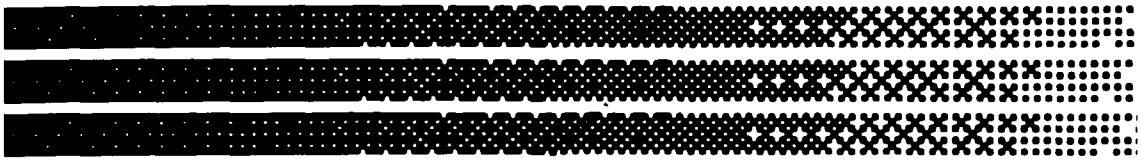
EK-KA630-UG-001



KA630-AA CPU Module User's Guide

Prepared by Educational Services
of Digital Equipment Corporation

1st Edition, February 1986

The material in this document is for informational purposes and is subject to change without notice; it 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.

FCC 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.

Printed in U.S.A.

The manuscript for this book was created on a VAX-11/780 system running WPS-PLUS. The book was produced by Educational Services Development and Publishing in Marlboro, MA.

Motorola is a registered trademark of Motorola, Inc.

The following are trademarks of Digital Equipment Corporation:

digital™	MASSBUS	RSTS
DEC	OMNIBUS	RSX
DECmate	OS/8	RT
DECNET	PDP	UNIBUS
DECUS	PDT	VAX
DECwriter	P/OS	VAXstation
DIBOL	Professional	VMS
IAS	Q-Bus	VT
LSI-11	Rainbow	Work Processor

Page

FIGURES

TABLES

This manual is intended for the design engineer or applications programmer who is familiar with Digital's extended LSI-11 bus (Q22-Bus) and the VAX instruction set. This manual is divided into the following chapters:

1 **OVERVIEW** -- Introduces the KA630-AA MicroVAX CPU module and MS630 memory modules, including module features and specifications.

2 **INSTALLATION** -- Describes the installation of the KA630-AA and MS630 modules in Q22-Bus backplanes and system enclosures.

3 **BOOTING AND CONSOLE PROGRAM INTERFACE** -- Describes the console program, device booting sequence and console commands.

4 **ARCHITECTURE** -- Provides a description of KA630-AA registers, instruction set and memory.

5 **DIAGNOSTICS** -- Describes the KA630-AA boot diagnostics.

CONVENTIONS

The following chart lists the conventions used in this manual.

Convention	Meaning
<mm:nn>	Read as "mm through nn," it indicates a bit field or a set of lines or signals. For example, A <17:00> is the mnemonic device that stands for address lines 17 through 00.
<CR>	A label enclosed by angle brackets represents a key (usually a control or special character key) on the keyboard (in this case, the carriage return).
NOTE	Contains general information.
CAUTION	Contains information to prevent damage to equipment.
XX	Boldface capital Xs indicate variables.

RELATED DOCUMENTS

The following is a list of related documentation.

Microcomputer Interfaces Handbook	EB-20175-20
Microcomputers and Memories Handbook	EB-18451-20
VAX Architecture Handbook	EB-19580-20
VAX-11 Architecture Reference Manual	EK-VAXAR-RM

You can order these documents from:

Digital Equipment Corporation
Accessories and Supplies Group
P. O. Box CS2008
Nashua, NH 03061

Attention: Documentation Products

1.1 INTRODUCTION

The KA630-AA (Figure 1-1) is a quad-height VAX processor module for the Q22-Bus (extended LSI-11 bus). It is designed for use in high speed, real-time applications and for multiuser, multitasking environments. It can be configured as an arbiter or auxiliary CPU. The major components of the KA630-AA are described in the following paragraphs.

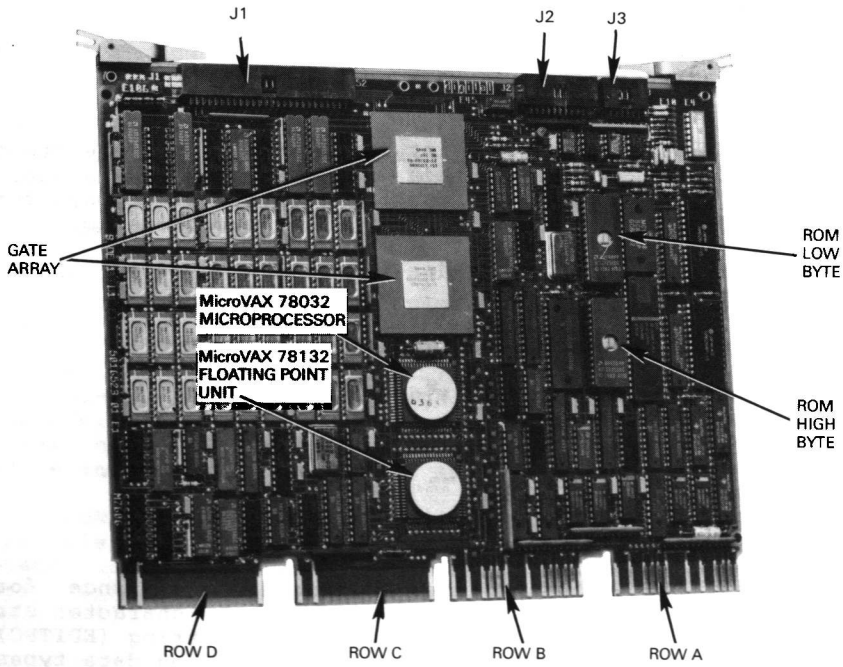


Figure 1-1 KA630-AA (MicroVAX 630) CPU Module

1.2 MicroVAX 78032 MICROPROCESSOR CHIP

The MicroVAX 78032 (referred to as the MicroVAX CPU chip in this manual) is a 32-bit virtual memory microprocessor packaged in a 68-pin ZMOS (double metal NMOS) chip. It requires no special clock generator or support chips. At its maximum frequency, the MicroVAX CPU chip achieves a 200 ns microcycle and a 400 ns I/O (memory) cycle. The MicroVAX CPU chip contains a 32-bit extension of the industry standard microprocessor interface.

The MicroVAX CPU chip includes a VAX compatible, demand-paged Memory Management Unit (MMU). This MMU provides direct access to four gigabytes (2**32) of virtual memory and one gigabyte (2**30) of physical memory. Virtual mapping of system space addresses is accomplished through single level page tables. Virtual mapping of process space addresses is accomplished through double level page tables.

The MicroVAX CPU chip provides the following subset of the VAX data types.

- Byte
- Word
- Longword
- Quadword
- Character string
- Variable length bit field

The MicroVAX 78132 Floating Point Unit (FPU) chip (referred to as the MicroVAX FPU chip in this manual) provides support for F_floating, D_floating and G_floating data types. Support for the remaining VAX data types can be provided by macrocode emulation.

The MicroVAX CPU chip provides the following subset of the VAX instruction set.

- Integer arithmetic and logical
- Address
- Variable length bit field
- Control
- Procedure call
- Miscellaneous
- Queue
- Character string moves (MOVC3 and MOVC5)
- Operating system support

The MicroVAX CPU chip provides microcode assistance for the emulation of the remaining VAX instructions: character string, decimal string, EDIT Packed to Character string (EDITPC) and Cyclic Redundancy Check (CRC). Support for floating data types and instructions is provided by the MicroVAX FPU chip (Section 1.3).

1.3 MicroVAX 78132 FPU CHIP

The MicroVAX FPU chip supports D_floating, F_floating and G_floating data types and instructions. It does not support H_floating data types or instructions. H_floating data types can be provided by macrocode emulation.

1.4 MicroVAX INTERFACE GATE ARRAY

The MicroVAX interface gate array consists of two custom Large Scale Integration (LSI) chips. The gate array includes the following features.

- Provides interface between the MicroVAX CPU and FPU chips and module logic
- Provides signals to the KA630-AA LEDs indicating console and diagnostic boot state
- Decodes signals from the KA630-AA connector J2 to determine module characteristics
- Contains a local decoder and address latch, for use by the memory subsystem and other KA630-AA logic elements

1.5 LOCAL MEMORY

The KA630-AA CPU contains 1 Mbyte of on-board local memory, and supports one or two MS630 memory expansion modules (Figure 1-2) for a maximum of 9 Mbytes of local memory. The KA630-AA communicates with MS630 memory modules through the CD interconnect of a system backplane and through a 50-conductor cable included with each memory module. MS630 memory modules are available in three variants (Table 1-1), all populated with 256 K RAM chips.

The KA630-AA provides byte parity generation and checking for all local memory. The memory mapping procedure is described in Chapter 4.

1.6 64 KBYTE BOOT AND DIAGNOSTIC ROM

The KA630-AA boot and diagnostic ROM provides power-up diagnostics, boot programs for standard devices, and a subset of the VAX console program. The power-up diagnostics, booting procedure and console program are described in Chapter 3.

1.7 CONSOLE SERIAL LINE UNIT (SLU)

The console SLU, described in Chapter 2, is accessed by the processor using four VAX Internal Processor Registers (IPRs), and features externally selectable baud rates. The IPRs are described in Chapter 4.

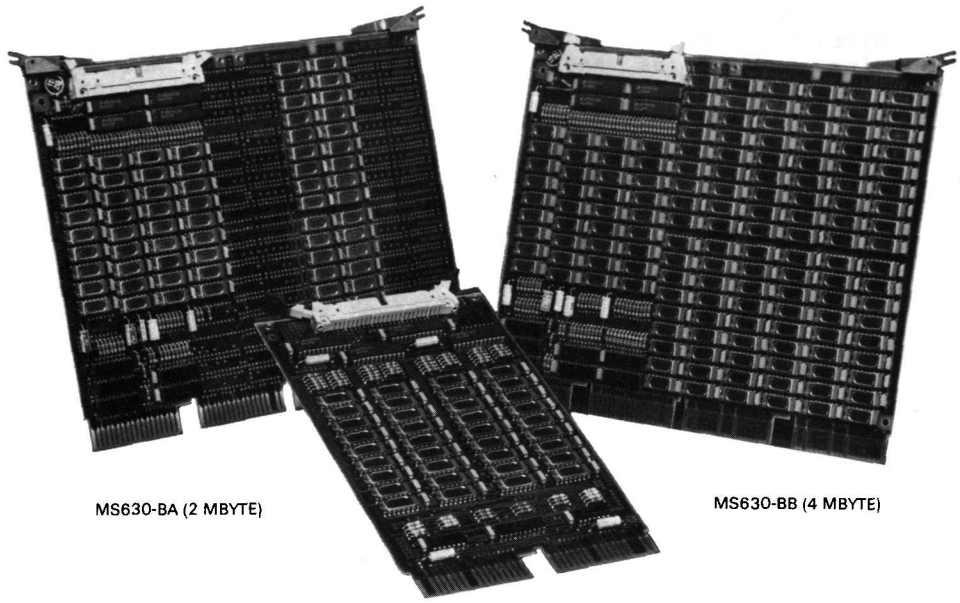


Figure 1-2 MS630 Memory Modules

Table 1-1 MS630 Memory Module Variants

Variant	Storage (Mbytes)	Module Height	Module Number	Current at +5 Vdc (Max)
MS630-AA	1	Dual	M7607-AA	1.0 A
MS630-BA	2	Quad	M7608-AA	1.3 A
MS630-BB	4	Quad	M7608-BA	1.8 A

1.8 Q22-BUS INTERFACE

The Q22-Bus interface provides the following features.

- Block mode and single transfer Direct Memory Access (DMA)
- Q22-Bus I/O map, which allows DMA devices to access local memory through a 4 Mbyte window divided into 8192 independent pages
- Q22-Bus interrupt requests BIRQ7 through 4 (when configured as an arbiter CPU)
- 240 Ω termination

Figure 1-3 is a block diagram of the KA630-AA. Figure 1-4 is a system level block diagram.

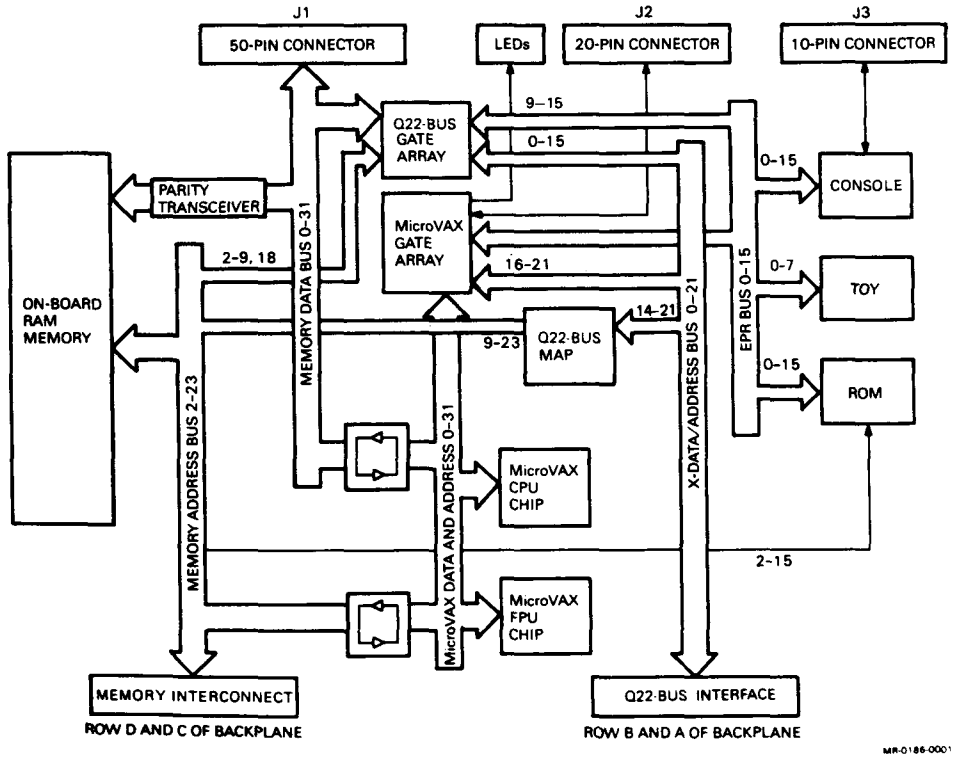


Figure 1-3 KA630-AA Block Diagram

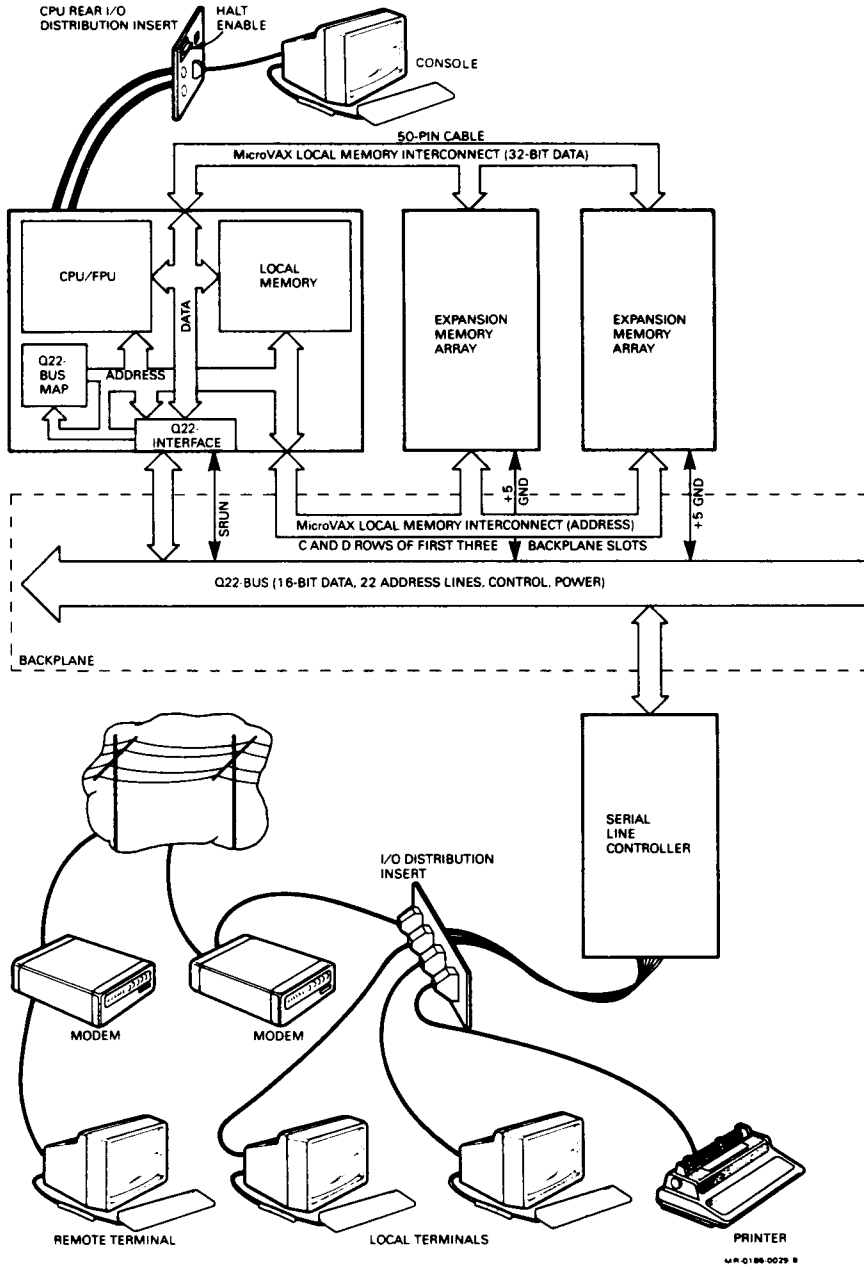


Figure 1-4 MicroVAX II System Level Block Diagram

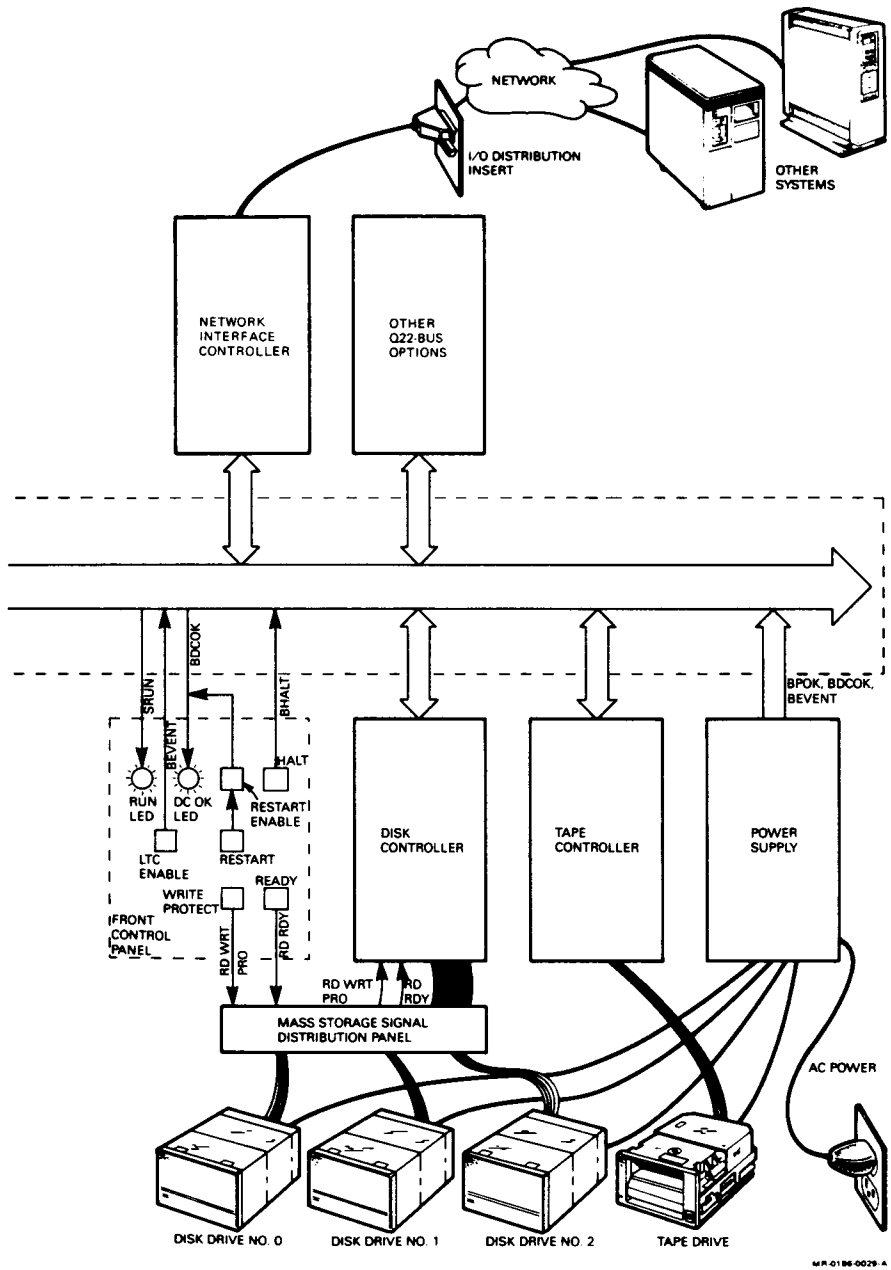


Figure 1-4 (Cont)

1.9 KA630-AA OPERATION MODES

When configured as an arbiter CPU, the KA630-AA must be installed in the first slot of a Q22-Bus backplane containing the CD interconnect. It arbitrates bus mastership and fields Q22-Bus interrupt requests BIRQ7 through 4. It also responds to interrupt requests from its own interval timer, console SLU and interprocessor doorbell. The interprocessor doorbell provides a means for auxiliary CPUs to request control of the Q22-Bus.

When configured as an auxiliary CPU, the KA630-AA can be installed in any backplane slot containing the CD interconnect. The arbiter may be a Q22-Bus PDP-11 CPU or another KA630-AA. The auxiliary KA630-AA requests bus mastership to access the Q22-Bus. It does not field Q22-Bus interrupt requests, but can respond to interrupt requests from its own interval timer, console SLU and interprocessor doorbell.

1.10 KA630-AA SPECIFICATIONS

The KA630-AA CPU module electrical and environmental specifications are listed in Tables 1-2 and 1-3, respectively.

Table 1-2 Electrical Specifications

Module Height	Maximum Currents +5 Vdc	Maximum Currents +12 Vdc	Q22-Bus Loads AC	Q22-Bus Loads DC
Quad	6.2 A	0.14 A	2.7	1.0

Table 1-3 Environmental Specifications

Specification	Range
Ambient storage temperature	-40 to +65°C (-40 to +149°F)
Operating temperature (CPU mounted in an enclosure):	
150 ft/min air flow	5 to 40°C (41 to 104°F)
250 ft/min air flow	5 to 50°C (41 to 122°F)
Relative humidity:	
Storage	10 to 90% noncondensing, altitude to 9.1 km (50,000 ft). Derate maximum temperature by 1°C for each 1000 m (1 ft for each 1000 ft) of altitude.
Operating	10% to 90% noncondensing

2.1 INTRODUCTION

This chapter contains information required to install the KA630-AA in a system. It describes the following.

- KA630-AA connectors
- Configuration board
- CPU distribution panel
- Compatible system enclosures

2.2 KA630-AA CONNECTORS

The KA630-AA communicates with local memory, the console device, and the Q22-Bus through three J connectors and through its four module fingers. The user can configure the KA630-AA through a CPU distribution panel insert or a configuration board. The slot pinouts on the fingers of the KA630-AA are listed in Appendix A.

The KA630-AA has three connectors (Figure 2-1), J1 through J3.

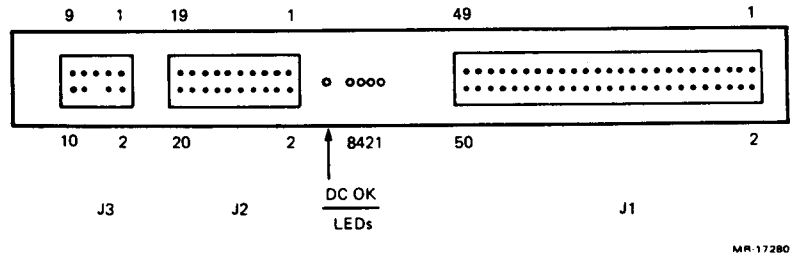


Figure 2-1 KA630-AA Pin and LED Orientation

2.2.1 Memory Expansion Connector (J1)

The 50-pin memory expansion connector provides the interface between the KA630-AA and MS630 memory modules installed in the CD rows of slots 2 and 3 of a Q22-Bus backplane containing the CD interconnect. Table 2-1 lists J1 pinouts. The memory expansion connector contains the following control, data and ground signals.

- BUFENL <01:00> (2 pins)
- BDIRTL (1 pin)
- PE <03:00> (4 pins)
- MD <31:00> (memory data lines -- 32 pins)
- GND (ground -- 11 pins)

2.2.2 Configuration and Display Connector (J2)

The KA630-AA has no jumper or switch settings to change or set. Module configuration is done using switches on the CPU distribution panel insert or the KA630CNF configuration board. The 20-pin configuration and display connector is connected to the inside of the CPU distribution panel insert by a 20-conductor cable, or directly to connector J2 of the KA630CNF configuration board. Table 2-2 lists J2 pinouts.

Table 2-1 Memory Expansion Connector (J1) Pinouts

Pin	Mnemonic	Pin	Mnemonic
01	GND	26	GND
02	MD0	27	PEL3
03	MD1	28	BUFENL 0
04	MD2	29	MD15
05	MD3	30	GND
06	GND	31	GND
07	MD5	32	PEL2
08	MD4	33	MD17
09	MD7	34	MD18
10	MD6	35	MD19
11	MD9	36	MD20
12	MD8	37	MD21
13	GND	38	GND
14	MD10	39	MD23
15	MD11	40	MD22
16	MD12	41	MD25
17	MD13	42	MD24
18	MD14	43	MD27
19	PEL0	44	MD26
20	GND	45	GND
21	BDIRTL	46	MD28
22	MD16	47	MD29
23	BUFENL 1	48	MD30
24	PEL1	49	MD31
25	GND	50	GND

Table 2-2 Configuration and Display Connector (J2) Pinouts*

Pin	Mnemonic	Meaning
01	GND	Ground.
02	GND	Ground.
03	GND	Ground.
04 05	CPU CD0 L CPU CD1 L	CPU Code <01:00>. This 2-bit code can be configured only by using switches 7 and 8 on the KA630CNF configuration board. (See Table 2-4.) It determines whether the KA630-AA is configured as the arbiter or as one of three auxiliaries. CPU Code <01:00> Configuration 00 Arbiter 01 Auxiliary 1 10 Auxiliary 2 11 Auxiliary 3 CPU Code <01:00> is read by software from the BDR. If the CPU distribution panel insert is used, no connections are made to pins 4 and 5. In that case, signal levels are negated by pull-up resistors on the KA630-AA, making it the arbiter CPU.
06	GND	Ground.
07 08 09 11	DSPL 00 L DSPL 01 L DSPL 02 L DSPL 03 L	Display Register Bits <03:00>. When asserted each of these four output signals lights a corresponding LED on the module. DSPL <03:00> are asserted (low) by power-up and by the negation of DC OK. They are updated by boot and diagnostic programs from the BDR.
10	BTRY VCC	Battery backup voltage for TOY clock.
12	GND	Ground.
13 14 14	BDG CD0 L BDG CD1 L BDG CD1 L	Boot and Diagnostic Code <01:00>. This 2-bit code indicates power-up mode, and is read by software from the BDR.

Table 2-2 Configuration and Display Connector (J2) Pinouts* (Cont)

Pin	Mnemonic	Meaning
15	HLT ENB L	Halt Enable. This input signal controls the response to an external halt condition. If HLT ENB is asserted (low), then the KA630-AA halts and enters the console program if any of the following occur. • The program executes a halt instruction in kernel mode. • The console detects a break character. • The KA630-AA is configured as an arbiter CPU and the Q22-Bus halt line is asserted. • The KA630-AA is configured as an auxiliary CPU and the interprocessor communication register AUX HLT bit is set. If HLT ENB is negated (high), then the halt line and break character are ignored and the ROM program responds to a halt instruction by restarting or rebooting the system. If HLT ENB is negated and the KA630-AA is configured as an auxiliary CPU, then the ROM program responds to assertion of the ICR AUX HLT bit by rebooting. HLT ENB is read by software from the BDR.
16	GND	Ground.
17 18 19	CSBR 02 L CSBR 01 L CSBR 00 L	Console Baud Rate <02:00>. These three bits are configured by using either the baud rate select switch on the CK-KA630-A distribution panel, or switches 2, 3 and 4 of the KA630CNF configuration board.
20	+5 V	Fused +5 volts.

* The KA630-AA module has 10 K pull-up resistors for the 8 input signals (pins 4 through 5, 13 through 15 and 17 through 19).

2.2.3 Console SLU Connector (J3)

The 10-pin console SLU connector provides the connection between the KA630-AA and the console terminal. It is connected to the inside of the CPU distribution panel by a 10-conductor cable, or directly to connector J3 of the KA630CNF configuration board. A cable from the outside of the distribution panel or J1 of the KA630CNF provides the external connection to the console terminal. Table 2-3 lists J3 pinouts.

Table 2-3 Console SLU Connector (J3) Pinouts

Pin	Mnemonic	Meaning
01		EIA signal out.
02	GND	Ground.
03	SLU OUT L	Console SLU output from the KA630-AA.
04	GND	Ground.
05	GND	Ground.
06		Key (no pin).
07 08	SLU IN + SLU IN -	Console SLU differential inputs to the KA630-AA.
09	GND	Ground.
10	+12 V	Fused +12 volts.

2.3 KA630CNF CONFIGURATION BOARD

A KA630CNF (H3263-00) configuration board (Figures 2-2, 2-3, 2-4) is provided with each KA630-AA. The KA630CNF plugs directly into connectors J2 and J3 on the KA630-AA. It allows the user to configure the KA630-AA by setting the 10 switches on SW1 as listed in Table 2-4.

Connector J1 is used to connect a cable to the console SLU. Connector J4 is for a Battery Backup Unit (BBU). The J4 pin closest to connector J1 is the positive pin.

Table 2-5 lists the pins on the KA630-AA J2 and J3, and the corresponding KA630CNF connectors and switches on SW1. Note that connectors J2 and J3 both have more connectors than there are pins on the corresponding KA630-AA connector. The two left and two right side connectors on J2 and J3 of the KA630CNF are unused. Switches 1 through 8 on SW1 set values that enable or disable halts; and determine CPU operation mode, power-up mode, and console baud rate. SW1 switches 9 and 10 connect transmit and receive lines as required for normal operation or loopback testing.

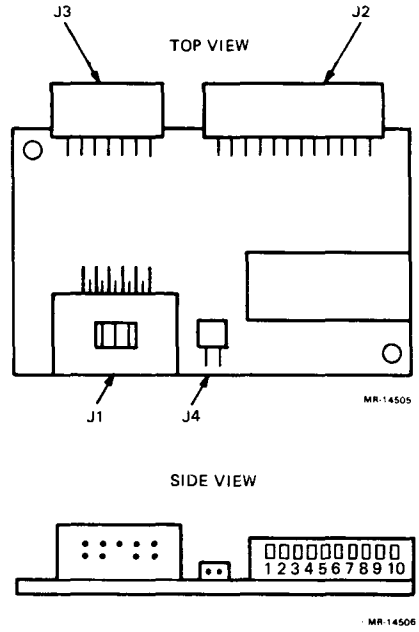


Figure 2-2 KA630CNF Configuration Board

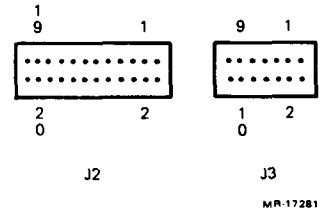


Figure 2-3 KA630CNF J2 and J3 Pin Orientation

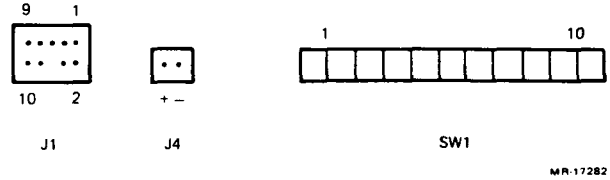


Figure 2-4 KA630CNF J1 and J4 Pin Orientation

Table 2-4 KA630CNF Switch Selections

Switch/Setting				Mode/Function
1				**Halt Mode**
Off				Disabled
On				Enabled
2	**3**	**4**		**Console Baud Rate**
Off	Off	Off		300
On	Off	Off		600
Off	On	Off		1,200
On	On	Off		2,400
Off	Off	On		4,800
On	Off	On		9,600
Off	On	On		19,200
On	On	On		38,400
5	**6**	**9**	**10**	**Power-Up Mode**
Off	Off	On	Off	Normal operation. Transmit line connected. Receive line connected.
On	Off	On	Off	Language inquiry mode. Transmit line connected. Receive line connected.
Off	On	Off	On	Loopback test mode (maintenance). Transmit line connected to receive line and console.
On	On	On	Off	Manufacturing use only. Bypasses memory test.
Note: Other settings for switches 5, 6, 9 and 10 should not be used.				
7	**8**			**CPU Operation Mode**
Off	Off			Arbiter
On	Off			Auxiliary 1
Off	On			Auxiliary 2
On	On			Auxiliary 3

Table 2-5 KA630CNF Connector and Switches

CPU J2 Pin	Mnemonic	CNF J2 Connector	CNF SW1 Switch	CNF J4 Pin	CPU J3 Pin	Mnemonic	CNF J3 Connector	CNF SW1 Switch	CNF J1 Pin
		1					1		
		2					2		
1	GND	3			1	EIA OUT	3		
2	GND	4			2	GND	4		2, 4, 5, 9
3	GND	5			3	SLU OUT L	5	10	3
4	CPU CD0 L	6	7		4	GND	6		2, 4, 5, 9
5	CPU CD1 L	7	8		5	GND	7		2, 4, 5, 9
6	GND	8			6	Key (no pin)	8		
7	DSPL 00 L	9			7	SLU IN +	9		7
8	DSPL 01 L	10			8	SLU IN -	10	9	
9	DSPL 02 L	11			9	GND	11		2, 4, 5, 9
10	BTRY VCC	12		1*	10	+12 V	12		10
11	DSPL 03 L	13					13		
12	GND	14					14		
13	BDG CD0 L	15	5						
14	BDG CD1 L	16	6						
15	HLT ENB L	17	1						
16	GND	18							
17	CSBR 02 L	19	2						
18	CSBR 01 L	20	3						
19	CSBR 00 L	21	4						
20	+5 V	22							
		23							
		24							

* +10 V from BBU to TOY clock chip on CPU

2.4 CK-KA630-A CPU DISTRIBUTION PANEL INSERT

When the KA630-AA is installed in a MicroVAX II system, the CK-KA630-A CPU distribution panel insert (Figures 2-5, 2-6, 2-7) in the rear I/O distribution panel is used to select configuration settings. The KA630-AA can only function as an arbiter when it is connected to the CK-KA630-A.

The CK-KA630-A is available in two variants: the CK-KA630-AB and CK-KA630-AF. The difference is in the cable length for the J2 and J3 cables. The CK-KA630-AB is used in Digital BA23-A and BA123-A enclosures. The CK-KA630-AF is used in the H9642 and BA11-S enclosures.

2.4.1 Time-Of-Year (TOY) Clock BBU

The CK-KA630-A also contains a BBU for the TOY clock chip. The BBU is located on the back of the CK-KA630-A. It consists of three nickel-cadmium batteries connected in series for a combined voltage of 3.75 Vdc. The minimum required voltage is 3.6 Vdc. The BBU provides power for the TOY clock chip when power is not supplied to the KA630-AA from the system power supply. The BBU recharges when dc power is applied to the KA630-AA. In addition to the time-of-year data, the TOY clock contains four Control and Status Registers (CSRs) and 50 bytes of RAM used by the console program (described in Chapter 3) to store information required to restart the processor following a halt. The TOY clock chip and four CSRs are described in detail in Chapter 4.

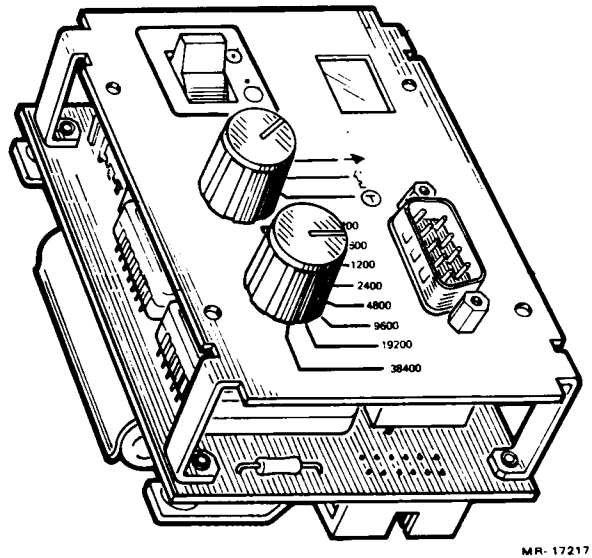


Figure 2-5 CK-KA630-A Distribution Panel Insert

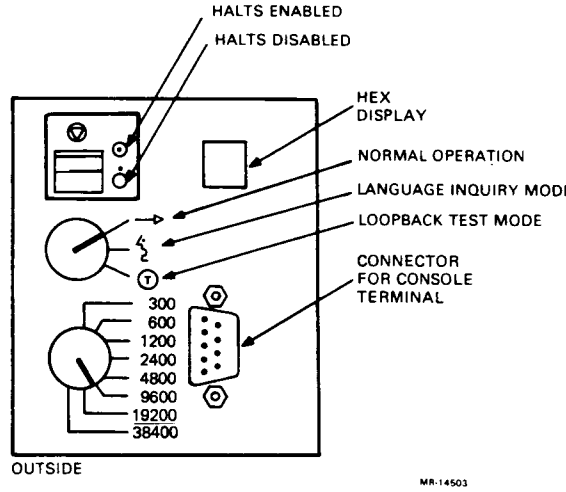


Figure 2-6 CK-KA630-A Connectors (Front View)

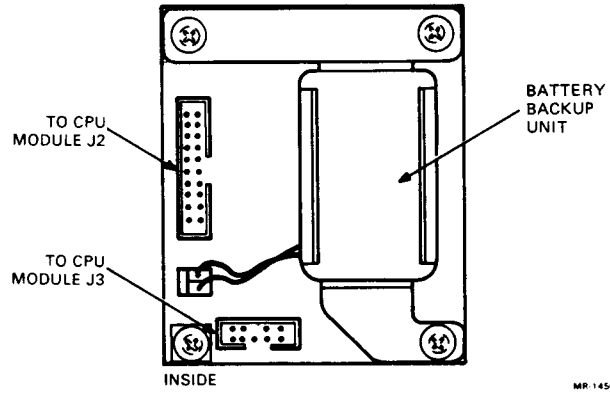


Figure 2-7 CK-KA630-A Connectors (Rear View)

2.5 COMPATIBLE SYSTEM ENCLOSURES

The KA630-AA is compatible with the following Digital enclosures.

BA11-S The BA11-S contains a 4 row X 9 slot backplane with 22-bit addressing on slots A/B. The C/D rows contain the CD interconnect. The backplane can contain up to nine dual-height or nine quad-height modules. Dimensions are 13.2 X 48.3 X 57.8 cm (5.2 X 19 X 22.7 in). The power supply includes a master console and provides 36 A at +5 V and 5 A at +12 V.

BA23-A The BA23-A contains a 4 row X 8 slot 22-bit address backplane. Slots 1 through 3 provide 22-bit addressing on the A/B rows and the CD interconnect on the C/D rows. Slots 4 through 8 provide 22-bit addressing on both the A/B and C/D rows. Up to 8 quad-height, or 3 quad-height and 10 dual-height modules can be mounted. The BA23-A has mounting space for 2 13.2 cm (5.25 in) mass storage devices. The power supply includes a master console and provides 36 A at +5 V and 7 A at +12 V.

The BA23-A is also available in an H9642 cabinet, which provides 8 additional backplane slots and space for 2 26.5 cm (10.5 in) mass storage devices.

BA123-A The BA123-A contains a 4 row X 12 slot 22-bit address backplane. Slots 1 through 4 provide 22-bit addressing on the A/B rows and the CD interconnect on the C/D rows. Slots 5 through 12 provide 22-bit addressing on both the A/B and C/D rows. The BA123-A has mounting space for 5 13.2 cm (5.25 in) mass storage devices. The power supply includes a master console and 2 regulators that provide 36 A at +5 V and 7 A at +12 V per regulator. Total power from each regulator must not exceed 230 W.

CHAPTER 3
BOOTING AND CONSOLE PROGRAM INTERFACE

3.1 INTRODUCTION

This chapter describes the KA630-AA console program and booting sequence. The console program, in conjunction with the KA630-AA hardware, gains control whenever the KA630-AA halts. For the KA630-AA, halting means only that control is transferred to this program, not that the processor stops executing instructions.

The console program is located in ROM on the KA630-AA. The ROM address range is located in the KA630-AA local I/O space. The console program uses the KA630-AA LEDs and console terminal output to communicate diagnostic progress and error reports to the user. In order for the console program to operate, the processor must be functioning at a level able to execute instructions from the console program ROM.

The console program provides the following services.

- Automatic restart or bootstrap following processor halts or initial power-up
- Interactive command language allowing the user to examine and alter the state of the processor
- Diagnostic tests executed on power-up that perform checks on the CPU, memory system and Q22-Bus I/O map
- Support of a video or hard-copy terminal as the console terminal

Users are not assumed to speak English. The console program can output console messages in 11 languages. If there is no language specified when the system powers up, the console program prompts the user for a language. The user language is then recorded (CPMBX <07:04>) in battery backed up RAM on the TOY clock chip. The preferred language is thus retained when the system is turned Off.

The KA630-AA decodes the ROM addresses so that the same ROM appears more than once in the address space. The console program is written in position-independent code so that it can be executed from any address range. The KA630-AA uses this feature to selectively enable and disable the external halt circuitry. If the console program is executing in the first address range (20040000 to 2004FFFF hex), external halt conditions are ignored. If the console program is executing in the second address range (20050000 to 2005FFFF hex), external halt conditions are honored, the console program halts, and immediately starts again (at its beginning) to process the halt. The console program normally executes from the first address range, while the diagnostics software normally executes from the second address range.

A console terminal is not required for operation, but halts should not be enabled on a system not having a console terminal.

The console program is divided into the following major sections.

- Power-up
- Entry/dispatch
- Diagnostics
- Restart
- Bootstrap
- Console I/O mode (system halted)
- Console I/O mode (system running)

The console program receives control whenever the processor halts, which occurs as a result of any of the following conditions.

- Power-up
- External halt signal
- Execution of a halt instruction
- Serious system error

When any halt occurs, the processor performs the following.

- Switches to physical addressing
- Saves the Program Counter (PC), Processor Status Longword (PSL), and Interrupt Stack Pointer (ISP) internally
- Encodes and saves the condition that caused the halt in a halt code
- Branches to the start of the console program ROM

If the DC OK signal is present, the hex value F is displayed on the KA630-AA LEDs. Upon entry, the console program outputs the hex value E to the console LEDs, indicating that at least one instruction has been executed. It then loops until Boot and Diagnostic Register (BDR) bit 15 (PWR OK) is set, indicating that power is stable.

The console program then checks bits <14:08> in IPR 43, noting if the halt is a power-up halt. If it is a power-up halt, the console program begins the power-up sequence described in Section 3.2. If the halt is the result of a condition other than power-up, control passes to the entry and dispatch code described in Section 3.3.

3.2 POWER-UP

At power-up, the console initializes the KA630-AA by performing a variety of operations unique to the power-up process.

3.2.1 Power-Up Mode

At power-up, BDR <10:09> is interpreted as a power-up mode field (Table 3-1). Several power-up operations are dependent on the power-up mode.

Table 3-1 Power-Up Modes

Mode	Language Prompt	Diagnostics
0	Prompt for language only if TOY battery backup failed.	Run full diagnostics.
1	Prompt for language on every power-up.	Run full diagnostics.
2	Set language to English.	Run console terminal loopback tests.
3	Set language to English.	Run abbreviated diagnostics.

3.2.2 Power Stabilization and ROM Checksum

The console program outputs the hex value D to the LEDs, indicating that the power stabilization wait is over. It then calculates a checksum of the console program ROM and checks it against the valid checksum stored in the ROM itself. If the computed checksum differs from the stored checksum, the console program hangs in a loop. If the checksum is the same, the power-up code proceeds to the next step.

3.2.3 Console Program Initialization

The next step of the power-up initialization is location and initialization of the memory needed for the console program itself. The hex value C is output to the LEDs at the beginning of this step.

During this step, the console ROM code searches top-down through available memory for a contiguous block to be used by the console program for writeable storage. This block consists of two pages for the console's direct use and additional pages for use to store a bitmap of available memory. The amount of memory allocated to the bitmap varies according to the amount of memory available.

Following initialization, memory appears as shown in Figure 3-1. The console program memory is used by the console program for its stack and other data structures.

The bitmap is filled in at a later time with a map of valid memory pages by the power-up memory diagnostics. This bitmap is passed to the bootstrap as a map of valid memory. Beginning from the base of the bitmap, the first bit corresponds to the first page of low memory, the second bit to the second page, and so on. If the bit is set, the page is good; if the bit is clear, the page failed the memory test. The bitmap does not map itself or any other memory that follows it in the console program.

Since system software is expected to use only pages marked as good in the bitmap, it is not expected to modify the bitmap or the console program memory. However, the console program memory pages and the bitmap are checksummed by the console program to guard against accidental modification by system software.

If the console program cannot locate enough memory for its own use and for the bitmap, it hangs. Following initialization of its memory, the console program clears the following Console Program Mailbox (CPMBX) register bits.

- <01:00> -- Processor halt action
- 2 -- "Bootstrap in progress" flag
- 3 -- "Restart in progress" flag

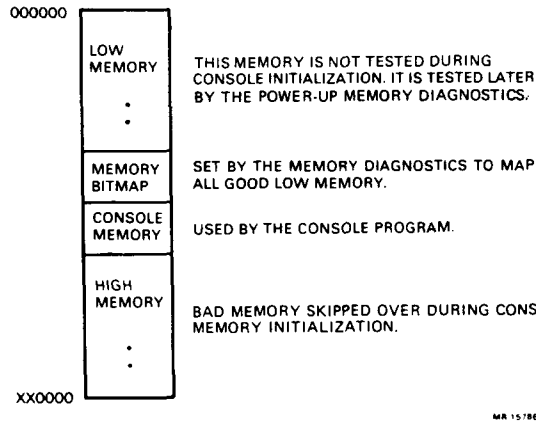


Figure 3-1 Console Memory Map After Initialization

3.2.4 Battery Backup Check

The console program then checks the TOY clock to determine if the battery backup has failed. If this has happened, time-of-year data has been lost along with the contents of all the TOY clock RAM. If the battery backup has failed, the console program performs the following steps.

- Stops the TOY clock
- Zeros the time and all TOY RAM
- Initializes the four TOY clock CSRs

The operating system must check TOY clock CSR B and determine if the clock is stopped to know if the TOY clock contains a valid time. No change is made to the LEDs during this operation.

3.2.5 InterProcessor Communication Register (IPCR) Test

Next, the IPCR is tested. The hex value B is output to the LEDs during this test. The test determines whether the Q22-Bus is arbitrating properly. If the CPU module is not arbitrating, the console program hangs at this point.

3.2.6 Determining the Console Terminal Type

3.2.6.1 Alternate Console Device Hardware Determination -- If the processor is an arbiter, the console program next checks for the presence of a VCB01 or VCB02 as the console device. If the KA630-AA is an auxiliary processor, this test is skipped. The hex value A is output to the LEDs during the test.

VCB01 and VCB02 alternate console devices are determined by testing for the presence of the CSR address first at 20001E92 hex (for VCB01), and then at 20001F00 hex (for VCB02). If there is no response at either location, the console program assumes that alternate console devices are not present and moves to the console terminal determination code (Section 3.2.6.2).

If a VCB01 or VCB02 video subsystem is detected, it is initialized and a short diagnostic is executed. If the initialization and diagnostics succeed, the console uses the VCB01 or VCB02 as the console terminal, skips the next step and moves directly to the console message language check (Section 3.2.7). If either the initialization or the diagnostic fails, the system hangs at this point.

3.2.6.2 **Console Terminal Determination** -- When VCB01 or VCB02 alternate console devices are not detected, it is assumed that a normal console terminal is connected to the console port, or that no terminal is connected. The console program then attempts to determine the type of terminal connected. This information is used when in console I/O mode to govern how command line editing is performed. The console program sends the console port a device attribute request escape sequence. If the device responds with a recognizable response, the terminal is classified as a video terminal. The terminal must respond in 1 second to the device query. When there is no response or the response is not recognized, the test is repeated twice. If the device still does not respond or the response is not recognized, the terminal is classified as a hard-copy terminal. Terminal response is recognized in either 8- or 7-bit mode.

The information obtained in this procedure is also used to determine if the terminal supports the Digital Multinational Character Set (MCS). The console program assumes that all new terminals (VT200 series and beyond) support MCS. If the terminal does not support MCS, CPMBX <07:04> is set to 2, selecting English as the console display language. The value 9 is output to the LEDs during this test.

3.2.7 Console Message Language Check

The console next outputs the value 8 to the LEDs and then determines the appropriate language to use for all console messages. The console language is stored in CPMBX <07:04>. The algorithm used to determine the language follows.

1. If power-up mode (Table 3-1) is 2 or 3, set the console language to English and exit.

2. If power-up mode is 1 and the terminal supports MCS, or if the value of CPMBX <07:04> is 0, solicit the language from the user. If the user does not respond within 30 seconds, set the language to English (mode 2) and exit.

Note that when the terminal is queried, if it is not recognized as one that supports MCS, CPMBX <07:04> is set to 2, forcing English as the console language. English messages use the 7-bit subset of MCS. If a loss of power to the TOY clock chip is detected, the contents of the TOY RAM are zeroed. This means that step 2 above causes the user to be prompted for language if the terminal supports the MCS.

If the console program determines that a VCB01 video display system is being used as the console, a step in addition to selecting one of the languages is required. The VCB01 display system uses the DEC LK201 keyboard, which comes in 16 national variants (Table 3-2). The keyboard variant cannot be determined by querying the keyboard itself; it must be determined either from the language selected or by means of an additional menu selection. If French, German or English is selected, the keyboard variant is ambiguous and the additional menu is displayed. The user is prompted to specify which national keyboard variant is in use. If the user does not respond in 30 seconds, the last selection is assumed.

3.3 ENTRY/DISPATCH

Following the determination of the console language on power-up, or directly on entry from any other halt condition, the console dispatches to the appropriate code to service the halt.

To determine what action to take, the console program examines the halt error code (IPR 43 <14:08>), the halt enable bit (BDR 14), and the processor halt action (CPMBX <01:00>). It then acts in accordance with the decision table shown in Table 3-3.

Table 3-2 Additional Language Selections (VCB01 Only)

ROM Language Selected/ Additional Selections	French	German	English
1	Canada	Germany/Austria	United Kingdom
2	France/Belgium	Switzerland	United States/ Canada
3	Switzerland		

Table 3-3 Console Entry Decision Table*

Halt Enable (BDR 14)	Power-Up Halt	Processor Halt Action (CPMBX <01:00>)	Functions
T	T	X	Diagnostics, halt.
T	F	0	Halt.
F	T	X	Diagnostics, bootstrap, halt.
F	F	0	Restart, bootstrap, halt.
X	F	1	Restart, halt.
X	F	2	Bootstrap, halt.
X	F	3	Halt.

* T = true, F = false, X = condition of the bit(s) does not matter.

If a power-up halt (second column) is true, it is one in which the halt error code contained in IPR 43 <14:08> equals 3. When the processor halt action is 1, 2 or 3, the condition of BDR bit 14 is ignored. When the processor halt action is 0, the action is determined by the condition of HLT ENB (BDR bit 14). Multiple actions mean that the first action is taken, and if and only if it fails, the next action is taken. Diagnostics are an exception. If diagnostics fail, the console program hangs without attempting to bootstrap the processor. If they succeed, then the next action is taken.

Note that because the KA630-AA does not support battery backup for main memory, it examines the halt code and does not attempt to perform restart operations following power-up.

3.4 DIAGNOSTICS

On power-up, the console outputs the message "Performing normal diagnostic tests of system" to the console terminal. The Entry/Dispatch code dispatches the diagnostics to check the processor and memory before proceeding. As each test in the diagnostics is run, it is output to the console terminal, causing a "countdown" to be displayed on the processor LEDs.

The first diagnostic LED code is 8. Executing the diagnostics continues the LED countdown. The diagnostic codes are listed in Chapter 5.

At the conclusion of all tests, the message "Tests successfully completed" is output to the console terminal. If a diagnostic test detects a fatal error, an error message is displayed on the console, along with a summary message indicating that continued operation is not possible. The console program then hangs there, leaving the test code on the LEDs. If halts are disabled, the only way to clear the system is to turn it Off and then On again. If halts are enabled, the system can be cleared by manually halting it, causing it to enter console command mode. Additional information on the diagnostics is located in Chapter 5.

3.5 RESTART

The console can restart a halted operating system. To do so, the console searches system memory for the Restart Parameter Block (RPB, Figure 3-2), a page-aligned control block created for this purpose by the operating system. If a valid RPB is found, the console restarts the operating system at an address specified in the RPB.

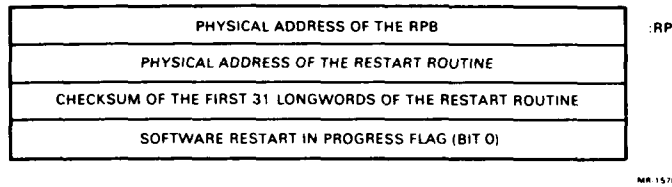


Figure 3-2 RPB Format

The console uses the following sequence to find an RPB:

1. Searches for a page of memory that contains its address in the first longword. If none is found, the search for an RPB fails.

2. Reads the second longword in the page (the physical address of the restart routine). If it is not a valid physical address, or if it is 0, the console program returns to step 1. The check for 0 is necessary to ensure that a page of 0s does not pass the test for a valid RPB.

3. Calculates the 32-bit 2's complement sum (ignoring overflows) of the first 31 longwords of the restart routine. If the sum does not match the third longword of the RPB, the console program returns to step 1. If the sum does match, a valid RPB exists and has been found.

The same algorithm is used for both arbiter and auxiliary processors.

The console keeps a "Restart in progress" flag in CPMBX bit 3, which it uses to avoid repeated attempts to restart a failing operating system. An additional "Restart in progress" flag may be maintained by software in the RPB.

The console uses the following sequence to restart the operating system:

1. Checks the "Restart in progress" flag in CPMBX bit 3. If it is set, restart fails.

2. Prints the message "Restarting the operating system" on the console terminal.

3. Sets CPMBX bit 3.

4. Looks for an RPB left in memory by the operating system. If none is found, restart fails.

5. Reads the software "Restart in progress" flag from bit 0 of the fourth longword of the RPB. If it is set, restart fails.

6. Loads the Stack Pointer (SP) with the address of the RPB plus 512 bytes.

7. Loads the Argument Pointer (AP) with the halt code (IPR 43 <14:08>).

8. Displays 0 on the console LEDs.

9. Starts the processor at the restart address, which is read from the second longword in the RPB.

If restart fails, the console program prints "Attempt to restart operating system failed" on the console terminal. If the restart is successful, the operating system must clear CPMBX bit 3.

3.6 BOOTSTRAP

The console program can load and start (bootstrap) an operating system. To do so, it performs the following steps:

1. Searches for a 64 Kbyte segment of correctly functioning system memory.

2. Sets SP equal to the base address of the segment plus 512 bytes.

3. Copies the primary bootstrap, called Virtual Memory Bootstrap (VMB), from the console program ROM to the segment starting at the location specified by the SP.

4. Branches to the first location in VMB, which then loads and starts the operating system.

To prevent a situation in which the console program repeatedly tries and fails to bootstrap the operating system, the console program maintains a "Bootstrap in progress" flag in CPMBX bit 2.

The console uses the following sequence to bootstrap the operating system:

1. Begins at step 4, if the bootstrap is the result of a console Bootstrap command.

2. Checks the "Bootstrap in progress" flag in CPMBX bit 2. If it is set, the bootstrap fails.

3. Prints the message "Starting the operating system" on the console terminal.

4. Sets CPMBX bit 1.

5. Locates a page-aligned, 64 Kbyte segment of good memory. If such a segment cannot be found, the bootstrap fails.

6. Initializes the Q22-Bus I/O map. The main function of this initialization is to preset the arbiter processor I/O map so that all unoccupied pages of the Q22-Bus are mapped to the corresponding pages in the first 4 Mbytes of local memory. This is a MicroVAX I compatibility feature, and is not done for auxiliary processors. Any auxiliary Q22-Bus I/O mapping must be coordinated with all other processors, so that all auxiliary processor I/O map registers are marked invalid. The bitmap is rebuilt during boot, as follows.

   a. Turn on IPCR bit 8, the halt flag.

   b. Disable the I/O map by clearing IPCR bit 5.

   c. If the KA630-AA is an arbiter processor, do the following for each I/O map register:

      (1) Set the map register address bits to map the Q22-Bus page to the corresponding local memory page.

      (2) If the corresponding Q22-Bus page is unoccupied, turn on the valid bit.

      (3) If the page is occupied, turn off the valid bit.

      If the KA630-AA is an auxiliary processor, turn off the valid bit in all map registers.

   d. Enable the I/O map by setting IPCR bit 5.

   e. If the KA630-AA is an auxiliary processor, loop until IPCR bit 8 is cleared.

      (Note that steps a. and e. are present to perform a secondary function while the Q22-Bus I/O map is initialized, namely, to synchronize an auxiliary processor with its bootstrap host.)

7. Loads the general registers for VMB as shown in Table 3-4.

8. Copies VMB from the console ROM to an address 512 bytes past the base of the good segment.

9. Invokes VMB. If VMB fails, the bootstrap fails.

Table 3-4 VMB Register Usage

Register	Description
R0	ASCII device name (from Bootstrap command) or 0
R1	Contents of BDR
R2	Memory bitmap size in bytes
R3	Address of memory bitmap
R4	Unused
R5	Software boot control flags (from Bootstrap command only)
R10	Halt PC value
R11	Halt PSL value
AP	Halt code (argument pointer)
SP	512 bytes past base of 64 Kbytes of good memory (stack pointer)

If bootstrap fails, the console prints "Attempt to start operating system failed" on the console terminal.

If the bootstrap is successful, the operating system must clear the "Bootstrap" and "Restart in progress" flags in CPMBX <03:02>, and clear the LED display by depositing a value of 0 in BDR <03:00>.

3.6.1 Primary Bootstrap Program (VMB)

VMB is the KA630-AA primary bootstrap. It is executed as the first part of a two-part system bootstrap operation. VMB contains the code that executes the following operations.

- Initialization of System Control Block (SCB)
- Initialization of an extended RPB
- Initialization of a Page Frame Number (PFN) bitmap and the relevant extended RPB fields
- Selection of a bootstrap device
- Performance of a Files-11 ODS2, boot block, ROM, or down-line load of the secondary bootstrap

The secondary bootstrap continues the bootstrap operation. For KA630-AA systems, primary bootstrap operations are defined by VMB, and secondary bootstrap operations are defined by the operating system being booted.

VMB finds the bootstrap device in one of three ways.

1. If the bootstrap is the result of a console Bootstrap command and a device name is specified in the command, that device is searched for the secondary bootstrap.

2. If the bootstrap is not the result of a console Bootstrap command or if no device name is specified, VMB searches the following devices, in the order shown.

   a. A bootable removable disk
   b. A bootable fixed disk
   c. TK50 tape unit
   d. MRV11 PROM
   e. DEQNA, for a down-line bootstrap

3. If the bootstrap is the result of a halt with CPMBX <01:00> equal to 2, that is, a request from the operating system to reboot the system, the device used previously to bootstrap the operating system is used (as well as the same command flags).

When a VMB attempt fails, the console program halts.

3.6.1.1 Bootstrap Devices -- The following bootstrap devices are supported by the console program.

- RQDX2, RQDX3, KDA, and RC25 MSCP disk controllers. VMB can boot from any disk unit supported by an MSCP disk controller. Units supported by RQDX2 and RQDX3 are RX50, RD51, RD52 and RD53. Units supported by KDA are RA63 and RA81. The unit supported by the KLESI is the RC25. The Bootstrap command designation for these units is DUA0, DUA1, etc. The first controller must be configured at Q22-Bus address 17772150 (octal) and interrupt vector 154 (octal). Additional controllers are located in floating CSR and vector space.

- DEQNA Ethernet adapter. This controller connects to an Ethernet cable. The Bootstrap command designation for this device is XQA0. The controller must be configured at Q22-Bus address 17774440 (octal) and vector 124 (octal, for one unit or the first unit).

- MRV11 Programmable Read-Only Memory (PROM) board. The Bootstrap command designation is PRA0.

- TMSCP tape controller. The Bootstrap command designation is MUA0. The TQK50 controller must be configured at Q22-Bus address 17774500 (octal).

3.6.1.2 Bootstrap Command F1 -- When a bootstrap is invoked using the Bootstrap command, the user can specify several Bootstrap command flags by bit encoding the flags in a flag word specified with the /R5: qualifier. These command flags are described in Table 3-5.

3.6.1.3 Booting from Disk -- For VMB to boot using an MSCP disk controller, the first controller must be configured at 17772150 (octal) and subsequent controllers must be configured in their appropriate floating CSRs and vectors. When VMB determines that a controller is present, it searches for an accessible unit attached to the controller that has a removable volume. The search is made in order of increasing unit number (DUA0, DUA1, etc.). If it finds such a unit with a removable volume, VMB proceeds as described below. If it finds no such volume, it searches the same controller again, but this time checking for nonremovable volumes. If by this time no accessible volume is found, it checks for the next controller and repeats the process. If no more controllers are found, the disk boot fails.

If an accessible volume is located, VMB then determines if it is a Files-11 volume. If it is, it searches the volume for file [SYS0.SYSEXE] SYSBOOT.EXE, which contains the secondary bootstrap. If this file is found, VMB loads and executes it (performs a secondary bootstrap).

If the volume is not a Files-11 volume, VMB then checks logical block 0 of the volume for a valid bootblock (Figure 3-3). If the bootblock is a valid bootblock, VMB loads and executes the secondary bootstrap specified in the bootblock. If there is no valid bootblock present, the search resumes for the next accessible volume.

Note that the bootstrap process can be altered by Bootstrap command flags, as described in Table 3-5.

Table 3-5 VMB Bootstrap Command Flags

Bit Number(s)	Value (Hex)	Flag Word	Description
00	00000001	Conversation	Conversational bootstrap.
03	00000008	Bootblock	Secondary bootstrap from bootblock. When this bit is set, VMB reads logical block number 0 of the boot device and tests it for conformance with the bootblock format. If in conformance, the block is executed to continue the bootstrap. No attempt to perform a Files-11 bootstrap is made.
04	00000010	Diagnostic	Diagnostic bootstrap. When this bit is set the secondary bootstrap is file [SYS0.SYSMAINT]DIAGBOOT.EXE.
06	00000040	Header	Image header. If this bit is not set, VMB transfers control to the first location of the secondary bootstrap. If this bit is set, VMB transfers control to the address specified by the file's image header.
08	00000100	Solicit	File name Solicit. When this bit is set, VMB prompts the operator for the name of the secondary bootstrap file.
09	00000200	Halt	Halt before transfer. When this bit is set, VMB halts before transferring control to the secondary bootstrap.
<31:28>	X0000000	Topsys	X can be any value from 0 through F (hex). This flag changes the top level directory name for system disks with multiple operating systems. For example, if X = 1, the top level directory name is [SYS1....].

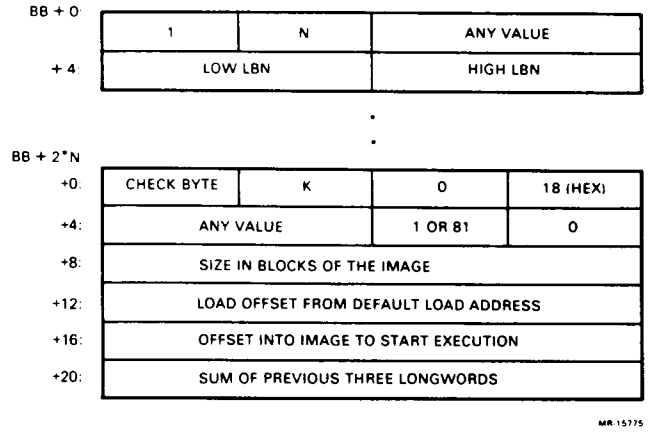


Figure 3-3 Bootblock Format

BB + 0: These two bytes can have any value.

BB + 2: This value is the word offset from the start of the bootblock to the identification area described below.

BB + 3: This byte must be 1.

BB + 4: This longword contains the logical block number (word swapped) of the secondary image.

BB + (2*n) + 0: This byte defines the expected instruction set. (18 hex = VAX instruction set.)

BB + (2*n) + 1: This byte defines the expected controller type. (0 = unknown.)

BB + (2*n) + 2: This byte defines the file structure on the volume. It may be any value.

BB + (2*n) + 3: This byte must be the 1's complement of the sum of the previous three bytes.

BB + (2*n) + 4: This byte must be 0.

BB + (2*n) + 5: This byte must be 1 or 81 (hex). This byte defines the version number of the format standard and the type of disk. The version is 1; the high bit is 0 for single-sided, 1 for double-sided volumes.

BB + (2*n) + 6: These two bytes may be any value, but generally they are 0.

BB + (2*n) + 8: This entry is a longword containing the size (in blocks) of the secondary bootstrap image.

BB + (2*n) + 12: This entry is a longword containing a load offset (usually 0) from the default load address of the secondary bootstrap.

BB + (2*n) + 16: This entry is a longword containing the byte offset into the secondary bootstrap where execution is to begin.

BB + (2*n) + 20: This entry is a longword containing the sum of the previous three longwords.

3.6.1.4 Booting from Tape -- If no bootable disk is found, VMB attempts to bootstrap from a TK50 tape.

If a TK50 is present, VMB determines if a tape is loaded and if the unit is on-line. If so, VMB rewinds the tape and searches for the file TAPEBOOT.EXE. (The user may specify an alternative file name by setting the Solicit bit in the software command register.) If this file is found, VMB loads and executes it. Normally this file would contain a program to load an operating system from tape onto a system disk.

If a user has both disks and tape and a disk is bootable, to boot from tape the user must either take all bootable disks off-line, or explicitly boot the TK50 using the console Bootstrap command.

3.6.1.5 Booting from PROM -- If neither disk nor tape is bootable, VMB checks for a PROM bootstrap. To locate a PROM bootstrap, VMB searches the Q22-Bus address range from high to low addresses by page, looking for readable memory. If the first six longwords of any such page contain a valid PROM signature block (Figure 3-4), VMB passes control directly to the bootstrap code in the PROM. It does not copy the PROM code to local memory for execution, as it does for all other secondary bootstraps.

Note that while defined as an MRV11 PROM or equivalent bootstrap, VMB does not actually require that the signature block or the bootstrap code be in PROM. The signature block or bootstrap code may be in ROM, nonvolatile RAM, or it could be loaded into another KA630-AA's RAM and mapped to the Q22-Bus.

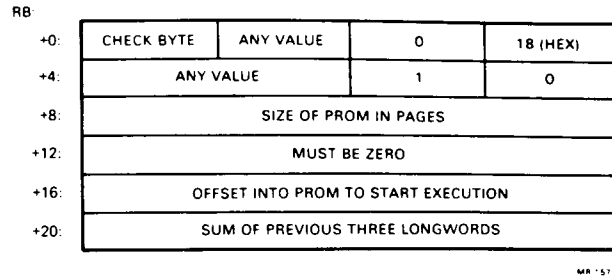


Figure 3-4 PROM Bootstrap Memory Format (Signature Block)

RB + 0: This byte must be 18 (hex).

RB + 1: This byte must be 0.

RB + 2: This byte may be any value.

RB + 3: This byte must be the 1's complement of the sum of the previous three bytes.

RB + 4: This byte must be 0.

RB + 5: This byte must be 1.

RB + 6: These two bytes may be any value.

RB + 8: This longword contains the size (in pages) of the PROM.

RB + 12: This longword must be 0.

RB + 16: This longword contains the byte offset into the PROM where execution is to begin.

RB + 20: This entry is a longword containing the sum of the previous three longwords.

3.6.1.6 Booting from DEQNA -- If no other bootstrap device is found, VMB attempts to bootstrap from the DEQNA Ethernet controller. In this case, the secondary bootstrap is down-line loaded from a host on the Ethernet, using DECnet low-level Maintenance Operation Protocol (MOP) Version 3.0. The DEQNA module must be configured at Q22-Bus address 17774440 (octal).

The down-line load process consists of the following steps:

1. VMB performs local testing of the DEQNA. If the tests fail, the bootstrap attempt fails and the three LEDs on the DEQNA are set according to the problem detected. The LED settings and their interpretations are as follows.

   - 3 LEDs on: DEQNA initialization failure
   - 2 LEDs on: internal loopback failure
   - 1 LED on: external loopback failure

2. VMB transmits a program request MOP message over the Ethernet. The message destination is the load assistant multicast address AB-00-00-01-00-00. The message source address is the DEQNA station address (from DEQNA PROM). The MOP program type is operating system.

3. VMB waits approximately 30 seconds to receive a response. If it does not receive a response, it retransmits the request every 30 seconds for a total of 2 minutes. If a response is not received in two minutes, the bootstrap fails.

4. VMB accepts MOP load messages and loads the data into memory, terminating when the final message is received as indicated in the MOP message protocol. If the interval between load messages exceeds 30 seconds, VMB restarts the DEQNA bootstrap at step 2.

3.6.1.7 Booting an Auxiliary Processor -- VMB bootstraps an auxiliary processor by using the ROM bootstrap protocol. Refer to the Q22-Bus initialization algorithm in Section 3.6.

Note that whenever the console program is entered, it turns off IPCR bit 8. Steps 1 and 5 ensure that an auxiliary processor loops until some other processor clears IPCR bit 8. When another processor -- the bootstrap host, clears IPCR bit 8, the auxiliary proceeds with the bootstrap. This synchronization gives the arbiter processor control over the bootstrapping of all auxiliary processors.

An auxiliary processor cannot directly bootstrap itself from any of the normal bootstrap devices, so VMB on an auxiliary checks only for the ROM bootstrap described above. The ROM bootstrap may be either a block of nonvolatile memory on the Q22-Bus, or the bootstrap host can construct an equivalent bootstrap in RAM. In either case, the auxiliary does not proceed with the bootstrap until the bootstrap host clears IPCR bit 8. The bootstrap host, in turn, should not clear the auxiliary IPCR bit 8 unless IPCR bit 5 is clear.

3.6.2 Secondary Bootstrap Program

The secondary bootstrap program is invoked as the second part of a system bootstrap. Following successful execution of the primary bootstrap, the secondary bootstrap has either been loaded into memory or located in ROM. It is the responsibility of the secondary bootstrap to complete the bootstrap of the processor.

VMB calls the secondary bootstrap with the processor in the following state.

- The processor is running in kernel mode on the interrupt stack at IPL 31 (hex).
- R11 contains the base address of the extended RPB (Figure 3-5) created by VMB.

OFFSET (HEX):	
00:	ADDRESS OF THE EXTENDED RPB
04:	0
08:	0
0C:	0
10:	PC AT RESTART/HALT
14:	PSL AT RESTART/HALT
18:	HALT CODE
1C:	VMB INPUT REGISTER R0
20:	VMB INPUT REGISTER R1
24:	VMB INPUT REGISTER R2
28:	VMB INPUT REGISTER R3
2C:	VMB INPUT REGISTER R4
30:	VMB INPUT REGISTER R5
34:	TWO LONGWORDS RESERVED
3C:	DISK BLOCK ADDRESS OF SECONDARY BOOTSTRAP
40:	SIZE OF SECONDARY BOOTSTRAP FILE IN BLOCKS
44:	DESCRIPTOR OF PFN BITMAP (TWO LONGWORDS)
48:	NUMBER OF GOOD PHYSICAL PAGES
4C:	RESERVED
50:	PHYSICAL CSR ADDRESS OF BOOT DEVICE
54:	FOUR LONGWORDS RESERVED
68:	SECONDARY BOOTSTRAP FILE NAME (40 CHARACTERS)
90:	EIGHT LONGWORDS RESERVED
B0:	SYSTEM CONTROL BLOCK BASE ADDRESS

MR-15784

Figure 3-5 Extended RPB

- AP contains the address of the secondary bootstrap argument list (Figure 3-6).
- SP contains the address of the top of the stack plus 4, which is also the address of the beginning of the secondary bootstrap (Figure 3-7).
- System Control Block Base (SCBB, an internal processor register) contains the address of the SCB created by VMB.

Note that the first four longwords of the VMB-created, extended RPB would not be recognized as a valid RPB by the console restart algorithm. It is up to the secondary bootstrap or the operating system itself to complete the RPB if automatic restart is desired.

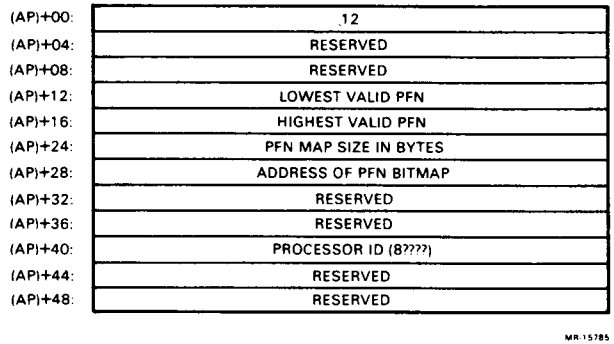


Figure 3-6 Secondary Bootstrap Argument List

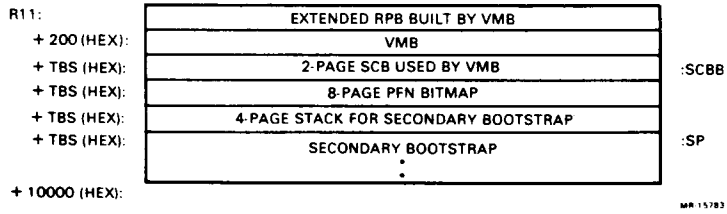


Figure 3-7 Secondary Bootstrap Memory Map

3.7 CONSOLE I/O MODE (SYSTEM HALTED)

When the KA630-AA is halted, the operator controls the system through the console terminal using the console command language. The console terminal is in console I/O mode. The console prompts the operator for input with the string >>>.

3.7.1 Console Control Characters

In console I/O mode, several keys have special functions. Note that the control characters are typed by pressing the character key while holding down the Control key (<CTRL>).

- <CR> -- The carriage return ends a command line. No action is taken on a command until after it is terminated by pressing the Carriage Return key. A null line terminated by a carriage return is treated as a valid, null command. No action is taken, and the console reprompts for input. Carriage return is echoed as carriage return, line feed.

- <Rubout> -- Pressing the Rubout key deletes the previously typed character. What appears on the console terminal depends on whether the terminal is a video or hard-copy terminal.

  On hard-copy terminals, when <Rubout> is pressed, the console echoes with a backslash (\), followed by the character being deleted. If the operator types additional rub-outs, the additional characters deleted are echoed. When the operator types a nonrub-out character, the console echoes another backslash, followed by the character typed. The result is to echo the characters deleted, surrounding them with backslashes. For example:

  ```
  The operator types: EXAMI;E <Rubout> <Rubout> NE <CR>
  The console echoes: EXAMI;E\E\;\NE <CR>
  The console sees the command line: EXAMINE <CR>
  ```

  On video terminals, when <Rubout> is pressed, the previous character is erased from the screen and the cursor is restored to its previous position.

  The console does not delete characters past the beginning of a command line. If the operator types more rub-outs than there are characters on the line, the extra rub-outs are ignored. If a rub-out is typed on a blank line, it is ignored.

- <CTRL> U -- The console echoes U <CR>, and deletes the entire line. If <CTRL> U is typed on an empty line, it is echoed, and otherwise ignored. The console prompts for another command.

- <CTRL> S -- This stops output to the console terminal until <CTRL> Q is typed. <CTRL> S and <CTRL> Q are not echoed. <CTRL> C, <CTRL> O, and <Break> also clear <CTRL> S.

- <CTRL> Q -- This resumes output to the console terminal. Additional <CTRL> Qs are ignored. <CTRL> S and <CTRL> Q are not echoed.

- <CTRL> O -- The console throws away transmissions to the console terminal until the next <CTRL> O is entered. <CTRL> O is echoed as O <CR> when it disables output, but is not echoed when it reenables output. Output is reenabled if the console prints an error message, or if it prompts for a command from the terminal. Displaying a Repeat command does not reenable output. When output is reenabled for reading a command, the console prompt is displayed. Output is also enabled by entering program I/O mode, by <Break> and by <CTRL> C. <CTRL> O clears <CTRL> S.

- <CTRL> R -- This causes the console to echo <CR> <LF> followed by the current command line. This function can be used to improve the readability of a command line that has been heavily edited.

- <CTRL> C -- The console echoes C and aborts processing a command. <CTRL> C has no effect as part of a binary load data stream. <CTRL> C clears <CTRL> S, and reenables output stopped by <CTRL> O. When <CTRL> C is typed as part of a command line, the console deletes the line as it does with <CTRL> U.

- <Break> -- If the console is in console I/O mode, <Break> is equivalent to <CTRL> C, but is not echoed at all. If the console is in program I/O mode and halt is disabled, <Break> is ignored. If the console is in program I/O mode and halt is not disabled, <Break> causes the processor to halt and enter console I/O mode.

If an unrecognized control character is typed, it is echoed as a caret (^) followed by the ASCII code character plus 64. A control character here, means a character with an ASCII code less than 32 decimal [C0], or between 128 and 159 decimal [C1]. For example, BEL (ASCII code 7) is echoed as ^G, since capital G is ASCII code 7 + 64 = 71. When a control character is deleted by rubout, it is echoed the same way. After echoing the control character, the console processes it like a normal character. Unless the control character is part of a comment, the command is invalid, and the console responds with an error message.

3.7.2 Console Command Syntax

The console accepts commands up to 80 characters long. Longer commands are responded to with an error message. The count does not include rub-outs, rubbed out characters, or the terminating carriage return.

Commands may be abbreviated. Abbreviations are formed by dropping characters from the end of a keyword. All commands are recognized from their first character.

Multiple adjacent spaces and tabs are treated as a single space by the console. Leading and trailing spaces and tabs are ignored.

Command qualifiers can appear after the command keyword, or after any symbol or number in the command.

All numbers (addresses, data, counts) are in hexadecimal. (Note, though, that symbolic register names include decimal digits.) Hex digits are 0 through 9, and A through F. The console does not distinguish between upper and lower case either in hex numbers (A through F) or in commands. Both are accepted.

3.7.3 References to Processor Registers and Memory

The KA630-AA console is implemented by macrocode executing from ROM. For this reason, the actual processor registers cannot be modified by the command interpreter. When console I/O mode is entered, the console saves the processor registers in a scratch page and all command references to them are directed to the corresponding scratch page locations, not to the registers themselves. When the console reenters program mode, the saved registers are restored and any changes then become operative. References to processor memory are handled normally except where noted below.

Generally, a free page on the interrupt stack is used for the scratch page, so the console does not modify the machine state. If a free page on the interrupt stack cannot be located, the console program uses the last valid page in contiguous physical memory and the original machine state is lost. This should occur only on power-up.

References to the console scratch page by Examine and Deposit commands must be qualified by the /U qualifier. Access is primarily to simplify debugging of the console program. The binary load and unload commands cannot reference the console scratch page.

3.7.4 Console Commands

3.7.4.1 Binary Load and Unload (X)

Command Syntax:

X <address> <count> <CR> <checksum>

The X command is for use by automatic systems communicating with the console. It is not intended for use by operators. The console loads or unloads (that is, writes to memory, or reads from memory) the specified number of data bytes, starting at the specified address.

If bit 31 of the count is clear, data is to be received by the console, and deposited into memory. If bit 31 of the count is set, data is to be read from memory and sent by the console. The remaining bits in the count are a positive number indicating the number of bytes to load or unload.

The console accepts the command upon receiving the carriage return. The next byte the console receives is the command checksum, which is not echoed. The command checksum is verified by adding all command characters, including the checksum, (but not including the terminating carriage return or rub-outs or characters deleted by rub out), into an 8-bit register initially set to zero. If no errors occur, the result is zero. If the command checksum is correct, the console responds with the input prompt and either sends data to the requester or prepares to receive data. If the command checksum is in error, the console responds with an error message. The intent is to prevent inadvertent operator entry into a mode where the console is accepting characters from the keyboard as data, with no escape sequence possible.

If bit 31 of the count is clear (binary load commands), the console responds with the input prompt, then accepts the specified number of data bytes for depositing to memory, and an additional byte of received data checksum. The data is verified by adding all data characters and the checksum character into an 8-bit register initially set to zero. If the final contents of the register is nonzero, the data or checksum are in error, and the console responds with an error message.

If bit 31 of the count is set (binary unload commands), the console responds with the input prompt, followed by the specified number of bytes of binary data. As each byte is sent it is added to a checksum register initially set to zero. At the end of the transmission, the 2's complement of the low byte of the register is sent.

If the data checksum is incorrect on a load, or if memory errors or line errors occur during the transmission of data, the entire transmission is completed, and then the console issues an error message. If an error occurs during loading, the contents of the memory being loaded are unpredictable.

Echo is suppressed during the receiving of the data string and checksums.

It is possible to control the console using the console control characters (<CTRL> C, <CTRL> S, <CTRL> O, etc.) during binary unload commands. It is not possible during binary load commands, as all received characters are valid binary data.

Data being loaded with a binary load command must be received by the console at a rate of at least one byte per second. The command checksum that precedes the data must be received by the console within 10 seconds of the <CR> that terminates the command line. The data checksum must be received within 10 seconds of the last data byte. If any of these timing requirements are not met, the console aborts the transmission by issuing an error message and prompting for input.

The entire command, including the checksum, may be sent to the console as a single burst of characters at the console's specified character rate. The console is able to receive at least 4 Kbytes of data in a single X command.

3.7.4.2 Boot

Command Syntax:

BOOT [<qualifier list>] [<device>]

The device specification is of the format **ddcu**, where **dd** is a two-letter device mnemonic, **c** is an optional one-digit controller number, and **u** is a one-digit unit number.

The console initializes the processor and starts VMB running. VMB boots the operating system from the specified device. The default bootstrap device is determined as described in Section 3.6.

Qualifier:

- /R5:<data> -- After initializing the processor and before starting VMB, R5 is loaded with the specified data. This allows a console user to pass a parameter to VMB. (To remain compatible with previous processors, /<data> is also recognized and has the same result.)

3.7.4.3 Comment (!)

Command Syntax:

! <comment>

The comment command is ignored. It is used to annotate console I/O command sequences.

3.7.4.4 Continue

Command Syntax:

CONTINUE

The processor begins instruction execution at the address currently contained in the program counter. Processor initialization is not performed. The console enters program I/O mode.

3.7.4.5 Deposit

Command Syntax:

DEPOSIT [<qualifier list>] <address> <data>

This command deposits the data into the address specified. If no address space or data size qualifiers are specified, the defaults are the last address space and data size used in a Deposit or Examine command. After processor initialization, the default address space is physical memory, the default data size is long, and the default address is zero.

If the specified data is too large to fit in the data size to be deposited, the console ignores the command and issues an error response. If the specified data is smaller than the data size to be deposited, it is extended on the left with zeros.

The address may also be one of the following symbolic addresses:

- PSL -- The processor status longword. No address space qualifier is legal. When PSL is examined, the address space is identified as M (machine dependent).
- PC -- The program counter (general register 15). The address space is set to /G.
- SP -- The stack pointer (general register 14). The address space is /G.

- Rn -- General register n. The register number is in decimal. The address space is /G. For example:

  D R5 1234 is equivalent to D/G 5 1234.

  D R10 6FF00 is equivalent to D/G A 6FF00.

- + -- The location immediately following the last location referenced in an Examine or Deposit command. For references to physical or virtual memory spaces, the location referenced is the last address plus the size of the last reference (1 for byte, 2 for word, 4 for longword). For other address spaces, the address is the last address referenced plus one.

- - -- The location immediately preceding the last location referenced in an Examine or Deposit command. For references to physical or virtual memory spaces, the location referenced is the last address minus the size of this reference (1 for byte, 2 for word, 4 for longword). For other address spaces, the address is the last address referenced minus one.

- * -- The location last referenced in an Examine or Deposit command.

- @ -- The location addressed by the last location referenced in an Examine or Deposit command.

Qualifiers:

- /B -- The data size is byte.

- /W -- The data size is word.

- /L -- The data size is longword.

- /V -- The address space is virtual memory. All access and protection checking occurs. If the access would not be allowed to a program running with the current PSL, the console issues an error message. Virtual space Deposits cause PTE bit M to be set. If memory mapping is not enabled, virtual addresses are equal to physical addresses.

- /P -- The address space is physical memory.

- /I -- The address space is internal processor registers. These are the registers addressed by the MTPR and MFPR instructions.

- /G -- The address space is the general register set, R0 through PC.

- /U -- Access to console program memory is allowed. This qualifier also disables virtual address protection checks.

- /N:<count> -- The address is the first of a range. The console deposits to the first address, then to the specified number of succeeding addresses. Even if the address is the symbolic address '-', the succeeding addresses are at larger addresses. The symbolic address specifies only the starting address, not the direction of succession. For repeated references to preceding addresses, use Repeat Deposit - <data>.

NOTE

Only memory may be accessed as bytes or words. General registers, the PSL and IPRs must be accessed using the longword reference. This means that the /B and /W qualifiers may not be used with the /I and /G qualifiers.

For example:

D/P/B/N:1FF 0 0	Clears the first 512 bytes of physical memory.
D/V/L/N:3 1234 5	Deposits 5 into four longwords starting at virtual address 1234.
D/N:8 R0 FFFFFFFF	Loads general registers R0 through R8 with -1.
D/N:200 - 0	Starting at previous address, clears 513 bytes.

If conflicting address space or data sizes are specified, the console ignores the command and issues an error response.

3.7.4.6 Examine

Command Syntax:

EXAMINE [<qualifier list>] [<address>]

Examines the contents of the specified address. If no address is specified, '+' is assumed. The address may also be one of the symbolic addresses described under Section 3.7.4.5, Deposit.

Qualifiers:

The same qualifiers used with Examine may be used with Deposit.

Response:

<tab> <address space identifier> <address> <tab> <data>

The address space identifier can be:

- P -- Physical memory. Note that when virtual memory is examined, the address space and address in the response are the translated physical address.
- G -- General register.
- I -- Internal processor register.
- M -- Machine dependent address (used only for display of the PSL).

3.7.4.7 Find

Command Syntax:

FIND [<qualifier list>]

The console searches main memory starting at address zero for a page-aligned, 64 Kbyte segment of good memory, or an RPB. If the segment or block is found, its address plus 512 bytes is left in SP. If the segment or block is not found an error message is issued, and the contents of SP are unpredictable. If no qualifier is specified, /RPB is assumed.

Qualifiers:

- /Memory -- Searches memory for a page-aligned segment of good memory, 64 Kbytes in length. The search includes a read/write test of memory and leaves the contents of memory unpredictable.
- /RPB -- Searches memory for a restart parameter block. The search leaves the contents of memory unchanged.

3.7.4.8 Initialize

Command Syntax:

INITIALIZE

A processor initialization is performed. The following registers are set (all values are hexadecimal):

PSL	041F0000
IPL	1F
ASTLVL	4
SISR	0
ICCS	0
RXCS	0
TXCS	80
MAPEN	0

All other registers are unpredictable.

The previous console reference defaults (the defaults used to fill in unsupplied qualifiers for Deposit and Examine commands) are set to physical address, longword size and address 0.

3.7.4.9 Halt

Command Syntax:

HALT

The Halt command has no effect; the processor is already halted when in console I/O mode.

3.7.4.10 Repeat

Command Syntax:

REPEAT <command>

The console repeatedly displays and executes the specified command. The repeating is stopped by typing <CTRL> C. Any valid console command may be specified for the command with the exception of Repeat.

3.7.4.11 Start

Command Syntax:

START [<address>]

The console starts instruction execution at the specified address. If no address is given, the current PC is used. If no qualifier is present, macroinstruction execution is started. If memory mapping is enabled, macroinstructions are executed from virtual memory. The Start command is equivalent to a Deposit to PC, followed by a Continue. No Initialize is performed.

3.7.4.12 Test

Command Syntax:

TEST [<test number>]

The console invokes a diagnostic test program denoted by <test number>. Valid test numbers are 3 through 7 and B. If no test number is supplied, no test is performed.

3.7.4.13 Unjam

Command Syntax:

An I/O bus reset is performed.

3.7.5 Console Errors and Error Messages

Some console commands result in errors. For example, if a memory error occurs as the result of a console command, the console responds with an error message. The error messages are listed in Table 3-6.

3.7.6 Halts and Halt Messages

Whenever the processor halts, the console prints the halt code, error message, and the hex value contained in the program counter. For example:

```
06 HLT INST
PC = 800050D3
```

The halt code is passed to the operating system on a restart. The halt messages are listed in Table 3-7.

Table 3-6 Console Error Messages

Halt Code	Message	Explanation
	FNF	VMB could not find the secondary bootstrap file.
16	ILL REF	The requested reference would violate virtual memory protection, the address is not mapped, the reference is invalid in the specified address space, or the value is invalid in the specified destination.
17	ILL CMD	The command string cannot be parsed.
18	INV DGT	A number has an invalid digit.
19	LTL	The command was too large for the console to buffer. The message is issued only after the console receives the terminating carriage return.
1A	ILL ADR	The address specified falls outside the limits of the address space.
1B	VAL TOO LRG	The value specified does not fit in the destination.
1C	SW CONF	Switch conflict. For example, two different data sizes are specified with an Examine command.
1D	UNK SW	The switch is unrecognized.
1E	UNK SYM	The symbolic address in an Examine or Deposit command is unrecognized.
1F	CHKSM	The command or data checksum of an X command is incorrect.
20	HLTED	The operator entered a Halt command.
21	FND ERR	A Find command failed either to find the RPB or 64 Kbytes of good memory.
22	TMOUT	During an X command, data failed to arrive in the time expected.
23	MEM ERR	Parity error detected.

Table 3-7 KA630-AA Halt Messages

Halt Code	Message	Explanation
02	EXT HLT	<Break> was typed on the console, QBINIT or QBHALT was asserted.
04	ISP ERR	In attempting to push state onto the interrupt stack during an interrupt or exception, the processor discovered that the interrupt stack was mapped No Access or Not Valid.
05	DBL ERR	The processor attempted to report a machine check to the operating system, and a second machine check occurred.
06	HLT INST	The processor executed a halt instruction in kernel mode.
07	SCB ERR3	The vector had bits <01:00> equal to 3.
08	SCB ERR2	The vector had bits <01:00> equal to 2.
0A	CHM FR ISTK	A change mode instruction was executed when PSL bit IS was set.
0B	CHM TO ISTK	The exception vector for a change mode had bit 0 set.
0C	SCB RD ERR	A hard memory error occurred while the processor was trying to read an exception or interrupt vector.
10	MCHK AV	An access violation or an invalid translation occurred during machine check exception processing.
11	KSP AV	An access violation or an invalid translation occurred during processing of an invalid kernel stack pointer exception.

3.8 CONSOLE I/O MODE (SYSTEM RUNNING)

When the processor is not executing instructions from the console program ROM, it is in program I/O mode, in which all terminal interaction is handled by the operating system. In program I/O mode, the console terminal behaves like any other operating system terminal. If halts are disabled, break is ignored. If halts are enabled, break causes the processor to halt, that is, to enter console I/O mode.

On successful power-up, the first line of the console display identifies the processor and version number (XX) of the console program ROM. The next line explains that the system is performing normal tests. The countdown sequence assures the user that the system is progressing through its tests, and documents which tests are executed. When diagnostics are complete, the console notifies the user that the tests completed successfully. The "Loading system software" message indicates the beginning of the bootstrap sequence. The execution of the bootstrap sequence causes the remaining digits of the countdown to be displayed. Because successful completion of a bootstrap occurs in the context of the operating system bootstrapped, a confirming message indicating that the system power-up has completed can only be issued by the bootstrapped operating system.

When fatal problems are detected by the diagnostics, the countdown sequence is interrupted and a diagnostic message is displayed. The diagnostic message is composed of a question mark, a subtest code number, and up to three parameters for use by diagnostic personnel. More than one such error message is possible, but unlikely. The summary message that follows indicates that the test failed and that normal operation is not possible. The console program then hangs.

Catastrophic errors are errors of such severe magnitude that the program cannot continue. When a catastrophic error is detected, the program attempts to display an error message on the console terminal. Following that attempt, the processor goes into an infinite loop at IPL 31 (there is no halt state for MicroVAX). An example of catastrophic error is when the console program is unable to locate any working memory.

It is possible to bypass all diagnostic tasks. This may be done by enabling halts and by manually halting the processor following power-up or reset. The diagnostics are then halted and the processor enters console I/O mode. This option allows a field service engineer to bypass a failing test and enter console I/O mode, where the console commands can be used to further diagnose the problem.

As part of each diagnostic subtest, a test code is displayed on the console and on the LEDs, making it possible to monitor the progress of the diagnostics. The two display mechanisms use unrelated logic, providing a high probability that at least one is currently operative. If a hard error is detected by a test, a diagnostic message is displayed on the console. If a catastrophic error occurs, it may not be possible to display a diagnostic message on the console, but the most recent test code is left on the LEDs.

The following significant console features are omitted by the KA630-AA console program.

- Microstep command -- Not supported by the MicroVAX CPU chip.
- Load command -- No console storage device is supported.
- Set command -- No set options are defined.
- Next command -- Not supported by the MicroVAX CPU chip.
- @ command -- No console storage device is supported.

The console supports the Digital MCS. This support extends to displaying foreign language messages with MCS, accepting and echoing MCS characters, and accepting a device attributes report (the console queries the terminal to determine if it is a CRT) using the C1 control characters of MCS. However, all console commands must be entered using the American National Standards Institute (ANSI) subset.

If the terminal does not support MCS, the console uses English message texts.

The console program uses four characters that are national replacement characters, the caret (^), the backslash (\) and the right and left square brackets ([ ]). The caret is used by the console to denote control characters. The backslash is used to delimit text deletions when editing console input. The square brackets are used to denote directory specifications when the user directs the bootstrap to solicit a secondary bootstrap file name. No provision is made for terminals that replace any of these characters.

4.1 INTRODUCTION

This chapter contains a list of the data types, instruction groups and processor registers implemented by the KA630-AA. Register structures and formats, as well as MicroVAX memory management, are also described.

4.2 PROCESSOR STATE

The processor state is stored in processor registers, rather than in memory. This section describes the processor registers, the general purpose register set, and the Processor Status Longword (PSL). Nonprivileged software can access the general purpose register set and the Processor Status Word (PSW, bits <15:00> within the PSL). The processor registers and bits <31:16> of the PSL can only be accessed by privileged software, using the Move To Processor Register (MTPR) and Move From Processor Register (MFPR) instructions.

4.2.1 General Purpose Registers

There are 16 general purpose registers, R0 through R15. The bits of a register are numbered from right to left, 0 through 31.

The following registers are defined by the VAX architecture.

- R15 is the Program Counter (PC). The PC contains the address of the next instruction byte of the program.
- R14 is the Stack Pointer (SP). The SP contains the address of the top of the processor-defined stack.
- R13 is the current Frame Pointer (FP). The VAX procedure call convention builds a data structure on the stack called a stack frame. The FP contains the address of the base of the stack frame.
- R12 is the Argument Pointer (AP). The VAX procedure call convention uses a data structure termed an argument list. The AP contains the address of the base of this data structure.

4.2.2 Processor Status Longword

The PSL (Table 4-1, Figure 4-1) determines the execution state of the processor at any time.

Table 4-1 Processor Status Longword Description

Bit(s)	Mnemonic	Name/Meaning
31	CM	Compatibility Mode.* This bit always reads as 0. Loading a 1 into this bit has no effect.
30	TP	Trace Pending.
<29:28>		Must Be Zero.
27	FPD	First Part Done.
26	IS	Interrupt Stack.
<25:24>	CUR	Current Mode.
<23:22>	PRV	Previous Mode.
21		Must Be Zero.
<20:16>	IPL	Interrupt Priority Level.
<15:08>		Must Be Zero.
07	DV	Decimal Overflow Trap Enable. This read/write bit has no effect on MicroVAX hardware. It can be used by macrocode that emulates VAX decimal instructions.
06	FU	Floating Underflow Fault Enable.
05	IV	Integer Overflow Trap Enable.
04	T	Trace Trap Enable.
03	N	Negative Condition Code.
02	Z	Zero Condition Code.
01	V	Overflow Condition Code.
00	C	Carry Condition Code.

* Note that compatibility mode instructions can be emulated by macrocode. Since the emulation software runs in native mode, the CM bit is never actually set.

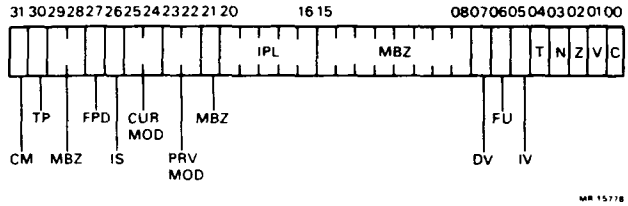


Figure 4-1 Processor Status Longword (PSL)

4.2.3 Processor Registers

The processor registers can be accessed through the MFPR and MTPR privileged instructions. Each of the processor registers listed in Table 4-2 falls into one of the following numbered categories.

1. VAX processor registers implemented as described in the VAX Architecture Reference Manual (EK-VAXAR-RM). These registers are implemented by the MicroVAX CPU chip.

2. VAX processor registers implemented external to the MicroVAX CPU chip by the KA630-AA logic.

3. Processor registers read as 0, no operation (NOP) on write.

4. Processor registers implemented by MicroVAX CPU chip uniquely (that is, registers not described in the VAX Architecture Reference Manual).

5. Processor register access not allowed. Attempted access results in reserved operand fault.

Table 4-2 Processor Register Summary

Number	Register Name	Mnemonic	Type	Category*
0	Kernel Stack Pointer	KSP	R/W	1
1	Executive Stack Pointer	ESP	R/W	1
2	Supervisor Stack Pointer	SSP	R/W	1
3	User Stack Pointer	USP	R/W	1
4	Interrupt Stack Pointer	ISP	R/W	1
5	Reserved			5
6	Reserved			5
7	Reserved			5
8	P0 Base Register	P0BR	R/W	1
9	P0 Length Register	P0LR	R/W	1
10	P1 Base Register	P1BR	R/W	1
11	P1 Length Register	P1LR	R/W	1
12	System Base Register	SBR	R/W	1
13	System Length Register	SLR	R/W	1
14	Reserved			5
15	Reserved			5
16	Process Control Block Base	PCBB	R/W	1
17	System Control Block Base	SCBB	R/W	1
18	Interrupt Priority Level	IPL	R/W	1R
19	AST Level	ASTLVL	R/W	1R
20	Software Interrupt Request	SIRR	W	1
21	Software Interrupt Summary	SISR	R/W	1R
22	Interprocessor Interrupt	IPIR	R/W	5
23	CMI Error Register	CMIERR	R/W	5
24	Interval Clock Control/Status	ICCS	R/W	4R
25	Next Interval Count Register	NICR	W	3
26	Interval Count Register	ICR	R	3
27	TOY Register	TODR	R/W	3
28	Console Storage Receiver Status	CSRS	R/W	3
29	Console Storage Receiver Data	CSRD	R	3
30	Console Storage Transmit Status	CSTS	R/W	3
31	Console Storage Transmit Data	CSTD	W	3
32	Console Receiver Control/Status	RXCS	R/W	2R
33	Console Receiver Data Buffer	RXDB	R	2R
34	Console Transmit Control/Status	TXCS	R/W	2R
35	Console Transmit Data Buffer	TXDB	W	2R
36	Translation Buffer Disable	TBDR	R/W	3
37	Cache Disable Register	CADR	R/W	3
38	Machine Check Error Summary	MCESR	R/W	3
39	Cache Error Register	CAER	R/W	3
40	Accelerator Control/Status	ACCS	R/W	5
41	Console Saved ISP	SAVISP	R/W	4
42	Console Saved PC	SAVPC	R/W	4
43	Console Saved PSL	SAVPSL	R/W	4
44	WCS Address	WCSA	R/W	5

Table 4-2 Processor Register Summary (Cont)

Number	Register Name	Mnemonic	Type	Category*
45	WCS Data	WCSB	R/W	5
46	Reserved			5
47	Reserved			5
48	SBI Fault/Status	SBIFS	R/W	3
49	SBI Silo	SBIS	R	3
50	SBI Silo Comparator	SBISC	R/W	3
51	SBI Maintenance	SBIMT	R/W	3
52	SBI Error Register	SBIER	R/W	3
53	SBI Timeout Address Register	SBITA	R	3
54	SBI Quadword Clear	SBIQC	W	3
55	I/O Bus Reset	IORESET	W	2
56	Memory Management Enable	MAPEN	R/W	1R
57	TB Invalidate All	TBIA	W	1
58	TB Invalidate Single	TBIS	W	1
59	Translation Buffer Data	TBDATA	R/W	3
60	Microprogram Break	MBRK	R/W	3
61	Performance Monitor Enable	PMR	R/W	3
62	System Identification	SID	R	1
63	Translation Buffer Check	TBCHK	W	1
64--127	Reserved			5

* An R following the category number indicates that the register is cleared by power-up and by the negation of DC OK.

4.3 INSTRUCTION SET

The MicroVAX CPU chip implements all instructions in the following VAX instruction groups.

- Integer arithmetic and logical
- Address
- Variable length bit field
- Control
- Procedure call
- Miscellaneous
- Queue
- Character string moves (MOVC3 and MOVC5)
- Operating system support

The MicroVAX CPU chip provides special microcode "hooks" to aid the emulation of the following instruction groups by macrocode.

- Character string moves (except MOVC3 and MOVC5)
- Decimal string
- CRC
- Edit

The following instruction groups are implemented by the MicroVAX FPU chip.

- F_floating
- G_floating
- D_floating

The following instruction groups are not implemented, but may be emulated by macrocode.

- H_floating
- Octaword
- Compatibility mode instructions

4.4 EXCEPTIONS AND INTERRUPTS

Both exceptions and interrupts divert execution from the normal flow of control. An exception is typically handled by the current process (for example, an arithmetic overflow), while an interrupt typically transfers control outside the process (for example, an interrupt from an external hardware device).

4.4.1 Interrupts

The MicroVAX architecture specifies 31 interrupt priority levels (IPLs), as follows.

IPL	Condition
Nonmaskable	HALT L asserted
1F	Unused
1E	PWRFL L asserted
19--1D	Unused
18	Unused
17	BIRQ7 L asserted
16	Interval timer interrupt, BIRQ6 L asserted
15	BIRQ5 L asserted
14	Console terminal interrupts, interprocessor doorbell, or BIRQ4 L asserted
10--13	Unused
01--0F	Software interrupt request

The Q22-Bus requests of levels 4 through 7 set IPL equal to 17, since the Q22-Bus has only one grant line. The single grant does not differentiate between the different request levels and grants the first requesting device it finds. The IPL is set to 14 after a console terminal or interprocessor doorbell interrupt. It is set to 16 after an interval timer interrupt.

When the KA630-AA is configured as an auxiliary CPU it ignores BIRQ7 through 4 interrupt requests, but does respond to IPL 14 requests from its own console SLU and from its interprocessor doorbell (in that order of priority). It also responds to interrupt requests from its own interval timer.

The interrupt system is controlled by the IPL register (IPL corresponds to PSL <20:16>.), the Software Interrupt Request Register (SIRR), and the Software Interrupt Summary Register (SISR), all shown in Figure 4-2.

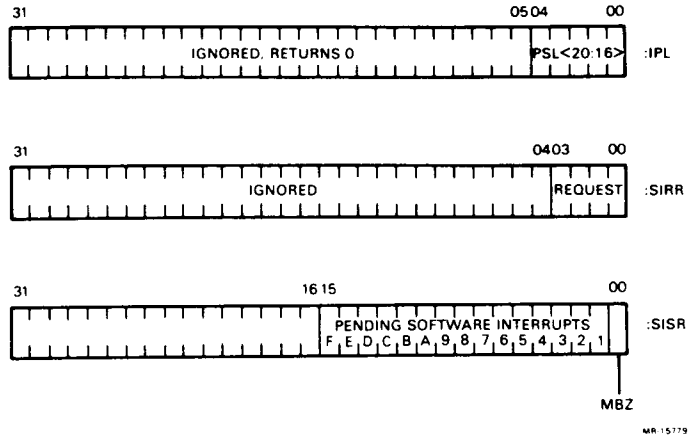


Figure 4-2 Interrupt Registers

4.4.2 Exceptions

The MicroVAX architecture recognizes six classes of exceptions, as follows.

Exception Class	Instances
Arithmetic trap/fault	Integer overflow trap Integer divide by zero trap Subscript range trap Floating overflow fault Floating divide by zero fault Floating underflow fault
Memory management	Access control violation fault Translation not valid fault
Operand reference	Reserved addressing mode fault Reserved operand fault or abort
Instruction execution	Reserved/privileged instr. fault Emulated instruction fault Extended function fault Breakpoint fault
Tracing	Trace trap
System failure	Memory read error abort Memory write error abort Kernel stack not valid abort Interrupt stack not valid abort Machine check abort

4.4.3 Machine Check Parameters

In response to a machine check, the parameters shown in Figure 4-3 are pushed onto the stack.

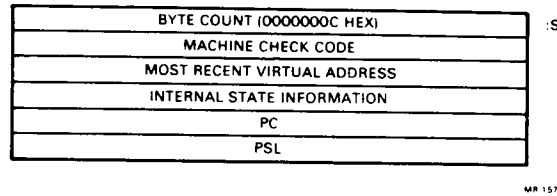


Figure 4-3 Machine Check Parameters

Machine check code (hex):

1 = Impossible microcode state (FSD)
2 = Impossible microcode state (SSD)
3 = Undefined FPU error code 0
4 = Undefined FPU error code 7
5 = Undefined memory management status (TB miss)
6 = Undefined memory management status (M = 0)
7 = Process PTE address in P0 space
8 = Process PTE address in P1 space
9 = Undefined interrupt ID code
80 = Read bus error, VAP is virtual address
81 = Read bus error, VAP is physical address
82 = Write bus error, VAP is virtual address
83 = Write bus error, VAP is physical address

Most recent virtual address:

<31:00> = Current contents of VAP register

Internal state information:

<28:24> = Current contents of ATDL register
<23:20> = Current contents of STATE <03:00>
<19:16> = Current contents of ALU condition codes
14 = Current contents of VAX restart bit
<07:00> = PC increment at the time of the exception (reported as zero if FPD set in saved PSL)

PC: <31:00> = PC at the start of the current instructions

PSL: <31:00> = Current contents of PSL

4.4.4 Halt Conditions

If the hardware or kernel software environment becomes severely corrupted, the chip may be unable to continue normal processing. In this case, the chip passes control to recovery code (the console program described in Chapter 3) beginning at physical address 20040000 (hex). The previous state of the machine is stored in temporary registers that are read as processor registers using the MFPR instruction. The previous state of the machine is as follows.

1. IPR 42 contains the saved PC.

2. IPR 43 contains the saved PSL, the saved memory management (MAPping) ENable bit (MAPEN) and the error code.

   a. IPR console.psl bits <31:16> and <07:00> contain the saved PSL.

   b. IPR console.psl bit 15 contains the saved MAPEN bit.

   c. IPR console.psl bits <14:08> contain the error code.

3. IPR 41 contains the previous interrupt stack pointer.

NOTE

There are severe restrictions on using these saved values. For example, they must be accessed before executing instructions that use the registers for temporary storage.

The halt process sets the state of the chip, as follows.

```
PSL    = 041F0000 (hex)
PC     = 20040000 (hex)
MAPEN  = 0
ASTLVL = Unchanged (set to 4 by power-up)
SISR   = Unchanged (cleared by power-up)
```

The error codes that indicate the reason for the halt are as follows.

Error Code	Condition
2	Assertion of external halt
3	Initial power-up
4	Interrupt stack not valid during exception
5	Machine check during machine check or kernel stack not valid exception
6	Halt instruction executed in kernel mode
7	SCB vector bits <01:00> = 11
8	SCB vector bits <01:00> = 10
A	CHMx executed while on interrupt stack
10	ACV or TNV during machine check exception
11	ACV or TNV during kernel stack not valid exception

4.4.5 System Control Block

The SCB consists of two pages that contain the vectors for servicing interrupts and exceptions. The SCB is pointed to by the SCBB (Figure 4-4). The KA630-AA uses SCB device vector 204 (hex) for the interprocessor doorbell interrupt. The SCB format is described in Table 4-3.

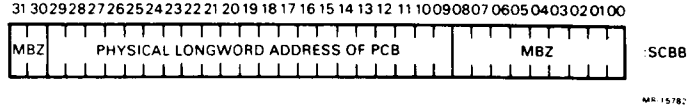


Figure 4-4 System Control Block Base Register (SCBB)

Table 4-3 System Control Block Format

Vector	Name	Type	Number of Parameters	Notes
00	Unused			
04	Machine Check	Abort		
08	Kernel Stack Not Valid	Abort	0	Serviced on interrupt stack, IPL is raised to 1F
0C	Power Fail	Interrupt	0	IPL is raised to 1E
10	Reserved/ Privileged Instruction	Fault	0	
14	Extended Instruction	Fault	0	XFC instruction
18	Reserved Operand	Fault/ Abort	0	Not always recoverable
1C	Reserved Addressing Mode	Fault	0	
20	Access Control Violation	Fault	2	Parameters are virtual address, status code

Table 4-3 System Control Block Format (Cont)

Vector	Name	Type	Number of Parameters	Notes
24	Translation Not Valid	Fault	2	Parameters are virtual address, status code
28	Trace Pending	Fault	0	
2C	Breakpoint Instruction	Fault	0	
30	Unused			Compatibility mode in VAX
34	Arithmetic	Trap/ Fault	1	Parameter is type code
38--3C	Unused			
40	CHMK	Trap	1	Parameter is operand word
44	CHME	Trap	1	Parameter is operand word
48	CHMS	Trap	1	Parameter is operand word
4C	CHMU	Trap	1	Parameter is operand word
50--5C	Unused			
60--80	Unused			
84	Software Level 1	Interrupt	0	
88	Software Level 2	Interrupt	0	Ordinarily used for AST delivery
8C	Software Level 3	Interrupt	0	Ordinarily used for process scheduling
90--BC	Software Levels 4 through 15	Interrupt	0	

Table 4-3 System Control Block Format (Cont)

Vector	Name	Type	Number of Parameters	Notes
C0	Interval Timer	Interrupt	0	IPL is 16 (INTTIM L)
C4	Unused			
C8	Emulation Start	Fault	10	Same mode exception; FPD = 0; parameters are opcode, PC, specifiers
CC	Emulation Continue	Fault	0	Same mode exception; FPD = 1; no parameters
D0--F4	Unused			
F8	Console Receive	Interrupt	0	IPL is 14
FC	Console Transmit	Interrupt	0	IPL is 14
100--1FC	Adapter Vectors	Interrupt	0	Not used by KA630-AA
200--3FC	Device Vectors	Interrupt	0	Correspond to bus vectors 000 -- 1FC; KA630-AA appends the assertion of bit 9

4.5 HARDWARE DETECTED ERRORS

The KA630-AA detects certain error conditions during program execution. These conditions and the resultant actions are described below.

4.5.1 Nonexistent Memory Errors

If the processor attempts a read or write to a nonexistent address in local memory or I/O space, then a nonexistent memory error occurs.

If the processor attempts a read or write (by asserting BDIN or BDOUT) to a device on the Q22-Bus, and if BRPLY is not asserted by that device within 10 µs, a bus timeout error occurs. This results in a machine check abort and trap through vector 4.

4.5.2 Parity Error Detection

Parity errors can be detected during read operations from local memory address space, Q22-Bus memory address space and Q22-Bus I/O page address space.

Memory System Error Register (MSER) bit 0 enables parity error detection for all reads from local memory, whether it is accessed through local memory address space or through the Q22-Bus memory address space (through the bus map). MSER bit 0 has no effect on parity error detection for reads from external Q22-Bus memory or Q22-Bus devices.

During read operations from the local memory address space, parity is checked only for those bytes designated by the processor as Byte Mask signals (BM, <03:00>). Because the MicroVAX chip must receive a stable ERRor signal (ERR) at least 150 ns before it requires stable data, performance considerations dictate that parity errors occurring during reads from local memory address space do not cause ERR assertion during the cycle for which the parity error was detected. Instead, the KA630-AA asserts ERR for the next cycle, and if that cycle was a prefetch read cycle, for the cycle after that as well.

When a parity error occurs during a local memory access through local memory address space, the processor is allowed to complete that cycle and may execute an instruction that alters the processor's internal state. However, the processor recognizes a machine check and traps through vector 4 when it attempts the next external cycle.

During read operations from Q22-Bus space (including the access of local memory through the Q22-Bus map), a parity error is detected if both BDAL bits 17 and 16 are asserted. ERR is then asserted.

When the processor reads local memory from the Q22-Bus memory space, parity is checked on both bytes of each word accessed, even if the processor only requested a single byte.

When ERR is asserted, the processor responds as follows.

- For nonprefetch reads, the processor recognizes a machine check and traps through vector 4.
- For prefetch operations, the processor aborts the prefetch cycle and performs a nonprefetch read if an instruction fetch is required from that location.

4.5.3 Interrupt Vector Timeouts

An interrupt vector timeout occurs when BRPLY L is not asserted by a device within 10 µs after an interrupt is acknowledged (BIAK L) by the processor. The ERR is asserted. The processor aborts the interrupt cycle and continues as though the interrupt request did not occur.

4.5.4 No Sack Timeouts

A No Sack timeout occurs when a device does not assert BSACK L within 10 µs after it has been granted bus mastership (received BDMG L). The KA630-AA continues as though the DMA request did not occur.

4.6 LATENCY

4.6.1 Interrupt Latency

Interrupt latency is defined as the time between receiving an interrupt request (BIRQ L) and acknowledging the request (BIAK L). Interrupt latency can be divided into the following three segments.

- The length of time the processor runs at an interrupt priority level that masks out the interrupt. This time period is highly software dependent.
- The length of time the processor takes to execute the last instruction after the interrupt.
- The length of time it takes the KA630-AA to gain bus mastership. Because the arbiter KA630-AA is the highest priority DMA device, this period is equal to the time required for the previous bus master to finish its data transfer(s) and relinquish the bus. (This represents a change from the traditional priority structure where DMA devices have a higher priority than either CPU fetches or interrupts.) Eight block mode transfers typically require about 5 µs. The KA630-AA asserts DMA when it needs the bus, limiting a block mode device to no more than eight additional transfers. A nonexistent memory timeout typically requires 10 to 15 µs.

4.6.2 DMA Latency

DMA latency is defined as the time between receiving a DMA request (BDMR L) and granting the request (BDMG L). This calculation is made assuming that the DMA request occurs while the CPU has control of the bus, and that there are no conflicting DMA requests. The result of this calculation is the CPU-induced latency. The DMA latency seen by any device is a combination of the CPU-induced latency and the latency induced by other DMA devices.

The CPU-induced DMA latency is the time required to complete the longest CPU operation that retains control of the bus. The longest KA630-AA operation that retains control of the bus is a 32-bit read-lock/write-unlock to non-block-mode memory.

When a CPU that is the lowest priority device relinquishes control of the bus, it does not regain control of the bus until all DMA requests have been honored. Thus, two high bandwidth devices could exchange control of the bus, effectively locking out the CPU until one of them has completed its set of transfers.

The arbiter KA630-AA CPU is the highest priority bus device in the system. After it has relinquished control of the bus, it can regain control of the bus during the next bus arbitration.

System level DMA latency calculations must take into account the fact that the arbiter KA630-AA can request the bus as the highest priority bus device.

4.7 SYSTEM IDENTIFICATION REGISTER (SID)

The read-only SID (Figure 4-5 and Table 4-4), processor register 62, is implemented by the MicroVAX CPU chip. On the KA630-AA, and on all other processors that use the MicroVAX CPU chip, the SID always reads 00080000.

The KA630-AA implements a 32-bit System Identification Extension register (SIE) at physical location 20040004. This 32-bit register exists within the KA630-AA console program ROM.

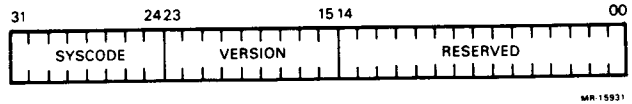


Figure 4-5 System Identification Register (SID)

Table 4-4 System Identification Register Format

Bits	Mnemonic	Name/Meaning
<31:24>	SYSCODE	System Code. This field reads as 1 for the KA630-AA.
<23:16>		Version number of console program ROM.
<15:00>		Reserved.

4.8 MEMORY MANAGEMENT

4.8.1 Physical and Virtual Address Space

The virtual address space is four gigabytes (2**32), as shown in Figure 4-6. The physical address space is one gigabyte (2**30), as shown in Figure 4-7.

4.8.2 Memory Management Control Registers

Memory management is controlled by three processor registers: MAPEN, Translation Buffer Invalidate Single (TBIS), and Translation Buffer Invalidate All (TBIA). MAPEN contains one bit, 0, as shown in Figure 4-8.

Translation buffer invalidation is controlled by TBIS (Figure 4-9) and TBIA (Figure 4-10). Writing a virtual address into TBIS invalidates any entry that maps that virtual address. Writing a 0 into TBIA invalidates the entire translation buffer.

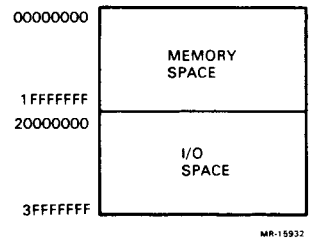


Figure 4-6 Virtual Address Space

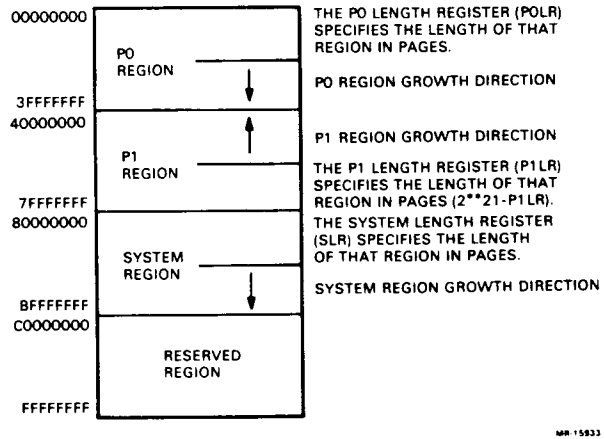


Figure 4-7 Physical Address Space

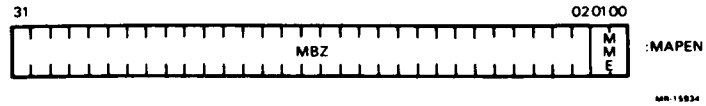


Figure 4-8 Memory Management (Mapping) Enable Register (MAPEN)

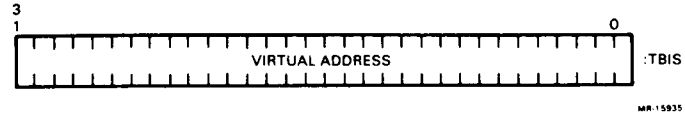


Figure 4-9 Translation Buffer Invalidate Single Register (TBIS)

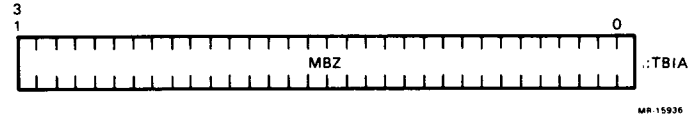


Figure 4-10 Translation Buffer Invalidate All Register (TBIA)

4.8.3 System Space Address Translation

A virtual address with bits <31:30> = 2 is an address in the system virtual address space. Refer to Figure 4-11.

System virtual address space is mapped by the System Page Table (SPT), which is defined by the System Base Register (SBR) and the System Length Register (SLR). The SBR contains the physical address of the SPT. The SLR contains the size of the SPT in longwords, that is, the number of Page Table Entries (PTEs). The PTE addressed by the SBR maps the first page of system virtual address space, that is, virtual byte address 80000000 (hex).

4.8.4 Process Space Address Translation

A virtual address with bit 31 = 0 is an address in the process virtual address space. Process space is divided into two equally sized, separately mapped regions. If virtual address bit 30 = 0, the address is in region P0. If virtual address bit 30 = 1, the address is in region P1.

4.8.4.1 P0 Region Address Translation -- Refer to Figure 4-12. The P0 region of the address space is mapped by the P0 Page Table (P0PT), which is defined by the P0 Base Register (P0BR) and the P0 Length Register (P0LR). The P0BR contains the system virtual address of the P0PT. The P0LR contains the size of the P0PT in longwords, that is, the number of PTEs. The PTE addressed by the P0BR maps the first page of the P0 region of the virtual address space, that is, virtual byte address 0.

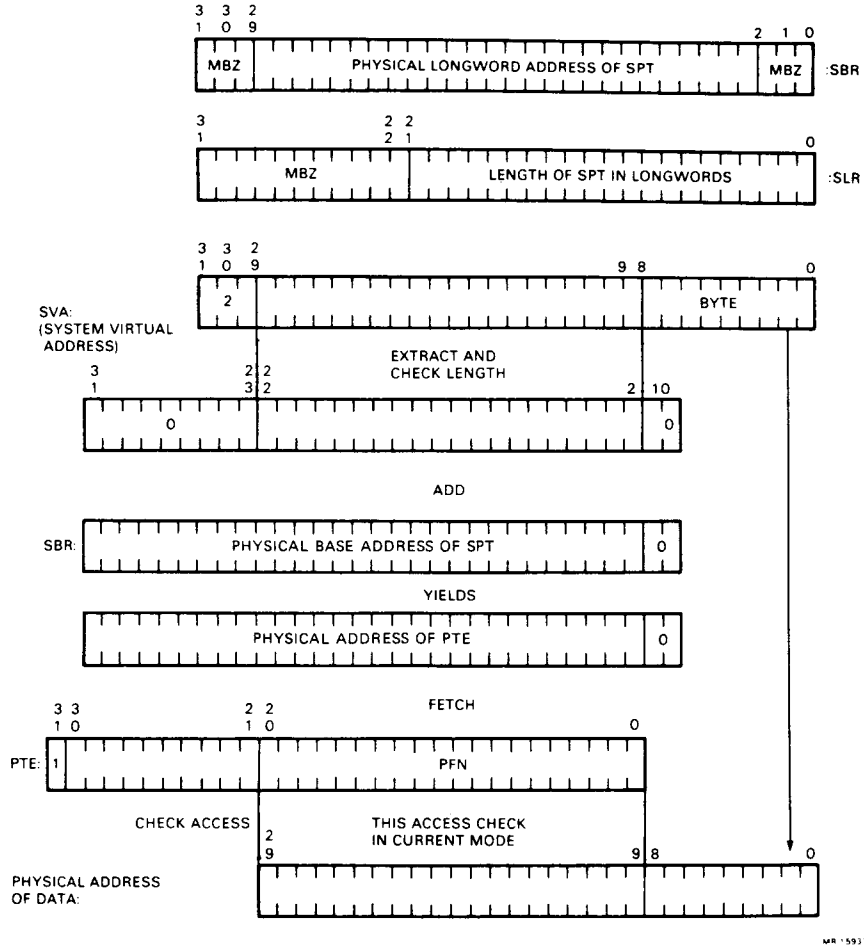


Figure 4-11 System Space Virtual to Physical Address Translation

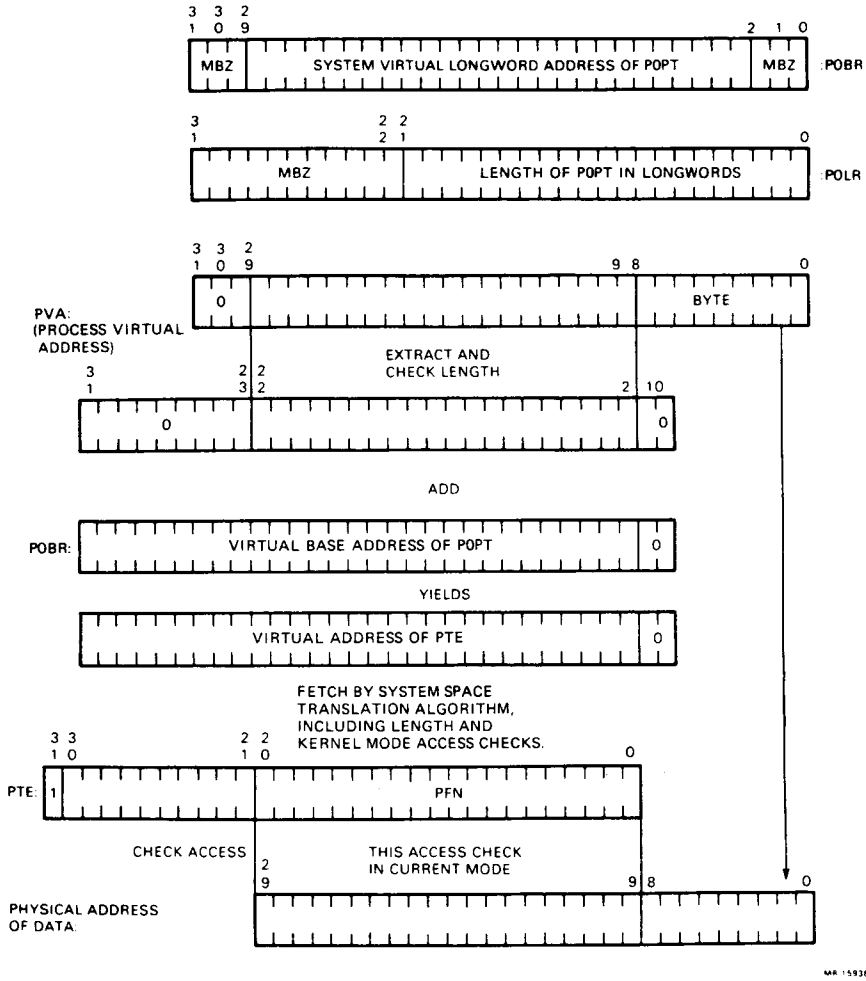


Figure 4-12 P0 Virtual to Physical Address Translation

4.8.4.2 P1 Region Address Translation -- Refer to Figure 4-13. The P1 region of the address space is mapped by the P1 Page Table (P1PT), which is defined by the P1 Base Register (P1BR) and the P1 Length Register (P1LR). Because P1 space grows toward smaller addresses, and because a consistent hardware interpretation of the base and length registers is desirable, P1BR and P1LR describe the portion of P1 space that is not accessible. Note that P1LR contains the number of nonexistent PTEs. P1BR contains the virtual address of what would be the PTE for the first page of P1, that is, virtual byte address 40000000 (hex). The address in P1BR is not necessarily a valid physical address, but all the addresses of PTEs must be valid physical addresses.

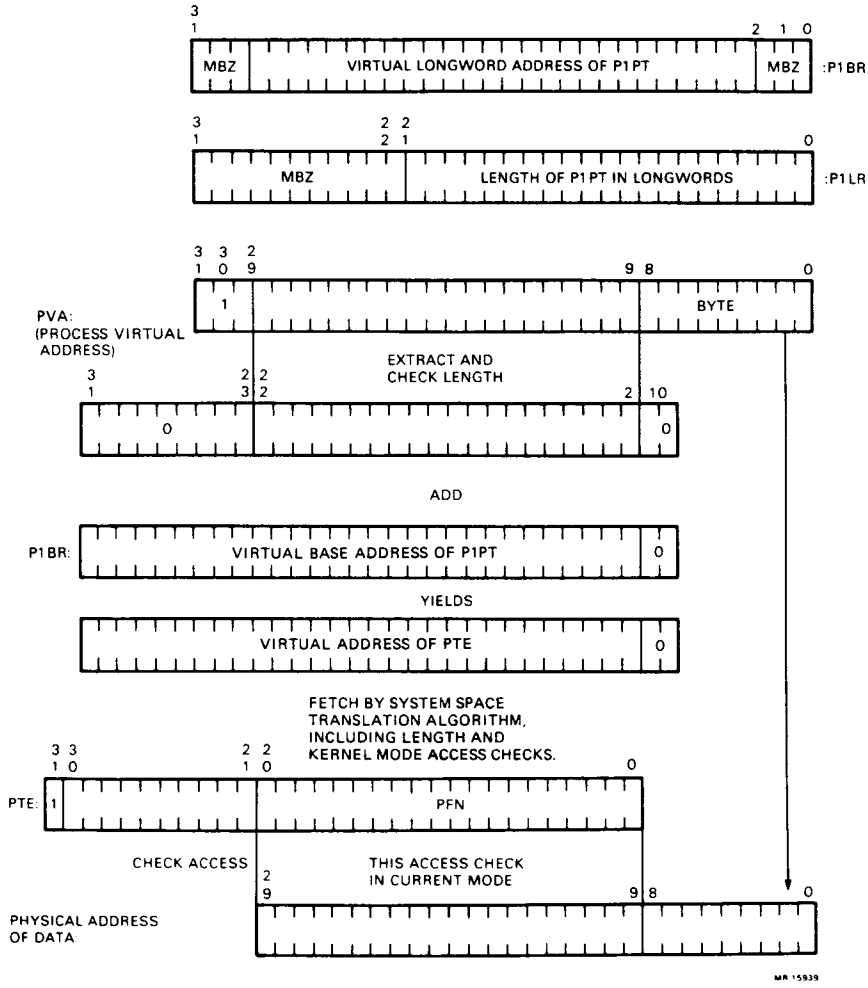


Figure 4-13 P1 Virtual to Physical Address Translation

4.8.5 Page Table Entry

The format of a valid PTE is shown in Figure 4-14, where:

V = Valid bit (must be set)
PROT = Protection code
M = Modify bit
OWN = Owner bits
PFN = Page frame number.

If bit 31 (the V bit) is clear, the format of the remaining bits is not examined by the hardware.

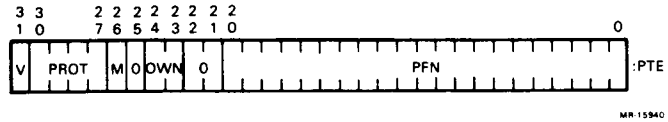


Figure 4-14 Page Table Entry (PTE)

4.9 KA630-AA MEMORY SYSTEM

The KA630-AA memory system consists of the local memory and a Q22-Bus map that allows Q22-Bus master devices to access this local memory. The memory system also includes two registers that are used primarily for diagnostic purposes. The KA630-AA supports up to 9 Mbytes of local memory. The KA630-AA typically accesses this memory directly, through physical addresses 00000000 to 00FFFFFF. Any Q22-Bus master device, including the KA630-AA, can access this memory indirectly through the Q22-Bus map. Mapping can be independently enabled and disabled for each page.

The KA630-AA accesses the Q22-Bus memory address space through physical addresses 30000000 to 303FFFFF. It accesses the Q22-Bus I/O space through physical addresses 20000000 to 20001FFF.

4.9.1 Local Memory Mapping Register Format

The Q22-Bus map contains 8192 mapping registers. Each of these mapping registers is a 32-bit longword with the format shown in Figure 4-15 and Table 4-5, and each one can map a page (512 bytes) of Q22-Bus space into a selected page of local memory.

Each mapping register is located on a longword boundary and must be written using longword instructions. Byte and word instructions can only load these registers with undefined data.

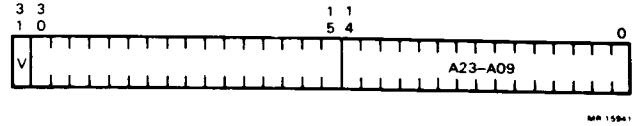


Figure 4-15 Mapping Register

Table 4-5 Memory Register Format

Bit(s)	Mnemonic	Name/Meaning
31	V	Valid. When a mapping register is selected by a Q22-Bus address, the valid bit determines whether the Q22-Bus map is enabled for that address. If the valid bit is set, the map is enabled. If the valid bit is clear, the map is disabled and the KA630-AA does not respond to that address.
<30:15>		Unused. These bits always read as 0.
<14:00>	A23--A09	Address bits <23:09>. When a mapping register is selected by a Q22-Bus address, and if that register's valid bit is set, then these 15 bits are used as local memory address bits 23 through 9. Q22-Bus address bits 8 through 0 are used as local memory address bits 8 through 0.

4.9.2 Mapping Register Addresses (2008XXXX Hex)

The mapping registers (Table 4-6) are located within the local register space at physical addresses 20088000 through 2008FFFC. They can only be accessed from the processor.

The physical address of each register was chosen so that register address bits <14:02> are identical to Q22-Bus address bits <2]:09> of the page they map.

Table 4-6 Mapping Register Addresses

Register Address	Q22-Bus Addresses Mapped (Hex)	Q22-Bus Addresses Mapped (Octal)
20088000	000000--0001FF	00000000--00000777
20088004	000200--0003FF	00001000--00001777
20088008	000400--0005FF	00002000--00002777
2008800C	000600--0007FF	00003000--00003777
20088010	000800--0009FF	00004000--00004777
20088014	000A00--000BFF	00005000--00005777
20088018	000C00--000DFF	00006000--00006777
2008801C	000E00--000FFF	00007000--00007777
.	.	.
.	.	.
.	.	.
.	.	.
2008FFF0	3FF800--3FF9FF	17774000--17774777
2008FFF4	3FFA00--3FFBFF	17775000--17775777
2008FFF8	3FFC00--3FFDFF	17776000--17776777
2008FFFC	3FFE00--3FFFFF	17776000--17777777

4.9.3 Q22-Bus Map Operation

At power-up, the Q22-Bus mapping registers, including the valid bits, are undefined. External access to local memory is disabled as long as the IPCR LM EAE bit is cleared.

After completing the ROM diagnostics that are part of its console program, an arbiter KA630-AA enables the mapping registers to map sufficient local memory space to boot the system, and then sets the LM EAE bit. When the operating system gains control, it may either invalidate or reassign various pages, as required.

Upon completion, the ROM programs in an auxiliary KA630-AA clear all mapping register valid bits, and then set the LM EAE bit.

The Q22-Bus map monitors each Q22-Bus cycle and responds if the following three conditions are met:

1. The IPCR LM EAE bit is set.
2. The valid bit of the selected mapping register is set.
3. For read operations, the mapping register must have mapped into existent local memory. (During write operations, the response depends only on conditions 1 and 2 because the KA630-AA returns Q22-Bus BRPLY before checking for existent local memory.)

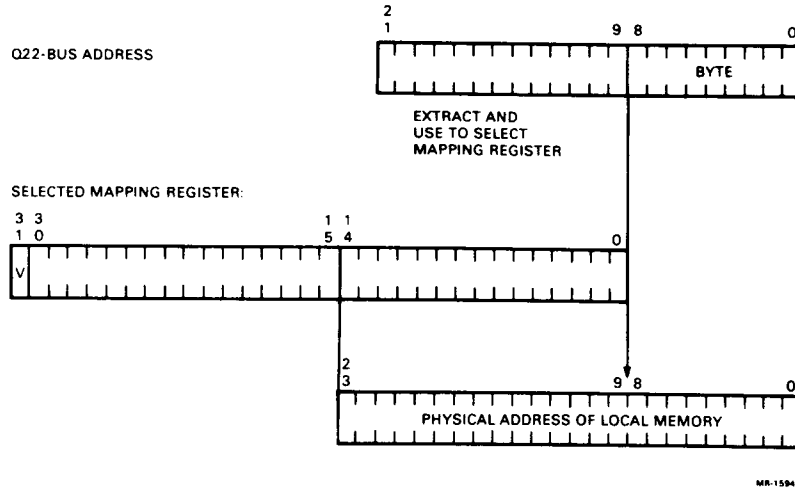


Figure 4-16 Q22-Bus to Local Memory Physical Address Translation

The translation from Q22-Bus address to local memory physical address is shown in Figure 4-16.

4.9.4 Memory System Registers

The three registers associated with the memory system are located in the local register I/O address space and can only be accessed by the on-board processor. Software uses the MSER to monitor parity and nonexistent memory errors, and to control parity generation and checking for the local memory. The CPU Error Address Register (CEAR) contains the address of the page in local memory that caused a parity error during an access by the on-board CPU. The DMA Error Address Register (DEAR) contains the address of the page in local memory that caused a parity error during an access by an external device.

4.9.4.1 Memory System Error Register (20080004 Hex) -- The MSER (Figure 4-17, Table 4-7) is located in the local register I/O address space at physical address 20080004. It can only be accessed by the on-board processor.

MSER bits <07:05> and 3 indicate the status of machine check traps through SCB vector 4. MSER bit 4 is set if an external Q22-Bus device receives a parity error while reading KA630-AA local memory.

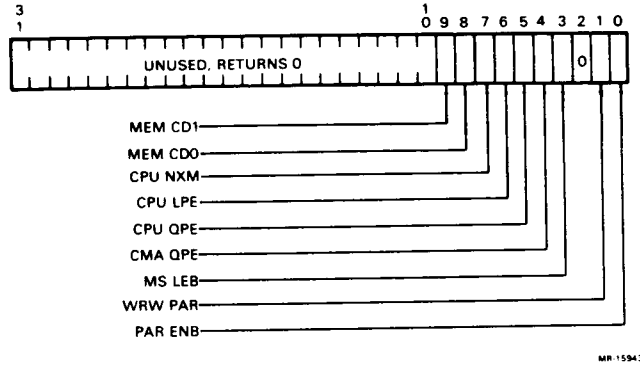


Figure 4-17 Memory System Error Register (MSER)

Table 4-7 Memory System Error Register Format

Bit(s)	Mnemonic	Name/Meaning
<31:10>		Unused. Reads as 0.
09 08	MEM CD1 MEM CD0	Memory Code <01:00>. When one of the two CPU parity error bits (MSER <06:05>) is set, the two read-only MEM CD bits are loaded with a 2-bit code indicating the source of the parity error, as follows. MEM CD <09:08> Source 00 Q22-Bus memory or device 01 KA630-AA on-board memory 10 Memory expansion module 1 11 Memory expansion module 2 A second parity error does not update this code unless software has cleared the CPU parity error bits. MEM CD1 and MEM CD0 are cleared by power-up, by the negation of DC OK, and by writes to the BIR.
07	CPU NXM	CPU Nonexistent Memory Error. This bit is set by any CPU nonprefetch read or write operation that references nonexistent memory, causing a trap through SCB vector 4. Writing a 1 to this bit clears it; writing a 0 to this bit has no effect. CPU NXM is cleared by power-up, by the negation of DC OK, and by writes to the BIR.

Table 4-7 Memory System Error Register Format (Cont)

Bit(s)	Mnemonic	Name/Meaning
06	CPU LPE	CPU Local Address Space Parity Error. If parity error detection is enabled (MSER bit 0 set), then CPU LPE is set by any CPU read access (prefetch or nonprefetch) to local memory address space that causes a parity error. The processor does not receive an error indication on that cycle. The next MicroVAX cycle is aborted, causing a trap through SCB vector 4. Writing a 1 to this bit clears it; writing a 0 has no effect. CPU LPE is cleared by power-up, by the negation of DC OK, and by writes to the BIR. Only those memory bytes selected by processor outputs BM <03:00> can cause a CPU LPE parity error. Because the fetch that caused the parity error is not aborted, it could be difficult for software to determine the result of the error. For this reason, parity errors that set this bit are generally treated as fatal errors.
05	CPU QPE	CPU Q22-Bus Address Space Parity Error. CPU QPE is set by any CPU nonprefetch read access to the Q22-Bus address space that results in a parity error causing a CPU trap through SCB vector 4. If the CPU is accessing local memory through the Q22-Bus map, parity detection is enabled only if MSER bit 0 is set. If the CPU is accessing the Q22-Bus, parity detection is enabled or disabled at the external Q22-Bus memory or device. Writing a 1 to this bit clears it; writing a 0 to this bit has no effect. CPU QPE is cleared by power-up, by the negation of DC OK, and by writes to the BIR.
04	DMA QPE	DMA Q22-Bus Address Space Parity Error. If parity error detection is enabled (MSER bit 0 set), then DMA QPE is set by any external read access to KA630-AA local memory that results in a parity error. This type of parity error does not cause the CPU trap through SCB vector 4. (The DMA device typically interrupts with an error indication.) Writing a 1 to this bit clears it; writing a 0 to this bit has no effect. DMA QPE is cleared by power-up, by the negation of DC OK, and by writes to the BIR.

Table 4-7 Memory System Error Register Format (Cont)

Bit(s)	Mnemonic	Name/Meaning
03	MS LEB	Memory System Lost Error Bit. This bit is set by an operation that sets MSER bit 6 or bit 5 after one or both of those bits has already been set. Writing a 1 to this bit clears it; writing a 0 to this bit has no effect. MS LEB is cleared by power-up, by the negation of DC OK, and by writes to the BIR.
02		Unused. Read as 0s.
01	WRW PAR	Write Wrong Parity. If this read/write bit is set, and either the CPU or a DMA device writes to local memory, then wrong parity is written into the parity bits of the RAMs. If this bit is clear, correct parity is written. WRW PAR is cleared by power-up, by the negation of DC OK, and by writes to the BIR.
00	PAR ENB	Parity Enable. If this read/write bit is set, local memory parity error detection is enabled. If this bit is clear, parity errors are ignored during all CPU and DMA reads from local memory. PAR ENB is cleared by power-up, by the negation of DC OK, and by writes to the BIR. PAR ENB controls parity detection for all CPU reads from local memory, including accesses through the Q22-Bus map. PAR ENB has no effect on CPU reads from external Q22-Bus memory.

4.9.4.2 CPU Error Address Register -- The CEAR (Figure 4-18) is located in the local register I/O address space at physical address 20080008. It can only be accessed by the on-board processor and contains valid information only when either MSER bit 6 (CPU LPE) or MSER bit 5 (CPU QPE) is set.

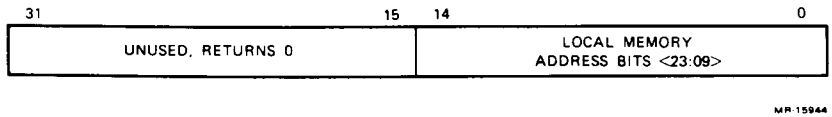


Figure 4-18 CPU Error Address Register (CEAR)

The CEAR contains the address of the page in local memory that caused a parity error during an access by the on-board CPU. The contents of this register are latched when either MSER bit 6 or MSER bit 5 is set. Additional local memory parity errors have no effect on the CEAR until software clears MSER <06:05>.

Local memory address bits <23:09> are loaded into CEAR bits <14:00>. CEAR bits <31:15> always read as 0.

4.9.4.3 DMA Error Address Register -- The DEAR (Figure 4-19) is located in the local register I/O address space at physical address 2008000C. It can only be accessed by the on-board processor and contains valid information only when MSER bit 4 (DMA QPE) is set.

The DEAR contains the address of the page in local memory that caused a parity error during an access by an external device. The contents of this register are latched when MSER bit 4 is set. Additional local memory parity errors have no effect on the DEAR until software clears MSER bit 4.

Local memory address bits <23:09> are loaded into DEAR bits <14:00>. DEAR bits <31:15> always read as 0.

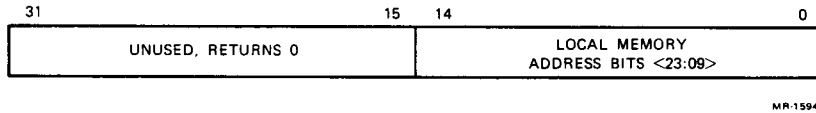


Figure 4-19 DMA Error Address Register (DEAR)

4.9.5 Memory System Operation

The KA630-AA memory system can perform the following data transfer and memory refresh cycles.

- MicroVAX accesses local memory directly.
- MicroVAX accesses local memory through Q22-Bus map.
- MicroVAX accesses on-board registers in the local or Q22-Bus I/O address space.
- MicroVAX accesses Q22-Bus memory or registers.
- External Q22-Bus device accesses local memory through Q22-Bus map.
- External Q22-Bus device accesses the KA630-AA IPCR in the Q22-Bus I/O space.
- KA630-AA performs memory refresh cycle.

NOTE

The KA630-AA does not require access to the memory system when it accesses internal processor registers (including those implemented external to the MicroVAX CPU chip).

Access to the local memory data/address paths and control of the Q22-Bus are arbitrated independently. An external device can gain control of the Q22-Bus and access a KA630-AA register (in the Q22-Bus address space) while the MicroVAX CPU chip is accessing local memory. When an external device accesses KA630-AA local memory, MicroVAX cycles are not interrupted until the Q22-Bus map has decoded the local memory address.

When the local memory is between cycles, it continually decodes the address on the MicroVAX address/data lines. When it receives a refresh or external device request, it switches off the MicroVAX address/data lines and monitors the address lines from the refresh logic or from the Q22-Bus map.

When the local memory is between cycles, the arbitrator for the local memory responds to requests in the following order of priority:

1. MicroVAX request
2. External device request
3. Refresh request

When the local memory is completing a cycle, the arbitrator for the local memory responds to requests in the following order of priority:

1. External device request
2. Refresh request
3. MicroVAX request

The local memory cycle time is 400 ns for all read, write or refresh cycles.

The MicroVAX must have control of the Q22-Bus before it can carry out the following actions.

- Perform any read-lock cycle
- Access local memory through the Q22-Bus map
- Access the IPCR or one of the Q22-Bus map registers
- Perform Q22-Bus cycles

On an arbiter KA630-AA, which contains the Q22-Bus arbitrator, the MicroVAX has control of the Q22-Bus except when the KA630-AA has granted an external DMA request. An auxiliary KA630-AA can gain control of the Q22-Bus only by posting a DMA request.

The MicroVAX accesses Q22-Bus memory and registers by using the following Q22-Bus cycles.

- DATa-Out-Byte (DATOB) for 8-bit writes
- DATa-In (DATI) for 16-bit reads (nonlocked)
- DATa-Out (DATO) for 16-bit writes
- DATa Block In (DATBI) for 32-bit reads (nonlocked)
- DATa Block Out (DATBO) for 32-bit writes
- DATI followed by DATO for a 16-bit read-lock followed by a 16-bit write-unlock
- DATBI followed by DATBO for a 32-bit read-lock followed by a 32-bit write-unlock

When performing 32-bit reads from non-block-mode memory, two successive DATI cycles are substituted for the DATBI cycle. When performing 32-bit writes to non-block-mode memory, the two successive DATO cycles are substitued for the DATBO cycle. The same substitutions are made for the 32-bit read-lock followed by a 32-bit write-unlock. In all three cases, the KA630-AA retains control of the bus between successive DATI and/or DATO cycles.

When the processor reads a byte from the Q22-Bus, the KA630-AA performs a DATI cycle with address bit 0 correctly reflecting the byte address.

4.10 KA630-AA BOOT AND DIAGNOSTIC FACILITY

The KA630-AA boot and diagnostic facility features one 16-bit register and two 28-pin ROM sockets for 16, 32 or 64 Kbytes of read-only memory. The ROM memory is located on consecutive word boundaries and may be accessed through longword, word or byte references.

The KA630-AA populates each of the two ROM sockets with 32 K X 8 bytes of ROM (or EPROM). These two ROMs contain the 64 Kbyte console program. If these ROMs are replaced for special applications, the new ROM must contain some version of the console program.

4.10.1 Boot and Diagnostic Register

The 16-bit BDR (Table 4-8, Figure 4-20) is located at physical address 20080000. It can be accessed by KA630-AA software, but not by external Q22-Bus devices. The BDR allows the boot and diagnostic ROM programs to read various KA630-AA configuration bits and to load the 4-bit error display.

Table 4-8 Boot and Diagnostic Register Format

Bit(s)	Mnemonic	Name/Meaning
15	PWR OK	Power OK. This read-only bit is set if the Q22-Bus BP OK signal is asserted and clear if BP OK is negated.
14	HLT ENB	Halt Enable. This read-only bit reflects the status of external connector pin 15. The set condition of this signal enables the various external halts. Also, following the execution of a halt instruction in kernel mode, the KA630-AA ROM program reads the HLT ENB bit to decide whether to enter the console program (HLT ENB set) or to restart the operating system (HLT ENB clear).
<13:12>		Unused. Always reads as 0.
11 10	CPU CD1 CPU CD0	CPU Code <01:00>. These two read-only bits originate from connector pins 4 and 5. They indicate whether the KA630-AA is configured as the arbiter or as one of the three auxiliaries, as follows. CPU CD <11:10> Configuration 00 Arbiter 01 Auxiliary 1 10 Auxiliary 2 11 Auxiliary 3
09 08	BDG CD1 BDG CD0	Boot and Diagnostic Code <01:00>. This 2-bit read-only code reflects the status of configuration and display connector pins <14:13>.
<07:04>		Unused. Always reads as 0.
03 02 01 00	DSPL 03 DSPL 02 DSPL 01 DSPL 00	Display <03:00>. These four write-only bits update an external LED display. Writing a 1 to a bit lights the corresponding LED; writing a 0 to a bit turns its LED off. The display bits are set (all LEDs are lit) by power-up and by the negation of DC OK.

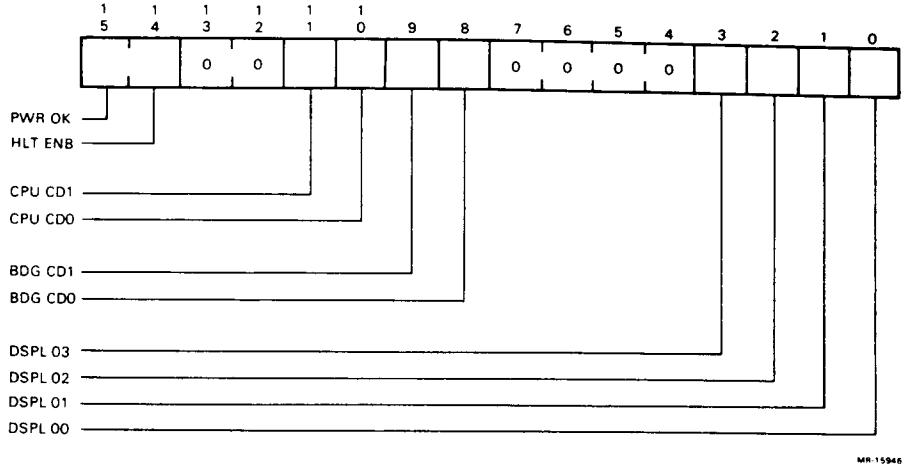


Figure 4-20 Boot and Diagnostic Register (BDR)

4.10.2 ROM Memory

4.10.2.1 ROM Sockets -- The two ROM sockets are compatible with 8, 16 and 32 K X 8 byte ROMs. A machine-inserted jumper selects the pin 27 input to be either +5 V (for 8 and 16 K parts) or ROM address bit 14 (for 32 K parts).

The KA630-AA is shipped with 32 K X 8 byte ROMs, and with the jumper in the address bit 14 position.

4.10.2.2 ROM Address Space -- The entire boot and diagnostic ROM can be read from either the 64 Kbyte halt mode ROM space or the 64 Kbyte run mode ROM space. Writes to either of these address spaces result in a nonexistent memory trap.

Any I-Stream read from the halt mode ROM space places the KA630-AA in halt mode. The front panel Run light is off and the halt input to the MicroVAX CPU chip is disabled.

Any I-Stream read that does not access the halt mode ROM space, including reads from the run mode ROM space, places the KA630-AA in run mode. The front panel Run light is lit and the halt input to the MicroVAX CPU chip is reenabled.

Writes and D-Stream reads to any address space have no effect on run mode/halt mode status.

I-Stream reads include all instruction fetches (except when the MicroVAX CPU chip retries a fetch following a nonexistent memory or parity error) and certain character string data fetches (again, except for those retries that follow an error). All reads that are not I-Stream reads are D-Stream reads. When running in halt mode, the ROM programs cannot use character string instructions to fetch data from outside the halt mode ROM address space.

When in halt mode, the KA630-AA always responds to the full 64 Kbyte halt mode ROM space (hex addresses 20040000 through 2004FFFF). When the KA630-AA contains 16 Kbytes of ROM memory, it appears four times, once within each 16 Kbytes of halt mode ROM space. When the KA630-AA contains 32 Kbytes of ROM memory, it appears twice, once within each 32 Kbytes of ROM space.

When in run mode, the KA630-AA always responds to the full 64 Kbyte run mode ROM space (hex addresses 20050000 through 2005FFFF). When the KA630-AA contains 16 Kbytes of ROM memory, it appears four times within the run mode ROM space. When the KA630-AA contains 32 Kbytes of ROM memory, it appears two times within the ROM memory space. Note that the run mode ROM space accesses the same ROM code as the halt mode ROM space.

4.10.2.3 KA630-AA Console Program Operation -- The console program is entered by transferring program control to location 20040000. There are various halt conditions that cause the MicroVAX to transfer program control to location 20040000. These conditions include the kernel mode halt instruction, assertion of the external halt input to the MicroVAX CPU chip, and certain fatal machine checks. When DC OK has been negated, either at power-up or by reboot, the combined assertion of DC OK and P OK initiates program execution at location 20040000.

The KA630-AA console program provides the following services.

- Automatic restart or bootstrap following processor halts or initial power-up
- Interactive command language allowing the user to examine and alter the state of the processor
- Diagnostic tests executed on power-up that check the CPU, the memory system and the Q22-Bus map
- Support of video or hard-copy console terminals, as well as support of VCB01-based bitmapped terminals

The KA630-AA console program is described in detail in Chapter 3.

4.11 KA630-AA TOY CLOCK

4.11.1 Battery Backed-Up Watch Chip

The KA630-AA contains a Motorola* MC146818 CMOS watch chip and battery backup circuitry that are connected to three batteries mounted on the CPU distribution panel through KA630-AA connector J2. The battery backup for this chip is specified to be greater than 240 hours when using three nickel-cadmium batteries in series.

The operating system software must fetch the correct time from this chip whenever power is restored to the system. If the power was Off long enough for the battery voltage to go below specification, or if the battery was temporarily disconnected while the system power was Off, the time in the watch chip is undefined. If the operating system detects a cleared Valid RAM and Time bit (VRT, in watch chip register CSR D), it must prompt the system operator for the time, and then load this time into the watch chip.

Although the MicroVAX interval timer interrupts have a resolution of 10 ms, the watch chip only has a resolution of seconds. Therefore, the time resolution is as follows.

- While under full power supply -- 10 ms
- After power-down for less than 240 hours (while watch chip is powered by battery) -- 1 second

4.11.2 Watch Chip Registers

The watch chip contains 64 8-bit registers (Table 4-9). Ten of these registers contain time of day data and 4 are CSRs. The remaining 50 provide 50 bytes of battery backed-up RAM. They are addressed from a base address of 200B8000, as described in the following sections.

Even though the addressing is on word boundaries, the TOY data and RAM locations are loaded into or read out of the chip a byte at a time.

4.11.2.1 TOY Data Registers -- Software reads the TOY data registers (Table 4-10) only after reading a cleared Update In Progress bit (UIP, CSR A bit 7) and only when all interrupts are disabled. (This assures that reading of the registers is not delayed beyond the time for which they are valid.) When the UIP bit is clear, the contents of these registers is guaranteed to be stable for at least 244 µs.

Software loads the TOY data registers only after setting the SET bit (SET, CSR B bit 7). After loading the correct time and date into the TOY data registers, software loads 20 (hex) into CSR A and then clears SET by loading 6 into CSR B.

---

* Motorola is a registered trademark of Motorola, Inc.

Table 4-9 Watch Chip Registers

Number	Function	Address Offset From Base Address	Comments
0	Seconds	00	Used on reads only
1	Second alarm	02	Not used
2	Minutes	04	Loaded and read
3	Minute alarm	06	Not used
4	Hours	08	Loaded and read
5	Hour alarm	0A	Not used
6	Day of week	0C	Not used
7	Date of month	0E	Loaded and read
8	Month	10	Loaded and read
9	Year	12	Loaded to produce 28th or 29th day Feb. (not read by VMS)
10	CSR A	14	Loaded and read
11	CSR B	16	Loaded and read
12	CSR C	18	Not used
13	CSR D	1A	Read-only
14	1st byte of RAM	1C	Uses assigned by the ROM code
	.		
	.		
63	50th byte of RAM	7E	

Table 4-10 Time-of-Year Data Register Addresses

Address	Units	Decimal Range	Hexadecimal Range
200B8000	Seconds	0--59	00--3B
200B8004	Minutes	0--59	00--3B
200B8008	Hours	0--23	00--17
200B800E	Day of month	1--31	01--1F
200B8010	Month	1--12	01--0C
200B8012	Year	0--99	00--63

4.11.2.2 Control and Status Register A -- CSR A (Figure 4-21) contains the UIP, the divider selection bits (DV <02:00>), and the rate selection bits (RS <03:00>).

The UIP is a read-only bit that is set when there is an update in progress within the chip. This bit must be read prior to reading the time. If the UIP is a 1, an update is in progress and the time registers are undefined. If the UIP is a 0, there are at least 244 µs available prior to the next update cycle. If interrupts are disabled, the time required to read the 5 time registers does not exceed 40 µs.

CSR A is undefined after battery power has been lost. Whenever the operating system software loads the TOY data registers, it must also load 20 (hex) into CSR A. Setting DV <02:00> = 2 sets up the timer for operation with the 32.768 kHz oscillator. Setting RS <03:00> = 0 disables the unused interrupt and square wave outputs from the chip.

CSR A is not affected by the chip going into or out of the normal BBU mode, as long as the battery voltage remains within specification.

4.11.2.3 Control and Status Register B -- CSR B (Figure 4-22) contains four bits that enable functions not used in the KA630-AA design, SET, and three bits that control timer format and operation.

SET is a read-write bit used to enable and disable clock operation. When written with a 0, the internal time updates occur every second. When written with a 1, the updates are disabled so that the program may load the TOY data registers. This bit must be set prior to setting the time. If the chip is in the middle of an update, setting this bit aborts the update.

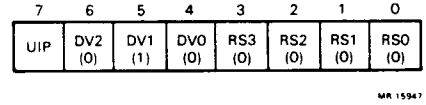


Figure 4-21 Control and Status Register A (CSR A)

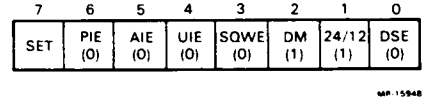


Figure 4-22 Control and Status Register B (CSR B)

Periodic Interrupt Enable (PIE), Alarm Interrupt Enable (AIE), Update ended Interrupt Enable (UIE) and SQuare-Wave Enable (SQWE) are not used.

Data Mode (DM) is a read-write bit that controls whether the time and date registers use binary or Binary Coded Decimal (BCD) formats. This bit is loaded with a 1 to select binary format.

24/12 is a read-write bit that controls whether the hour register operates in 24- or 12-hour mode. This bit is loaded with a 1 to select 24-hour mode.

Daylight Saving Enable (DSE) is a read-write bit that enables or disables special daylight saving time changes for the last Sunday in April and the last Sunday in October. This bit is loaded with a 0 to disable this function.

CSR B is undefined after battery power has been lost. Whenever the operating system software loads the TOY data registers, it must restart the timer by loading 6 (hex) into CSR B. Loading 6 into CSR B clears SET, and correctly loads the DM, 24/12 and DSE bits.

CSR B is not affected by the chip going into or out of the normal BBU mode, as long as the battery voltage remains within specification.

4.11.2.4 Control and Status Register C -- This register is not used.

4.11.2.5 Control and Status Register D -- CSR D (Figure 4-23) is a read-only register that contains the VRT. The remaining seven bits always read as 0s.

The VRT is read by software, before reading the time registers, to verify the validity of the time. If the battery voltage goes below specification while in BBU mode, this bit is reset to 0 by the hardware sensing circuitry during power-up, indicating that the time registers are undefined. If the VRT is 0, the time registers must be updated immediately. VRT is automatically set to 1 when CSR D is read, indicating that the chip contains a valid time setting.

If battery voltages are removed and then restored during power-down, KA630-AA logic guarantees that VRT = 0.

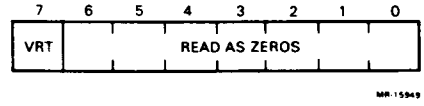


Figure 4-23 Control and Status Register D (CSR D)

4.11.2.6 RAM Memory -- The 50 bytes of RAM memory are used by the KA630-AA console program to store information required to restart the machine following a halt. Halts transfer program control to location 20040000 (hex). One of these RAM locations -- CPMBX, is used for communication between the operating system and the console program.

The CPMBX (Figure 4-24, Table 4-11) contains the console message text language, "Restart in Progress" and "Bootstrap in Progress" flags, and the processor halt action. The CPMBX is TOY register 14. Its address is 200B801C.

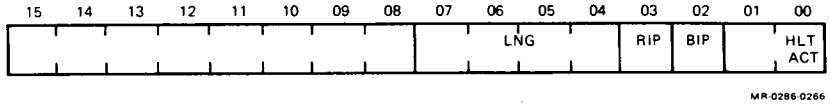


Figure 4-24 Console Program Mailbox (CPMBX)

4.11.3 Power-Up
Following a power-up, the KA630-AA console program reads the VRT bit in CSR D. If this bit is set, the RAM and time data are valid. If this bit is clear, RAM and time data are invalid, and the console program disables the clock by setting CSR B SET.

When the operating system gains control of the machine, it checks CSR B SET. If that bit is set, the operating system must request the correct time of year from the operator.

4.11.3.1 Valid RAM and Time -- If the VRT is set, RAM and time data are valid. The operating system reads the UIP in CSR A to assure that an update is not in progress. If this bit is read as 1, the watch chip is doing an update and the data is invalid until the update is complete. The maximum time for the update is 1.984 ms.

If the UIP is read as 0, the clock registers can be read by the operating system. The operating system reformats the time into a 32-bit count and loads it into the memory location that contains the time of day count during system operation.

4.11.3.2 Invalid RAM and Time -- If the VRT is clear, RAM and time data are invalid. The operating system stops timer operation by setting CSR B bit 7 (SET), and then requests the time of year from the operator. After loading the correct time and date into the TOY data registers, the operating system loads 10 (hex) into CSR A and then clears SET by loading 6 into CSR B.

The operating system also reformats the time into a 32-bit count, and loads it into the memory location that contains the time of day count during system operation.

Table 4-11 Console Program Mailbox Format

Bit(s)	Mnemonic	Name/Meaning
<07:04>	LNG	Console Message Text Language. This field controls the output of message texts to the console terminal. When set to 0, the console program prompts the user to set the field on power-up. Other settings are as follows. Setting — National Variant 1 — German 2 — English 3 — Spanish 4 — French 5 — Italian 6 — Danish 7 — Dutch 8 — Finnish 9 — Norwegian 10 — Swedish 11 — Portuguese
03	RIP	If set, a restart attempt is in progress. This flag must be cleared by the operating system when the restart succeeds.
02	BIP	If set, a bootstrap attempt is in progress. This flag must be cleared by the operating system when the bootstrap succeeds.
<01:00>	HLT ACT	Processor Halt Action. This field is used to control the automatic restart/bootstrap procedure. This mailbox allows operating system software to override the BDR HLT ENB field. Both bits are cleared on power-up and when the console program exits. The bits may be set as follows. HLT ACT <01:00> — Action 00 — Use HLT ENB (BDR <14>) to determine action. 01 — Restart, if that fails, halt. 10 — Reboot, if that fails, halt. 11 — Halt.

4.12 INTERVAL TIMER

The KA630-AA interval timer is contained within the MicroVAX CPU chip. When it is enabled, the interval timer posts an interrupt request every 10 ms.

4.12.1 Interval Clock Control and Status Register (ICCS)

The ICCS (Figure 4-25) is accessed as IPR 24. ICCS implementation is unique to the MicroVAX CPU chip and consists of a minimal interval timer control.

ICCS bit 6 (IE) is a read-write bit that enables and disables the interval timer interrupts. When this bit is set, an interval timer interrupt is requested every 10 ms. When ICCS bit 6 is clear, interval timer interrupts are disabled. ICCS bit 6 is cleared by power-up and by the negation of DC OK.

4.12.2 Interval Timer Operation

When ICCS bit 6 is set, the interval timer posts an interrupt request every 10 ms. The interval timer is the highest priority device at IPL 16 (hex). The interrupt vector for the interval timer is C0 (hex).

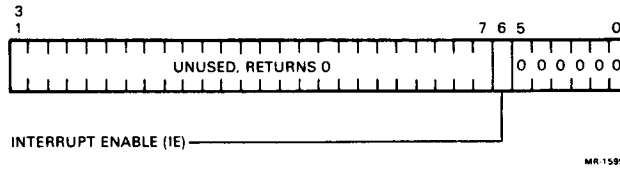


Figure 4-25 Interval Clock Control and Status Register (ICCS)

4.13 CONSOLE SLU

4.13.1 Console Functionality

The console serial line provides the KA630-AA processor with a full-duplex serial interface for the console terminal. It provides an RS-423-A EIA interface that is also RS-232-C compatible.

The serial data format of the console SLU contains 8-bit data, no parity, and one stop bit. The interrupt vectors of the console SLU are F8 (hex) for the receiver and FC (hex) for the transmitter. Its IPL is described in Section 4.4.1.

The receive and transmit baud rates are always identical and are determined by the Baud Rate Select signals (BRS <02:00> L), which are received from an external 8-position switch through a connector mounted at the top of the module.

The baud rate is selected as follows.

BRS02 L	BRS01 L	BRS00 L	Baud Rate
H	H	H	300
H	H	L	600
H	L	H	1,200
H	L	L	2,400
L	H	H	4,800
L	H	L	9,600
L	L	H	19,200
L	L	L	38,400

4.13.2 Console Registers

There are four registers (Table 4-12) associated with the console SLU. They are accessed through IPRs 32 to 35 (decimal).

Table 4-12 SLU Console Registers

Number	Mnemonic	Register Name
32	RXCS	Console Receiver Control/Status
33	RXDB	Console Receiver Data Buffer
34	TXCS	Console Transmit Control/Status
35	TXDB	Console Transmit Data Buffer

4.13.2.1 Console Receiver CSR (IPR 32) -- The contents of the console receiver CSR are shown in Figure 4-26 and Table 4-13.

4.13.2.2 Console Receiver Data Buffer (IPR 33) -- The contents of the console receiver data buffer are shown in Figure 4-27 and Table 4-14.

NOTE

Error conditions remain present until the next character is received, at which point the error bits are updated. The error bits are cleared by power-up and by the negation of DC OK.

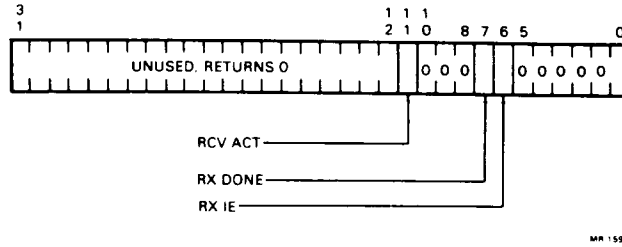


Figure 4-26 Console Receiver CSR

Table 4-13 Console Receiver CSR Format

Bit(s)	Mnemonic	Name/Meaning
<31:12>		Unused. Read as 0s.
11	RCV ACT	Receiver Active. This read-only bit is set at the center of the start bit of the serial input data, and is cleared at the expected center (per DLART timing) of the stop bit at the end of the serial data. RX DONE is set one bit time after RCV ACT clears.
<10:08>		Unused. Read as 0s.
07	RX DONE	Receiver Done. This read-only bit is set when an entire character has been received and is ready to be read from the RBUF register. This bit is automatically cleared when RBUF is read. It is also cleared by power-up, by the negation of DC OK, and by writes to the BIR.
06	RX IE	Receiver Interrupt Enable. This read/write bit is cleared by power-up, by the negation of DC OK, and by writes to the BIR. If RX DONE and RX IE are both set, a program interrupt is requested.
<05:00>		Unused. Read as 0s.

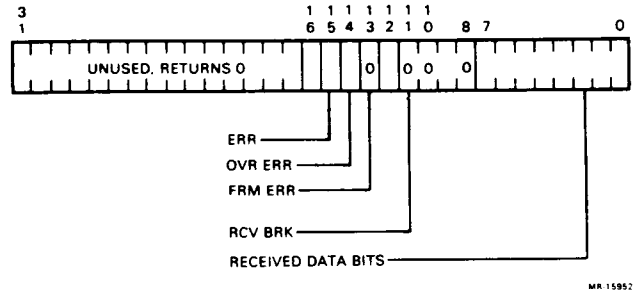


Figure 4-27 Console Receiver Data Buffer

Table 4-14 Console Receiver Data Buffer Format

Bit(s)	Mnemonic	Name/Meaning
<31:16>		Unused. Always read as 0.
15	ERR	Error. This read-only bit is set if RBUF bit 14 or 13 is set. ERR is clear if these two bits are clear. This bit cannot generate a program interrupt.
14	OVR ERR	Overrun Error. This read-only bit is set if a previously received character was not read before being overwritten by the present character.
13	FRM ERR	Framing Error. This read-only bit is set if the present character has no valid stop bit.
12		Unused. This bit always reads as 0.
11	RCV BRK	Received Break. This read-only bit is set at the end of a received character for which the serial data input remained in the space condition for all 11 bit times. RCV BRK then remains set until the serial data input returns to the mark condition. RCV BRK is also cleared by power-up and by the negation of DC OK.
<10:08>		Unused. These bits always read as 0.
<07:00>		Received Data Bits. These read-only bits contain the last received character.

4.13.2.3 Console Transmitter CSR (IPR 34) -- The contents of the console transmitter CSR are shown in Figure 4-28 and Table 4-15.

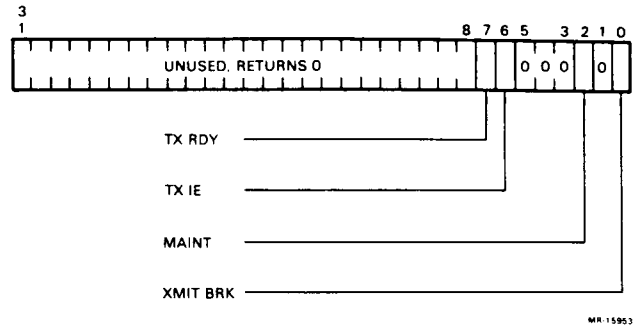


Figure 4-28 Console Transmitter CSR

Table 4-15 Console Transmitter CSR Format

Bit(s)	Mnemonic	Name/Meaning
<31:08>		Unused. Read as 0s.
07	TX RDY	Transmitter Ready. This read-only bit is clear when XBUF is loaded and sets when XBUF can receive another character. XMT RDY is set by power-up, by the negation of DC OK, and by writes to the BIR.
06	TX IE	Transmitter Interrupt Enable. This read/write bit is cleared by power-up, by the negation of DC OK, and by writes to the BIR. If both TX RDY and TX IE are set, a program interrupt is requested.
<05:03>		Unused. Read as 0s.
02	MAINT	Maintenance. This read/write bit is used to facilitate a maintenance self-test. When MAINT is set, the external serial input is disconnected and the serial output is used as the serial input. This bit is cleared by power-up, by the negation of DC OK, and by writes to the BIR.
01		Unused. Read as 0.
00	XMIT BRK	Transmit Break. When this read/write bit is set, the serial output is forced to the space condition. XMIT BRK is cleared by power-up, by the negation of DC OK, and by writes to the BIR.

4.13.2.4 Console Transmitter Data Buffer (IPR 35) -- XBUF bits <31:08> are not used. XBUF bits <07:00> are write-only bits used to load the transmitted character.

4.13.3 Break Response

The KA630-AA console SLU may be configured either to perform a halt operation or to have no response when a break condition is received. A halt operation causes the processor to transfer program control to ROM location 20040000 (hex).

The halt on break option is enabled if the connector HLT ENB signal is asserted.

The DLART recognizes a break condition at the end of a received character for which the serial data input remained in the space condition for all 11 bit times. The break recognition line remains asserted until software reads the RBUF.

4.14 Q22-BUS CONTROL

4.14.1 Bus Initialize Register (IPR 55)

The BIR is accessed as IPR 55 (decimal). On an arbiter KA630-AA, writing to this register asserts the Q22-Bus BINIT signal for 10 µs (+ 20%) and clears all on-board register bits that are specified to clear on writes to the BIR. On an auxiliary KA630-AA, writing to this register does not assert the Q22-Bus BINIT signal, but it does clear all on-board register bits specified to clear on writes to the BIR. For either configuration (arbiter or auxiliary), this register always reads as 0.

NOTE

An auxiliary KA630-AA module receives BINIT from the Q22-Bus and uses that signal to initialize the MicroVAX CPU chip, and to clear all internal register bits that are specified to clear on the negation of DC OK. Stated another way, the assertion of the Q22-Bus BINIT signal has the same effect as the negation of DC OK on auxiliary modules.

4.14.2 Multilevel Interrupts

When the KA630-AA is configured as the arbiter CPU, it responds to interrupt requests BIRQ7 through 4 with the standard Q22-Bus interrupt acknowledge protocol (DIN followed by IAK). The console SLU and the interprocessor doorbell can request interrupts at BIRQ4 and have priority over all Q22-Bus BIRQ4 interrupt requests. After responding to any interrupt request BIRQ7 through 4, the KA630-AA sets the processor priority to IPL 17. All BIRQ7 through 4 interrupt requests are disabled unless software lowers the processor priority.

When the KA630-AA is configured as an auxiliary, it does not respond to interrupt requests from the Q22-Bus. However, it does respond to the BIRQ4 interrupt requests from its console SLU and interprocessor doorbell.

Interrupt requests from the KA630-AA interval timer are handled internally by the MicroVAX CPU chip. Interval timer interrupt requests have a higher priority than BIRQ6 interrupt requests. After responding to an interval timer interrupt request, the MicroVAX CPU chip sets the processor priority to IPL 16. Thus, BIRQ7 interrupt requests remain enabled.

4.14.3 Interprocessor Communications Facility

The KA630-AA interprocessor communication facility allows other processors on the system to request program interrupts from the KA630-AA without using the Q22-Bus interrupt request lines. It also controls external access to local memory by means of the Q22-Bus map, and allows other processors to halt an auxiliary CPU.

4.14.3.1 Interprocessor Communication Register -- The IPCR (Figure 4-29, Table 4-16) resides in the Q22-Bus I/O page address space and can be accessed by any device that can become Q22-Bus master, including the KA630-AA itself. The IPCR is byte accessible, meaning that a write-byte instruction can write to either the low or high byte without affecting the other byte.

The I/O page address of the IPCR varies with the four configurations of arbiter and auxiliary KA630-AA, as follows.

Hex 32-Bit Address	Octal 22-Bit Address	Register
20001F40	17777500	IPCR (Arbiter CPU)
20001F42	17777502	IPCR (Auxiliary 1)
20001F44	17777504	IPCR (Auxiliary 2)
20001F46	17777506	IPCR (Auxiliary 3)

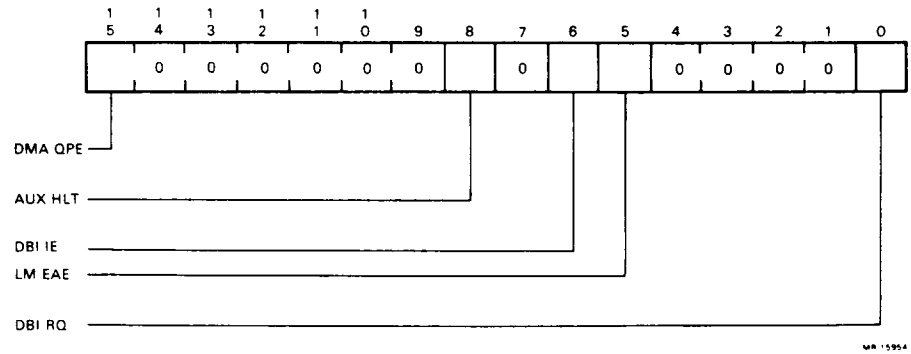


Figure 4-29 Interprocessor Communication Register (IPCR)

Table 4-16 Interprocessor Communication Register Format

Bit(s)	Mnemonic	Name/Meaning
15	DMA QPE	DMA Q22-Bus Address Space Parity Error. This read-only bit is set if MSER bit 4 (DMA QPE) is set. The DMA QPE bit indicates that a parity error occurred when an external device (or CPU) was accessing the KA630-AA local memory.
<14:09>		Unused. Read as 0s.
08	AUX HLT	Auxiliary Halt. On an auxiliary KA630-AA, AUX HLT is a read/write bit. When set, typically by the arbiter CPU, it causes the on-board CPU to transfer program control to the halt mode ROM code. On an arbiter KA630-AA, AUX HLT is a read-only bit that always reads as 0. It has no effect on arbiter CPU operation.
07		Unused. Read as 0.
06	DBI IE	Doorbell Interrupt Enable. This bit, when set, enables interprocessor doorbell interrupt requests through IPCR bit 0. When the on-board CPU is Q22-Bus master, DBI IE is a read/write bit. When an external device (or CPU) is bus master, DBI IE is a read-only bit. DBI IE is cleared by power-up, by the negation of DC OK, and by writes to the BIR.
05	LM EAE	Local Memory External Access Enable. This bit, when set, enables external access to local memory (by means of the Q22-Bus map). When the on-board CPU is Q22-Bus master, LM EAE is a read/write bit. When an external device (or CPU) is bus master, LM EAE is a read-only bit. LM EAE is is cleared by power-up and by the negation of DC OK.
<04:01>		Unused. Read as 0s.
00	DBI RQ	Doorbell Interrupt Request. If IPCR bit 6 (DBI IE) is set, writing a 1 to DBI RQ sets DBI RQ, thus requesting a doorbell interrupt. If IPCR bit 6 is clear, writing a 1 to DBI RQ has no effect. Writing a 0 to DBI RQ has no effect. DBI RQ is cleared when the CPU grants the doorbell interrupt request. DBI RQ is held clear whenever DBI IE is clear.

4.14.3.2 Interprocessor Doorbell Interrupts -- If the IPCR DBI IE bit is set, any Q22-Bus master can request an interprocessor doorbell interrupt by writing a 1 into IPCR bit 0.

The interprocessor doorbell interrupt vector is 204 (hex). Its interrupt priority is described in Section 4.4.1.

NOTE

Following an interprocessor doorbell interrupt, the KA630-AA sets the IPL = 14. The IPL is set = 17 for external Q22-Bus BIRQ4 interrupts.

4.15 MULTIPROCESSOR CONSIDERATIONS

4.15.1 Auxiliary/Arbiter Differences

When the KA630-AA is configured as an auxiliary, its operation differs from operation as an arbiter in several important areas:

1. The arbiter KA630-AA arbitrates bus mastership per the Q22-Bus DMA protocol. The arbitration logic is disabled on an auxiliary KA630-AA.

2. Both the arbiter and auxiliary KA630-AA request bus mastership using the Q22-Bus DMA request protocol, as follows.

   a. They both assert BDMR on the Q22-Bus.

   b. The arbiter KA630-AA receives DMGI from its arbitration logic. The auxiliary receives DMGI from its Q22-Bus BDMGI pin.

   c. Only the auxiliary KA630-AA actually asserts BSACK on the Q22-Bus.

3. The arbiter KA630-AA asserts the Q22-Bus BINIT signal when DC OK is negated and when its CPU software writes to its BIR. The auxiliary KA630-AA never asserts Q22-Bus BINIT, but receives BINIT and uses it to initialize the MicroVAX CPU chip and to clear all internal registers that are specified to clear on the negation of DC OK.

4. The physical address of the IPCR is different for each of the four KA630-AA arbiter/auxiliary configurations.

5. An auxiliary KA630-AA can be halted by setting bit 8 (AUX HLT) of its IPCR. On an arbiter KA630-AA, this feature is disabled and AUX HLT is a read-only bit that always reads as 0.

6. The CPU halts are controlled by the external connector HLT ENB input. However, the external halts that are affected differ somewhat for the arbiter and auxiliary KA630-AA modules.

7. Each arbiter or auxiliary KA630-AA module can field interrupt requests from its interval timer, from its console device, and from its interprocessor doorbell. Only the arbiter KA630-AA can field interrupts from Q22-Bus interrupt request lines BIRQ7 through 4.

8. The arbiter asserts BIAKO to the Q22-Bus when it responds to a Q22-Bus interrupt request. The auxiliary asserts BIAKO to the Q22-Bus when it receives the assertion of BIAKI from the Q22-Bus.

9. Although both arbiter and auxiliary KA630-AA modules contain the same TOY clock and battery backup circuitry, it is assumed that the auxiliary will be configured without batteries and that its clock will never actually be enabled.

4.15.2 Multiprocessor Features

The following features have been added to the KA630-AA to allow its use in multiprocessor systems.

- A 2-bit code, received at the external connector, allows the KA630-AA module to be configured as the arbiter or as one of three auxiliaries.

- The IPCR provides a mechanism for interprocessor interrupts, for enabling and disabling external access to local memory, and for flagging local memory parity errors caused by external references. On auxiliary KA630-AA modules, it also provides a mechanism for halting the CPU.

4.15.3 KA630-AA Based Multiprocessor Systems

The KA630-AA multiprocessor features were designed for use in a message passing environment similar to the System Communications Architecture (SCA) that is currently layered on the CI port architecture.

Each KA630-AA processor in a system fetches instructions and data primarily from its own local memory. The various processors communicate by way of message queues stored in local memory that has been mapped to the Q22-Bus address space. Typically, the processors use the interprocessor doorbell feature to interrupt each other after placing a message in an empty queue.

In most systems, all Q22-Bus devices would be under the direct control of the arbiter processor, which fields all interrupts. When a disk controller is under the direct control of the arbiter CPU, the arbiter must set up the transfer of program and data information between the corresponding disks and the auxiliary processors. The auxiliary processor is responsible for setting up its own Q22-Bus map to point to the local memory space that is a target of that transfer.

Following a power-up or system restart, the auxiliary CPU runs its self-test diagnostics, clears the valid bits in its Q22-Bus map mapping registers, enters halt mode ROM space, and then sets its own IPCR bits 8 (AUX HLT) and <06:05> (DBI IE and LM EAE). The arbiter CPU waits for the auxiliary's LM EAE bit to set and then boots the auxiliary CPU by loading the appropriate programs and data into the arbiter's own local memory. These programs and data are mapped to an assigned Q22-Bus address space through the Q22-Bus map. The arbiter then clears the auxiliary's AUX HLT bit. The auxiliary CPU, still in halt mode ROM space, waits for its AUX HLT bit to clear and then begins auxiliary execution at a specified location in the Q22-Bus address space (referencing local memory in the arbiter).

4.15.4 PDP-11 Based Multiprocessor Systems

Up to three auxiliary KA630-AA modules can be added to a KDF11-B or KDJ11-B based Q22-Bus system. Operation of a PDP-11 based system is similar to that of a KA630-AA based system. However, the following issues must be addressed.

- When a PDP-11 processor is arbiter, its "local" memory is actually Q22-Bus memory. This appears to present no special problems. A portion of the Q22-Bus memory address space must be reserved for mapping the auxiliary KA630-AA modules' local memory.

- Since the PDP-11 processor does not contain an IPCR, an external device must be added that allows the auxiliary KA630-AA modules to interrupt the processor. Since the KA630-AA console program does not interrupt the arbiter CPU, the auxiliary KA630-AA modules do not require modification if this external device is not compatible with the KA630-AA IPCR.

5.1 INTRODUCTION

The diagnostics test the basic functionality of an arbiter KA630-AA. The KA630-AA diagnostics operate in the following two modes.

- Power-up mode
- Console I/O mode

In power-up mode, the diagnostics, in conjunction with the boot program, test the KA630-AA's ability to load and run a typical operating system or diagnostic supervisor. Seven diagnostic tests are performed. They cover CPU functionality, system pathing and memory. The boot program tests the Q22-Bus interface.

In console I/O mode, each of the seven tests can be selected using the Test command. Before each test is executed, its test number is output by the console program to the LEDs on the KA630-AA and to the console terminal. This provides an external indication of testing progress, so that a loss of control may be traced to the failing test.

The diagnostics are located in the console program ROM on the KA630-AA. They do not test all the functional areas of the KA630-AA. The areas not tested are as follows.

- TOY clock
- Bus reset register (IPR 55)
- Console saved ISP (IPR 41)
- Console saved PC (IPR 42)
- Console saved ISL (IPR 43)
- ASTLVL register (IPR 19)
- DMA related circuitry
- Category 3 registers

5.2 FUNCTIONAL DESCRIPTION

The KA630-AA diagnostics include seven tests that are run in a sequence designed to localize failures. The tests run from 8 to 3 (test 3 covering two functional areas), and are described in Table 5-1.

Table 5-1 Diagnostic Tests

Test	Coverage	Error Type*
8	IPCR (no interrupts)	F
7	Memory data	F, H
6	Memory address	F
5	Q22-Bus mapping registers (no interrupts), MSER and CEAR (no wrong parity)	F
4	CPU chip	C, F
3	Interrupts and traps TOY MSER CEAR Q22-Bus Interval timer Console	C, F

* C (Catastrophic) -- The state of KA630-AA is unpredictable.

F (Fatal) -- KA630-AA cannot operate, but console commands can be used.

H (Hard) -- Memory error.

5.2.1 IPCR Test

This test reads the IPCR. No interrupts are generated. No traps should occur. An IPCR test checks the interprocessor communication register and interprocessor doorbell interrupt.

5.2.2 Memory Data Test

This test checks all the memory for shorted and stuck conditions, and builds a memory bitmap. This bitmap does not include the VMB map or the console program scratch pages. The memory data test is executed with PAR ENB set. Therefore, MSER and CEAR are checked for 0.

A fatal error exists when 64 Kbytes of contiguous memory are not found within the first 4 Mbytes of local memory.

5.2.3 Memory Address Test

This test checks every cell of the memory to be sure each address is unique.

5.2.4 Q22-Bus Mapping Registers Test

This test checks Q22-Bus mapping registers for shorted and stuck bits.

5.2.5 MicroVAX CPU Chip Test

This test checks data paths between the console program ROM and MicroVAX CPU chip. Since it is a basic path for the diagnostics, the test assumes that only the MicroVAX CPU chip can malfunction.

The following checks are performed:

1. Checks general registers and bits <07:04> of PSL for shorted and stuck bits.

2. Executes a limited set of instructions. This set includes all instructions in all modes that are necessary to load software, plus all instructions that are used in the ROM diagnostic. This forces a functional check of bits <03:00> of PSL as well.

3. Checks Process Control Block Base (PCBB); kernel, executive, supervisor, and user stack pointers; and TBIS for shorted and stuck bits.

4. Checks TBIA for a 0.

5. Checks SBR, P0BR, and PCBB bits <01:00> for a 0, <29:02> for shorted and stuck bits, and <31:30> for a proper state.

6. Checks SLR, P0LR, and P1LR bits <31:22> for a 0, and <21:00> for shorted and stuck bits.

7. Checks MAPEN bit 0 for 0.

8. Checks P1BR bits <01:00> for 0s, and <31:02> for shorted and stuck bits.

9. Checks SCBB bits 31, 30, and <08:00> for 0s, and <29:09> for shorted and stuck bits.

10. Checks the MMU.

11. Checks the FPU.

5.2.6 Software Interrupts and Traps

This group consists of data path tests between the ROM and MicroVAX CPU chip. It checks exceptions and software interrupts.

The following algorithm is used:

1. Ensure that SISR is a 0.
2. Walk through SIRR bits <03:00> and check SISR bits <16:01>.
3. Decrease IPL to 10 (hex) using bits <04:00>, and compare to PSL bits <20:16>.
4. Drop IPL to 1 (hex) and start responding to the software interrupts.
5. Increase IPL to 1E (hex).
6. Check traps by violating limits of SLR, P0LR and P1LR.
7. Drop mode to executive, check PSL bits <25:22> and check protection trap.
8. Drop mode to supervisor, check PSL bits <25:22> and check protection trap.
9. Drop mode to user, check PSL bits <25:22> and check protection trap.

5.2.7 System Interrupts and Data Paths

These tests verify system interrupts as well as several processor board data paths. They check the TOY register, interval timer, CEAR and MSER.

The following tests are performed:

1. Check BDR bits <13:11> and <07:04> for 0, bit 15 for 1 and bits <03:00> for proper code.
2. Check interval timer.
3. Map two pages of good memory into Q22-Bus map registers.
4. Check CEAR for 0.
5. Check MSER for 0, and set PAR ENB bit 0.
6. Write data into good pages of memory. Read it locally. No trap should occur.

7. Set WRW bit 1. Write data into good pages.

8. Read data locally. Since wrong parity was used, the test traps through vector 4 of the SCB.

9. Check bits <09:03> of MSER for proper code.

10. Check bits <14:00> of CEAR for proper error address.

11. Clear bits <01:00> of MSER.

12. Write data to memory through Q22-Bus.

13. Read data from memory locally. No trap should occur.

14. Read data from memory through Q22-Bus.

15. Set MSER bits <01:00>, WRW and PAR ENB. Write data through Q22-Bus.

16. Read data from memory locally. Since wrong parity was used, the test traps through vector 4 of the SCB.

17. Check bits <09:03> of MSER for proper code.

18. Check bits <14:00> of CEAR for proper error address.

19. Check bit 15 of IPCR.

20. Write data to memory through Q22-Bus.

21. Read data from memory through Q22-Bus. Since wrong parity was used, the test traps through vector 4 of the SCB.

22. Check bits <09:03> of MSER for proper code.

23. Check bits <14:00> of CEAR for proper error address.

24. Check bit 15 of IPCR.

25. Write data into nonexistent I/O register and trap.

5.2.7.1 Console Test -- The diagnostic sets maintenance bit 2 of the console transmitter CSR, and outputs four null characters. This test checks the following.

- Receiver and transmitter interrupts
- Receiver Done and Active bits
- Receiver buffer Error and Overrun Error flags

5.2.8 Console Program/ROM Diagnostic Interface

In either power-up or console I/O mode, the console program invokes one test at a time. Each test number is displayed on the LEDs and the console terminal. No carriage returns are output as part of the code display, so if all tests complete normally, the string 8...7...6...5...4...3... is seen on the console terminal. If the diagnostic detects a catastrophic or fatal error, it passes the status and error number to the console program immediately, and no further testing is done. If hard (memory) errors are detected, the diagnostic passes the status and number of bad pages to the console program only at the end of the test.

At the end of the test, the diagnostic passes status to the console program. This way Repeat Test (an optional specifier) can be easily implemented. Before each test, the diagnostic sets the "Test in Progress" safeguard flag.

NOTE

Test commands may include a read/write test of memory that leaves memory contents in an unpredictable state. The RPB may be lost. In this case, the system must be rebooted to restore the RPB.

5.3 ERROR OUTPUT

If a diagnostic detects a catastrophic or fatal error, it passes the status and error number to the console program, and no further testing is done. Neither text nor error messages are printed by the diagnostic program. All text and error messages are printed by the console in the appropriate language.

5.4 KA630-AA LED DISPLAY

There are four red LEDs on the KA630-AA. These four LEDs, interpreted as a 4-bit hexadecimal digit, are described in Table 5-2. All LED codes are displayed during power-up. A problem is indicated only if the CPU stops at a particular LED code.

An LED is illuminated if its control bit in the BDR is 1, and is not illuminated if its control bit is 0.

It is possible for privileged code to output to the LEDs when the processor is in program mode. Because this may cause confusion in the interpretation of the LEDs, this use of the LEDs is discouraged.

When executing the power-up sequence, the codes 9 through 0 are also output to the console terminal.

Table 5-2 KA630-AA LED Interpretation

Code	Activity	Exit Criteria
F	Electrical power-up.	MicroVAX starts execution from console program ROM.
E	Wait for PWR OK.	BDR bit 15 set.
D	Perform ROM checksum and TOY RAM tests.*	Test success.
C	Initialize console program memory.	Console memory and bitmap initialized, registers saved.
B	Run IPCR tests.*	Test success.
A	Test for and check VCB01 video console display, if present.*	VCB01 operational or not present.
9	Perform console port tests and terminal identification.*	Console terminal type determined.
8	Query console language,* then enter console command mode.	Exits power-up automatically, otherwise exits on console Continue, Start, Boot or Test commands.
7	Run memory pattern tests.†	At least 64 Kbytes of contiguous good memory found.
6	Run memory address tests.*	All address tests passed.
5	Run I/O map tests.*	Test success.
4	Run CPU tests.*	Test success.
3	Run interrupt tests.*	Test success.
2	Search for bootstrap device.‡	Valid bootstrap device located.
1	Load bootstrap.‡	Bootstrap successfully loaded.
0	Program mode.	Not applicable.

* Performed only on power-up entry into console program.

† Performed on power-up entry and on operator-requested bootstrap.

‡ Performed only during bootstrap.

APPENDIX A
Q22-BUS SPECIFICATION

A.1 GENERAL DESCRIPTION

The Q22-Bus, also known as the extended LSI-11 Bus, is the low-end member of Digital's bus family. All of Digital's microcomputers, such as the MicroVAX I, MicroVAX II, and MicroPDP-11, use the Q22-Bus.

The Q22-Bus consists of 42 bidirectional and 2 unidirectional signal lines. These form the lines along which the processor, memory, and I/O devices communicate with each other.

Addresses, data, and control information are sent along these signal lines, some of which contain time-multiplexed information. The lines are divided as follows.

- Sixteen multiplexed data/address lines -- BDAL<15:00>
- Two multiplexed address/parity lines -- BDAL<17:16>
- Four extended address lines -- BDAL<21:18>
- Six data transfer control lines -- BBS7, BDIN, BDOUT, BRPLY, BSYNC, BWTBT
- Six system control lines -- BHALT, BREF, BEVNT, BINIT, BDCOK, BPOK
- Ten interrupt control and direct memory access control lines -- BIAKO, BIAKI, BIRQ4, BIRQ5, BIRQ6, BIRQ7, BDMGO, BDMR, BSACK, BDMGI

In addition, a number of power, ground, and space lines are defined for the bus. Refer to Table A-1 for a detailed description of these lines.

The discussion in this Appendix applies to the general 22-bit physical address capability. All modules used with the KA630-A CPU module must use 22-bit addressing.

Most Q22-Bus signals are bidirectional and use terminations for a negated (high) signal level. Devices connect to these lines via high-impedance bus receivers and open collector drivers. The asserted state is produced when a bus driver asserts the line low.

Although bidirectional lines are electrically bidirectional (any point along the line can be driven or received), certain lines are functionally unidirectional. These lines communicate to or from a bus master (or signal source), but not both. Interrupt acknowledge (BIAK) and direct memory access grant (BDMG) signals are physically unidirectional in a daisy-chain fashion. These signals originate at the processor output signal pins. Each is received on device input pins (BIAKI or BDMGI) and is conditionally retransmitted via device output pins (BIAKO or BDMGO). These signals are received from higher-priority devices and are retransmitted to lower-priority devices along the bus, establishing the position-dependent priority scheme.

A.1.1 Master/Slave Relationship

Communication between devices on the bus is asynchronous. A master/slave relationship exists throughout each bus transaction. Only one device has control of the bus at any one time. This controlling device is termed the bus master, or arbiter. The master device controls the bus when communicating with another device on the bus, termed the slave.

The bus master (typically the processor or a DMA device) initiates a bus transaction. The slave device responds by acknowledging the transaction in progress and by receiving data from, or transmitting data to, the bus master. Q22-Bus control signals transmitted or received by the bus master or bus slave device must complete the sequence according to bus protocol.

The processor controls bus arbitration, that is, which device becomes bus master at any given time. A typical example of this relationship is a disk drive, as master, transferring data to memory as slave. Communication on the Q22-Bus is interlocked so that, for certain control signals issued by the master device, there must be a response from the slave in order to complete the transfer. It is the master/slave signal protocol that makes the Q22-Bus asynchronous. The asynchronous operation precludes the need for synchronizing with, and waiting for, clock pulses.

Since bus cycle completion by the bus master requires response from the slave device, each bus master must include a timeout error circuit that aborts the bus cycle if the slave does not respond to the bus transaction within 10 μs. The actual time before a timeout error occurs must be longer than the reply time of the slowest peripheral or memory device on the bus.

A.2 Q22-BUS SIGNAL ASSIGNMENTS

Table A-1 lists the signal assignments for the data/address, control, power/ground, and spare functions of the Q22-Bus.

Table A-1 Signal Assignments

Name	Pin Assignment
DATA AND ADDRESS	
BDAL0	AU2
BDAL1	AV2
BDAL2	BE2
BDAL3	BF2
BDAL4	BH2
BDAL5	BJ2
BDAL6	BK2
BDAL7	BL2
BDAL8	BM2
BDAL9	BN2
BDAL10	BP2
BDAL11	BR2
BDAL12	BS2
BDAL13	BT2
BDAL14	BU2
BDAL15	BV2
BDAL16	AC1
BDAL17	AD1
BDAL18	BC1
BDAL19	BD1
BDAL20	BE1
BDAL21	BF1
CONTROL	
Data Control	
BDOUT	AE2
BRPLY	AF2
BDIN	AH2
BSYNC	AJ2
BWTBT	AK2
BBS7	AP2
Interrupt Control	
BIRQ7	BP1
BIRQ6	AB1
BIRQ5	AA1
BIRQ4	AL2
BIAKO	AN2
BIAKI	AM2
DMA Control	
BDMR	AN1
BSACK	BN1
BDMGO	AS2
BMDGI	AR2

Table A-1 Signal Assignments (Cont)

Name	Pin Assignment
System Control	
BHALT	AP1
BREF	AR1
BEVNT	BR1
BINIT	AT2
BDCOK	BA1
BPOK	BB1
POWER AND GROUND	
+5B (battery) or +12B (battery)	AS1
+12B	BS1
+5B	AV1
+5	AA2
+5	BA2
+5	BV1
+12	AD2
+12	BD2
+12	AB2
-12	AB2
-12	BB2
GND	AC2
GND	AJ1
GND	AM1
GND	AT1
GND	BC2
GND	BJ1
GND	BM1
GND	BT1
SPARES	
SSpare1	AE1
SSpare3	AH1
SSpare8	BH1
SSpare2	AF1
MSpareA	AK1
MSpareB	AL1
MSpareB	BK1
MSpareB	BL1
PSpare1	AU1
ASpare2	BU1

A.3 DATA TRANSFER BUS CYCLES

Data transfer bus cycles are listed and defined in Table A-2.

These bus cycles, executed by bus master devices, transfer 32-bit words or 8-bit bytes to or from slave devices. In block mode, multiple words may be transferred to sequential word addresses, starting from a single bus address. The bus signals listed in Table A-3 are used in the data transfer operations described in Table A-2.

Table A-2 Data Transfer Operations

Bus Cycle Mnemonic	Description	Function (with Respect to the Bus Master)
DATI	Data word input	Read
DATO	Data word output	Write
DATOB	Data byte output	Write-byte
DATIO	Data word input/output	Read-modify-write
DATIOB	Data word input/byte output	Read-modify-write byte
DATBI	Data block input	Read block
DATBO	Data block output	Write block

Table A-3 Bus Signals for Data Transfers

Mnemonic	Description	Function
BDAL<21:00> L	22 Data/address lines	BDAL<15:00> L are used for word and byte transfers. BDAL<17:16> L are used for extended addressing, memory parity error (16), and memory parity error enable (17), functions. BDAL<21:18> L are used for extended addressing beyond 256 Kbytes.
BSYNC L	Bus cycle control	Indicates bus transaction in progress.
BDIN L	Data input indicator	Strobe signals.
BDOUT L	Data output indicator	Strobe signals.
BRPLY L	Slave's acknowledge of bus cycle	Strobe signals.
BWTBT L	Write/byte control	Control signals.
BBS7	I/O device select	Indicates address is in the I/O page.

Data transfer bus cycles can be reduced to five basic types: DATI, DATO(B), DATIO(B), DATBI, and DATBO. These transactions occur between the bus master and one slave device selected during the addressing portion of the bus cycle.

A.3.1 Bus Cycle Protocol

Before initiating a bus cycle, the previous bus transaction must have been completed (BSYNC L negated) and the device must become bus master. The bus cycle can be divided into two parts: an addressing portion, and a data transfer portion. During the addressing portion, the bus master outputs the address for the desired slave device, memory location, or device register. The selected slave device responds by latching the address bits and holding this condition for the duration of the bus cycle until BSYNC L becomes negated. During the data transfer portion, the actual data transfer occurs.

A.3.2 Device Addressing

The device addressing portion of a data transfer bus cycle comprises an address setup and deskew time, and an address hold and deskew time. During the address setup and deskew time, the bus master does the following.

- Asserts BDAL<21:00> L with the desired slave device address bits
- Asserts BBS7 L if a device in the I/O page is being addressed
- Asserts BWTBT L if the cycle is a DATO(B) or DATBO bus cycle

During this time, the address, BBS7 L, and BWTBT L signals are asserted at the slave bus receiver for at least 75 ns before BSYNC goes active. Devices in the I/O page ignore the 9 high-order address bits BDAL<21:13>, and instead, decode BBS7 L along with the 13 low-order address bits. An active BWTBT L signal during address setup time indicates that a DATO(B) or DATBO operation follows, while an inactive BWTBT L indicates a DATI, DATBI, or DATIO(B) operation.

The address hold and deskew time begins after BSYNC L is asserted.

The slave device uses the active BSYNC L bus received output to clock BDAL address bits, BBS7 L, and BWTBT L into its internal logic. BDAL<21:00> L, BBS7 L, and BWTBT L remain active for 25 ns (minimum) after BSYNC L bus receiver goes active. BSYNC L remains active for the duration of the bus cycle.

Memory and peripheral devices are addressed similarly, except for the way the slave device responds to BBS7 L. Addressed peripheral devices must not decode address bits on BDAL<21:13> L. Addressed peripheral devices may respond to a bus cycle when BBS7 L is asserted (low) during the addressing portion of the cycle. When asserted, BBS7 L indicates that the device address resides in the

I/O page (the upper 4 K address space). Memory devices generally do not respond to addresses in the I/O page; however, some system applications may permit memory to reside in the I/O page for use as DMA buffers, read-only memory bootstraps, diagnostics, etc.

DATI -- The DATI bus cycle, shown in Figure A-1, is a read operation. During DATI, data is input to the bus master. Data consists of 16-bit word transfers over the bus. During the data transfer portion of the DATI bus cycle, the bus master asserts BDIN L 100 ns (minimum) after BSYNC L is asserted. The slave device responds to BDIN L active as follows.

- Asserts BRPLY L 0 ns (minimum) (8 ns maximum to avoid bus timeout) after receiving BDIN L, and 125 ns (maximum) before BDAL bus driver data bits are valid.

- Asserts BDAL<21L00> L with the addressed data and error information 0 ns (minimum) after receiving BDIN, and 125 ns (maximum) after assertion of BRPLY.

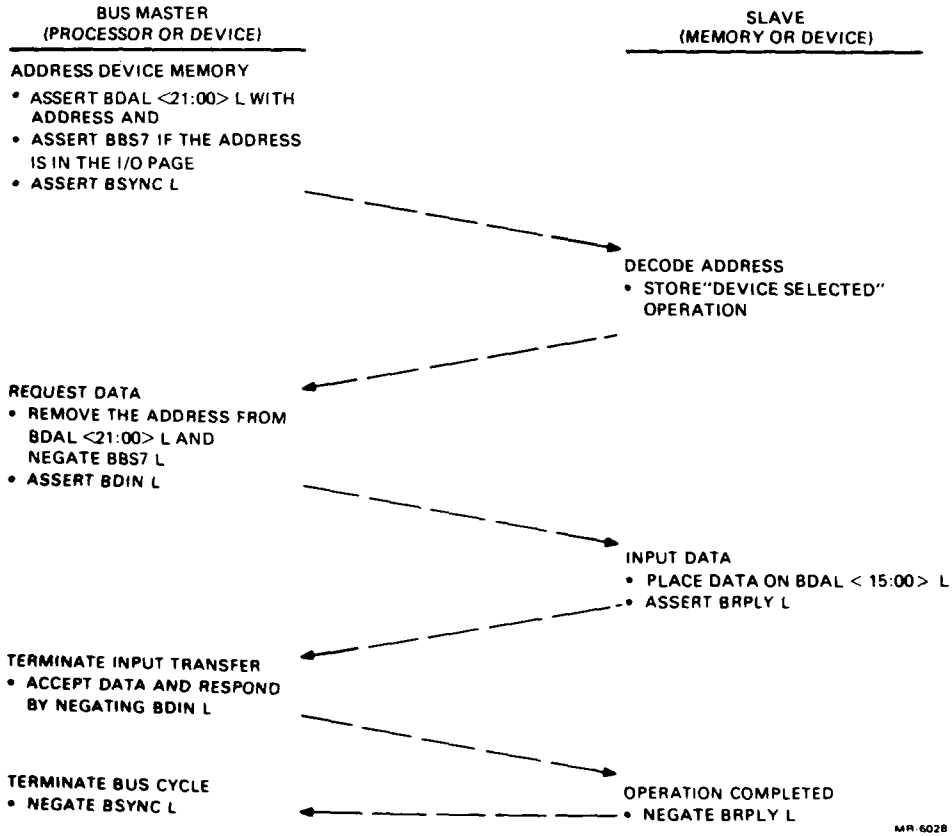


Figure A-1 DATI Bus Cycle

When the bus master receives BRPLY L, it does the following.

- Waits at least 200 ns deskew time and then accepts input data at BDAL<17:00> L bus receivers. BDAL <17:16> L are used for transmitting parity errors to the master.

- Negates BDIN L 200 ns (minimum) to 2 μs (maximum) after BRPLY L goes active.

The slave device responds to BDIN L negation by negating BRPLY L and removing read data from BDAL bus drivers. BRPLY L must be negated 100 ns (maximum) prior to removal of read data. The bus master responds to the negated BRPLY L by negating BSYNC L.

Conditions for the next BSYNC L assertion are as follows.

- BSYNC L must remain negated for 200 ns (minimum).

- BSYNC L must not become asserted within 300 ns of previous BRPLY L negation.

Figure A-2 shows DATI bus cycle timing.

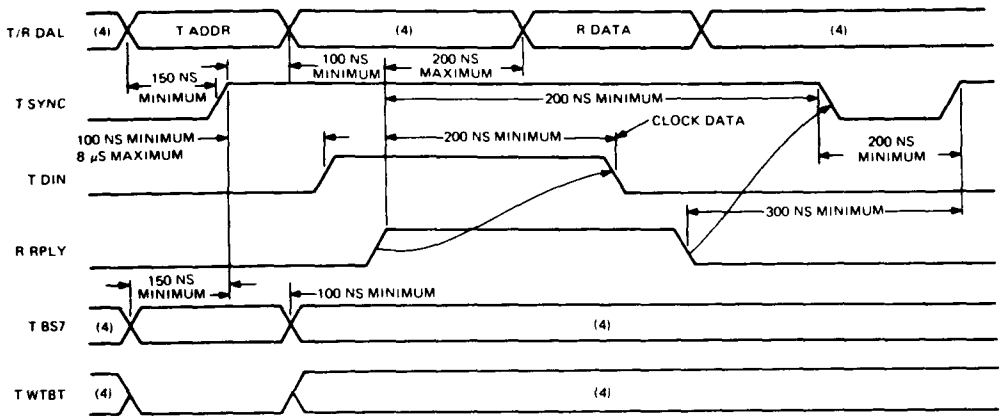
NOTE

Continuous assertion of BSYNC L retains control of the bus by the bus master, and the previously addressed slave device remains selected. This is done for DATIO(B) bus cycles where DATO or DATOB follows a DATI without BSYNC L negation and a second device addressing operation. Also, a slow slave device can hold off data transfers to itself by keeping BRPLY L asserted, which causes the master to keep BSYNC L asserted.

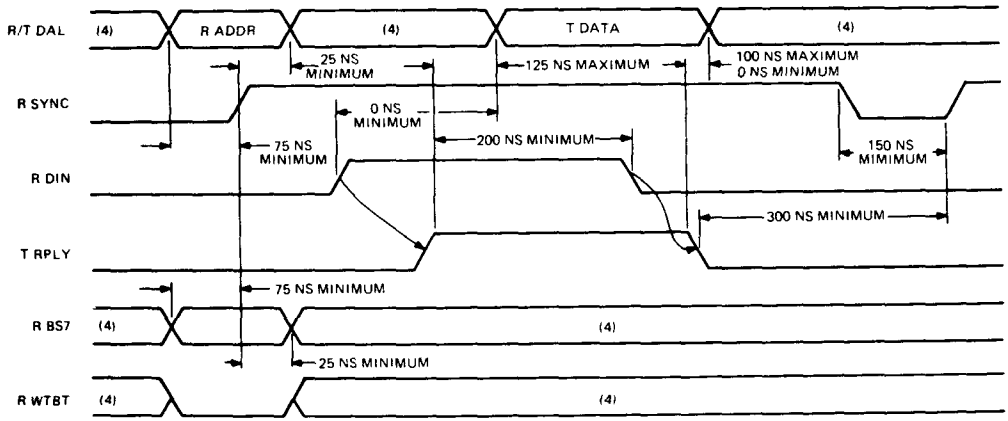
DATO(B) -- DATO(B), shown in Figure A-3, is a write operation. Data is transferred in 32-bit words (DATO) or 8-bit bytes (DATOB) from the bus master to the slave device. The data transfer output can occur after the addressing portion of a bus cycle when BWTBT L has been asserted by the bus master, or immediately following an input transfer part of a DATIO(B) bus cycle.

The data transfer portion of a DATO(B) bus cycle comprises a data setup and deskew time and a data hold and deskew time.

During the data setup and deskew time, the bus master outputs the data on BDAL<15:00> L at least 100 ns after BSYNC L assertion. BWTBT L remains negated for the length of the bus cycle. If the transfer is a byte transfer, BWTBT L remains asserted. If it is the output of a DATIOB, BWTBT L becomes asserted and lasts the duration of the bus cycle.



TIMING AT MASTER DEVICE



TIMING AT SLAVE DEVICE

NOTES

1. TIMING SHOWN AT MASTER AND SLAVE DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS
2. SIGNAL NAME PREFIXES ARE DEFINED BELOW
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT
3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX.
4. DON'T CARE CONDITION

MR 6037

Figure A-2 DATI Bus Cycle Timing

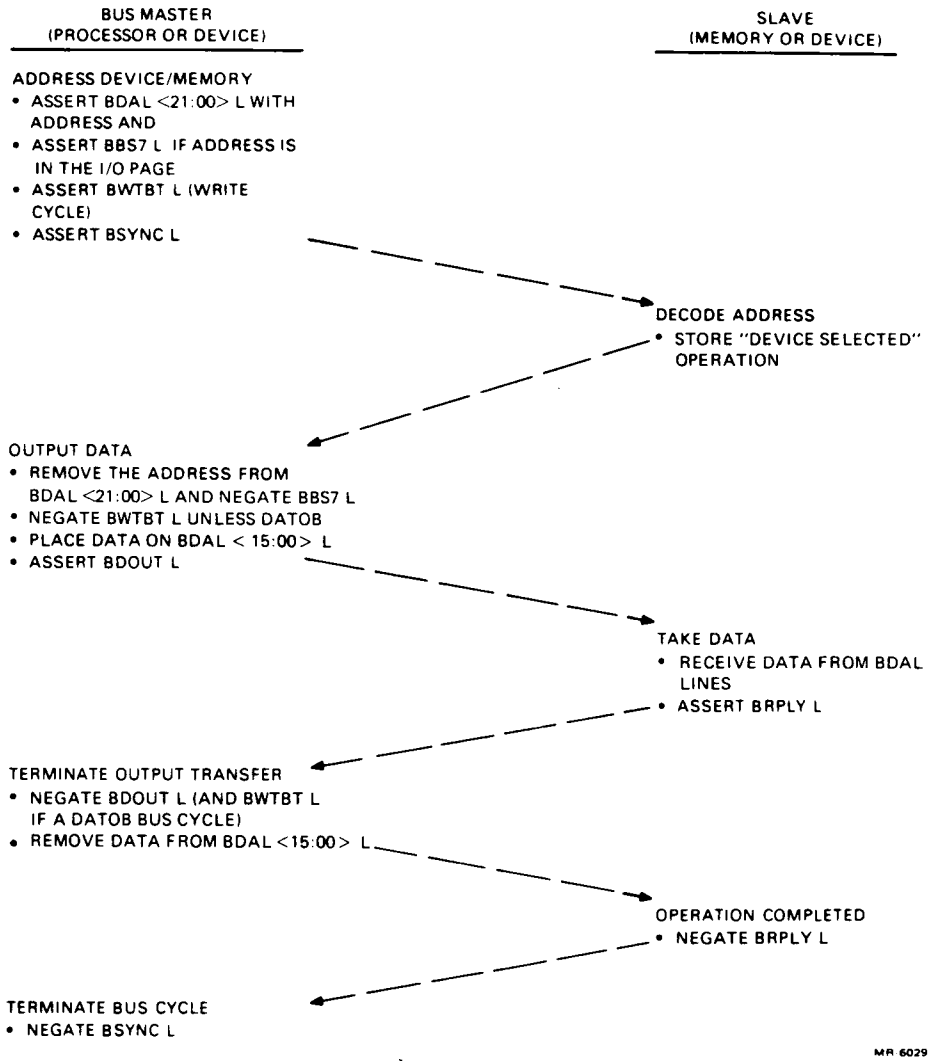


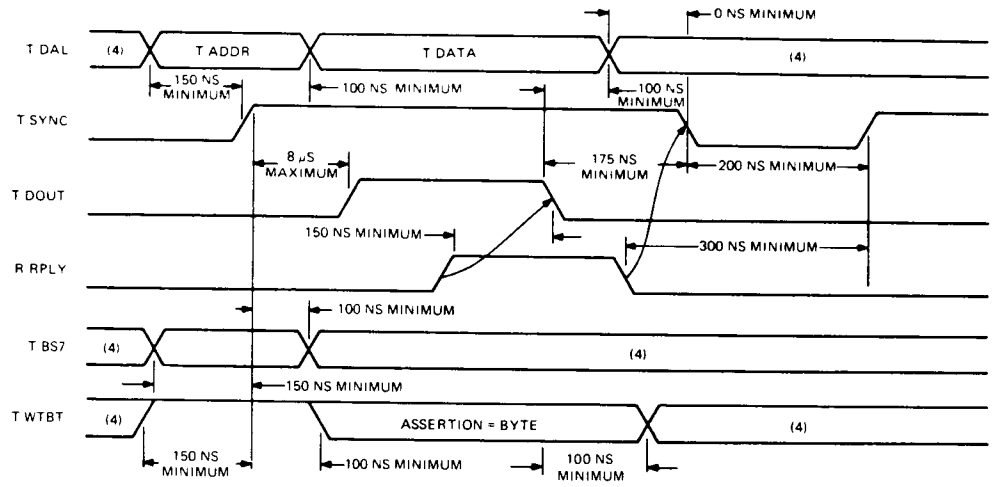
Figure A-3 DATO or DATOB Bus Cycle

During a byte transfer, BDAL<00> L selects the high or low byte. This occurs while in the addressing portion of the cycle. If asserted, the high byte (BDAL<15:08> L) is selected; otherwise, the low byte (BDAL<07:00> L) is selected. An asserted BDAL 16 L at this time forces a parity error to be written into memory if the memory is a parity-type memory. BDAL 17 L is not used for write operations. The bus master asserts BDOUT L at least 100 ns after BDAL and BDWTBT L bus drivers are stable. The slave device responds by asserting BRPLY L within 10 μs to avoid bus timeout. This completes the data setup and deskew time.

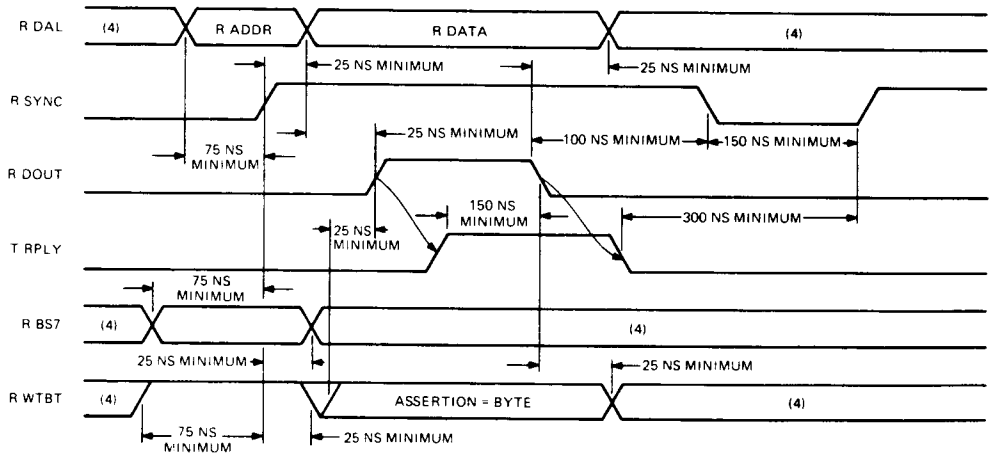
During the data hold and deskew time, the bus master receives BRPLY L and negates BDOUT L, which must remain asserted for at least 150 ns from the receipt of BRPLY L before being negated by the bus master. BDAL<17:00> L bus drivers remain asserted for at least 100 ns after BDOUT L negation. The bus master then negates BDAL inputs.

During this time, the slave device senses BDOUT L negation. The data is accepted and the slave device negates BRPLY L. The bus master responds by negating BSYNC L. However, the processor does not negate BSYNC L for at least 175 ns after negating BDOUT L. This completes the DATO(B) bus cycle. Before the next cycle, BSYNC L must remain unasserted for at least 200 ns. Figure A-4 shows DATO(B) bus cycle timing.

DAITO(B) -- The protocol for a DATIO(B) bus cycle is identical to the addressing and data transfer portions of the DATI and DATO(B) bus cycles, and is shown in Figure A-5. After addressing the device, a DATI cycle is performed as explained earlier; however, BSYNC L is not negated. BSYNC L remains active for an output word or byte transfer [DATO(B)]. The bus master maintains at least 200 ns between BRPLY L negation during the DATI cycle and BDOUT L assertion. The cycle is terminated when the bus master negates BSYNC L, as described for DATO(B). Figure A-6 illustrates DATIO(B) bus cycle timing.



TIMING AT MASTER DEVICE



TIMING AT SLAVE DEVICE

NOTES

1. TIMING SHOWN AT MASTER AND SLAVE DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS
2. SIGNAL NAME PREFIXES ARE DEFINED BELOW
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT
3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX.
4. DON'T CARE CONDITION.

MR-1179

Figure A-4 DATO or DATOB Bus Cycle Timing

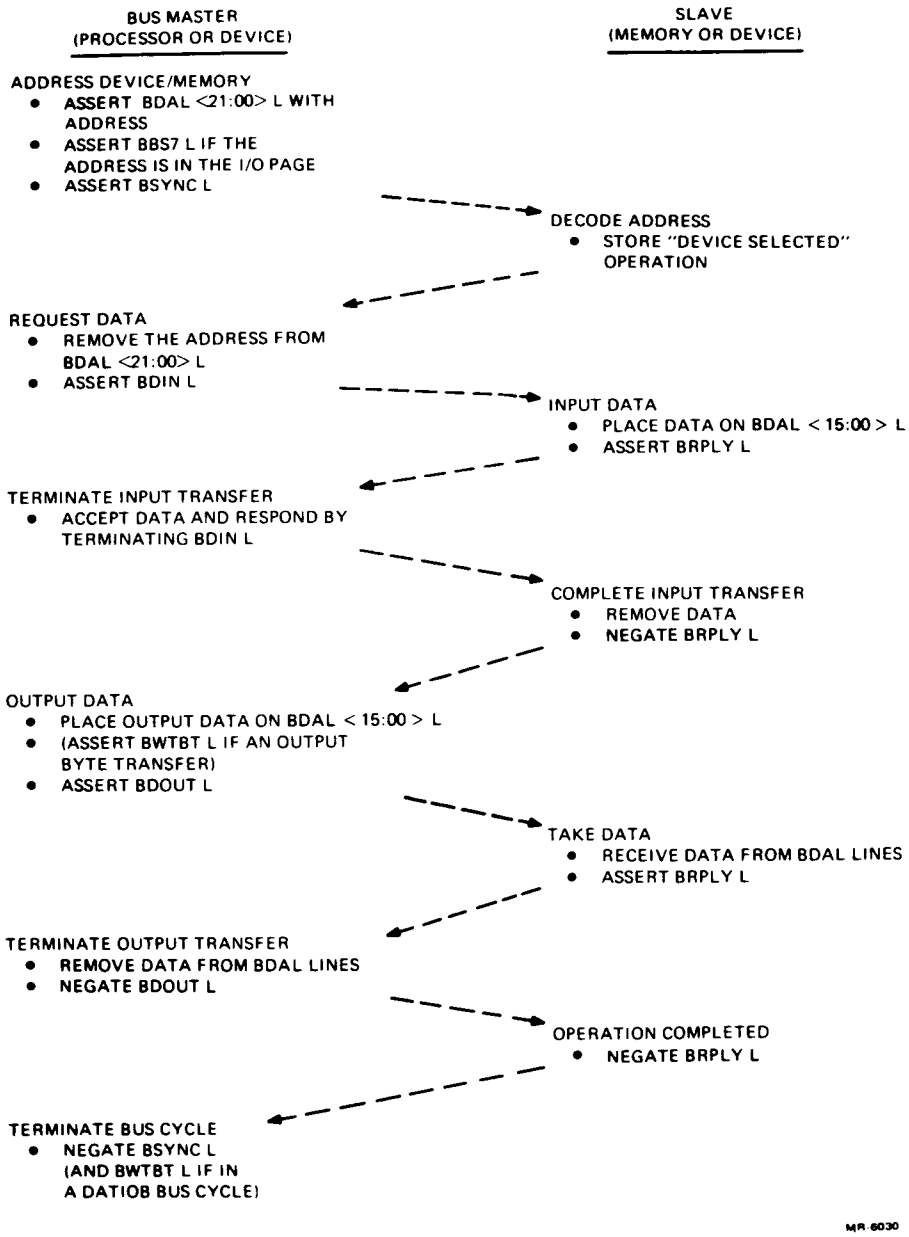
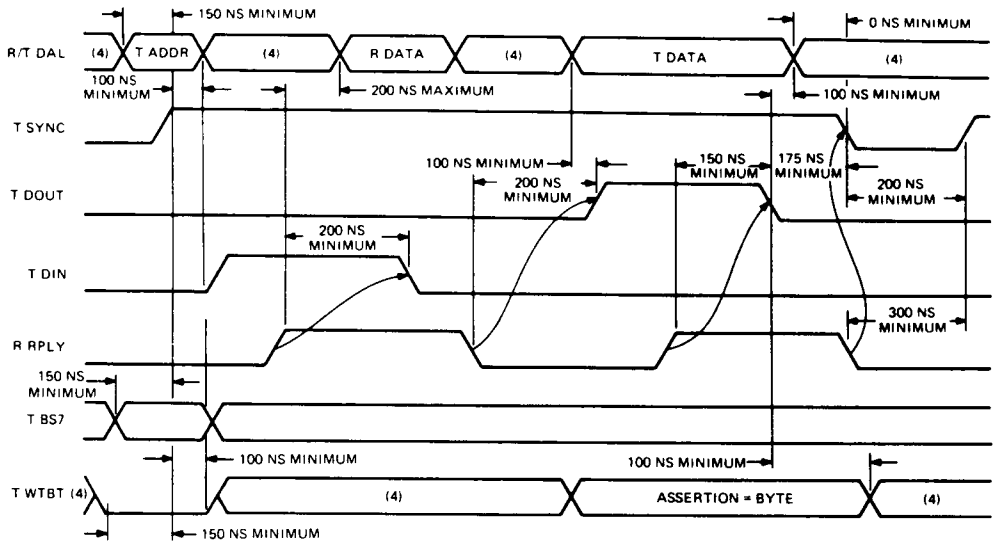
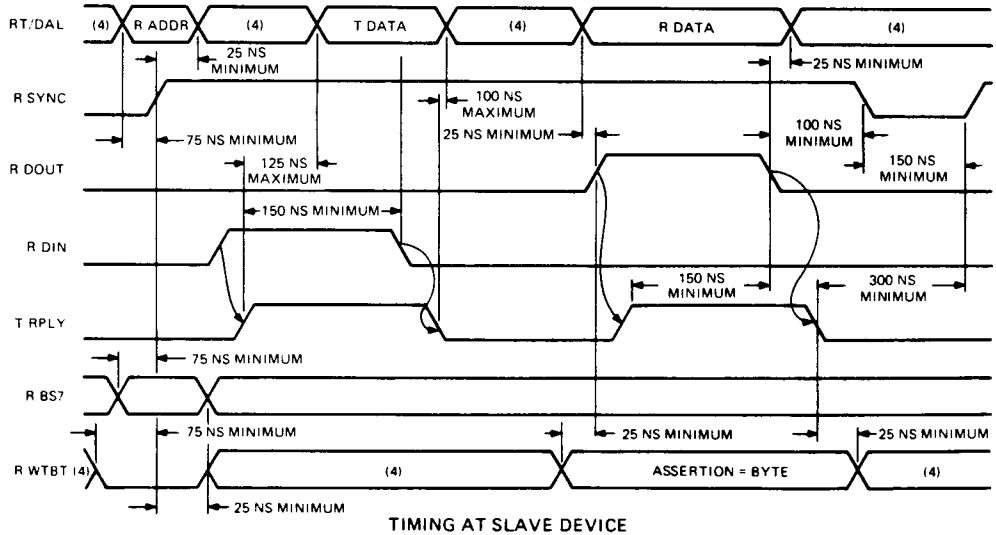


Figure A-5 DATIO or DATIOB Bus Cycle



TIMING AT MASTER DEVICE



TIMING AT SLAVE DEVICE

NOTES

1. TIMING SHOWN AT REQUESTING DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS
2. SIGNAL NAME PREFIXES ARE DEFINED BELOW:
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT
3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX.
4. DON'T CARE CONDITION.

MR 6036

Figure A-6 DATIO or DATIOB Bus Cycle Timing

A.4 DIRECT MEMORY ACCESS

The direct memory access (DMA) capability allows direct data transfer between I/O devices and memory. This is useful when using mass storage devices (for example, disks) that move large blocks of data to and from memory. A DMA device needs to know only the starting address in memory, the starting address in mass storage, the length of the transfer, and whether the operation is read or write. When this information is available, the DMA device can transfer data directly to or from memory. Since most DMA devices must perform data transfers in rapid succession or lose data, DMA devices are provided the highest priority.

DMA is accomplished after the processor (normally bus master) has passed bus mastership to the highest-priority DMA device that is requesting the bus. The processor arbitrates all requests and grants the bus to the DMA device electrically closest to it. A DMA device remains bus master until it relinquishes its mastership. The following control signals are used during bus arbitration.

BDMGI L	DMA grant input
BDMGO L	DMA grant output
BDMR L	DMA request line
BSACK L	Bus grant acknowledge

A.4.1 DMA Protocol

A DMA transaction can be divided into three phases:

1. Bus mastership acquisition phase
2. Data transfer phase
3. Bus mastership relinquishment phase.

During the bus mastership acquisition phase, a DMA device requests the bus by asserting BDMR L. The processor arbitrates the request and initiates the transfer of bus mastership by asserting BDMGO L.

The maximum time between BDMR L assertion and BDMGO L assertion is DMA latency. This time is processor-dependent. BDMGO L/BDMGI L is one signal that is daisy-chained through each module in the backplane. It is driven out of the processor on the BDMGO L pin, enters each module on the BDMGI L pin, and exits on the BDMGO L pin. This signal passes through the modules in descending order of priority until it is stopped by the requesting device. The requesting device blocks the output of BMDGO L and asserts BSACK L. If BDMR L is continuously asserted, the bus hangs.

During the data transfer phase, the DMA device continues asserting BSACK L. The actual data transfer is performed as described earlier.

The DMA device can assert BSYNC L for a data transfer 250 ns (minimum) after it received BDMGI L and its BSYNC L bus receiver becomes negated.

During the bus mastership relinquishment phase, the DMA device gives up the bus by negating BSACK L. This occurs after completing (or aborting) the last data transfer cycle (BRPLY L negated). BSACK L may be negated up to a maximum of 300 ns before negating BSYNC L. Figure A-7 shows the DMA protocol, and Figure A-8 shows DMA request/grant timing.

NOTE

If multiple data transfers are performed during this phase, consideration must be given to the use of the bus for other system functions, such as memory refresh (if required).

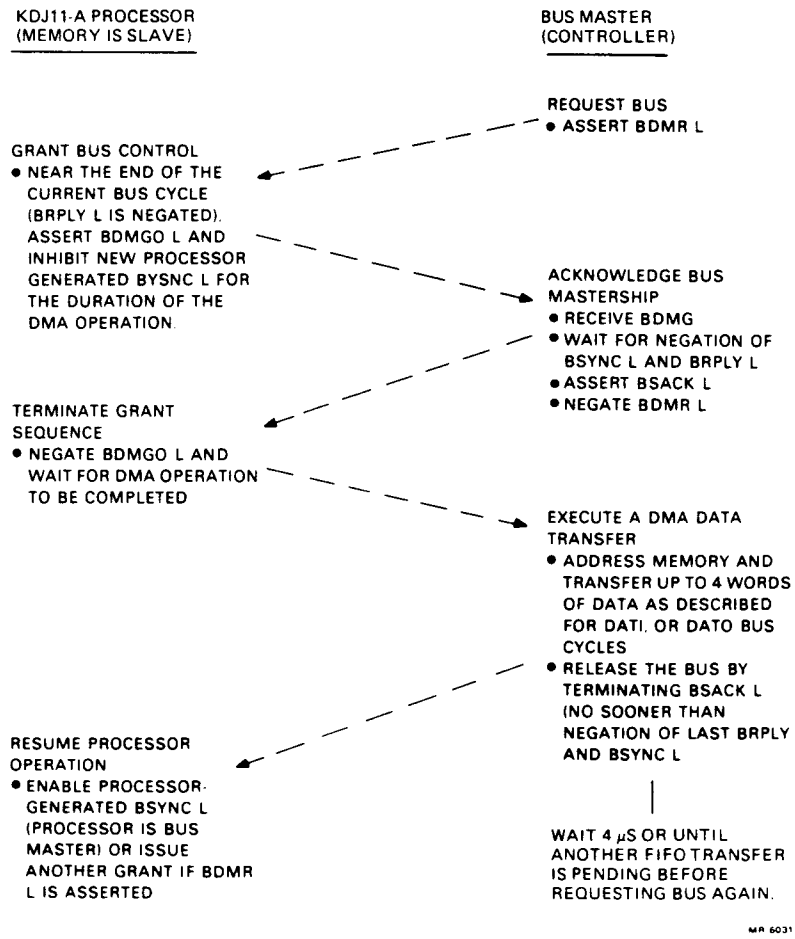


Figure A-7 DMA Protocol

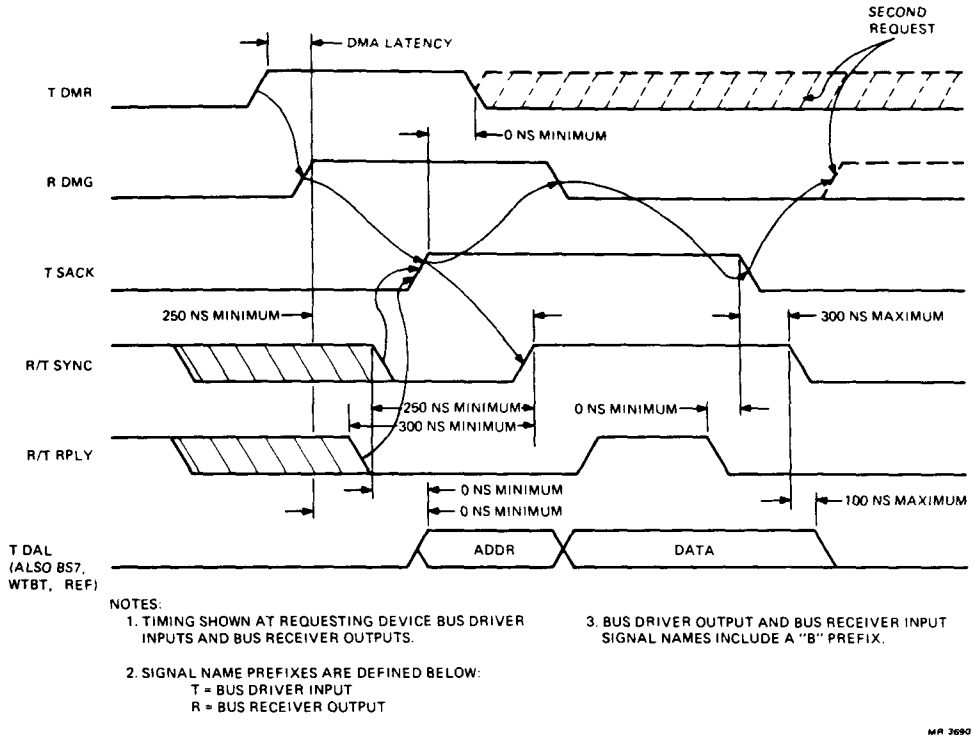


Figure A-8 DMA Request/Grant Timing

A.4.2 Block Mode DMA

For increased throughput, block mode DMA may be implemented on a device for use with memories that support this type of transfer. In a block mode transaction, the starting memory address is asserted, followed by data for that address, and data for consecutive addresses.

By eliminating the assertion of the address for each data word, the transfer rate is almost doubled. The DATBI and DATBO bus cycles are described below.

A.4.2.1 DATBI -- The device addressing portion of the cycle is the same as described earlier for other bus cycles. (See Figure A-9.) The bus master gates BDAL<21:00>, BBS7, and the negation of BWTBT onto the bus.

The master asserts the first BDIN 100 ns after BSYNC, and asserts BBS7 a maximum of 50 ns after asserting BDIN for the first time. BBS7 is a request to the slave for a block mode transfer. BBS7 remains asserted until a maximum of 50 ns after the assertion of BDIN for the last time. BBS7 may be gated as soon as the conditions for asserting BDIN are met.

The slave asserts BRPLY a minimum of 0 ns (8 ns maximum to avoid bus timeout) after receiving BDIN. It asserts BREF concurrently with BRPLY if it is a block mode device capable of supporting another BDIN after the current one. The slave gates BDAL<15:00> onto the bus 0 ns (minimum) after the assertion of BDIN, and 125 ns (maximum) after the assertion of BRPLY.

The master receives the stable data from 200 ns (maximum) after the assertion of BRPLY until 20 ns (minimum) after the negation of BDIN. It negates BDIN 200 ns (minimum) after the assertion of BRPLY.

The slave negates BRPLY 0 ns (minimum) after the negation of BDIN. If BBS7 and BREF are both asserted when BRPLY is negated, the slave prepares for another BDIN cycle. BBS7 is stable from 125 ns after BDIN is asserted until 150 ns after BRPLY is negated. The master asserts BDIN 150 ns (minimum) after BRPLY is negated, and the cycle is continued as before. (BBS7 remains asserted and the slave responds to BDIN with BRPLY and BREF.) BREF is stable from 75 ns after BRPLY is asserted until 20 ns (minimum) after BDIN is negated.

If BBS7 and BREF are not both asserted when BRPLY is negated, the slave removes the data from the bus 0 ns (minimum) and 100 ns (maximum) after negating BRPLY. The master negates BSYNC 250 ns (minimum) after the assertion of the last BRPLY, and 0 ns (minimum) after the negation of that BRPLY.

A.4.2.2 DATBO -- The device addressing portion of the cycle is the same as shown in Figure A-10. The bus master gates BDAL<21:00>, BBS7, and the assertion of BWTBT onto the bus.

A minimum of 100 ns after BSYNC is asserted, data on BDAL<15:00> and the negated BWTBT are put onto the bus. The master then asserts BDOUT a minimum of 100 ns after gating the data.

The slave receives stable data and BWTBT from 25 ns (minimum) before the assertion of BDOUT to 25 ns (minimum) after the negation of BDOUT. The slave asserts BRPLY 0 ns (minimum) after receiving BDOUT. It also asserts BREF concurrently with BRPLY if it is a block mode device capable of supporting another BDOUT after the current one.

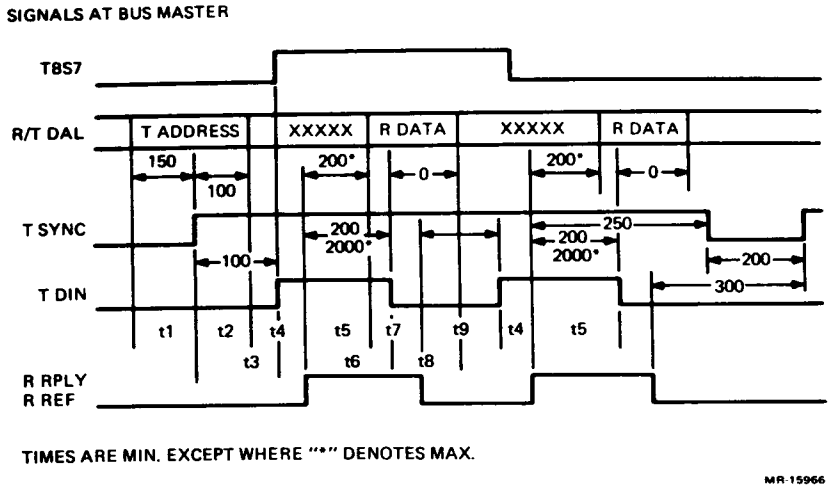


Figure A-9 DATBI Bus Cycle Timing

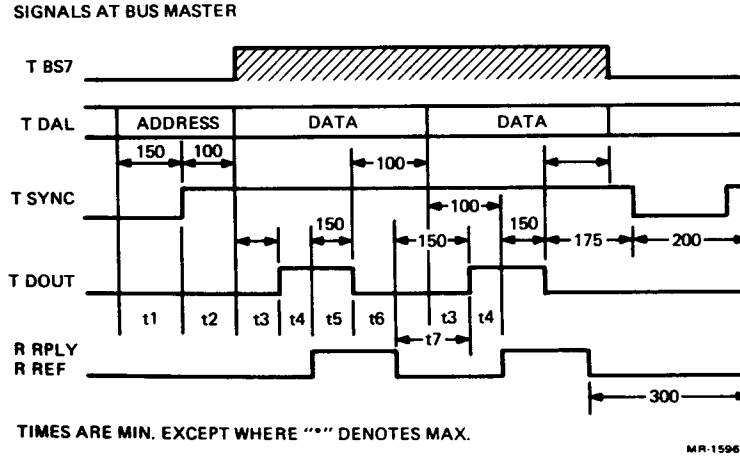


Figure A-10 DATBO Bus Cycle Timing

The master negates BDOUT 150 ns (minimum) after the assertion of BRPLY. If BREF was asserted when BDOUT was negated, and the master wants to transmit more data in this block mode cycle, the new data is gated onto the bus 100 ns (minimum) after BDOUT is negated. BREF is stable from 75 ns (maximum) after BRPLY is asserted until 20 ns (minimum) after BDOUT is negated. The master asserts BDOUT 100 ns (minimum) after gating new data onto the bus and 150 ns minimum after BRPLY negates. The cycle continues as before.

If BREF was not asserted when BDOUT was negated, or if the bus master does not want to transmit more data in this cycle, the master removes data from the bus 100 ns (minimum) after negating BDOUT. The slave negates BRPLY 0 ns (minimum) after negating BDOUT. The bus master negates BSYNC 175 ns (minimum) after negating BDOUT, and 0 ns (minimum) after the negation of BRPLY.

A.4.3 DMA Guidelines

1. Systems with memory refresh over the bus must not include devices that perform more than one transfer per acquisition.

2. Bus masters that do not use block mode are limited to four DATI, four DATO, or two DATIO transfers per acquisition.

3. Block mode bus masters that do not monitor BDMR are limited to eight transfers per acquisition.

4. If BDMR is not asserted after the seventh transfer, block mode bus masters that do monitor BDMR may continue making transfers until the bus slave fails to assert BREF, or until they reach the total maximum of 16 transfers. Otherwise, they stop after eight transfers.

A.5 INTERRUPTS

The interrupt capability of the Q22-Bus allows an I/O device to temporarily suspend (interrupt) current program execution and divert processor operation to service the requesting device. The processor inputs a vector from the device to start the service routine (handler). Like the device register address, hardware fixes the device vector at locations within a designated range below location 001000. The vector indicates the first of a pair of addresses. The processor reads the contents of the first address, the starting address of the interrupt handler. The contents of the second address is a new processor status word (PS).

The new PS can raise the interrupt priority level, thereby preventing lower-level interrupts from breaking into the current interrupt service routine. Control is returned to the interrupted program when the interrupt handler is ended. The original interrupted program's address (PC) and its associated PS are stored on a stack. The original PC and PS are restored by a return from interrupt (RTI or RTT) instruction at the end of the handler. The use of the stack and the Q22-Bus interrupt scheme can allow interrupts to occur within interrupts (nested interrupts), depending on the PS.

Interrupts can be caused by Q22-Bus options or the MicroVAX CPU. Those interrupts that originate from within the processor are called "traps". Traps are caused by programming errors, hardware errors, special instructions, and maintenance features.

The following are Q22-Bus signals used in interrupt transactions.

BIRQ4 L	Interrupt request priority level 4
BIRQ5 L	Interrupt request priority level 5
BIRQ6 L	Interrupt request priority level 6
BIRQ7 L	Interrupt request priority level 7
BIAKI L	Interrupt acknowledge input
BIAKO L	Interrupt acknowledge output
BDAL<21:00>	Data/address lines
BDIN L	Data input strobe
BRPLY L	Reply

A.5.1 Device Priority

The Q22-Bus supports the following two methods of device priority.

1. Distributed Arbitration -- Priority levels are implemented on the hardware. When devices of equal priority level request an interrupt, priority is given to the device electrically closest to the processor.

2. Position-Defined Arbitration -- Priority is determined solely by electrical position on the bus. The closer a device is to the processor, the higher its priority is.

A.5.2 Interrupt Protocol

Interrupt protocol on the Q22-Bus has three phases: the interrupt request phase, interrupt acknowledge and priority arbitration phase, and interrupt vector transfer phase. Figure A-11 shows the interrupt request/acknowledge sequence.

The interrupt request phase begins when a device meets its specific conditions for interrupt requests. For example, the device is ready, done, or an error has occurred. The interrupt enable bit in a device status register must be set. The device then initiates the interrupt by asserting the interrupt request line(s). BIRQ4 L is the lowest hardware priority level and is asserted for all interrupt requests for compatibility with previous Q22 processors. The level at which a device is configured must also be asserted. A special case exists for level 7 devices that must also assert level 6. For an explanation, refer to the discussion below on arbitration involving the 4-level scheme.

Interrupt Level	Lines Asserted by Device
4	BIRQ4 L
5	BIRQ4 L, BIRQ5 L
6	BIRQ4 L, BIRQ6 L
7	BIRQ4 L, BIRQ6 L, BIRQ7 L

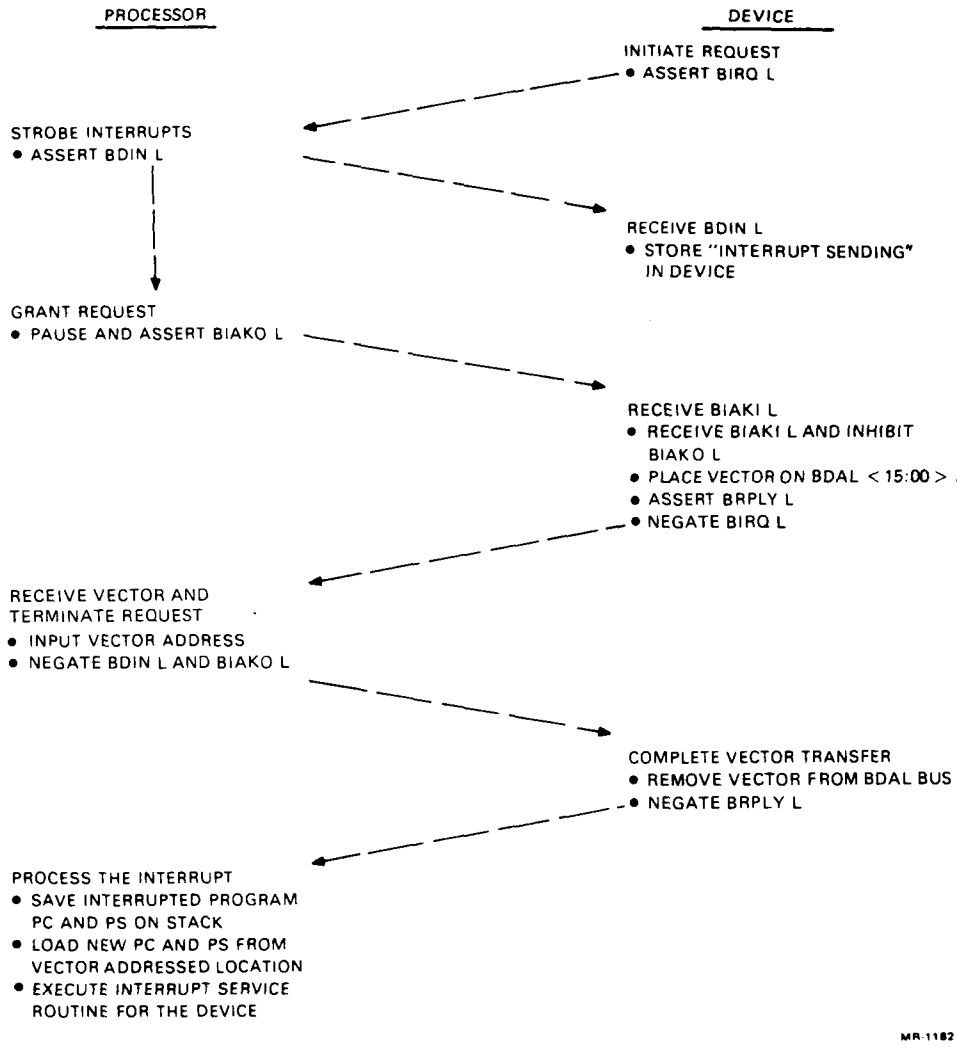


Figure A-11 Interrupt Request/Acknowledge Sequence

The interrupt request line remains asserted until the request is acknowledged.

During the interrupt acknowledge and priority arbitration phase, the LSI-11/23 processor acknowledges interrupts under the following conditions.

1. The device interrupt priority is higher than the current PS<7:5>.

2. The processor has completed instruction execution and no additional bus cycles are pending.

The processor acknowledges the interrupt request by asserting BDIN L, and 150 ns (minimum) later asserting BIAKO L. The device electrically closest to the processor receives the acknowledge on its BIAKI L bus receiver.

At this point, the two types of arbitration must be discussed separately. If the device that receives the acknowledge uses the 4-level interrupt scheme, it reacts as follows.

1. If not requesting an interrupt, the device asserts BIAKO L and the acknowledge propagates to the next device on the bus.

2. If the device is requesting an interrupt, it must check that no higher-level device is currently requesting an interrupt. This is done by monitoring higher-level request lines.The table below lists the lines that need to be monitored by devices at each priority level.

In addition to asserting levels 7 and 4, level 7 devices must drive level 6. This is done to simplify the monitoring and arbitration by level 4 and 5 devices. In this protocol, level 4 and 5 devices need not monitor level 7 because level 7 devices assert level 6. Level 4 and 5 devices become aware of a level 7 request because they monitor the level 6 request. This protocol has been optimized for level 4, 5, and 6 devices, since level 7 devices are very seldom necessary.

Device Priority Level	Line(s) Monitored
4	BIRQ5, BIRQ6
5	BIRQ6
6	BIRQ7
7	--

3. If no higher-level device is requesting an interrupt, the acknowledge is blocked by the device. (BIAKO L is not asserted.) Arbitration logic within the device uses the leading edge of BDIN L to clock a flip-flop that blocks BIAKO L. Arbitration is won, and the interrupt vector transfer phase begins.

4. If a higher-level request line is active, the device disqualifies itself and asserts BIAKO L to propagate the acknowledge to the next device along the bus.

Signal timing must be considered carefully when implementing 4-level interrupts. See Figure A-12.

If a single-level interrupt device receives the acknowledge, it reacts as follows.

1. If not requesting an interrupt, the device asserts BIAKO L and the acknowledge propagates to the next device on the bus.

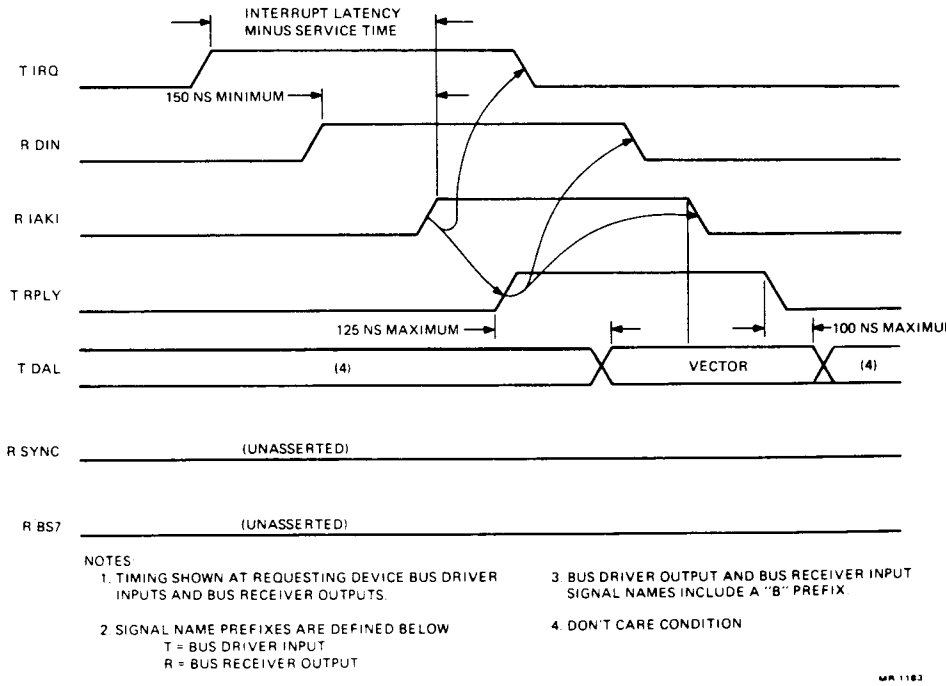


Figure A-12 Interrupt Protocol Timing

2. If the device was requesting an interrupt, the acknowledge is blocked using the leading edge of BDIN L, and arbitration is won. The interrupt vector transfer phase begins.

The interrupt vector transfer phase is enabled by BDIN L and BIAKI L. The device responds by asserting BRPLY L and its BDAL<15:00> L bus driver inputs with the vector address bits. The BDAL bus driver inputs must be stable within 125 ns (maximum) after BRPLY L is asserted. The processor then inputs the vector address and negates BDIN L and BIAKO L. The device then negates BRPLY L and 100 ns (maximum) later removes the vector address bits. The processor then enters the device's service routine.

NOTE

Propagation delay from BIAKI L to BIAKO L must not be greater than 500 ns per Q22-Bus slot.

The device must assert BRPLY L within 10 μs (maximum) after the processor asserts BIAKI L.

A.5.3 Q22-Bus 4-Level Interrupt Configurations

If you have high-speed peripherals and desire better software performance, you can use the 4-level interrupt scheme. Both position-independent and position-dependent configurations can be used with the 4-level interrupt scheme.

Figure A-13 shows the position-independent configuration. This allows peripheral devices that use the 4-level interrupt scheme to be placed in the backplane in any order. These devices must send out interrupt requests and monitor higher-level request lines as described. The level 4 request is always asserted from a requesting device regardless of priority. If two or more devices of equally high priority request an interrupt, the device physically closest to the processor wins arbitration. Devices that use the single-level interrupt scheme must be modified, or placed at the end of the bus, for arbitration to function properly.

Figure A-14 shows the position-dependent configuration. This configuration is simpler to implement. A constraint is that peripheral devices must be inserted with the highest-priority device located closest to the processor, and the remaining devices placed in the backplane in decreasing order of priority (with the lowest-priority devices farthest from the processor). With this configuration, each device has to assert only its own level and level 4. Monitoring higher-level request lines is unnecessary. Arbitration is achieved through the physical positioning of each device on the bus. Single-level interrupt devices on level 4 should be positioned last on the bus.

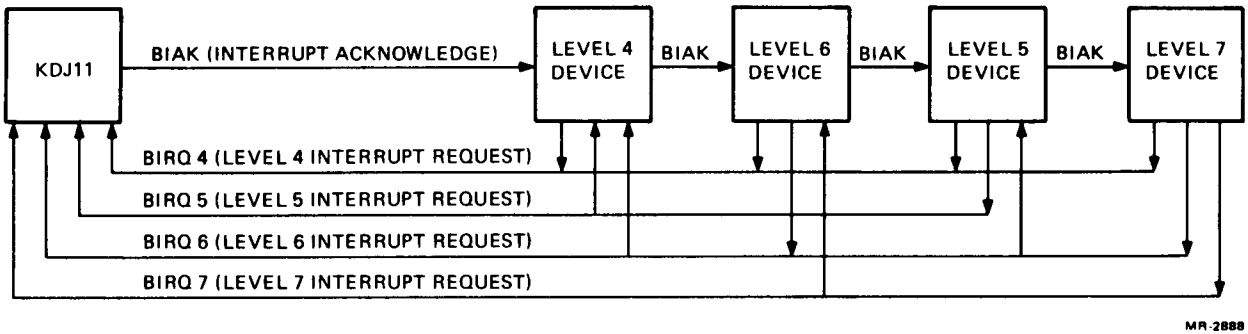


Figure A-13 Position-Independent Configuration

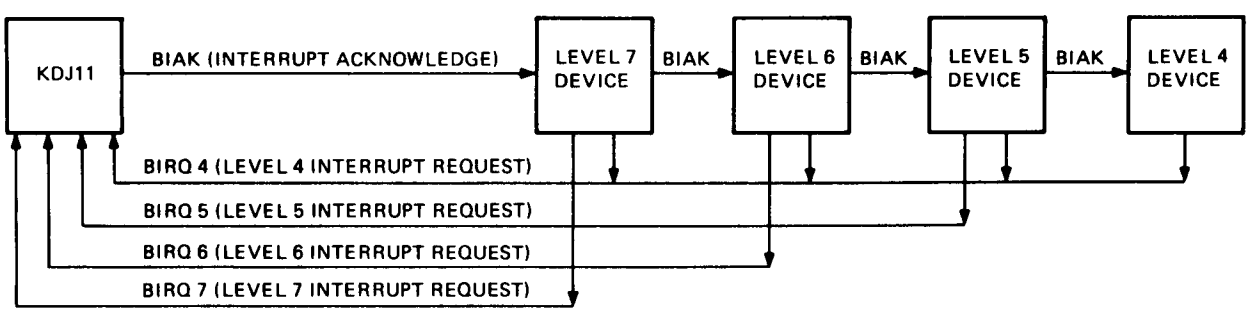


Figure A-14 Position-Dependent Configuration

A.6 CONTROL FUNCTIONS

The following Q22-Bus signals provide control functions.

BREF L	Memory refresh (also block mode DMA)
BHALT L	Processor halt
BINIT L	Initialize
BPOK H	Power OK
BDCOK H	DC power OK

A.6.1 Memory Refresh

If BREF is asserted during the address portion of a bus data transfer cycle, it causes all dynamic MOS memories to be addressed simultaneously. The sequence of addresses required for refreshing the memories is determined by the specific requirements for each memory. The complete memory refresh cycle consists of a series of refresh bus transactions. A new address is used for each transaction. A complete memory refresh cycle must be completed within 1 or 2 ms. Multiple data transfers by DMA devices must be avoided since they could delay memory refresh cycles. This type of refresh is done only for memories that do not perform on-board refresh.

A.6.2 Halt

Assertion of BHALT L for at least 25 ns interrupts the processor, which stops program execution and forces the processor unconditionally into console I/O mode.

A.6.3 Initialization

Devices along the bus are initialized when BINIT L is asserted. The processor can assert BINIT L as a result of executing a reset instruction as part of a power-up or power-down sequence. BINIT L is asserted for approximately 10 μs when reset is executed.

A.6.4 Power Status

Power status protocol is controlled by two signals, BPOK H and BDCOK H. These signals are driven by an external device (usually the power supply).

A.6.5 BDCOK H

When asserted, this indicates that dc power has been stable for at least 3 ms. Once asserted, this line remains asserted until the power fails. It indicates that only 5 μs of dc power reserve remains.

A.6.6 BPOK H

When asserted, this indicates there is at least an 8 ms reserve of dc power, and that BDCOK H has been asserted for at least 70 ms. Once BPOK has been asserted, it must remain asserted for at least 3 ms. The negation of this line, the first event in the power-fail sequence, indicates that power is failing and that only 4 ms of dc power reserve remains.

A.6.7 Power-Up/Down Protocol

Power-up protocol begins when the power supply applies power with BDCOK H negated. This forces the processor to assert BINIT L. When the dc voltages are stable, the power supply or other external device asserts BDCOK H. The processor responds by clearing the PS, floating point status register (FPS), and floating point exception register (FEC). BINIT L is asserted for 12.6 μs, and then negated for 110 μs. The processor continues to test for BPOK H until it is asserted. The power supply asserts BPIK H 70 ms (minimum) after BDCOK H is asserted. The processor then performs its power-up sequence. Normal power must be maintained at least 3.0 ms before a power-down sequence can begin.

A power-down sequence begins when the power supply negates BPOK H. When the current instruction is completed, the processor traps to a power-down routine at location 24. The end of the routine is terminated with a halt instruction to avoid any possible memory corruption as the dc voltages decay.

When the processor executes the halt instruction, it tests the BPOK H signal. If BPOK H is negated, the processor enters the power-up sequence. It clears internal registers, generates BINIT L, and continues to check for the assertion of BPOK H. If it is asserted and dc voltages are still stable, the processor performs the rest of the power-up sequence. Figure A-15 shows power-up/power-down timing.

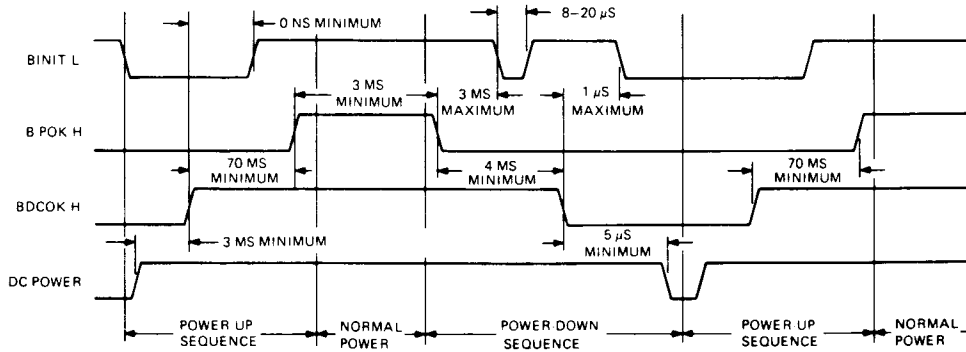


Figure A-15 Power-Up/Power-Down Timing

A.7 Q22-BUS ELECTRICAL CHARACTERISTICS

SIGNAL LEVEL SPECIFICATION

Input Logic Levels:

TTL Logical Low	0.8 Vdc (maximum)
TTL Logical High	2.0 Vdc (minimum)

Output Logic Levels:

TTL Logical Low	0.4 Vdc (maximum)
TTL Logical High	2.4 Vdc (minimum)

A.7.1 Load Definition

AC loads make up the maximum capacitance allowed per signal line to ground. A unit load is defined as 9.35 pF of capacitance. DC loads are defined as maximum current allowed with a signal line driver asserted or unasserted. A unit load is defined as 210 μA in the unasserted state.

A.7.2 120-Ohm Q22-Bus

The electrical conductors interconnecting the bus device slots are treated as transmission lines. A uniform transmission line, terminated in its characteristic impedance, propagates an electrical signal without reflections. Since bus drivers, receivers, and wiring connected to the bus have finite resistance and nonzero reactance, the transmission line impedance is not uniform, and introduces distortions into pulses propagated along it. Passive components of the Q22-Bus (such as wiring, cabling, and etched signal conductors) are designed to have a nominal characteristic impedance of 120 ohms.

The maximum length of interconnecting cable, excluding wiring within the backplane, is limited to 4.88 m (16 ft).

A.7.3 Bus Drivers

Devices driving the 120-ohm Q22-Bus must have open collector outputs and meet the following specifications.

DC SPECIFICATIONS

Output low voltage when sinking 70 mA of current: 0.7 V (maximum).

Output high leakage current when connected to 3.8 Vdc: 25 μA (even if no power is applied, except for BDCOK H and BPOK H).

These conditions must be met at worst-case supply temperature, and input signal levels.

AC SPECIFICATIONS

Bus driver output pin capacitance load: Not to exceed 10 pF.

Propagation delay: Not to exceed 35 ns.

Skew (difference in propagation time between slowest and fastest gate): Not to exceed 25 ns.

Rise/fall times: Transition time (from 10% to 90% for positive transition, 90% to 10% for negative transition) must be no faster than 10 ns.

A.7.4 Bus Receivers

Devices that receive signals from the 120-ohm Q22-Bus must meet the following requirements.

DC SPECIFICATIONS

Input low voltage (maximum): 1.3 V.

Input high voltage (minimum): 1.7 V.

Maximum input current when connected to 3.8 Vdc: 80 μA (even if no power is applied).

These specifications must be met at worst-case supply voltage, temperature, and output signal conditions.

AC SPECIFICATIONS

Bus receiver input pin capacitance load: Not to exceed 10 pF.

Propagation delay: Not to exceed 35 ns.

Skew (difference in propagation time between slowest and fastest gate): Not to exceed 25 ns.

A.7.5 Bus Termination

The 120-ohm Q22-Bus must be terminated at each end by an appropriate terminator, as shown in Figure A-16. This is to be done as a voltage divider with its Thevenin equivalent equal to 120 ohms and 3.4 V (nominal). This type of termination is provided by an REV11-A refresh/boot/terminator, BDV11-AA, KPV11-B, TEV11, or by certain backplanes and expansion cards.

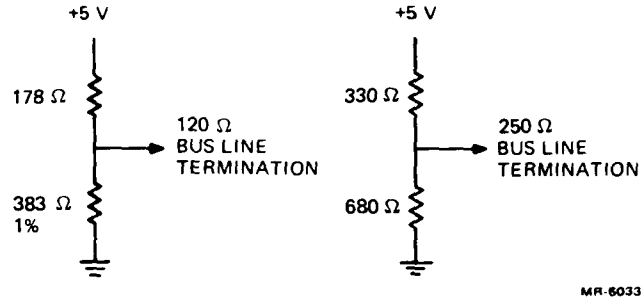


Figure A-16 Bus Line Terminations

Each of the several Q22-Bus lines (all signals whose mnemonics start with the letter B) must see an equivalent network with the following characteristics at each end of the bus.

Input impedance (with respect to ground)	120 ohm +5%, -15%
Open circuit voltage	3.4 Vdc +5%
Capacitance load	Not to exceed 30 pF

NOTE

The resistive termination may be provided by the combination of two modules. (The processor module supplies 220 ohms to ground. This, in parallel with another 220-ohm card, provides 120 ohms.) Both terminators must reside physically within the same backplane.

A.7.6 Bus Interconnecting Wiring

A.7.6.1 Backplane Wiring -- The wiring that connects all device interface slots on the Q22-bus must meet the following specifications.

1. The conductors must be arranged so that each line exhibits a characteristic impedance of 120 ohms (measured with respect to the bus common return).

2. Crosstalk between any two lines must be no greater than 5 percent. Note that worst-case crosstalk is manifested by simultaneously driving all but one signal line and measuring the effect on the undriven line.

3. DC resistance of the signal path, as measured between the near-end terminator and the far-end terminator module (including all intervening connectors, cables, backplane wiring, connector-module etch, etc.) must not exceed 20 ohms.

4. DC resistance of the common return path, as measured between the near-end terminator and the far-end terminator module (including all intervening connectors, cables, backplane wiring, connector-module etch, etc.) must not exceed an equivalent of 2 ohms per signal path. Thus, the composite signal return path dc resistance must not exceed 2 ohms divided by 40 bus lines, or 50 milliohms. Note that although this common return path is nominally at ground potential, the conductance must be part of the bus wiring. The specified low impedance return path must be provided by the bus wiring as distinguished from the common system or power ground path.

A.7.6.2 **Intra-Backplane Bus Wiring** -- The wiring that connects the bus connector slots within one contiguous backplane is part of the overall bus transmission line. Owing to implementation constraints, the nominal characteristic impedance of 120 ohms may not be achievable. Distributed wiring capacitance in excess of the amount required to achieve the nominal 120-ohm impedance may not exceed 60 pF per signal line per backplane.

A.7.6.3 **Power and Ground** -- Each bus interface slot has connector pins assigned for the following dc voltages. The maximum allowable current per pin is 1.5 A. +5 Vdc must be regulated to 5 percent with a maximum ripple of 100 mV pp. +12 Vdc must be regulated to 3 percent with a maximum ripple of 200 mV pp.

- +5 Vdc -- Three pins (4.5 A maximum per bus device slot)
- +12 Vdc -- Two pins (3.0 A maximum per bus device slot)
- Ground -- Eight pins (shared by power return and signal return)

NOTE

Power is not bussed between backplanes on any interconnecting bus cables.

A.8 SYSTEM CONFIGURATIONS

Q22-Bus systems can be divided into two types:

1. Systems containing one backplane
2. Systems containing multiple backplanes

Before configuring any system, three characteristics for each module in the system must be known:

- Power consumption -- +5 Vdc and +12 Vdc current requirements.
- AC bus loading -- The amount of capacitance a module presents to a bus signal line. AC loading is expressed in terms of ac loads, where one ac load equals 9.35 pF of capacitance.
- DC bus loading -- The amount of dc leakage current a module presents to a bus signal when the line is high (undriven). DC loading is expressed in terms of dc loads, where one dc load equals 210 μA (nominal).

Power consumption, ac loading, and dc loading specifications for each module are included in the Microcomputer Interface Handbook.

NOTE

The ac and dc loads and the power consumption of the processor module, terminator module, and backplane must be included in determining the total loading of a backplane.

Rules for configuring single-backplane systems:

1. When using a processor with 220-ohm termination, the bus can accommodate modules that have up to 20 ac loads (total) before additional termination is required. (See Figure A-17.) If more than 20 ac loads are included, the other end of the bus must be terminated with 120 ohms, and then up to 35 ac loads may be present.

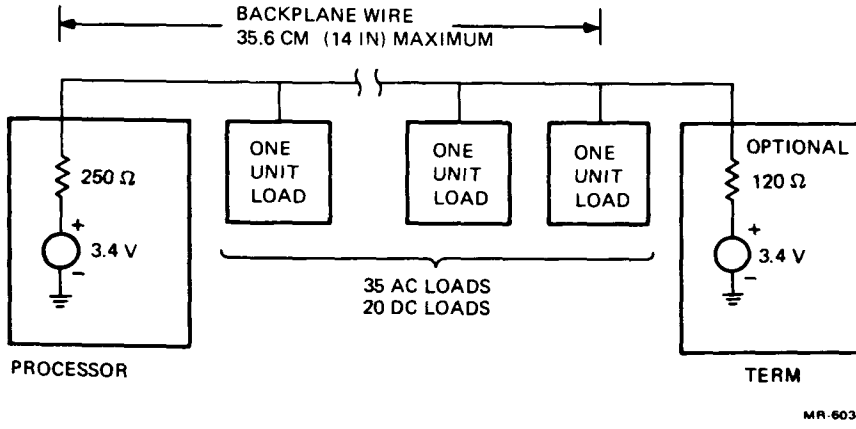


Figure A-17 Single-Backplane Configuration

2. With 120-ohm processor termination, up to 35 ac loads can be used without additional termination. If 120-ohm bus termination is added, up to 45 ac loads can be configured in the backplane.

3. The bus can accommodate modules up to 20 dc loads (total).

4. The bus signal lines on the backplane can be up to 35.6 cm (14 in) long.

Rules for configuring multiple-backplane systems:

1. Figure A-18 shows that up to three backplanes may make up the system.

2. The signal lines on each backplane can be up to 25.4 cm (10 in) long.

3. Each backplane can accommodate modules that have up to 22 ac loads (total). Unused ac loads from one backplane may not be added to another backplane if the second backplane loading will exceed 22 ac loads. It is desirable to load backplanes equally, or with the highest ac loads in the first and second backplanes.

4. DC loading of all modules in all backplanes cannot exceed 20 loads (total).

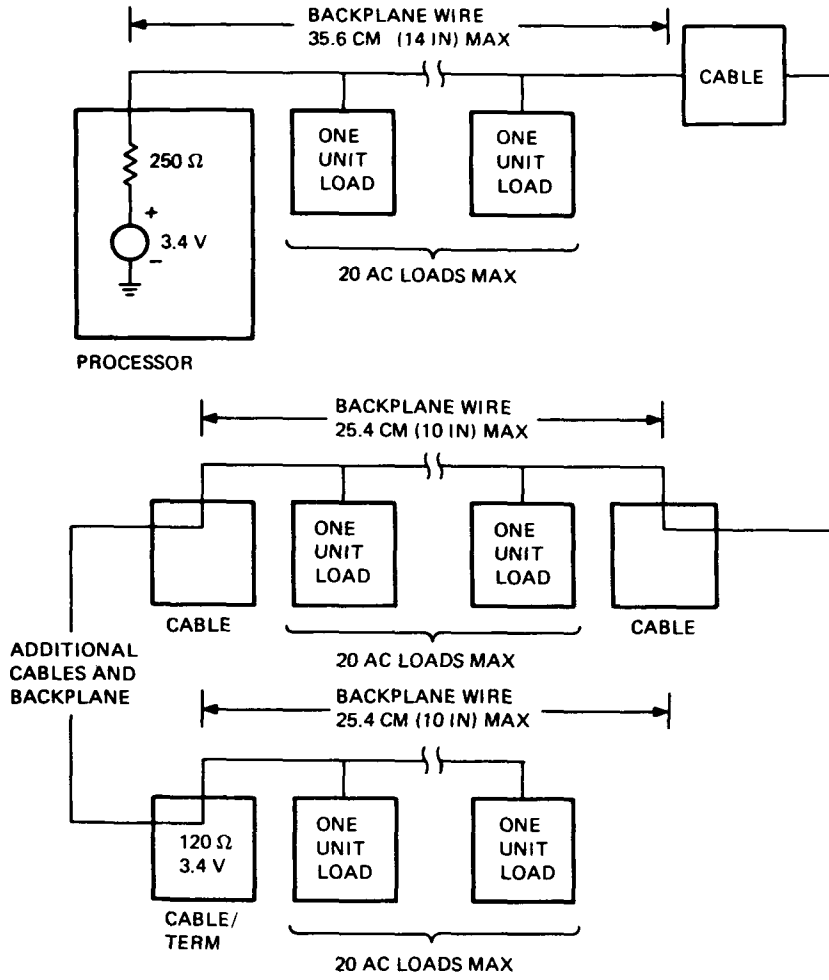
5. Both ends of the bus must be terminated with 120 ohms. This means the first and last backplanes must have an impedance of 120 ohms. To achieve this, each backplane may be lumped together as a single point. The resistive termination may be provided by a combination of two modules in the backplane - the processor providing 220 ohms to ground in parallel with an expansion paddle card providing 250 ohms to give the needed 120-ohm termination. Alternately, a processor with 120-ohm termination would need no additional termination on the paddle card to attain 120 ohms in the first box. The 120-ohm termination in the last box can be provided in two ways: the termination resistors may reside either on the expansion paddle card, or on a bus termination card (such as the BDV11).

6. The cable(s) connecting the first two backplanes is (are) 61 cm (2 ft) or more in length.

7. The cable(s) connecting the second backplane to the third backplane is (are) 122 cm (4 ft) longer or shorter than the cable(s) connecting the first and second backplanes.

8. The combined length of both cables cannot exceed 4.88 m (16 ft).

9. The cables used must have a characteristic impedance of 120 ohms.



NOTES:

1. TWO CABLES (MAX) 4.88 M (16 FT) (MAX) TOTAL LENGTH.
2. 20 DC LOADS TOTAL (MAX).

MR-6035

Figure A-18 Multiple-Backplane Configuration

A.8.1 Power Supply Loading

Total power requirements for each backplane can be determined by obtaining the total power requirements for each module in the backplane. Obtain separate totals for +5 V and +12 V power. Power requirements for each module are specified in the Microcomputer Interfaces Handbook.

When distributing power in multiple-backplane systems, do not attempt to distribute power via the Q22-Bus cables. Provide separate, appropriate power wiring from each power supply to each backplane. Each power supply should be capable of asserting BPOK H and BDCOK H signals according to bus protocol; this is required if automatic power-fail/restart programs are implemented, or if specific peripherals require an orderly power-down halt sequence. The proper use of BPOK H and BDCOK H signals is strongly recommended.

A.9 MODULE CONTACT FINGER IDENTIFICATION

Digital's plug-in modules all use the same contact finger (pin) identification system. A typical pin is shown in Figure A-19.

The Q22-Bus is based on the use of quad-height modules that plug into a 2-slot bus connector. Each slot contains 36 lines (18 lines on both the component side and the solder side of the circuit board).

Slots, row A, and row B include a numeric identifier for the side of the module. The component side is designated side 1, the solder side is designated side 2, as shown in Figure A-20. Letters ranging from A through V (excluding G, I, O, and Q) identify a particular pin on a side of a slot. Table A-4 lists and identifies the bus pins of the quad-height module. A bus pin identifier ending with a 1 is found on the component side of the board, while a bus pin identifier ending with a 2 is found on the solder side of the board.

The positioning notch between the two rows of pins mates with a protrusion on the connector block for correct module positioning.

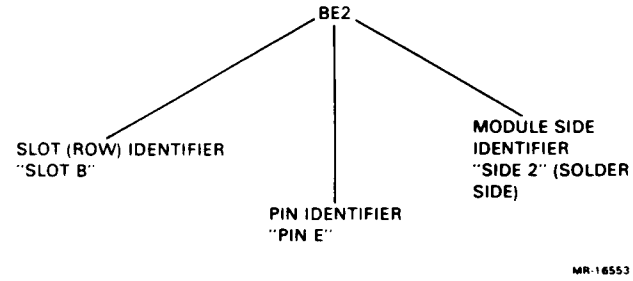


Figure A-19 Typical Pin Identification System

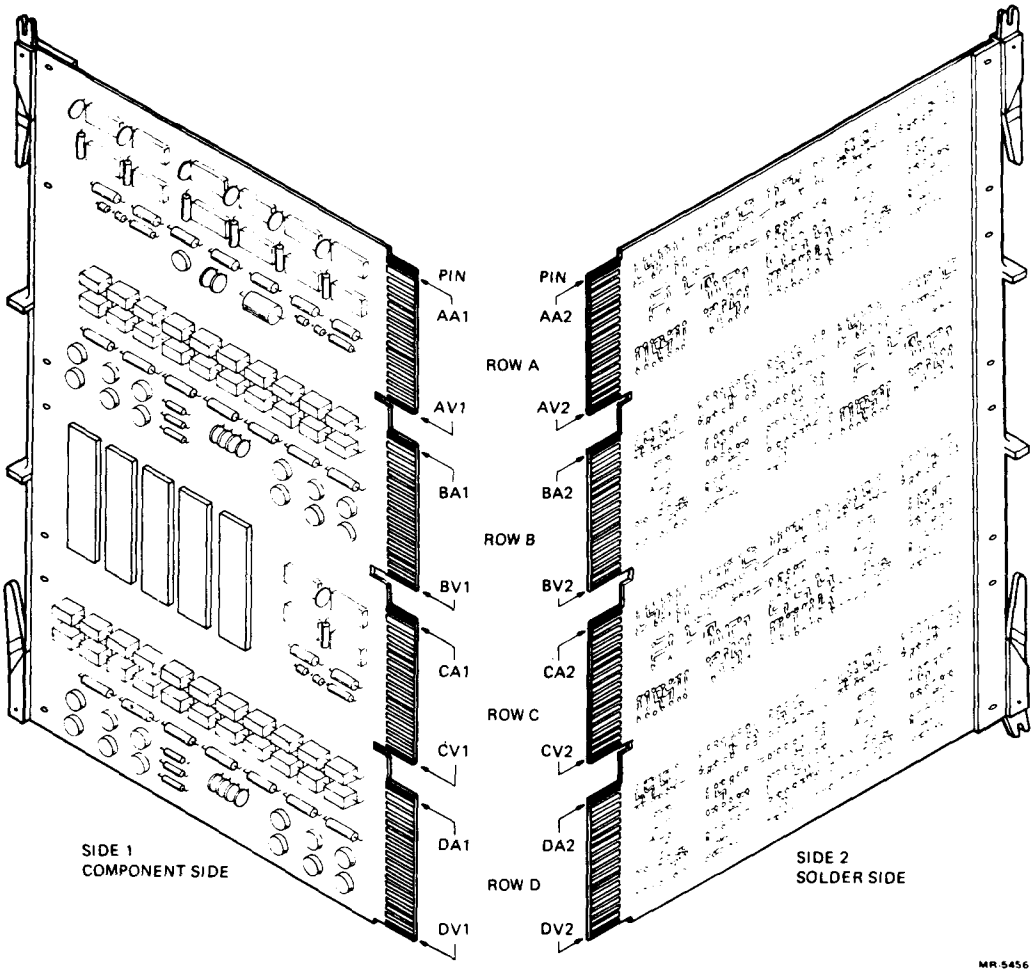


Figure A-20 Quad-Height Module Contact Finger Identification

Table A-4 Bus Pin Identifiers

Bus Pin	Mnemonic(s)	Description
AA1	BIRQ5 L	Interrupt request priority level 5.
AB1	BIRQ6 L	Interrupt request priority level 6.
AC1	BDAL16 L	Extended address bit during addressing protocol; memory error data line during data transfer protocol.
AD1	BDAL17 L	Extended address bit during addressing protocol; memory error logic enable during data transfer protocol.
AE1	SSPARE1 (alternate +5B)	Special Spare -- Not assigned or bussed in Digital's cable or backplane assemblies; available for user connection. Optionally, this pin may be used for +5 V battery (+5 B) backup power to keep critical circuits alive during power failures. A jumper is required on Q22-Bus options to open (disconnect) the +5 B circuit in systems that use this line as SSPARE1.
AF1	SSPARE2	Special Spare -- Not assigned or bussed in Digital's cable or backplane assemblies; available for user interconnection. In the highest-priority device slot, the processor may use this pin for a signal to indicate its RUN state.
AH1	SSPARE3 SRUN	Special Spare -- Not assigned or bussed simultaneously in Digital's cable or backplane assemblies; available for user interconnection. An alternate SRUN signal may be connected in the highest-priority set.
AJ1	GND	Ground -- System signal ground and dc return.
AK1	MSPAREA	Maintenance Spare -- Normally connected together on the backplane at each option location (not a bussed connection).
AL1	MSPAREB	Maintenance Spare -- Normally connected together on the backplane at each option location (not a bussed connection).

Table A-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic(s)	Description
AM1	GND	Ground -- System signal ground and dc return.
AN1	BDMR L	Direct Memory Access (DMA) Request -- A device asserts this signal to request bus mastership. The processor arbitrates bus mastership between itself and all DMA devices on the bus. If the processor is not bus master (it has completed a bus cycle and BSYNC L is not being asserted by the processor), it grants bus mastership to the requesting device by asserting BDMGO L. The device responds by negating BDMR L and asserting BSACK L.
AP1	BHALT L	Processor Halt -- When BHALT L is asserted for at least 25 μs, the processor services the halt interrupt and responds by halting normal program execution. External interrupts are ignored but memory refresh interrupts in Q22 are enabled if W4 on the M7264 and M7264-YA processor modules is removed and DMA request/grant sequences are enabled. The processor executes the ODT microcode, and the console device operation is invoked.
AR1	BREF L	Memory Refresh -- Asserted by a DMA device. This signal forces all dynamic MOS memory units requiring bus refresh signals to be activated for each BSYNC L/BDIN L bus transaction. It is also used as a control signal for block mode DMA. **CAUTION:** The user must avoid multiple DMA data transfers (burst or "hot" mode) that could delay refresh operation if using DMA refresh. Complete refresh cycles must occur once every 1.6 ms if required.

Table A-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic(s)	Description
AS1	+12 B or +5 B	+12 Vdc or +5 V battery backup power to keep critical circuits alive during power failures. This signal is not bussed to BS1 in all of Digital's backplanes. A jumper is required on all Q22-Bus options to open (disconnect) the backup circuit from the bus in systems that use this line at the alternate voltage.
AT1	GND	Ground -- System signal ground and dc return.
AU1	PSPARE 1	Spare -- Not assigned; customer usage not recommended. Prevents damage when modules are inserted upside down.
AV1	+5 B	+5 V Battery Power -- Secondary +5 V power connection. Battery power can be used with certain devices.
BA1	BDCOK H	DC Power OK -- A power supply-generated signal that is asserted when the available dc voltage is sufficient to sustain reliable system operation.
BB1	BPOK H	Power OK -- Asserted by the power supply 70 ms after BDCOK is negated when ac power drops below the value required to sustain power (approximately 75% of nominal). When negated during processor operation, a power-fail trap sequence is initiated.
BC1	SSPARE4 BDAL18 L (22-bit only)	Special Spare in the Q22-Bus -- Not assigned. Bussed in 22-bit cable and backplane assemblies; available for user interconnection.
BD1	SSPARE5 BDAL19 L (22-bit only)	**CAUTION:** These pins may be used by manufacturing as test points in some options.
BE1	SSPARE6 BDAL20 L	In the Q22-Bus, these bussed address lines are address lines <21:18>; currently not used during data time.
BF1	SSPARE7 BDAL21 L	In the Q22-Bus, these bussed address lines are address lines <21:18>; currently not used during data time.

Table A-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic(s)	Description
BH1	SSPARE8	Special Spare -- Not assigned or bussed in Digital's cable and backplane assemblies; available for user interconnection.
BJ1	GND	Ground -- System signal ground and dc return.
BK1 BL1	MSPAREB MSPAREB	Maintenance Spare -- Normally connected together on the backplane at each option location (not a bussed connection).
BM1	GND	Ground -- System signal ground and dc return.
BN1	BSACK L	This signal is asserted by a DMA device in response to the processor's BDMGO L signal, indicating that the DMA device is bus master.
BP1	BIRQ7 L	Interrupt request priority level 7.
BR1	BEVNT L	External Event Interrupt Request -- When asserted, the processor responds by entering a service routine via vector address 1008. A typical use of this signal is as a line-time clock interrupt.
BS1	+12 B	+12 Vdc battery backup power (not bussed to AS1 in all of Digital's backplanes).
BT1	GND	Ground -- System signal ground and dc return.
BU1	PSPARE2	Power Spare 2 -- Not assigned a function; not recommended for use. If a module is using -12 V (on pin AB2), and if the module is accidentally inserted upside down in the backplane, -12 Vdc appears on pin BU1.
BV1	+5	+5 V Power -- Normal +5 Vdc system power.
AA2	+5	+5 V Power -- Normal +5 Vdc system power.

Table A-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic(s)	Description
AB2	-12	-12 V Power -- -12 Vdc power for (optional) devices requiring this voltage. **NOTE:** Each Q22-Bus module that requires negative voltages contains an inverter circuit that generates the required voltage(s). Therefore, -12 V power is not required with Digital's options.
AC2	GND	Ground -- System signal ground and dc return.
AD2	+12	+12 V Power -- +12 Vdc system power.
AE2	BDOUT L	Data Output -- When asserted, BDOUT implies that valid data is available on BDAL<0:15> L and that an output transfer, with respect to the bus master device, is taking place. BDOUT L is deskewed with respect to data on the bus. The slave device responding to the BDOUT L signal must assert BRPLY L to complete the transfer.
AF2	BRPLY L	Reply -- BRPLY L is asserted in response to BDIN L or BDOUT L and during IAK transactions. It is generated by a slave device to indicate that it has placed its data on the BDAL bus or that it has accepted output data from the bus.
AH2	BDIN L	Data Input -- BDIN L is used for two types of bus operations: When asserted during BSYNC L time, BDIN L implies an input transfer with respect to the current bus master, and requires a response (BRPLY L). BDIN L is asserted when the master device is ready to accept data from a slave device. When asserted without BSYNC L, it indicates that an interrupt operation is occurring. The master device must deskew input data from BRPLY L.

Table A-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic(s)	Description
AJ2	BSYNC L	Synchronize -- BSYNC L is asserted by the bus master device to indicate that it has placed an address on BDAL<0:17> L. The transfer is in process until BSYNC L is negated.
AK2	BWTBT L	Write/Byte -- BWTBT L is used in two ways to control a bus cycle: It is asserted at the leading edge of BSYNC L to indicate that an output sequence (DATO or DATOB), rather than an input sequence, is to follow. It is asserted during BDOUT L, in a DATOB bus cycle, for byte addressing.
AL2	BIRQ4 L	Interrupt Request Priority Level 4 -- A level 4 device asserts this signal when its interrupt enable and interrupt request flips-flops are set. If the PS word bit 7 is 0, the processor responds by acknowledging the request by asserting BDIN L and BIAKO L.
AM2 AN2	BIAKI L BIAKO L	Interrupt Acknowledge -- In accordance with interrupt protocol, the processor asserts BIAKO L to acknowledge receipt of an interrupt. The bus transmits this to BIAKI L of the device electrically closest to the processor. This device accepts the interrupt acknowledge under two conditions: 1.) the device requested the bus by asserting BIRQXL, and 2.) the device has the highest-priority interrupt request on the bus at that time. If these conditions are not met, the device asserts BIAKO L to the next device on the bus. This process continues in a daisy-chain fashion until the device with the highest-interrupt priority receives the interrupt acknowledge signal.

Table A-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic(s)	Description
AP2	BBS7 L	Bank 7 Select -- The bus master asserts this signal to reference the I/O page (including that portion of the I/O page reserved for nonexistent memory). The address in BDAL<0:12> L when BBS7 L is asserted is the address within the I/O page.
AR2 AS2	BDMGI L BDMGO L	Direct Memory Access Grant -- The bus arbitrator asserts this signal to grant bus mastership to a requesting device, according to bus mastership protocol. The signal is passed in a daisy-chain from the arbitrator (as BDMGO L) through the bus to BDMGI L of the next priority device (the device electrically closest on the bus). This device accepts the grant only if it requested to be bus master (by a BDMR L). If not, the device passes the grant (asserts BDMGO L) to the next device on the bus. This process continues until the requesting device acknowledges the grant. **CAUTION:** DMA device transfers must not interfere with the memory refresh cycle.
AT2	BINIT L	Initialize -- This signal is used for system reset. All devices on the bus are to return to a known, initial state; that is, registers are reset to zero, and logic is reset to state 0. Exceptions should be completely documented in programming and engineering specifications for the device.
AU2 AV2	BDAL0 L BDAL1 L	Data/Address lines -- These two lines are part of the 16-line data/address bus over which address and data information are communicated. Address information is first placed on the bus by the bus master device. The same device then either receives input data from, or outputs data to, the addressed slave device or memory over the same bus lines.

Table A-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic(s)	Description
BA2	+5	+5 V Power -- Normal +5 Vdc system power.
BB2	-12	-12 V Power (voltage normally not supplied) -- -12 Vdc power for (optional) devices requiring this voltage.
BC2	GND	Ground -- System signal ground and dc return.
BD2	+12	+12 V Power -- +12 V system power.
BE2	BDAL2 L	Data/Address Lines -- These 14 lines are part of the 16-line data/address bus.
BF2	BDAL3 L	
BH2	BDAL4 L	
BJ2	BDAL5 L	
BK2	BDAL6 L	
BL2	BDAL7 L	
BM2	BDAL8 L	
BN2	BDAL9 L	
BP2	BDAL10 L	
BR2	BDAL11 L	
BS2	BDAL12 L	
BT2	BDAL13 L	
BU2	BDAL14 L	
BV2	BDAL15 L	

APPENDIX B
ACRONYMS

AIE -- Alarm Interrupt Enable

ANSI - American National Standards Institute

AP -- Argument Pointer

ASTLVL -- Asynchronous System Trap LeVeL

BBU -- Battery Backup Unit

BCD -- Binary Coded Decimal

BDR -- Boot and Diagnostic Register

BM -- Byte Mask

BRS -- Baud Rate Select signals

CEAR -- CPU Error Address Register

CPMBX -- Console Program MailBoX

CRC -- Cyclic Redundancy Check

CSR -- Control and Status Register

DEAR -- DMA Error Address Register

DM -- Data Mode

DMA -- Direct Memory Access

DSE -- Daylight Saving Enable

EDITPC -- EDIT Packed to Character string

EIA -- Electronic Industries Association

ERR -- ERRor signal

FP -- Frame Pointer

FPU -- Floating Point Unit

ICCS -- Interval Clock Control and Status register

IPCR -- InterProcessor Communication Register

IPL -- Interrupt Priority Level

IPR -- Internal Processor Register

ISP -- Interrupt Stack Pointer

LSI -- Large Scale Integration

MAPEN -- memory management (MAPping) ENable register

MBZ -- Must Be Zero

MCS -- Multinational Character Set

MFPR -- Move From Processor Register

MMU -- Memory Management Unit

MOP -- Maintenance Operation Protocol

MSER -- Memory System Error Register

MTPR -- Move To Processor Register

P0BR -- P0 (P zero) Base Register

P1BR -- P1 Base Register

PC -- Program Counter

PIE -- Periodic Interrupt Enable

P0LR -- P0 (P zero) Length Register

P1LR -- P1 Length Register

P0PT -- P0 (P zero) Page Table

P1PT -- P1 Page Table

PROM -- Programmable Read-Only Memory

PSL -- Processor Status Longword

PSW -- Processor Status Word

PTE -- Page Table Entry

RPB -- Restart Parameter Block

SBR -- System Base Register

SCA -- System Communications Architecture

SCB -- System Control Block

SCBB -- System Control Block Base

SID -- System IDentification register

SIE -- System Identification Extension

SIRR -- Software Interrupt Request Register

SISR -- Software Interrupt Summary Register

SLR -- System Length Register

SLU -- Serial Line Unit

SP -- Stack Pointer

SPT -- System Page Table

SQWE -- SQuare-Wave Enable

TBIA -- Translation Buffer Invalidate All

TBIS -- Translation Buffer Invalidate Single

TOY -- Time-of-Year

UIE -- Update Interrupt Enable

UIP -- Update in Progress bit

VRT -- Valid RAM and Time bit

VMB -- Virtual Memory Bootstrap