this case, the single-stepping is not controlled by a clock generator register; it is controlled by a bit in the VBus control register located in the console interface.

The clocks also stop together when a stop on micromatch occurs in either CPU. A micromatch is when the current microPC value in a CPU's IBox equals a preloaded microaddress (also in the IBox). This condition, in addition to stopping the clocks when enabled by the console, always causes the clock generator to generate a scope sync on the backplane.

Before the console unconditionally stops the clocks, or before it enables the clocks to stop on a micromatch or after a clock burst, it must place the memory controller and some I/O adapters, such as the NBI, into a slow (unpipelined) mode of operation. The console does this by asserting a SLOW MODE signal on the NMI. The signal is asserted when the console sets a control bit in a console interface register (control register 1).

Another early-warning signal, HARBINGER, is asserted on the NMI by the clock generator hardware just before the clocks actually stop. The memory controller, which is sequenced by the free-running clocks and not the gated clocks, uses the signal to block its free-running clocks and simulate the stopped clock condition occurring in the CPUs and other system components. This signal is also deasserted by hardware just before the clocks start to allow the memory controller to unblock its free-running clocks.

1.4 CLOCK STALLS

One type of system clock control does not affect both CPUs. This is the stalling of A CLK in some CPU (and console interface) logic when cache cannot be immediately read or written. That is, sequencing (by A CLK) is stopped temporarily until the access can be made. The CCS module in the stalling CPU inhibits the appropriate A CLKs by asserting a STALL signal. The gating of an A CLK with STALL to stop sequencing is not done in the clock generator. It is done in the clock distribution circuits for each module. These are the circuits that receive the system clocks transmitted on the backplane by the clock generator. Every module has its own clock distribution circuits including the CLK module itself. (Refer again to Figure 1-1.)

Although the console does not assert STALL, it can prevent its assertion by either of the CPUs by writing control bits in a console interface register (control register 0). This is done during system initialization to ensure that all the system clocks are generated when microcode is loaded. The STALL signal is generated by microcode, and unwanted stalls could occur during the loading process if not disabled.

1.5 CLOCK STATUS

The console determines clock generator status by reading the clock/timeout status register. Register bit format is shown in Figure 1-6. Bit definitions are given in Table 1-3.

There are four clock generator status bits. One indicates whether the main system clocks are stopped. The others, valid only when the clocks are stopped, indicate whether the clocks stopped at the end of a specified clock burst or because of a micromatch in the left or right CPU.

Two other bits in the register are used as flags by the console to determine if either the left or right CPU is hung in a stalled (A CLK) state. The flags are located in the console interface. Periodically, the console reads the flags and (if the flags are cleared) writes a control bit in control register 2. The control bit sets each flag if a stall in the associated CPU is in progress. Once set, a flag is cleared only when the stall ends. Thus, if it is still set when read again by the console (at the end of the timeout period), it indicates a hung state with the set bit indicating the faulty CPU.

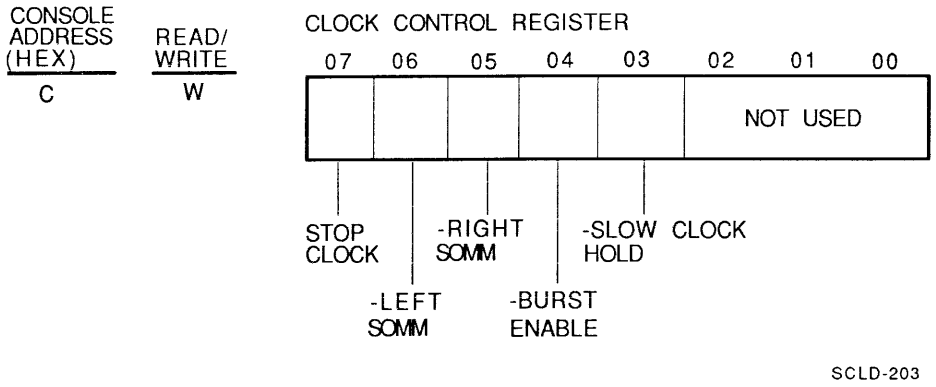


Figure 1-3 Clock Control Register

Table 1-2 Clock Control Register Bit Descriptions

Bit(s)	Description
<7>	Stops clock. When set to 1, stops main system clocks. When set to 0, starts main system clocks. Initialized to 0 by console.
<6>	Enables left CPU to stop on micromatch (when set to 0). Initialized to 1 by console.
<5>	Enables right CPU to stop on micromatch (when set to 0). Initialized to 1 by console.
<4>	Enables clock to stop at end of clock burst specified by burst count register (when set to 0). Initialized to 1 by console.
<3>	Holds (force assertion of) slow clock enable signal on NMI (when set to 0). Initialized to 1 by console.
<2:0>	Not used.

CONSOLE ADDRESS (HEX)	READ/ WRITE
D	W

BURST COUNT REGISTER

07	06	05	04	03	02	01	00
CLOCK BURST COUNT							
128	64	32	16	8	4	2	1

NOTE: BURST COUNT CAN BE 1 TO 255 FOR NORMAL OPERATION. A COUNT OF 0 BURSTS 256 CLOCKS BUT BURST DONE FLAG IN CLOCK/TIMEOUT STATUS REGISTER IS SET TO 1 BEFORE (AS WELL AS AFTER) CLOCK BURST.

SCLD-204

Figure 1-4 Burst Count Register

CONSOLE ADDRESS (HEX)	READ/ WRITE
E	W

CLOCK PERIOD REGISTER

07	06	05	04	03	02	01	00
	CLOCK PERIOD (N)						
	64	32	16	8	4	2	1

LOW FREQ SELECT

NOTES:
1. SYSTEM CLOCK FREQUENCY = (N + 1) X .25 MHZ
2. SYSTEM CLOCK PERIOD = $\frac{4000}{(N + 1)}$ NANOSECONDS
3. VALUES OF N FOR SPECIFIED LOW AND HIGH RANGE OF SYSTEM CLOCK FREQUENCIES ARE GIVEN BELOW.

LOW FREQ SELECT	N	CLOCK FREQUENCY	CLOCK PERIOD
0	56 TO 99	14.25 MHZ TO 25 MHZ	70 NS TO 40 NS
1	27 TO 99	7 MHZ TO 12.5 MHZ	140 NS TO 80 NS

SCLD-205

Figure 1-5 Clock Period Register

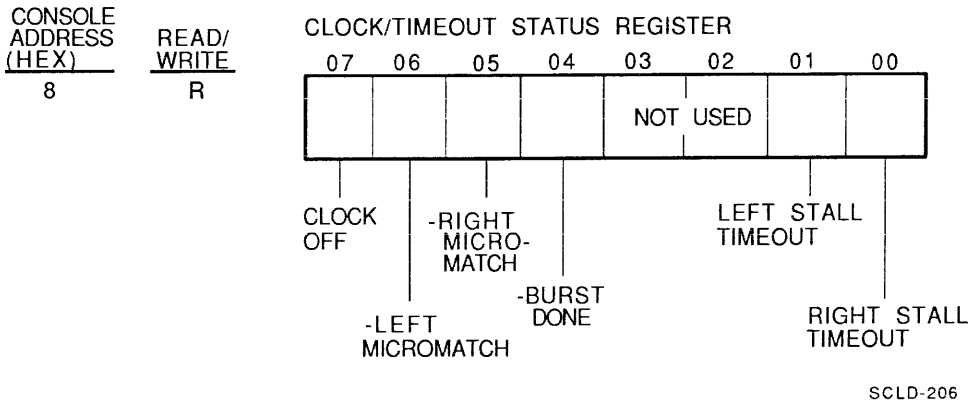


Figure 1-6 Clock/Timeout Status Register

Table 1-3 Clock/Timeout Status Register Bit Descriptions

Bit(s)	Description
<7>	Clock off. Indicates main system clocks are turned off.
<6>	When equal to 0, indicates micromatch in left CPU.
<5>	When equal to 0, indicates micromatch in right CPU.
<4>	Burst done. When equal to 0, indicates burst counter has reached a count of 0.
<3:2>	Not used.
<1>	Left stall timeout. Indicates left CPU hung in A CLK stall state.
<0>	Right stall timeout. Indicates right CPU hung in A CLK stall state.

CHAPTER 2
FUNCTIONAL DESCRIPTION

2.1 DETAILED BLOCK DIAGRAM

A detailed block diagram of the clock generator is shown in Figure 2-1. Clock generator input and output is defined in Tables 2-1 and 2-2.

2.1.1 Oscillator

The oscillator consists of a crystal-controlled low frequency source (250 kHz), and a phase-locked loop circuit. The phase-locked loop circuit multiplies the low frequency to produce the input (REFERENCE CLOCK) to the system clock phase generator. The low frequency is normally multiplied by a factor of (N 1) X 6, where N is the value loaded in the clock generator's clock period register.

Besides allowing different REFERENCE CLOCK frequencies (based on the value of N), the phase-locked loop closely controls the current REFERENCE CLOCK frequency by continuously comparing the phase of the crystal-controlled low frequency source with one of the system clocks derived from REFERENCE CLOCK. This clock sample, divided by N 1 in the phase-locked loop hardware, is a copy of the free-running A clock (F A CLK) received from the backplane with the rest of the system clocks.

In addition to its REFERENCE CLOCK output, the phase-locked loop circuit has seven INTERVAL SYNC outputs that connect to the system's interval timer located in the console interface. These signals, together with outputs from the clock period register, are used to maintain a constant interval timer frequency (1 MHz) even though the REFERENCE CLOCK (and, thus, the system clock) frequency is changed.

2.1.2 Phase Generator

The phase generator consists mainly of shift registers that divide the REFERENCE CLOCK frequency by six and generate the two nonoverlapping A and B clock phases, as well as the W phase. Latch circuits output four copies each of A PHASE and B PHASE plus a single copy of W PHASE. All copies are used as inputs to the clock distribution circuits. In addition, single copies of both A PHASE and B PHASE are produced that are used by the clock control logic.

2.1.3 Clock Control Logic

The clock control logic contains the clock control register, the clock period register, and the burst count register. It also contains a burst counter, the slow clock counter, and clock start/stop control logic. The burst counter is loaded from the burst count register and decrements to zero to signal the end of a clock burst. The slow clock counter produces the NMI signal (SLOW CLOCK ENAB) used to increment error timeout counters in the CPUs, memory controller, and I/O adapters. The start/stop logic generates the clock gating signals A GATE (turned on by B PHASE) and B GATE (turned on by A PHASE). These are the levels that enable the clock distribution circuits to transmit the main system clocks on the backplane.

2.1.4 Clock Distribution Circuits

The clock distribution circuits consist mainly of ECL drivers that transmit the system clocks to the CPUs, memory controller, I/O adapters, and the clock module itself. Several copies of each system clock are distributed (radially) with each copy consisting of a pair of differentially driven outputs that connect to a single destination module.

The free-running system clocks (F A CLK, F B CLK, and W CLK) are derived directly from the free-running clock phases (A PHASE, B PHASE, and W PHASE). The main system clocks are also derived from the free-running clock phases but are gated with A GATE and B GATE from the clock control logic to start, stop, and burst the clocks.

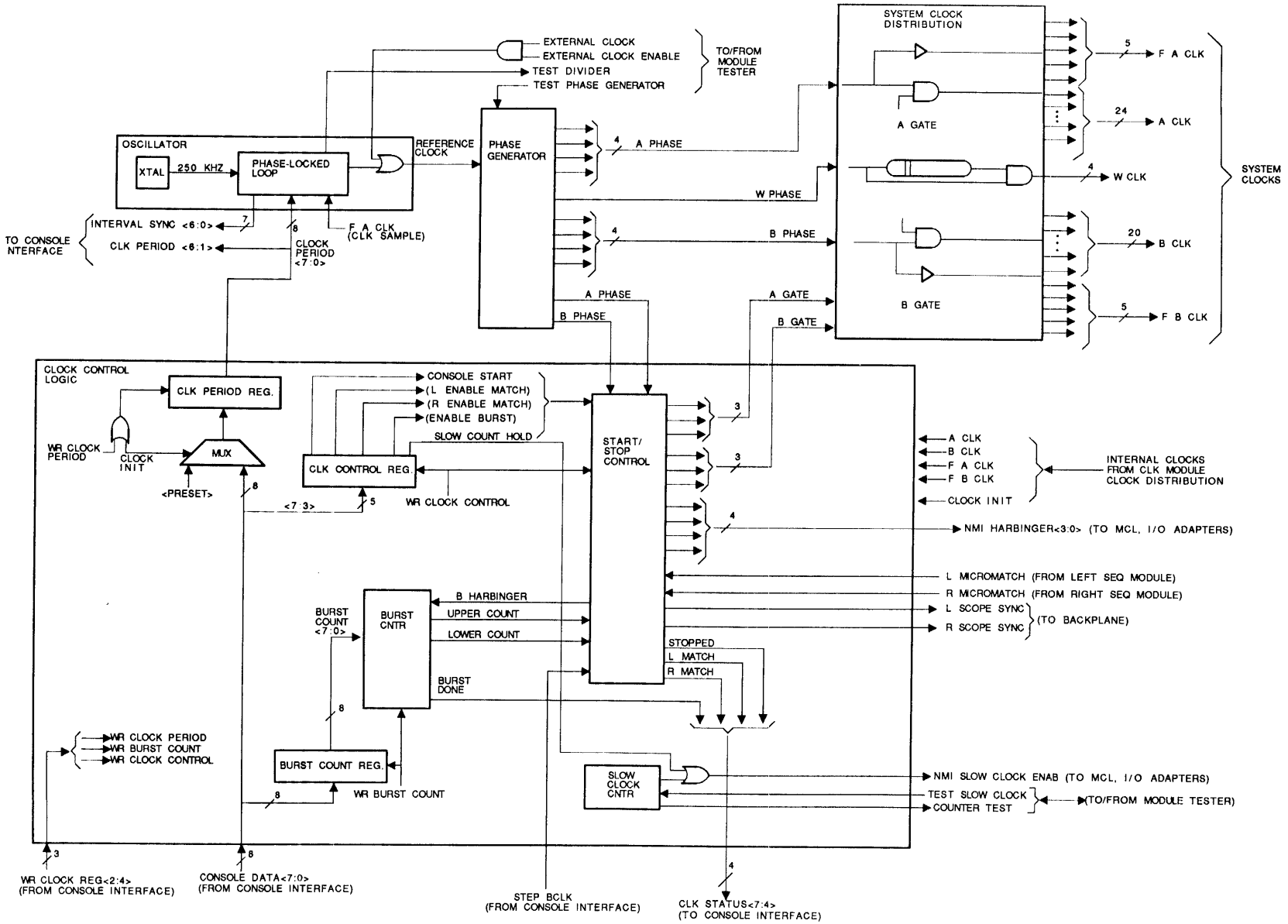


Figure 2-1 Clock Generator (Detailed Block Diagram)

Table 2-1 Clock Generator Inputs

Signal(s)	Number	Description
CONSOLE DATA <7:0>	8	Transfers register data from console interface when a clock register is written by the console.
WR CLOCK REG <2:0>	3	Loads addressed clock register when written by console. The three lines are outputs of address decoder in console interface. WR CLOCK REG — LOADS <2> — Clock period register <1> — Burst count register <0> — Clock control register
STEP BCLK	1	Causes one B CLK to be generated. Output of VBus control register in console interface.
CLOCK INIT	1	Initializes clock period register. Asserted by DC LO logic in console interface.
L MICROMATCH	1	Indicates micromatch detected in left CPU. Asserted by CPU's IBox.
R MICROMATCH	1	Indicates micromatch detected in right CPU. Asserted by CPU's IBox.
EXTERNAL CLOCK ENAB	1	Enables output from an external clock source to produce REFERENCE CLOCK and, thus, system clocks. Used by module test device during manufacturing test/repair procedures.

Table 2-1 Clock Generator Inputs (Cont)

Signal(s)	Number	Description
EXTERNAL CLOCK IN	1	External clock source from module test device. See EXTERNAL CLOCK ENAB.
TEST PHASE GENERATOR	1	Initializes phase generator. (An A CLK is generated after test signal is asserted and deasserted coincident with REFERENCE CLOCK, and five additional REFERENCE CLOCKS occur.) Used by module test device during manufacturing test/repair procedures.
TEST SLOW CLOCK	1	Initializes slow clock counter to zero. Used by module test device during manufacturing test/repair procedures.

Table 2-2 Clock Generator Outputs

Signal(s)	Number	Description
System Clocks		
A CLK	24	Gated A clock
B CLK	20	Gated B clock
F A CLK	5	Free-running A clock
F B CLK	5	Free-running B clock
W CLK	4	W clock (free-running)
INTERVAL SYNC <6:0>	7	Used to keep interval timer in console interface synchronized with system clocks.
CLK PERIOD <6:1>	6	Six high-order outputs of clock period register. Used with INTERVAL SYNC <6:0> to keep interval timer in console interface synchronized with system clocks.

Table 2-2 Clock Generator Outputs (Cont)

Signal(s)	Number	Description
CLK STATUS <7:4>	4	Clock generator status bits to clock/timeout status register (outputs of a data multiplexer) in console interface. CLK STATUS — STATUS BITS <7> CLOCK OFF <6> LEFT MICROMATCH <5> RIGHT MICROMATCH <4> BURST DONE
NMI HARBINGER <3:0>	4	Early-warning signal to memory controller that clocks are about to stop (when signal is asserted) or start (when signal is deasserted).
L SCOPE SYNC	1	Scope sync signal asserted on backplane when micromatch occurs in left CPU.
R SCOPE SYNC	1	Scope sync signal asserted on backplane when micromatch occurs in right CPU.
NMI SLOW CLOCK ENABLE	1	Slow clock used to increment timeout counters in CPU, memory controller, and I/O adapters.
TEST DIVIDER	1	Output of divider circuit in oscillator's phase-locked loop circuit. (Signal is a copy of DIVIDED ACLK.) Used by module test device during manufacturing test/repair procedures.
COUNTER TEST	1	Overflow (borrow output) from eight high-order stages of 16-bit slow clock counter. Used by module test device during manufacturing test/repair procedures.

2.2 CLOCK GENERATOR INITIALIZATION

At powerup, DC LO asserts CLOCK INIT, which loads a value of 56 into the clock period register. Also, at powerup, the oscillator starts to generate the system clock phases. The clock period, determined by the value preset in the clock period register, is 70 ns. The console lowers the clock period (increases clock frequency) to its normal value during the system initialization sequence that follows powerup.

With the system clock phases running, the clock generator is also producing the free-running clocks. This allows the console to access the clock (and console interface) registers during system initialization. (The console must start, stop, burst, and change the frequency of the clocks during the loading of microcode and the many other operations performed.) When system initialization ends, the clock generator is left in the following state:

- Clocks running (STOP CLOCK cleared in clock control register)
- Clock period set to 50 ns (N = 79 in clock period register)
- Micromatch stop disabled (-SOMM bits set in clock control register)
- Clock bursts disabled (-BURST ENABLE set in clock control register)
- Slow clock running (-SLOW CLOCK HOLD set in clock control register)

2.3 SYSTEM CLOCK PERIOD CONTROL

System clock frequency (and, thus, system clock period) is controlled by the oscillator's phase-locked loop circuit. The circuit is used as a frequency synthesizer to produce a wide range of frequencies determined by the value N in the clock period register. Refer to Figure 2-2.

2.3.1 Phase-Locked Loop Operation

The phase-locked loop circuit contains the following components.

- Phase detector
- Low pass filter
- Voltage controlled oscillator (VCO)
- Low range divider
- Divide by N 1 Counter

The phase detector compares the 250 kHz crystal-controlled low frequency source with a system clock sample that has been divided in frequency by the divide by N 1 counter. This 7-bit counter, loaded with the value N from the clock period register, is continuously decremented by the system clock sample (F A CLK). Whenever the counter reaches a 0 count (after N 1 counts), it reloads from the clock period register, starts decrementing again, and asserts the divided (by N 1) system clock connecting to the phase detector. This signal is called DIVIDED ACLK.

If a phase difference exists between the crystal-controlled frequency and DIVIDED ACLK, indicating that the two frequencies are not the same, the phase detector asserts one of two signals, UP or DOWN. The signal asserted depends on whether the divided system clock sample has a lower frequency (UP asserted) or a higher frequency (DOWN asserted) than the crystal-controlled frequency. The duration of the UP and DOWN outputs is proportional to the phase difference detected.

The low pass filter integrates the UP/DOWN signals to produce a dc voltage input to the VCO. The dc voltage determines the frequency of the VCO output, which is a high frequency square wave. This VCO output (or the VCO output divided by two) is the REFERENCE CLOCK that the phase generator divides by six to produce the system clock phases. (The VCO output is divided by two when the clock period register's LOW RANGE control bit is set as described in the following paragraph.)

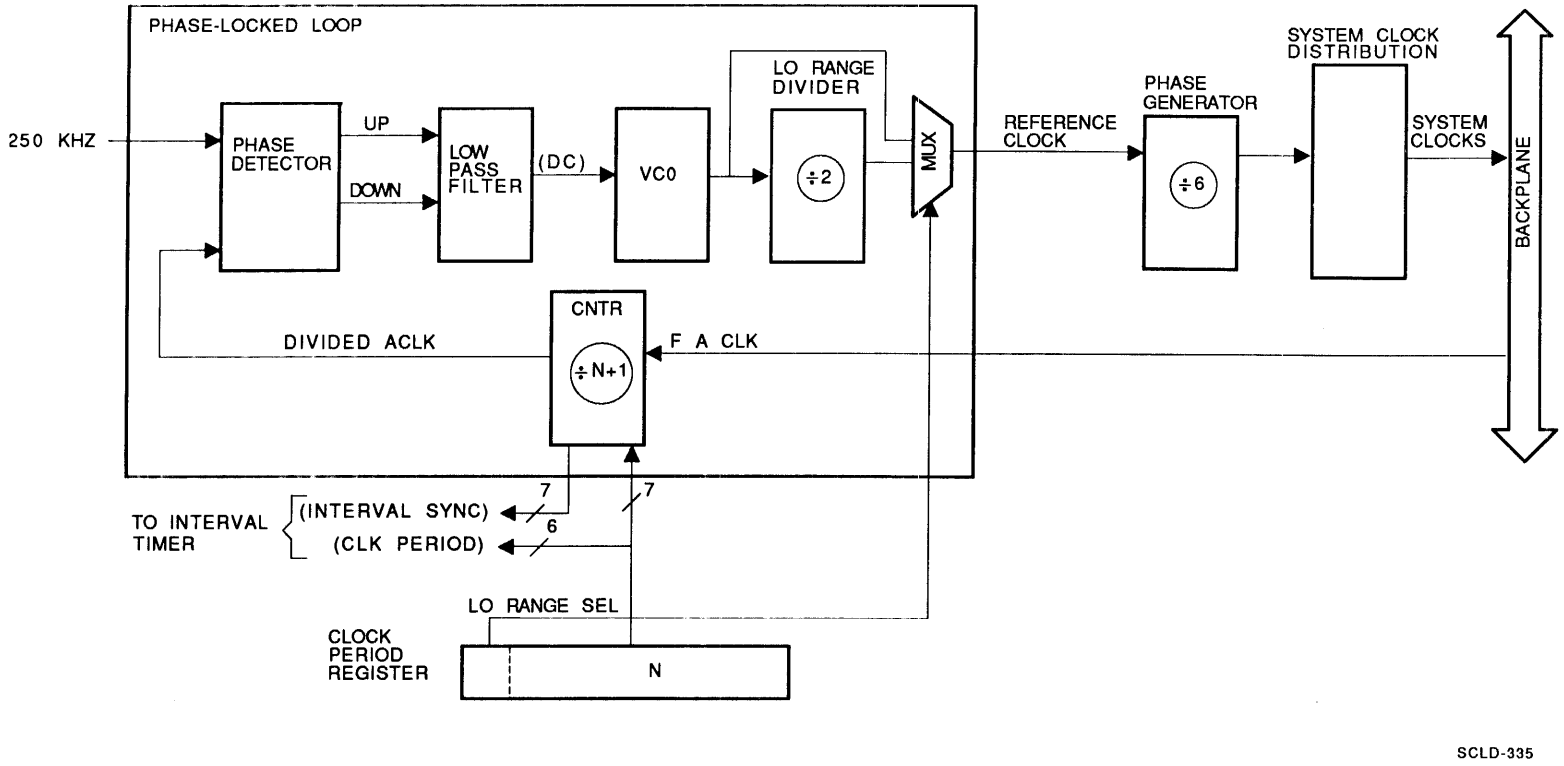


Figure 2-2 Simplified Clock Frequency Control Circuitry

At poweron, the clock period register is initialized to a value of 56 (by CLOCK INIT, which is asserted by DC LO). Also, the VCO begins generating the REFERENCE CLOCK; the phase detector starts generating UP or DOWN signals to bring the REFERENCE CLOCK frequency to a value where the divided system clock and the crystal-controlled frequency are the same. When this occurs, the phase-locked loop is locked and (for N = 56 and the LOW RANGE bit cleared) the REFERENCE CLOCK frequency is 85.5 MHz and the system clock frequency is 14.25 MHz (70 ns clock period). Further phase detector outputs are minimal, compensating only for slight drifts in the clock frequency. Of course, when the clock period register is loaded with a new value of N, the phase detector outputs (which are pulses) sharply increase in duration and then decrease over time as the clock frequency quickly approaches and locks to the new value.

The relationship between N, the low frequency source, and the system clock frequency is derived below.

At the input to the phase detector:

$$\text{DIVIDED A CLK FREQUENCY} = \text{LOW FREQUENCY SOURCE}$$

Substituting:

$$\frac{\text{SYSTEM CLOCK FREQUENCY} = 250\ \text{kHz}}{(N + 1)}$$

Thus:

$$\text{SYSTEM CLOCK FREQUENCY} = (N + 1) \times 250\ \text{kHz}$$

As can be seen, the low frequency source is not only the crystal-controlled reference frequency for the phase-locked loop; its value also determines the frequency step size. That is, the system clock can only be some multiple of 250 kHz and, of course, 250 kHz is the smallest increment or decrement that can be made in the system clock frequency by changing N.

In normal operation, the specified range of system clock frequencies that can be generated is 14.25 MHz to 25 MHz (N = 56 to 99). The limiting factor is mainly the operating range of the VCO. However, for diagnostic and debug purposes, the LOW RANGE control bit can be set in the clock period register causing the low range divider to divide the VCO frequency by two. This gives a lower range of system clock frequencies with the VCO still operating within its specified range. The low frequency range is 7 MHz to 12.5 MHz (N = 27 to 49). Note that the values of N are half of what they are in the normal operating range. That is, the system clock frequency is always 250 kHz X (N 1), regardless of how the VCO output is divided to produce the clocks.

2.3.2 Changing Clock Period

To change the clock period, the console simply loads a new value of N in the clock period register. The clocks do not need to be stopped first, and any sized increment or decrement may be made as long as the resulting clock frequency falls within the clock generator's specified operating range. The only restriction is that there should be no change from the low to normal, or normal to low, frequency range when a program is running. The changing of the LOW RANGE control bit in the clock period register can cause the low range divider to introduce a transient change in the system clock period. This clock period change can cause a sequencing CPU, memory, or other system component to malfunction.

2.4 SYSTEM CLOCK START/STOP/BURST CONTROL

Figure 2-3 shows the clock control logic that starts, stops, and bursts the main system clocks. (Timing is shown in Figure 2-4.) The major control element is a latch circuit that, when cleared, asserts the gating signals (A GATE and B GATE) that allow the clock distribution circuits to transmit the main system clocks on the backplane. This latch, when set, stops transmission of the clocks and asserts the CLOCK OFF bit in the clock/timeout status register that can be read by the console.

The first clock generated when the clocks are started is A CLK. The last clock generated when the clocks are stopped is B CLK. Before the last B CLK (and when the last A CLK occurs), the clock control logic asserts HARBINGER on the NMI. This is the signal used by the memory controller, which is sequenced by the free-running clocks, as a clock blocking signal. The controller can then simulate stopped clock operation even though the free-running clocks continue to be generated after the main clocks are stopped. HARBINGER, when it is deasserted, also serves as an early-warning signal that the clocks are about to start. When the clocks are restarted, HARBINGER is deasserted by a free-running clock (F A CLK) one system clock period before the first A CLK occurs.

Besides requiring the assertion of HARBINGER before the clocks stop, the memory controller must also be in a slow mode of operation as discussed in Chapter 1. That is, the console must assert SLOW MODE on the NMI by setting a control bit in a console interface register (control register 1). (Some I/O adapters such as the NBI must also be set to slow mode before the clocks stop.) The SLOW MODE signal must be asserted by the console at least four microseconds before the clocks stop.

2.4.1 Starting the Clocks

The console starts the clocks by clearing the STOP CLOCK bit in the clock control register. This generates a START pulse that clears the CLOCK OFF latch enabling the GATE A and GATE B signals to be asserted. The clocks will remain running until stopped unconditionally by the console, by a micromatch stop condition, or at the end of a burst. Once stopped, the only way to restart the clocks is for the console to again clear the STOP CLOCK bit.

2.4.2 Stopping the Clocks Unconditionally

The console can stop the clock at any time by just setting the STOP CLOCK bit in the clock control register. (SLOW MODE must be asserted first, however.) Setting STOP CLOCK asserts STOP, which sets the CLOCK OFF latch preventing the assertion of the GATE A and GATE B signals.

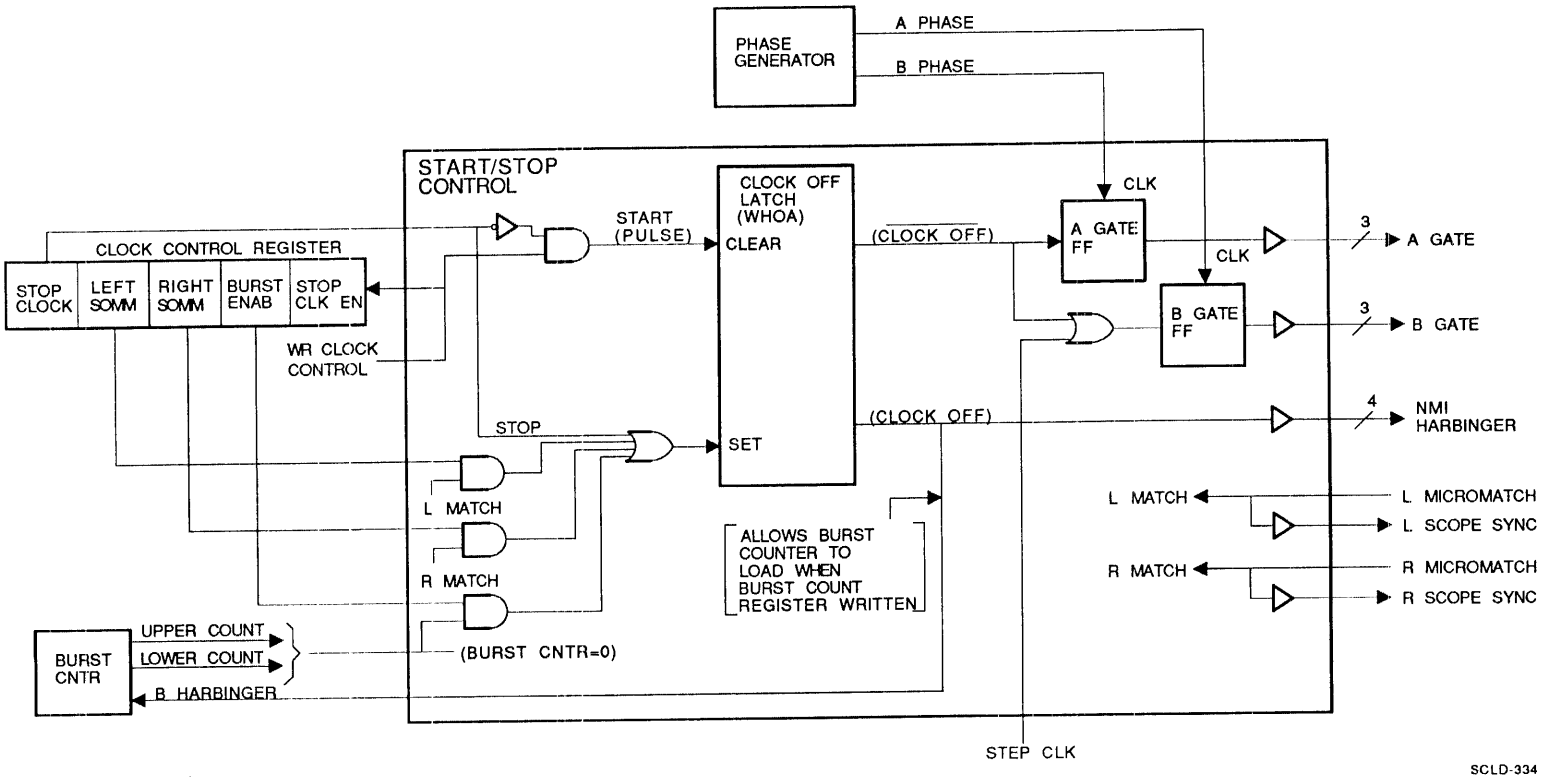


Figure 2-3 Simplified Clock Start/Stop/Burst Control Logic

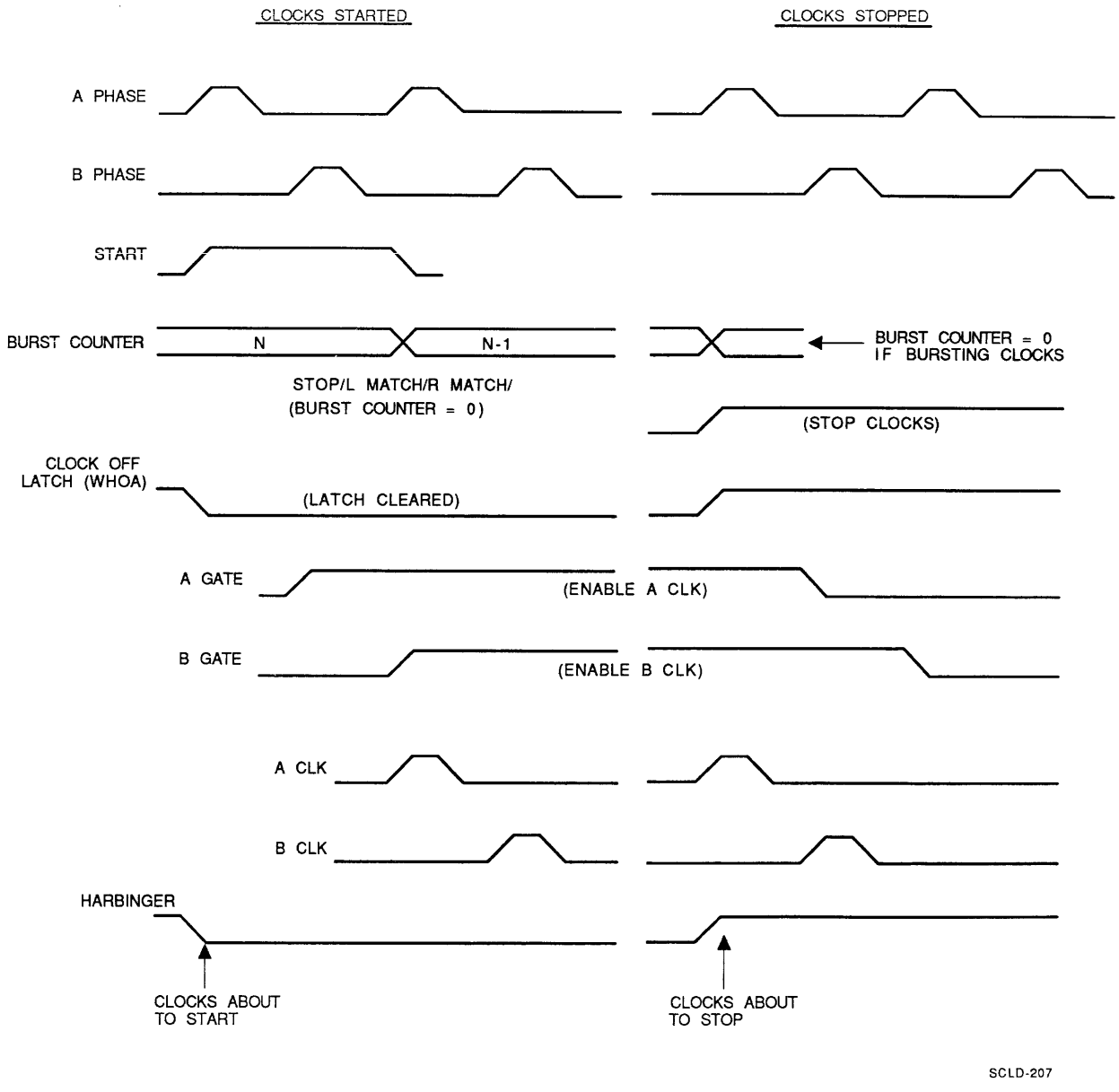


Figure 2-4 Start/Stop/Burst Control Timing Diagram

2.4.3 Stopping the Clocks on Micromatch/Scope Sync Generation

The console can enable the clock control logic to stop the clocks when the microPC in the left or right CPU is at a specified value. This condition, a micromatch, also causes the clock control logic to generate a scope sync. The scope sync is always generated by a micromatch even though the clocks are not enabled to stop.

To cause a micromatch (and scope sync) on a specific microPC value, the console must load that value in the CPU's micromatch register located in the IBox. If the console wants to stop the clocks when a micromatch occurs, it asserts SLOW MODE and clears the CPU's SOMM (stop on micromatch) bit in the clock control register. (A SOMM bit is asserted when equal to 0.) The loading of the microPC value, the assertion SLOW MODE, and the clearing of the SOMM bit are done with the clocks turned off.

Once the clock is turned on and if a micromatch occurs, the CPU's IBox asserts a MICROMATCH signal. This signal (one for each CPU) generates the scope sync (there is one for each CPU). The micromatch signal also sets the CLOCK OFF latch to stop the clocks, provided the corresponding SOMM bit has been asserted. Micromatch and scope sync timing is shown in Figure 2-5.

After the clocks stop on a micromatch, the MICROMATCH signal is normally deasserted. If it is still asserted when the clocks are restarted (indicating the microinstruction has branched to its own microaddress), the clock control logic generates one pair of system clocks, an A CLK and a B CLK, and stops the clock again.

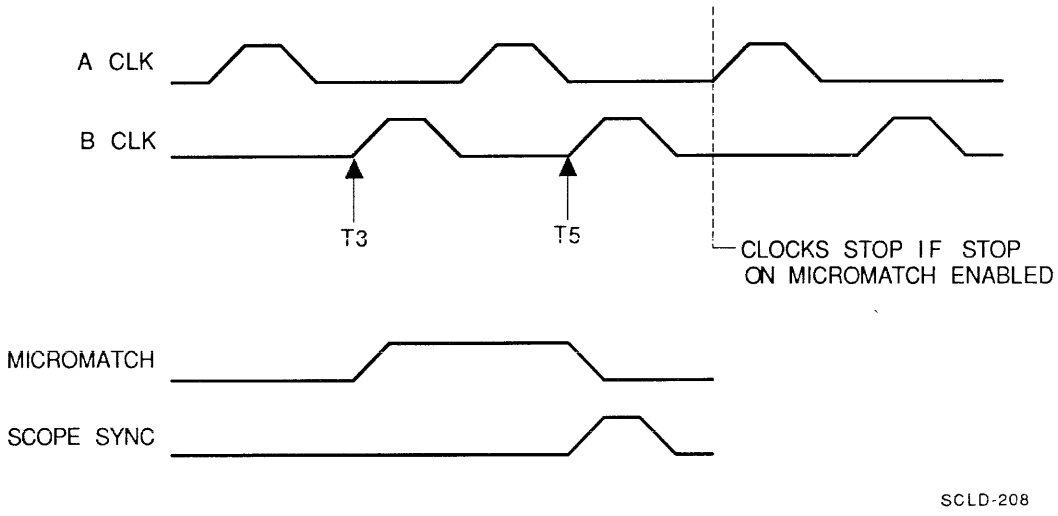


Figure 2-5 Micromatch and Scope Sync Timing Diagram

2.4.4 Bursting the Clocks

Bursting the clocks causes a specified number of clock cycles to be generated. (A clock cycle is one A CLK and one B CLK.) The clocks in the burst occur at the nominal clock frequency; that is, the frequency specified by the current contents of the clock period register.

To burst the clocks, with the clocks stopped, the console first asserts SLOW MODE (if not already asserted), loads a burst count in the burst count register, and clears the BURST ENABLE bit in the clock control register. (Like the SOMM bits, the BURST ENABLE bit is asserted when equal to 0.) The burst count, which can be any value from 1 to 255, is also loaded from the burst count register into the clock control's burst counter. (This is an automatic load performed by hardware whenever the burst count register is loaded.) The console starts the clock burst by simply starting the clocks as it does normally.

During a clock burst, the burst counter is decremented each clock cycle. When the count reaches zero (indicating that the specified number of clocks have been generated), and with the BURST ENABLE bit previously asserted by the console, the clock control's CLOCK OFF latch is set to stop the clocks and end the burst.

The console can tell when a clock burst has ended by reading the BURST DONE (burst counter = 0) status bit in the clock/timeout status register. It may then generate another burst by reloading the burst count register and restarting the clocks. If the clocks are stopped by the console or by a micromatch during a burst, the burst counter holds its current count and the interrupted burst completes when the clocks are restarted.

2.4.5 Single-Stepping the Clocks

The single-step operation is just a clock burst with the burst count equal to 1. A single clock cycle (one A CLK and one B CLK) is generated. The console loads the burst count and produces the clock burst as described in Section 2.4.4.

2.4.6 Single-Stepping the B CLK

During VBus operations, the console must single-step the B CLK to shift the VBus data address into the CPU modules. The single-stepping of the B CLK is not done by bursting the clock as when single-stepping a complete clock cycle (generating both an A CLK and a B CLK). Instead, the console sets a control bit in the VBus control register called STEP B CLK. The bit is immediately cleared by hardware so that it remains asserted for only one clock period. In the clock control logic, the STEP B CLK signal asserts the GATE B (but not the GATE A) clock enable signals causing one system B CLK to be generated. Timing is shown in Figure 2-6.

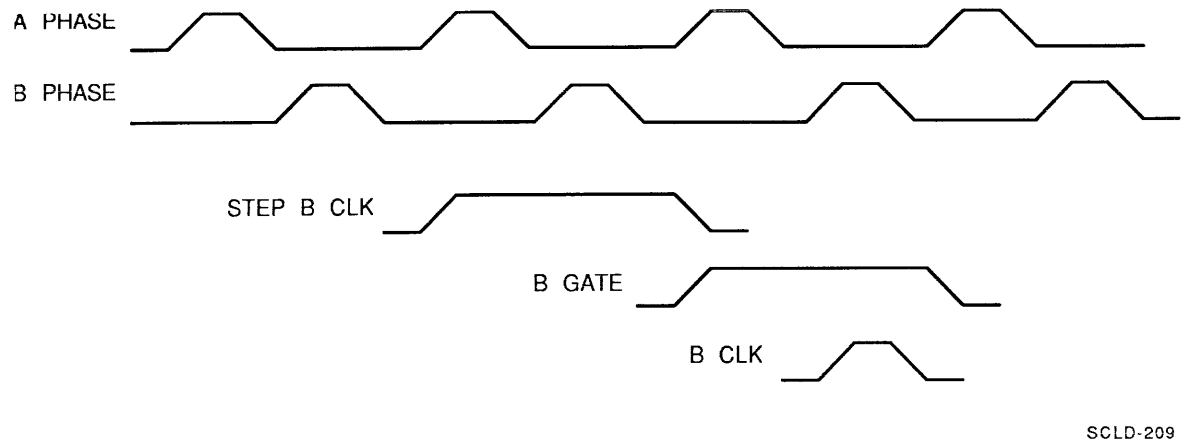


Figure 2-6 Single-Stepping B CLK Timing Diagram

2.5 SLOW CLOCK GENERATION AND CONTROL

SLOW CLOCK ENAB, an NMI signal, is asserted by the slow clock counter in the clock control logic. This 16-bit counter continuously decrements as long as the main system clocks are turned on (it is clocked by B CLK). The SLOW CLOCK ENAB signal is asserted just after the counter reaches its minimum value of 0 and when it begins a new count starting at its maximum value of 65,536 (64K). Timing is shown in Figure 2-7.

As can be seen, SLOW CLOCK ENAB is asserted for one B CLK period. NMI nexus use the signal to enable counters detecting timeout errors that are in the millisecond range (for example, no access to bus, no return read data). That is, the counters are clocked by the module's own system clocks but only when SLOW CLOCK ENAB has been asserted (once every 3.28 milliseconds at the normal system clock frequency). This clocking scheme allows the use of much smaller counters than would be necessary if only the system clocks were used to count through the comparatively long timeout periods.

Because of the way SLOW CLOCK ENAB is used, forcing the signal to its asserted state causes a timeout counter to count at the system clock rate and quickly set the associated error flag. The console can force the continuous assertion of the signal by clearing the SLOW CLOCK HOLD bit in the clock control register. (This control bit is asserted when equal to 0.) Diagnostics do this to simulate the timeout condition and check the error flag.

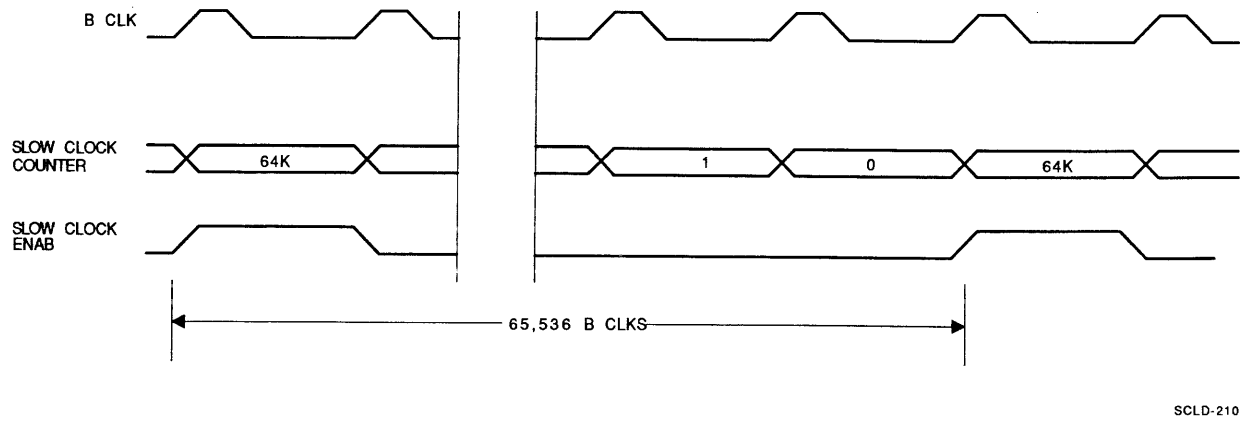


Figure 2-7 Slow Clock Timing Diagram

2.6 CLOCK CONSOLE COMMANDS

Three console commands control the main system clocks. The SET CLOCK command can stop and start the clocks and also change the clock period. The MICROSTEP command bursts the clocks. The SET SOMM command enables the clocks to stop (and a scope sync to be generated) on a micromatch. (A SET SYNC command enables only the scope sync to be generated on a micromatch, and a SET TOMM command enables a microtrap and a scope sync to be generated on a micromatch.) The three clock console commands affect the clocks in both CPUs.

Table 2-3 Clock Console Commands

Command	Description
SET CLOCK OFF	Turns off clocks. (Asserts SLOW MODE, sets CLOCK OFF in clock control register, deasserts SLOW MODE.)
SET CLOCK ON	Turns on clocks. (Asserts SLOW MODE, clears CLOCK OFF in clock control register, deasserts SLOW MODE.)
SET CLOCK NORMAL	Sets clock period to its normal value of 50 ns. (Loads N = 79 into clock period register.)
SET CLOCK SLOW	Sets clock period to its high margin value of 55 ns. (Loads N = 71 into clock period register.)
SET CLOCK FAST	Sets clock period to its low margin value of 45 ns. (Loads N = 87 into clock period register.)

Table 2-3 Clock Console Commands (Cont)

Command	Description
SET CLOCK <value>	Sets clock period. The value specifies clock period in nanoseconds. Restricted to a range of 140 to 40 ns. (Loads N = 27 to 99 into clock period register.)
MICROSTEP <stepcount>	Bursts the clocks. The stepcount specifies the number of clock cycles in the burst. Restricted to a range of 1 to 255 clock cycles. A burst of one clock cycle is generated if no stepcount is specified. (The clocks are single-stepped.) After the last clock cycle in a burst, pressing the space bar causes one more clock cycle to be generated. This space bar (single-step) mode can be exited by pressing carriage return (<cr>).
SET SOMM <microaddress>	Enables the clocks to stop and a scope sync to be generated when a micromatch occurs in the CPU; that is, when the microPC is equal to the specified microaddress. (Loads the microaddress in the CPU's micromatch register and asserts the SOMM bit in the clock control register.

Digital Equipment Corporation • Bedford, MA 01730