

DEC TRNcontroller 100

Hardware Description and Debugging

Order Number: EK-DEQRA-TM-001

December 1991

This guide describes the DEC™ TRNcontroller 100 Q-Bus-to-Token Ring Adapter (DEQRA), its architecture, and how it works in Digital systems. It also describes the on-line debugging tool, ODT68, for the DEC TRNcontroller 100.

Revision/Update Information: This is a new guide.

Operating System/Version: VMS V6.4

December 1991

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

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

© Digital Equipment Corporation 1991.
All Rights Reserved
Printed in U.S.A.

The following are trademarks of Digital Equipment Corporation: DEC, MicroVAX, PATHWORKS, PCSA, VAX, VMS, and the Digital logo.

The following are third-party trademarks: Motorola is a registered trademark of Motorola, Inc. PAL is a trademark of Monolithic Memories Inc. TMS380C16 and BUDS (software) are trademarks of Texas Instruments. Z-BUS is a trademark of Zilog Inc.

This guide was produced by Telecommunications and Networks Publications.

Contents

Preface	ix
1 Product Overview	
1.1 Purpose of DEQRA	1-1
1.2 DEQRA Applications	1-2
2 System Operations	
2.1 DEQRA Architecture Overview	2-1
2.2 General Operation	2-2
2.2.1 Start-up	2-3
2.2.2 Normal Operation	2-3
2.2.3 Reset	2-3
2.3 Operation in Application Environment	2-4
2.3.1 Host Performance	2-5
2.3.2 Communications Traffic	2-5
3 Hardware Description	
3.1 Architecture	3-1
3.1.1 Dual-Bus Architecture	3-3
3.1.2 Concurrency	3-5
3.1.3 Host Interface	3-5
3.2 Memory Map	3-5
3.2.1 Nonvolatile Memory Space	3-7
3.2.2 Peripheral Space	3-8
3.2.3 Volatile Memory Space	3-8
3.2.4 Z-BUS Space	3-8
3.3 Processor Bus	3-9
3.3.1 Microprocessor	3-9
3.3.2 Processor Bus Operation	3-9
3.3.3 Memory	3-12

3.3.4	EPROM	3-12
3.3.4.1	Diagnostics	3-12
3.3.4.2	Downloader	3-12
3.3.4.3	ODT68 Debugging Tool	3-13
3.3.5	Multifunction Peripheral	3-13
3.3.5.1	Console	3-13
3.3.5.2	Timers	3-14
3.3.5.3	General Purpose Port	3-15
3.3.6	Registers	3-16
3.3.6.1	LED Register	3-17
3.3.6.2	Board Configuration Register	3-17
3.4	Communications Bus	3-19
3.4.1	Z-BUS Operation	3-19
3.4.2	Token Ring Communications Processor	3-21
3.4.3	Shared Memory	3-21
3.4.3.1	Data Organization	3-22
3.4.3.2	Arbitration	3-23
3.4.3.3	Q-bus Support	3-24
3.4.3.4	Refresh	3-24
3.4.4	Timers	3-25
3.5	Host Interface	3-25
3.5.1	Command and Status Register	3-27
3.5.2	Interface Devices	3-29
3.5.3	Support Logic	3-29
3.6	Shared Memory Base Address Selection	3-29
3.7	CSR Address Selection	3-30
3.8	Interrupt Vector Address Selection	3-30
3.9	Shared Memory Jumper (JP1)	3-30
3.10	Token Ring Interface	3-30
3.10.1	TMS380C16	3-32
3.10.2	TMS380C16 Reset Operation	3-32
3.10.3	Bus Timing Circuitry	3-33
3.10.4	Token Ring Memory	3-33
3.10.5	Ring Interface Module	3-33
3.11	Host-to-Board-to-Token Ring Transfers	3-33
3.12	Details of Operation	3-34
3.12.1	Interrupts	3-34
3.12.1.1	Interrupt Structure	3-34
3.12.1.2	Interrupt Levels	3-35
3.12.2	Timing	3-36
3.12.2.1	Clock Generation	3-37
3.12.2.2	Transaction Timing	3-38

3.12.3	Z-BUS Arbitration	3-39
3.12.3.1	Z-Conversion State Machine	3-41
3.12.3.2	Arbitration State Machine	3-41
3.12.4	Reset	3-42
3.12.5	Bus Error	3-42
3.12.5.1	Bus Exception Controller Operation	3-42
3.12.5.2	Delayed Z-BUS Transactions	3-43
3.12.6	Non-Maskable Interrupt	3-44

4 Electrical Interfaces on the DEQRA

4.1	Token Ring Port	4-1
4.2	Console Port	4-2

5 Using the ODT68 Debugging Tool

5.1	Description of the ODT68	5-1
5.2	Required Equipment	5-2
5.3	Console and Terminal Installation	5-2
5.4	Accessing ODT68	5-2
5.5	Overview of ODT68 Command Processing	5-3
	EXIT	5-6
	HELP	5-7
	RECALL	5-8

6 Using the ODT68 Debug Command Set

6.1	Introduction	6-1
	ASCII	6-3
	AUX	6-4
	BREAKPT	6-5
	DISA	6-7
	DUMP	6-9
	FILL	6-10
	GOTO	6-11
	OFFSET	6-12
	PEEK	6-13
	POKE	6-14
	REGS	6-15
	SEARCH	6-18

TRACE	6-19
TTB	6-20
DIAGS	6-21

7 Using the ODT68 Auxiliary Command Subset

BIA	7-2
ENABLE	7-3
INHIBIT	7-4
SD	7-5

8 Using the ODT68 Execute Diagnostics Command Subset

CIO	8-3
INT	8-5
LOOP	8-7
MEMTEST	8-8
TMS	8-10
TRM	8-13
XBUSER	8-15
XEPROM	8-17
XMFP	8-19
XRAM and XZRAM	8-21

A PEEK and POKE Command Edit Functions

B Register Names

C Sample Power-Up Display

D LOOP Command Display

Glossary

Index

Figures

1-1	ISO OSI Reference Model	1-3
2-1	DEQRA Architecture Overview	2-2
2-2	Typical DEQRA Operating Environment	2-4
3-1	DEQRA PC Board Layout	3-2
3-2	DEQRA Architecture Block Diagram	3-4
3-3	Detailed Processor Bus Memory Map	3-6
3-4	Detailed Z-BUS Memory Map	3-7
3-5	Console Cable for Use with an EIA-232 Terminal	3-14
3-6	MFP General Purpose Port	3-16
3-7	LED Register	3-17
3-8	Board Configuration Register	3-18
3-9	Shared Memory Block Diagram	3-22
3-10	Data Organization	3-23
3-11	Host Interface Block Diagram	3-26
3-12	Command and Status Register	3-28
3-13	Token Ring Circuitry Block Diagram	3-31
3-14	Interrupt Priority Structure	3-36
3-15	DEQRA System Timing	3-38
3-16	DEQRA Transaction Timing	3-39
3-17	Z-BUS Arbitration Block Diagram	3-40
3-18	Z-BUS Arbitration State Diagram	3-41
3-19	Bus Exception State Diagram	3-43
4-1	Pin Assignments for the Token Ring Port Connector	4-2
4-2	Pin Assignments for the Console Port Connector	4-3

Tables

1	Document Conventions	xi
1-1	DEQRA Hardware Features and Benefits	1-2
3-1	Processor Bus Signal Description	3-10
3-2	Device Parameters	3-11
3-3	Functions of the Prescaler/Counter Timers	3-14
3-4	Functions of the General Purpose I/O Lines	3-15
3-5	NO_QMEM and INITDIS Bits	3-18
3-6	Z-BUS Signal Description	3-19
3-7	TMS380C16 Register Addresses	3-32
4-1	Console Cable Pin Assignment	4-3
5-1	Keyboard Functions for ODT68	5-3
6-1	Summary of Debug Commands	6-2
7-1	Auxiliary Command Subset	7-1
8-1	Diagnostics Command Subset	8-2
A-1	PEEK and POKE Command Edit Functions	A-1
B-1	DEQRA Registers	B-1

Preface

Purpose of this Guide

This guide describes the DECTM TRNcontroller 100 Q-Bus-to-Token Ring Adapter (DEQRA) communications controller, its architecture, and how it works in Digital systems. The DEQRA front-end processor belongs to the DEQRA series of products that includes the M7533-AB controller. The information in this guide supplements the basic information that appears in the *DEC TRNcontroller 100 Hardware Installation* guide.

This guide also describes the DEQRA on-line debugging tool, ODT68.

This guide may aid DEC TRNcontroller 100 programmers in debugging their Digital host-based application software.

Intended Audience

This guide is intended for maintenance technicians, computer system integrators, and software developers who need detailed information about the operating theory and features of the DEQRA hardware.

Organization of Document

If you are not familiar with front-end communications processors read Chapter 1 and 2. If you are familiar with front-end communications processors, you may want to skip these overviews and go directly to the DEQRA detailed technical descriptions in Chapter 3.

This guide contains the following chapters and appendixes:

Chapter 1	Product Overview
	Provides a functional overview of the DEQRA.

Chapter 2	System Operations Describes the DEQRA hardware and its operation in a host system.
Chapter 3	Hardware Description Describes the DEQRA hardware architecture, memory map, buses, and design.
Chapter 4	Electrical Interfaces on the DEQRA Describes the primary (token ring port) and secondary (console port) electrical interfaces on the DEQRA.
Chapter 5	Using the ODT68 Debugging Tool Provides basic information needed to run the ODT68 debugging tool. Also described, is how to attach a terminal to the DEQRA to run diagnostics routines.
Chapter 6	Using the ODT68 Debug Command Set Explains how to use the ODT68 debugging command set.
Chapter 7	Using the ODT68 Auxiliary Command Subset Explains how to use the ODT68 debugging auxiliary command subset.
Chapter 8	Using the ODT68 Execute Diagnostic Command Subset Explains how to use the ODT68 diagnostic command set.
Appendix A	PEEK and POKE Command Edit Functions Lists the PEEK and POKE command edit characters and a description of each.
Appendix B	Register Names Lists the DEQRA registers, the ODT68 recognized abbreviation for each register, and a description of each register.
Appendix C	Sample Power-Up Display Provides a sample console display, depicting a power-up routine.
Appendix D	LOOP Command Display Provides a sample display depicting the execution of the LOOP command.
Glossary	Provides definitions of the terms used throughout the DEQRA document set.

Reference Documents

Additional information about the DEC TRNcontroller 100 product can be found in the following documents:

- *DEC TRNcontroller 100 Hardware Installation*
- *DEC Token Ring Network Device Driver for VMS Installation*
- *DEC Token Ring Network Device Driver for VMS Use and Programming*
- *Token Ring Access Method, IEEE STD 802.5-1989*

Document Conventions

Table 1, Document Conventions, lists the conventions used in this guide.

Table 1 Document Conventions

Convention	Description
DEQRA	The term DEQRA refers to the M7533-AB controller board.
NOTE	Contains information that may be of special importance to the user.
CAUTION	Contains information to prevent damage to software or hardware.
Special Type	Shows program output displayed on the console screen. Special type in examples indicates user input.
Return	Indicates that you press the key labeled Return, in examples.
CtrlZ	Indicates that in examples you press the key labeled Ctrl and the key labeled Z, simultaneously.
\overline{AS}	Indicates the signal is asserted low true.
Communications Bus	Refers to Z-BUS and ZILOG Z8000™ bus.
[]	A bracket is used to indicate optional input.

Product Overview

The DEC TRNcontroller 100 Q-Bus-to-Token Ring Adapter (DEQRA) intelligent communications controller allows you to attach suitably configured Q-bus-based Digital Equipment Corporation VAX[™] computers directly to an industry-compatible token-ring network.

The DEQRA hardware is a single-board computer that has a central processing unit, random access memory, programmable read-only memory, token-ring interface circuitry, and host-interface circuitry. The software consists of an on-board operating system, diagnostic tests, host interface drivers, and application routines.

1.1 Purpose of DEQRA

The main purpose of the DEQRA is to provide an intelligent link to an IEEE 802.5 compatible token ring while improving the overall computing efficiency of the host computer. To do this, low-level communications tasks that are traditionally performed by the host central processor are migrated to the DEQRA.

The DEQRA increases overall system bandwidth by distributing the I/O processing away from the host CPU. In the traditional minicomputer architecture, the host services all I/O requests. This load on the CPU has grown steadily as computer peripherals have become increasingly more powerful. Modern operating systems allow intelligent front-end processors to perform these relatively simple tasks. The result is an overall increase in system throughput.

Table 1-1 list the attributes of the DEQRA.

Table 1-1 DEQRA Hardware Features and Benefits

Feature	Benefits
Dual-bus architecture	32-bit processor bus optimized for program execution. 16-bit communications bus optimized for data transfer and host interface. Concurrent operation improves performance.
68020 CPU	4 Gbyte linear address space. Allows use of familiar, powerful program development tools (assemblers, C compilers, and editors). Development can often be done on host system; separate system not required.
TMS380C16 Token Ring Communications Processor	Interfaces all control signals and data transfers to token ring. Protocol handler performs hardware-based protocol functions conforming to standard IEEE 802.5.
Large private RAM memory	Provides space for applications tasks, real-time executive, and so on. Applications are downloaded into RAM for ease of change.
Single-module form factor	Mounts easily in a standard backplane.

1.2 DEQRA Applications

The DEQRA board links the token ring to Digital's Q-bus-based VAX products. At system start-up, the VAX host downloads microcode containing the communication protocols. The application software may be customer-specific or part of a Digital developed turn-key product.

Communications software systems that use the DEQRA can fit the International Standards Organization (ISO) open system interconnection seven-layer reference model for communications protocols. Figure 1-1 shows this model. The DEQRA performs the tasks described by the first two layers of the model.

Once the software download has been completed, the DEQRA hardware and software combination performs its assigned task until a board-level reset initiates another start-up sequence.

Figure 1-1 ISO OSI Reference Model

	Layer Number	Function
Application	7	Services applications
Presentation	6	Code conversion, data format
Session	5	Coordinates interaction between applications
Transport	4	End-to-end data integrity
Network	3	Routes information
Data Link	2	Exchanges data with physical link
Physical	1	Transmits/receives bit stream to medium

LKG-4878-911

System Operations

This chapter contains a general description of the DEQRA hardware and its operation in host systems.

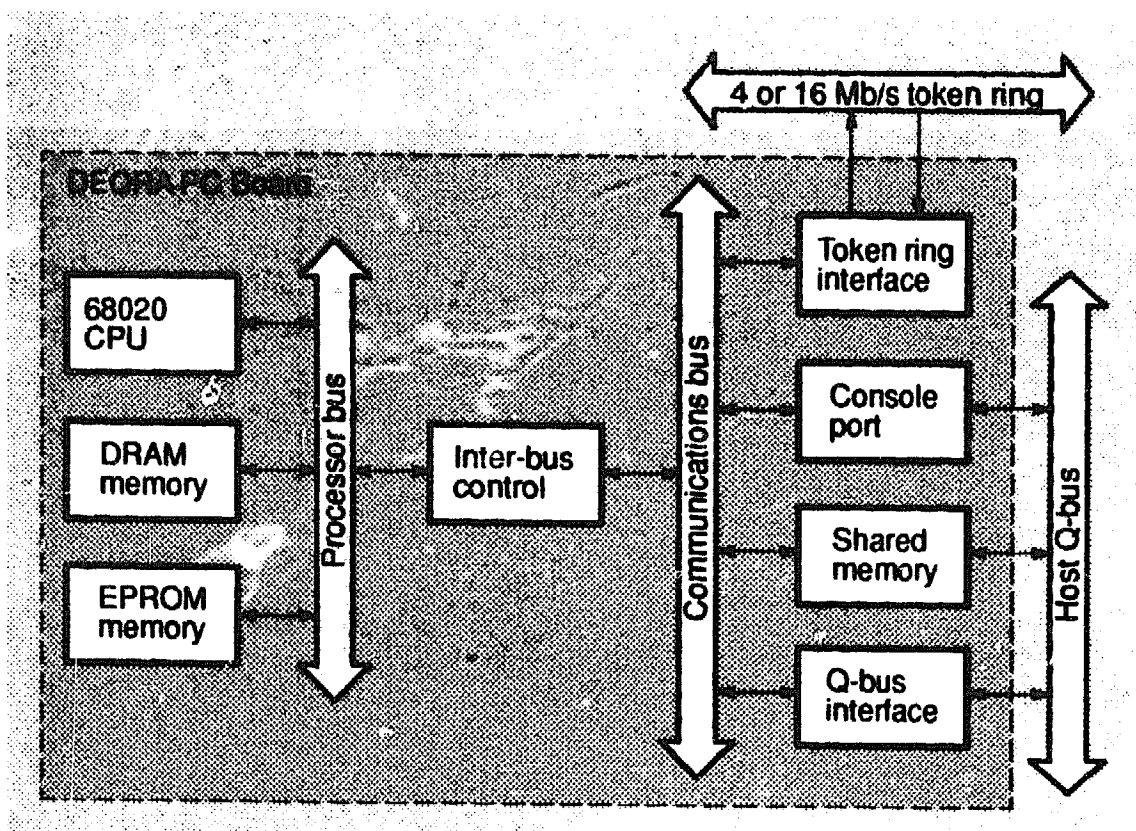
2.1 DEQRA Architecture Overview

The DEQRA controller uses a dual-bus architecture as shown in Figure 2-1, to provide high-performance, front-end processing in communication environments.

The processor bus consists of a CPU, memory, and support devices. This bus processes the application program and provides computing resources as required by high-level protocols.

The communications bus (the subsidiary bus) consists of a token-ring communication processor, memory, and host-interface circuits. The communications bus moves data between the token ring and the host. The processor and communications buses are two independent buses bridged by circuitry, which isolates them during concurrent operations and translates appropriate signals to control transactions between them during inter-bus operations.

Figure 2-1 DEQRA Architecture Overview



LKG-4941-911

2.2 General Operation

The DEQRA operations occur in two phases:

- Start-up
- Normal

When power is initially applied to the DEQRA, its CPU begins execution at a memory address in the PROM devices. This memory address contains instructions that set up the basic working environment for the CPU.

2.2.1 Start-up

Start-up begins when the on-board diagnostics test all major sections of the DEQRA circuitry. Upon the successful completion of these tests, the host interface is initialized and the DEQRA loads its operating system and application-program software from the host, through the Q-bus interface circuits into the shared memory. After these have been moved into processor memory, the CPU process the downloaded software.

2.2.2 Normal Operation

During normal operation, the DEQRA processes all communications-related work. The software program downloaded during start-up is used to operate the general-purpose DEQRA communications hardware in the mode required by the host system. This program is typically an implementation of some special-purpose communications protocol.

The DEQRA remains in this phase of operation until a board reset command is issued either by the host software, the host-bus hardware, or the reset switch.

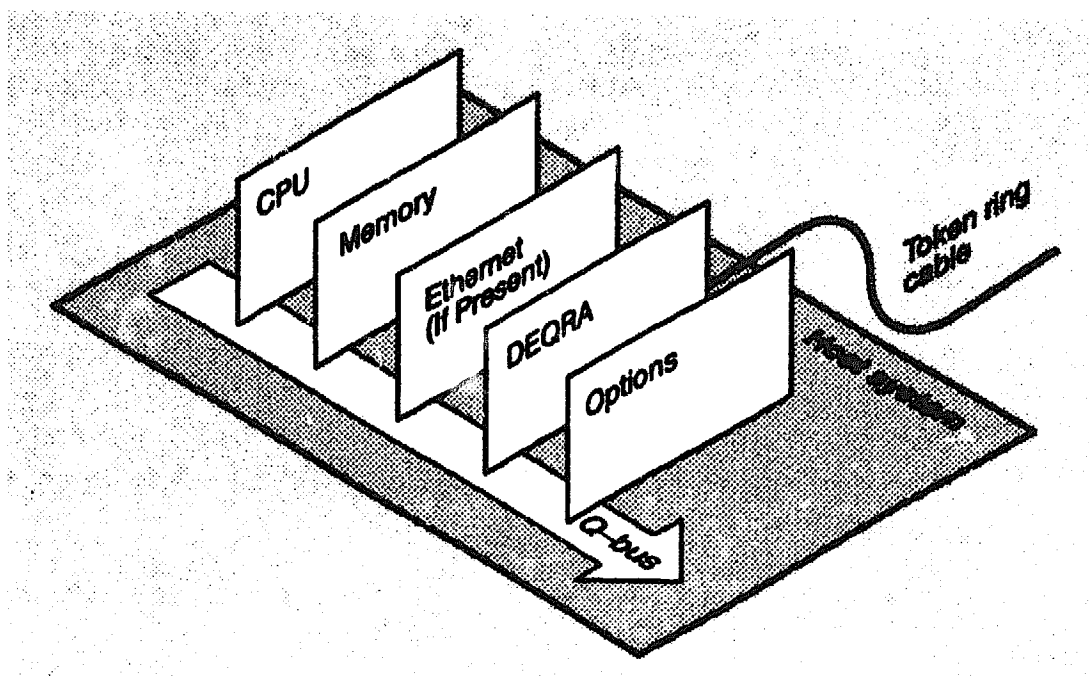
2.2.3 Reset

A hardware reset causes the DEQRA to stop processing program instructions. No attempt is made to save the current state of the CPU operation. The hardware reset causes the CPU to go back to the start-up phase.

2.3 Operation in Application Environment

The DEQRA hardware processes communications-related application software in a typical VAX operating environment. Figure 2-2 shows a typical DEQRA operating environment.

Figure 2-2 Typical DEQRA Operating Environment



LKG-4942-911

2.3.1 Host Performance

The DEQRA offloads the host, thus increasing overall VAX system performance. Traditionally, the host's CPU executes the tasks related to the operation of serial communications. These tasks can be extensive, and many require processing in real time. The DEQRA hardware, combined with application software, is capable of processing many of these protocol-dependent, low-level tasks. The application software is downloaded from the host system to the DEQRA during the system start-up phase.

2.3.2 Communications Traffic

When the software download is complete, communications traffic is accepted and processed for transfer to or from the host. In most applications, the data format used by the host processor is very different from the format transmitted on the token ring. Many additional structures must be added to the raw data processed by the host system in order to maintain the data's integrity during transmission. The software programs downloaded into the DEQRA perform this task.

Hardware Description

The DEQRA is an intelligent communications controller that is designed to interface Digital Equipment Corporation Q-bus-based VAX computers to an industry standard token-ring network.

This chapter describes the DEQRA architecture, memory map, buses, and design.

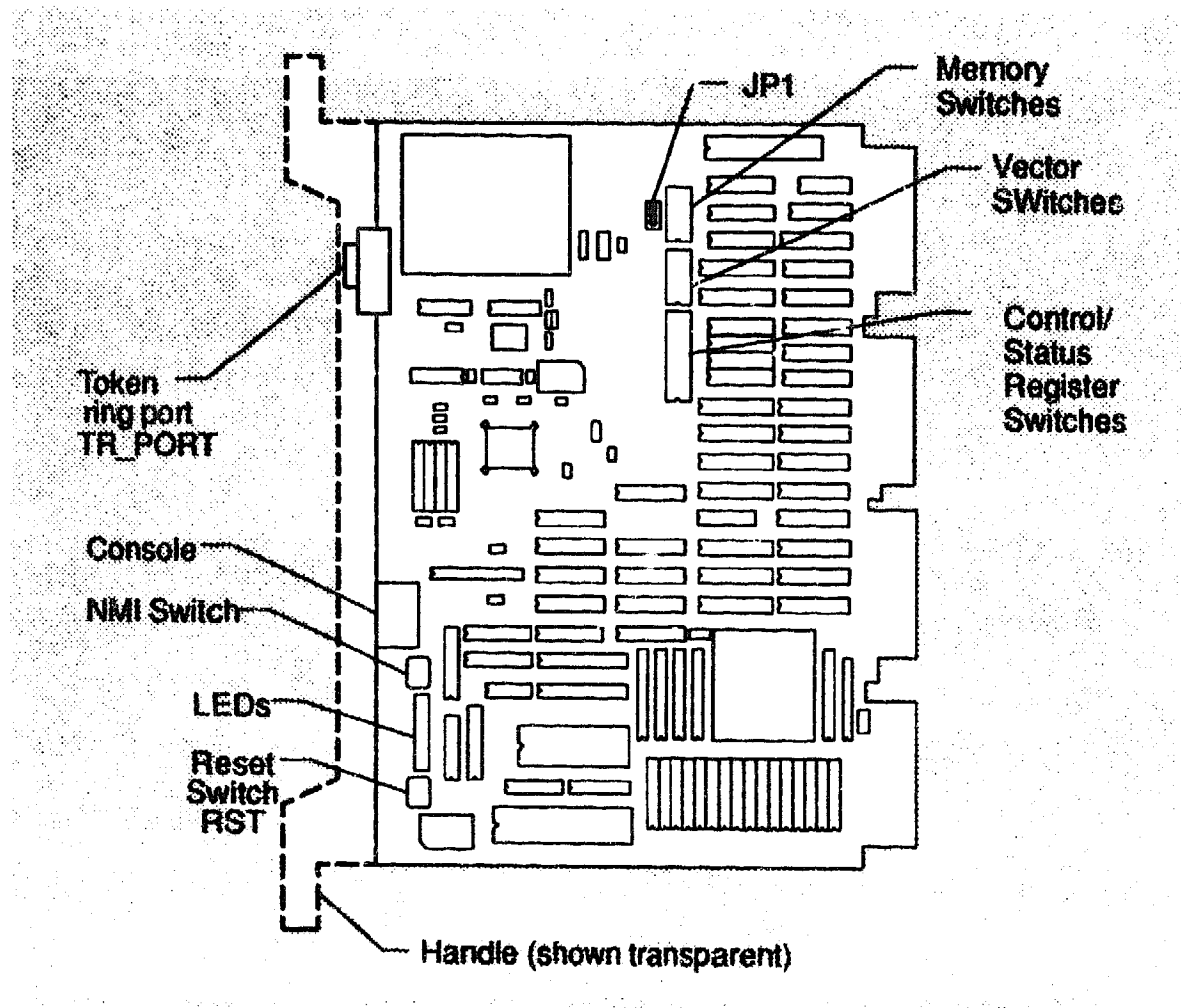
See Chapter 4 for the circuitry and pinout information on the electrical interfaces of the DEQRA.

3.1 Architecture

The DEQRA uses a dual-bus architecture to provide a high performance link.

Figure 3-1 shows the physical layout of the hardware components on the DEQRA board.

Figure 3-1 DEQRA PC Board Layout



LKG-4943-91

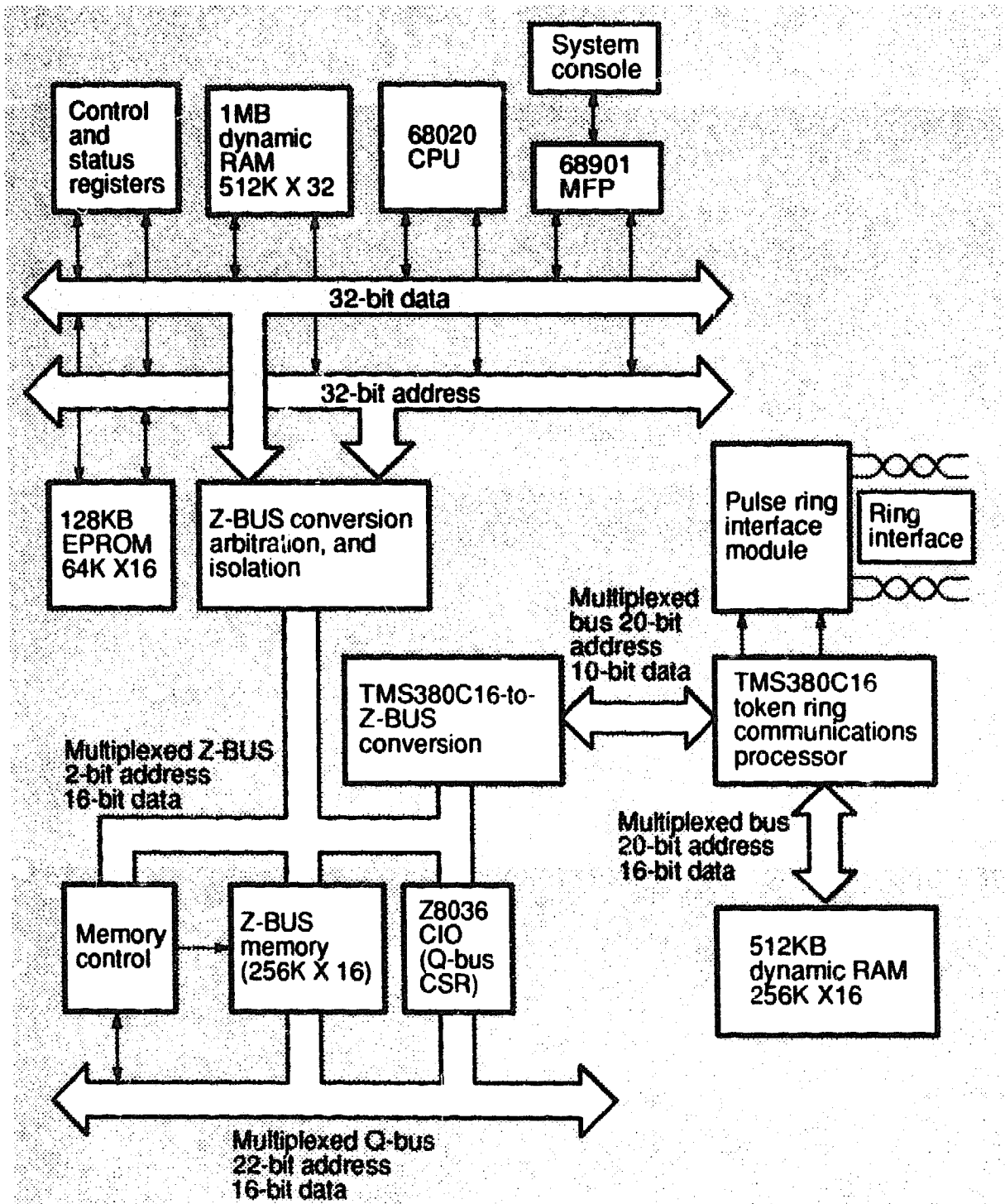
The following sections describe:

- Dual-bus architecture
- Concurrent operations
- Host interface

3.1.1 Dual-Bus Architecture

As shown in Figure 3-2, the DEQRA uses a dual-bus architecture to enhance data-transfer rates and allow the CPU more processing time. The processing engine resides on a processor bus while the token ring circuitry resides on an isolated communications bus. Each of these buses is optimized for its primary function. The processor bus has a Motorola 68020™ 32-bit processor, longword-wide memory, and a non-multiplexed bus architecture that is optimized for program execution. The communications bus is implemented as a Zilog™ multiplexed bus (Z-BUS™) with a word-wide memory system that is optimized for DMA-controlled data movement and host-interface transfers. The 68020 CPU has access to all of the memory and devices on both buses, whereas the TMS380C16™ on the Z-BUS has access only to the Z-BUS memory.

Figure 3-2 DEQRA Architecture Block Diagram



3.1.2 Concurrency

The dual-bus architecture optimizes the operational characteristics of each bus and enables the processing engine and the TMS380C16 to operate concurrently. This allows the on-board CPU to continue to execute application-level protocol programs and interrupt instruction streams on the processor bus while the TMS380C16 is moving data along the communications bus. Processing is suspended, or DMA transactions delayed, only when both the CPU and a TMS380C16 have transactions waiting for a Z-BUS device at the same time. This reduces the conflicts during normal operation and maximizes the use of each bus.

3.1.3 Host Interface

In most cases, the host machine is the ultimate source or destination of the data being transmitted and received. The board requires a mechanism for issuing commands, determining status, and controlling the data exchange with the host system in which it resides. The DEQRA communications bus memory, along with a command and status register, provides the host interface to the Q-bus. The communications bus memory is implemented as a shared memory that the 68020, the token ring communications processor, or a Q-bus master device on the host system can access. This optimizes data exchange between the board, the token ring, and the host. The control/status register (CSR) passes messages between the DEQRA and the host to coordinate data exchange. It is implemented as an eight-bit register in each direction. Writing to the CSR from either side can generate an interrupt on the other side. See Section 3.5 for details.

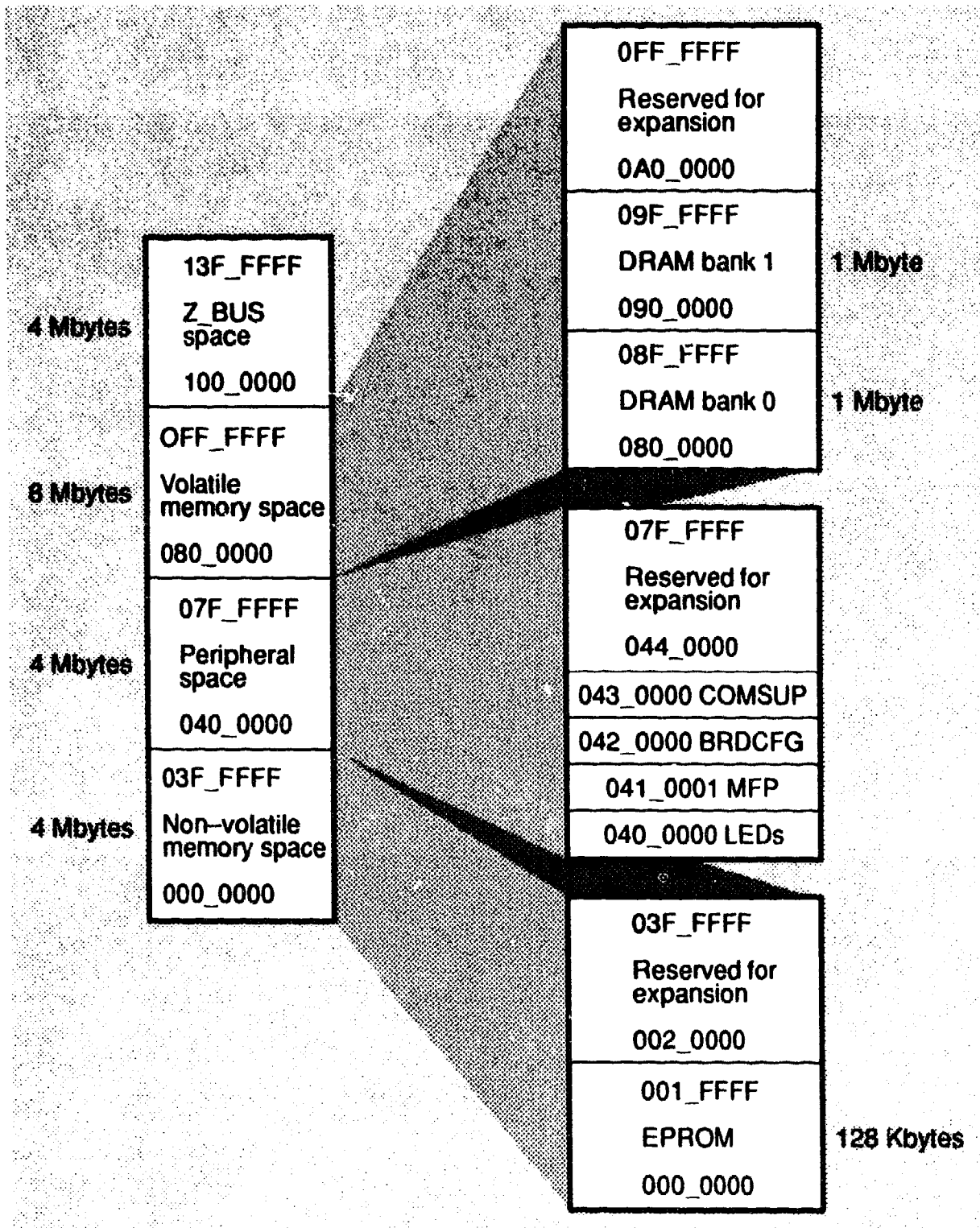
3.2 Memory Map

The DEQRA memory map is divided into four main memory spaces:

- Nonvolatile memory
- Peripheral
- Volatile memory
- Z-bus

Figure 3-3 shows the detailed addressing for the volatile memory space, peripheral space, and the nonvolatile memory space.

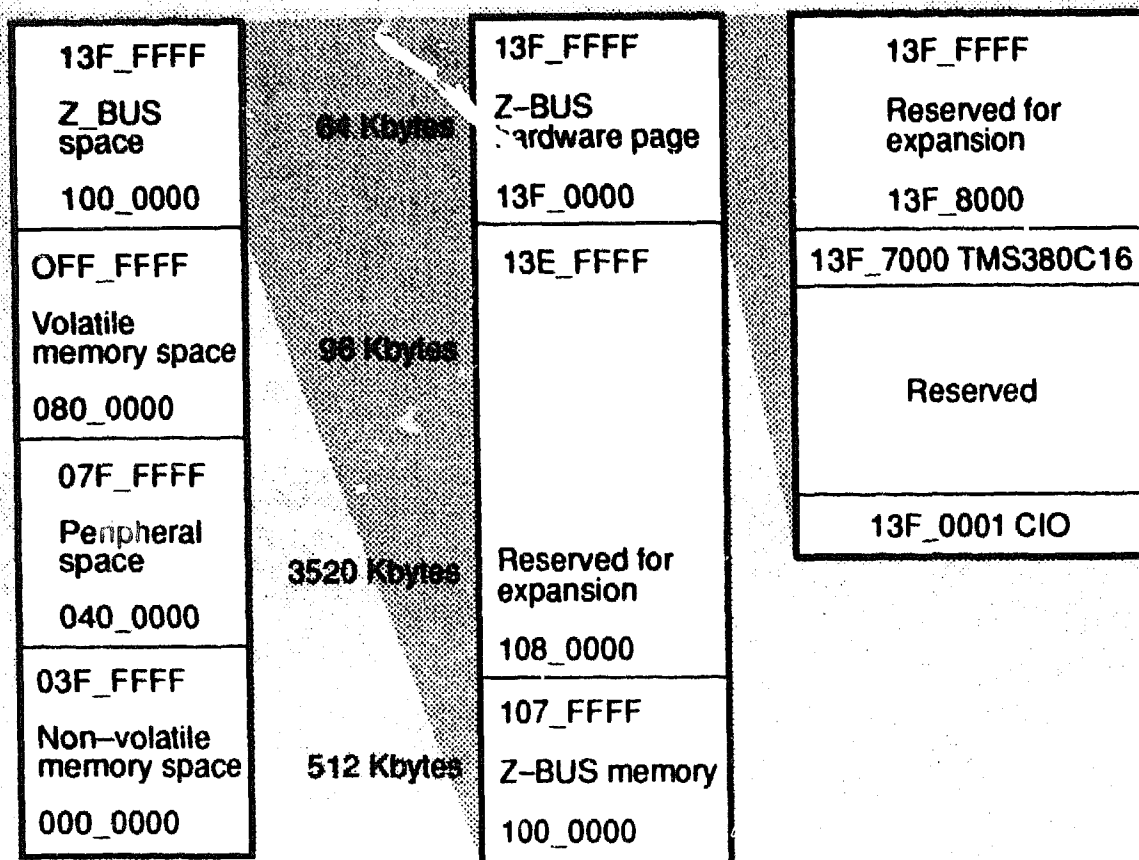
Figure 3-3 Detailed Processor Bus Memory Map



LKG-4946-911

Figure 3-4 shows the detailed addressing for the Z-bus memory space.

Figure 3-4 Detailed Z-BUS Memory Map



LKCL 4947-911

3.2.1 Nonvolatile Memory Space

The 4 Mbyte space beginning at address 000_0000₁₆ is the nonvolatile memory space. The DEQRA EPROM occupies the lowest 128 Kbytes. The additional space is reserved for future expansion.

3.2.2 Peripheral Space

The 4 Mbyte space beginning at address 040_0000_{16} is the peripheral device space. To simplify the address decoding logic on the board, each peripheral device is assigned a 64 Kbyte window. The DEQRA has three devices assigned: the LED register, the multifunction peripheral device, and the board configuration register. See Section 3.3.5 and Section 3.3.6 for descriptions of these peripheral devices. The additional space is reserved for future expansion.

3.2.3 Volatile Memory Space

The 8 Mbyte space beginning at address 080_0000_{16} is the volatile memory space. One bank of dynamic random access memory occupies the lowest 1 Mbyte of the volatile memory space. The additional space is reserved for future expansion.

3.2.4 Z-BUS Space

The 4 Mbytes beginning at address 100_0000_{16} is the Z-BUS space. This space is further divided into specific address sections:

- **Z-BUS memory**

The 512 Kbytes beginning at 100_0000_{16} is the Z-BUS shared memory. These Kbytes are mapped into the Q-bus memory space beginning at the address chosen by the board-selection switches.

- **Reserved**

The 3.424 Mbytes beginning at 108_0000_{16} are reserved for future expansion.

- **Peripheral device**

The 64 Kbytes, beginning at $13F_0000_{16}$, are used for Z-BUS peripheral devices. To simplify address decoding logic, each device is assigned a 16 Kbyte window. These devices are the Counter/Timer and the TMS380C16 token-ring communications processor. See Section 3.4 for descriptions of these peripheral devices. The additional space is reserved for future expansion.

3.3 Processor Bus

The DEQRA processor bus is composed of the CPU, EPROM, RAM, a multifunction peripheral device, control/status registers, diagnostic LEDs, and the support circuitry that is required for timing and control of the bus. Section 3.3.1 through Section 3.3.6 describe the major devices, support circuitry, bus signals, and the operation of the processor bus. For clarity, the processor description precedes the descriptions of the bus operation and other devices that connect to this bus.

3.3.1 Microprocessor

The DEQRA processing engine is a Motorola 68020. The 68020 address bus, data bus, and internal registers are all 32-bits wide. The microprocessor has a rich instruction set including versatile addressing modes that support high-level languages. The 68020 has a 256-byte internal instruction cache that may be disabled under program control, but is normally enabled to speed program processing. The internal operations are designed to operate in parallel using a three-stage pipeline. This allows multiple instructions to process concurrently. The processor also supports dynamic bus sizing that enables devices of different data widths to interface directly with the 68020 without data-alignment restrictions.

A partial list of microprocessor features include:

- Sixteen 32-bit general purpose registers
- Two 32-bit supervisor stack pointers, one user stack pointer
- One 32-bit program counter
- Five special control registers
- Eighteen addressing modes
- 4 Gbytes of address space
- One 256-byte instruction cache

3.3.2 Processor Bus Operation

The processor bus is an implementation of Motorola's 68020 bus architecture made up of address, data, control, and timing signals as shown in Table 3-1. Although the 68020 is capable of linearly addressing 4 Gbytes of address space, only the lower 25 address lines are decoded by the on-board logic. Detailed information about the memory map is found in Section 3.2.

Table 3-1 Processor Bus Signal Description

Signal Name	Mnemonic	I/O	Description
Address	A0-A31	O	Indicates processor address bus
Address Strobe	\overline{AS}	O	Indicates that function code, address, size, and R/\overline{W} signals are valid
Bus Error	\overline{BERR}	I	Indicates that the bus exception controller has timed out
Cache Disable	\overline{CDIS}	I	Disables 256-byte instruction cache (used for production testing only)
Clock	CLK	I	Defines the 10 MHz CPU clock used for internal processing and timing
Data	D0-D31	I/O	Indicates processor data bus
Data Transfer and Size	$\overline{DSACK0}-\overline{DSACK1}$	I	Acknowledges data transfer and size (used to end cycle and indicate slave data size)
External Cycle Start	\overline{ECS}	O	Indicates cycle is about to begin
Function Codes	FC0-FC2	O	Indicates processor state and address space identifiers
Halt	\overline{HALT}	O	Indicates a double bus fault has occurred
Interrupt Priority Level	$\overline{IPL0}-\overline{IPL2}$	I	Indicates the highest level interrupt request is active
Read/Write	R/\overline{W}	O	Determines the direction of data transfer
Reset	\overline{RESET}	I/O	Initiates an I/O Reset used to initiate board devices
Size	SIZE0-SIZE1	O	Indicates size of current transaction (number of bytes to be transferred)

The 68020 dynamic bus-sizing feature allows devices of different widths to reside on the 32-bit data bus; it supports byte, word, three-byte, and longword data transfers on any byte boundary without requiring special data alignment. Bus sizing is accomplished as follows:

1. The 68020 asserts the desired transfer size at the beginning of each transaction by encoding the SIZ0 and SIZ1 signals.
2. External logic returns \overline{DSACK} signals to the 68020 indicating what size transfer the addressed device can support.
3. The CPU uses internal byte-steering logic and multiple operations to complete the desired transaction.

As an example, when the CPU starts a longword transaction to a byte-wide device, it recognizes the byte-wide \overline{DSACK} s returned by the external logic and presents, or reads, only one byte of data. The CPU proceeds with three additional byte transactions in order to complete the longword transfer.

The DEQRA uses synchronous design techniques to decode and generate CPU control signals. State machines monitor the address bus, size codes, and function codes to generate \overline{DSACK} signals matching the size and timing requirements of each device. Table 3-2 shows the DEQRA devices, their widths, and \overline{DSACK} selection as decoded by programmable array logic. Automatic wait-state selection is generated for slave devices that do not provide \overline{DSACK} generation.

Table 3-2 Device Parameters

Device/Memory Space	Width	DSACK
Processor memory	longword	external
EPROM	word	three wait states
MFP register	byte	external
LED register	byte	one wait state
Board Configuration register	byte	one wait state
Communications Support register	longword	one wait state
Z-BUS space	word	external

3.3.3 Memory

The DEQRA processor bus is available with 1 Mbyte of zero-wait-state, longword-wide (32 bits), dynamic random access memory (DRAM). It consists of one bank of 256K by 32-bit DRAM beginning at address 080_0000₁₆. The memory controller state machine is implemented in programmable array logic devices and provides all of the logic required for address multiplexing, read-and-write control, and refresh timing. The memory controller works in conjunction with the 68020's dynamic bus sizing to fully support byte, word, three-byte, and longword access on any byte boundary. A memory refresh cycle is executed every 15.6 μ s to maintain data integrity.

3.3.4 EPROM

The 128 Kbytes of erasable programmable read only memory, beginning at address 000_0000₁₆, consists of a 64K by 16-bit EPROM device. The EPROM contains self-test diagnostics, a boot-loader program, and a debugging tool that includes a disassembler. For a detailed discussion of these tools, see Chapter 5 through Chapter 8.

3.3.4.1 Diagnostics

Self-test diagnostics begin when the board is first powered up. These tests validate the control circuitry, processor and Z-BUS memory, processor and Z-BUS peripheral devices, interrupt operation, bus error logic, EPROM checksum, and concurrent bus execution features of the DEQRA hardware. Test status and error reporting may be monitored either by viewing the LEDs or by connecting a terminal to the console port on the board edge.

Upon completion of these tests, an *inhibit diagnostics* pattern is written into memory. Succeeding hardware resets of the board cause a basic subset of the tests to execute. This prevents downloaded code and device setups from being overwritten or altered by a full reset. It also shortens the time required to download the board since the full diagnostics are not re-run for every reset. You may initiate and monitor menu-driven diagnostics testing from a console.

3.3.4.2 Downloader

The CPU uses the bootloader code and the host driver to download the operating system/impact executive and applications from the host to the board. The bootloader does this by moving the executable images through the shared memory to the processor memory. The CPU begins executing the downloaded code after the download sequence has completed.

3.3.4.3 ODT68 Debugging Tool

The ODT68 debugging tool enables you to connect a 9,600 bits/s terminal to the console port to communicate directly with the board during program-development or debugging sessions. It provides access and breakpointing throughout the entire DEQRA address space. It displays the 68020 address, data, and special registers. A disassembler is included and you can specify individual diagnostic tests for execution. See Chapter 5 for more information on the ODT68 debugging tool.

3.3.5 Multifunction Peripheral

Motorola's MC68901 multifunction peripheral is a byte-wide 68000 bus device that resides at address 41_0000_{16} . This large scale integration device provides a full-duplex asynchronous serial port, four eight-bit timers, and eight general-purpose, individually programmable, I/O lines with interrupt capabilities.

3.3.5.1 Console

The DEQRA uses the MFP serial port as a console device for direct EIA-232 communications. With a 9,600 bits/s terminal connected to the console port, the results of all the diagnostic self-tests will display.

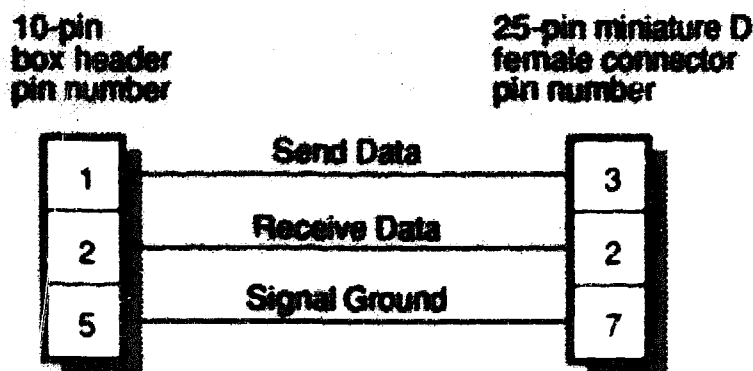
The console is also used while running the debugging tool during program development. Figure 3-5 shows the pinout for the console cable, BC29E-15.

During normal operation the console port is not used. To prevent RFI emissions the console cable should not be left plugged in.

NOTE

To provide ESD protection for the console interface, ensure the console cover plate is in place.

Figure 3-5 Console Cable for Use with an EIA-232 Terminal



KG-4879-91

3.3.5.2 Timers

The four eight-bit timers are prescaler/counter timers with a common 2.5MHz clock input. Table 3-3 lists the prescaler/counter timers and their functions.

Table 3-3 Functions of the Prescaler/Counter Timers

Timer	Function
A, B	Implement a 16-bit timer for use by the real-time operating system.
C	Used as the data-rate clock generator for the 9,600 bit/s console port.
D	Available as a general purpose timer for applications that require additional timing functions. It provides a timing resolution of 1.33 μ s, and may be used in delay, pulse-width-measurement, or event-count mode.

3.3.5.3 General Purpose Port

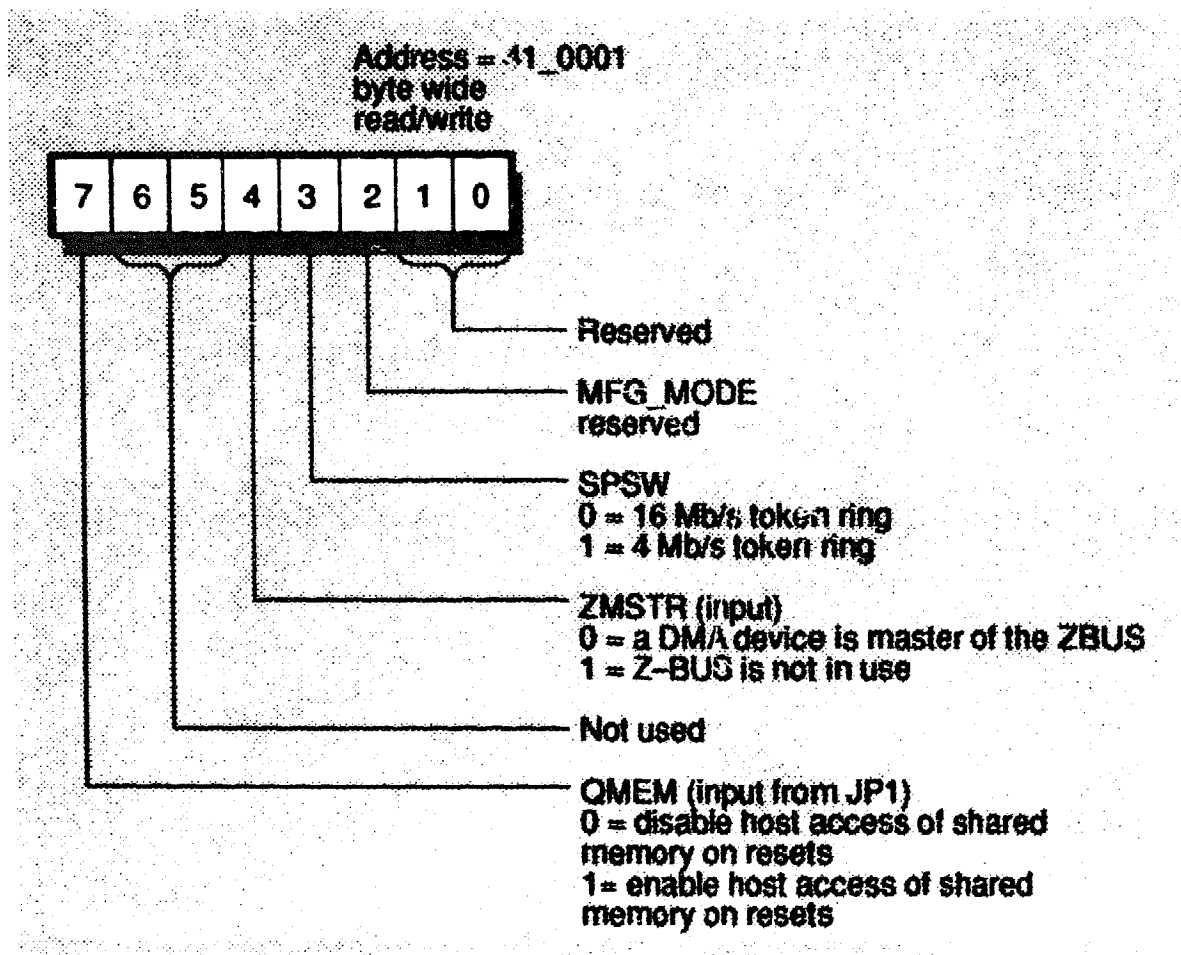
The eight general-purpose I/O lines may be individually operated as either inputs or outputs under software control. They may generate an interrupt on either transition direction of an input signal. Four of the eight I/O lines are currently allocated for use by the DEQRA. These I/O lines and their functions are listed in Table 3–4.

Table 3–4 Functions of the General Purpose I/O Lines

I/O Line	Function
ZMSTR	Input signifying that the TMS380C16 is actively using the communications bus.
QMEM	Input from JP1, used to set shared memory visibility at power-up.
SPSW	Input bit that sets the speed of the token ring at 4Mb/s or 16Mb/s.
MFG_MODE	Reserved for manufacturing purposes.

The other four I/O lines are reserved for future use. Figure 3-6 shows the register for bit assignments.

Figure 3-6 MFP General Purpose Port



LKG-4948-911

3.3.6 Registers

The processor bus includes two special-purpose registers. These are the LED and the board-configuration registers. These registers are used to select operational characteristics and maintain board-level configuration parameters. Read-back registers are used to enable the data from the last write (the current value of the register) to be read back by the CPU. This eliminates the need to maintain memory-resident images of the register's contents. The CPU must

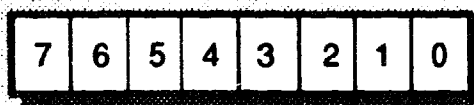
read these registers into an internal CPU register to manipulate bits, or bit fields, without affecting other bits.

3.3.6.1 LED Register

The byte-wide LED register, which resides at address 40_0000_{16} , is a read-back register whose outputs are tied directly to the LEDs on the DEQRA board's edge. These easily viewed LEDs show status and error information during the self-test diagnostics. They may also indicate the operational status of an application program. Clearing or writing a zero to a bit position illuminates the appropriate LED. See Figure 3-7.

Figure 3-7 LED Register

Address=40_0000
Byte wide
Read/Write
All bits are cleared to 0 at Reset time



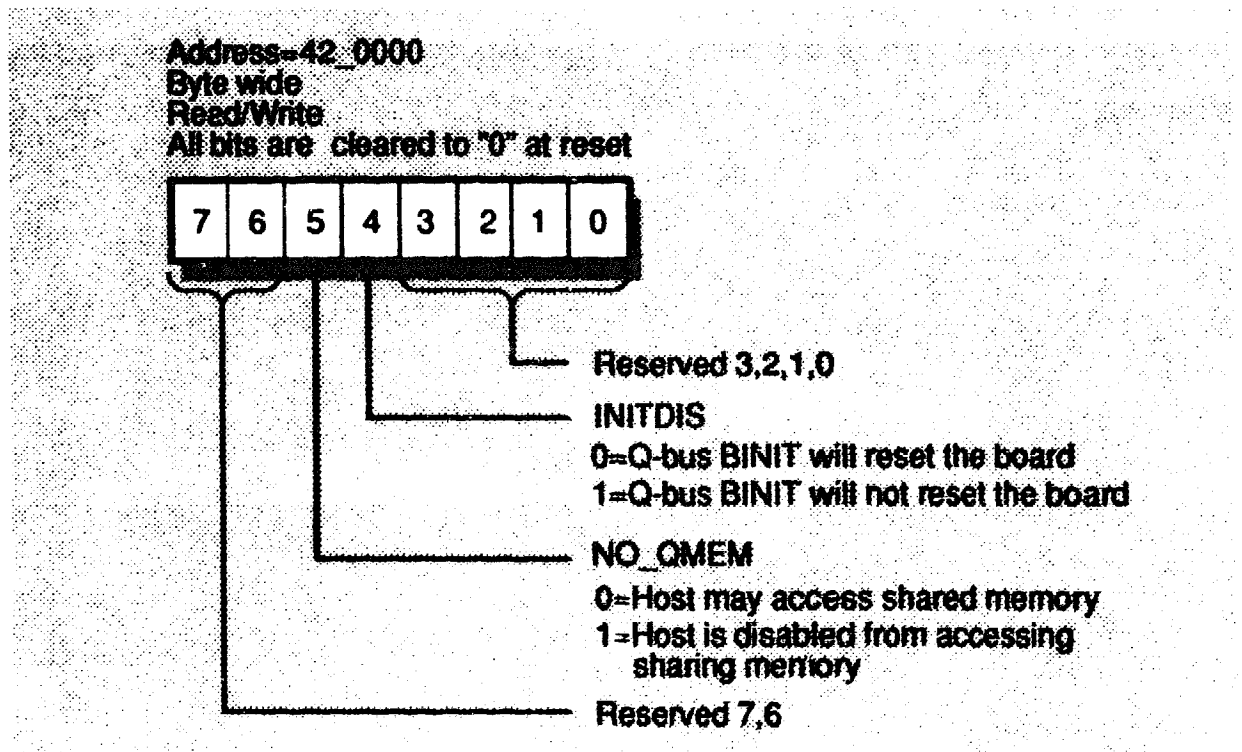
Write 00 hexadecimal to turn all LEDs on.
Write FF hexadecimal to turn all LEDs off.

LKG-4949-91

3.3.6.2 Board Configuration Register

The byte-wide board configuration register residing at address 42_0000_{16} , also a read-back register, is used to control board-level parameters such as reset options and Q-bus shared memory access enable. Figure 3-8 shows the register bit map.

Figure 3-8 Board Configuration Register



LKG-4950-91

Table 3-5 describes the NO_QMEM and INITDIS bits.

Table 3-5 NO_QMEM and INITDIS Bits

Bit	Description
NO_QMEM	Some operating systems or hardware configurations have requirements concerning memory visibility at host power-up time. This bit provides a mechanism for disabling the shared memory controller from generating Q-bus memory cycles. The EPROM power-up routine uses the position of jumper JP1 to set or clear the NO_QMEM bit at power-up time. The <i>DEC TRNcontroller 100 Hardware Installation</i> manual describes board configuration.
INITDIS	This bit provides the DEQRA with an option to ignore the Q-bus signal BINIT by disabling the reset controller state machine during BINIT sequences on the Q-bus.

3.4 Communications Bus

The DEQRA communications bus exists as a space in the CPU memory map and is composed of a token-ring communications processor, shared memory, and the support circuitry required for timing, data transfer, bus arbitration, and interrupt operations. The following sections describes the major devices, support circuitry, bus signals, and how the communications bus operates. The communications bus is implemented as a Zilog Z8000 bus and is referred to as the Z-BUS throughout these sections.

3.4.1 Z-BUS Operation

The Z-BUS is a multiplexed bus made up of 16 address/data lines, 6 extended-address lines, and 13 miscellaneous control lines. The possible bus masters on the Z-BUS are the 68020 processor or the token-ring communications processor. Table 3-6 shows the descriptions of the Z-BUS signals.

Table 3-6 Z-BUS Signal Description

Signal Name	Z-BUS Mnemonic	DEQRA Mnemonic	Description
Address Strobe	\overline{AS}	\overline{ZAS}	Timing signal indicating that the addresses and certain control signals are valid
Bus Acknowledge	\overline{BUSACK} , BAI, BAO	\overline{BUSACK}	Indicates that the bus arbiter has relinquished control of the bus in response to a request
Bus Request	\overline{BUSREQ}	\overline{BUSREQ}	Bus requester has, or is trying to obtain, control of the bus
Byte/Word	B/\overline{W}	ZB/\overline{W}	Indicates whether a byte or word of data is to be transferred on the bus
Data/Address	AD00-AD15	BDAH00-BDAH15	Multiplexed data and low order address lines
Data Strobe	\overline{DS}	\overline{ZDS}	Provides timing of data movement to or from the bus master

¹ \overline{IRQCIO} , \overline{IRQEGL} , and \overline{IRQEXC} are the three possible signals.

² $\overline{IACKCIO}$, $\overline{IACKEGL}$, and $\overline{IACKEXC}$ are the three possible signals.

(continued on next page)

Table 3-6 (Cont.) Z-BUS Signal Description

Signal Name	Z-BUS Mnemonic	DEQRA Mnemonic	Description
Extended Address	EA16–EA21	BDAH16–BDAH21	High order address lines
Interrupt	\overline{INT}	\overline{IRQ}^1	Shows that an interrupt request is being made
Interrupt Acknowledge	\overline{INTACK}	\overline{IACK}^2	Shows that an interrupt acknowledge transaction is in progress
Interrupt Enable In	IEI	IEI	Interrupts daisy chain input from a higher-priority device
Interrupt Enable Out	IEO	IEO	Interrupts daisy chain output to a lower-priority device
Peripheral Clock	none	PCLK	Clock timing signal used by peripherals for internal processes and timing
Read/Write	R/\overline{W}	ZR/\overline{W}	Determines the direction of the transfer on the bus
Wait	\overline{WAIT}	\overline{ZWAIT}	Shows that a responding device needs more time to complete a transaction

¹ \overline{IRQCIO} , \overline{IRQEGL} , and \overline{IRQEXC} are the three possible signals.

² $\overline{IACKCIO}$, $\overline{IACKEGL}$, and $\overline{IACKEXC}$ are the three possible signals.

An arbitration controller monitors the individual request lines and grants the bus to a requesting device when possible. CPU read, write, and interrupt transactions in the Z-BUS space require a Z-BUS conversion state machine to control address and data buffers for bus multiplexing, and to generate (using the Motorola signals) control signals that conform to the Z-BUS specification. This state machine also controls bus isolation to enable concurrent operation of the Z-BUS and the processor bus.

After the power-up test diagnostics have executed and the download sequence is completed, the Z-BUS becomes idle and starts to execute the arbitration scheme. Section 3.4.3.2 describes the arbitration.

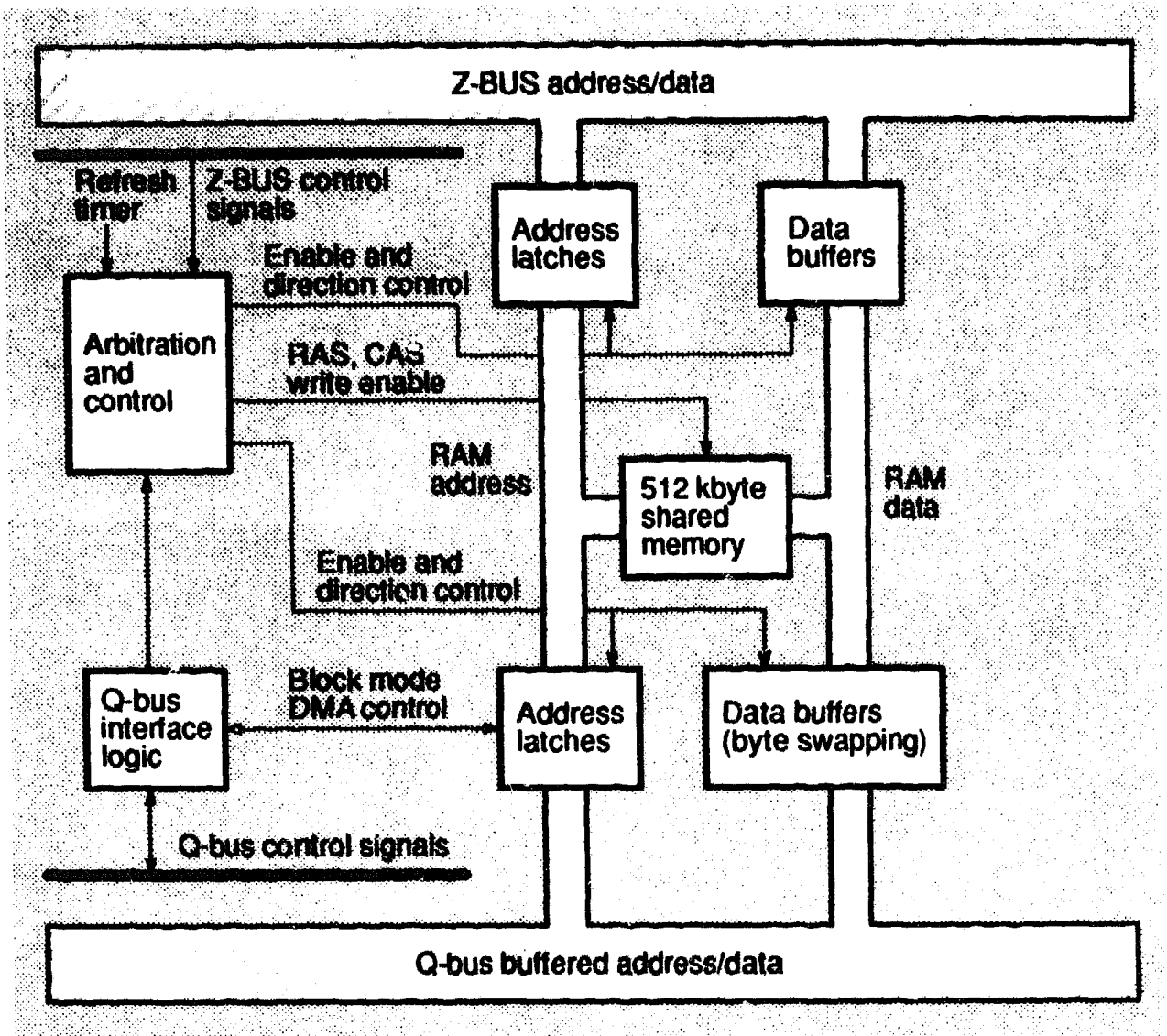
3.4.2 Token Ring Communications Processor

The TMS380C16, Texas Instruments' token-ring communications processor, is the interface between the communications bus and the token ring. The TMS380C16 handles all data transfers and control signals between the board and the token ring. Since it was designed to reside on a 68000 bus, the TMS380C16 requires some interface circuitry to make it part of the Z-BUS. For a detailed description of the TMS380C16, see Section 3.10.1.

3.4.3 Shared Memory

The 512 Kbyte Z-BUS memory is implemented as a 256K by 16-bit DRAM that may be accessed by the CPU, the TMS380C16, or the Q-bus host. The memory controller uses PAL devices to provide all of the control signals required for arbitration, address multiplexing, read-and-write control, and refresh timing. Figure 3-9 is a block diagram of the shared memory.

Figure 3-9 Shared Memory Block Diagram



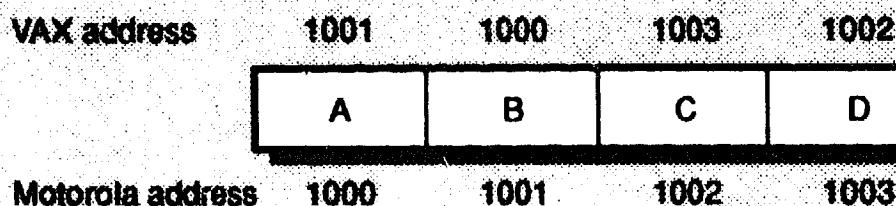
LKG-4951-911

3.4.3.1 Data Organization

Motorola and Zilog use a data structure different from the one used by Digital Equipment Corporation. When a word (16-bit) value is stored by a VAX processor, the low-order byte is stored at the lower address in memory, and the high-order byte is stored at the higher address in memory; however, a Motorola processor stores the high-order byte at the lower address and the low-order

byte at the higher address. As shown in Figure 3-10, if the VAX wrote the word "AB" at address 1000, the Motorola 68020 would read the word "AB" from address 1000, but when the VAX writes the byte "B" at address 1000, the 68020 must read address 1001 to access that "B," unless a byte-swapping mechanism is implemented.

Figure 3-10 Data Organization



LKG-4880-1

In communications applications, most data is byte-oriented, for example, a string of ASCII characters. Word and longword values are generally the exception, occurring only in headers. A 16-bit data count, or a 32-bit address are examples of header fields. To minimize data manipulation, the DEQRA hardware swaps all bytes as the VAX reads from, and writes to, the shared memory. This enables all byte-oriented data to be interpreted correctly without software intervention. Word values require byte-swapping, and longword values require both byte- and word-swapping. By convention, DEQRA-resident software accepts full responsibility for swapping data fields where required. This makes byte-swapping functions transparent to all host-resident software.

3.4.3.2 Arbitration

A Z-BUS master, the host, and the refresh timer may have simultaneous requests for memory transactions; therefore, arbitration logic must select the active memory-cycle type, execute a memory cycle for that requester, and generate appropriate signals to delay memory transactions from the other requesters. At the end of the current transaction, the arbiter selects the next active cycle, and initiates a new memory cycle to complete the stalled transaction. This arbitration scheme ensures that the Q-bus host and the DEQRA devices alternate cycles when both have continuous requests for memory usage.

3.4.3.3 Q-bus Support

The 512 Kbyte area of shared memory is mapped into the Q-bus address space for use as the shared-memory window by the host. The memory controller supports Q-bus block-mode transfers by providing the \overline{BREF} control signal required by Q-bus masters, and the address-incrementing function required for contiguous memory access without explicit address overhead for each Q-bus cycle. The DEQRA forces all Q-bus master devices to strictly follow the 16-word boundary, as specified by Digital Equipment Corporation, by not asserting \overline{BREF} across this boundary.

The memory-controller arbitration unit continues to interleave Z-BUS memory cycles with Q-bus cycles during host block-mode transfers. This ensures that neither the real-time communications data nor the 68020 program execution are locked out during Q-bus block-mode transfers.

3.4.3.4 Refresh

The DRAM requires refresh cycles every 15.6 μ s to maintain data integrity. Refresh cycles have highest priority and the arbitration unit schedules them for execution while Q-bus, Z-BUS, or both, transactions wait to use the memory. The refresh overhead uses approximately 2 percent of the memory bandwidth.

3.4.4 Timers

The Z-BUS includes a Z8036 counter/timer and parallel I/O device. The CIO contains two general purpose byte-wide parallel I/O ports, one four-bit I/O port, and three 16-bit counter/timers. The parallel I/O ports are used for the host interface and are described in Section 3.5.

The three 16-bit counter/timers are available to applications for general purpose use.

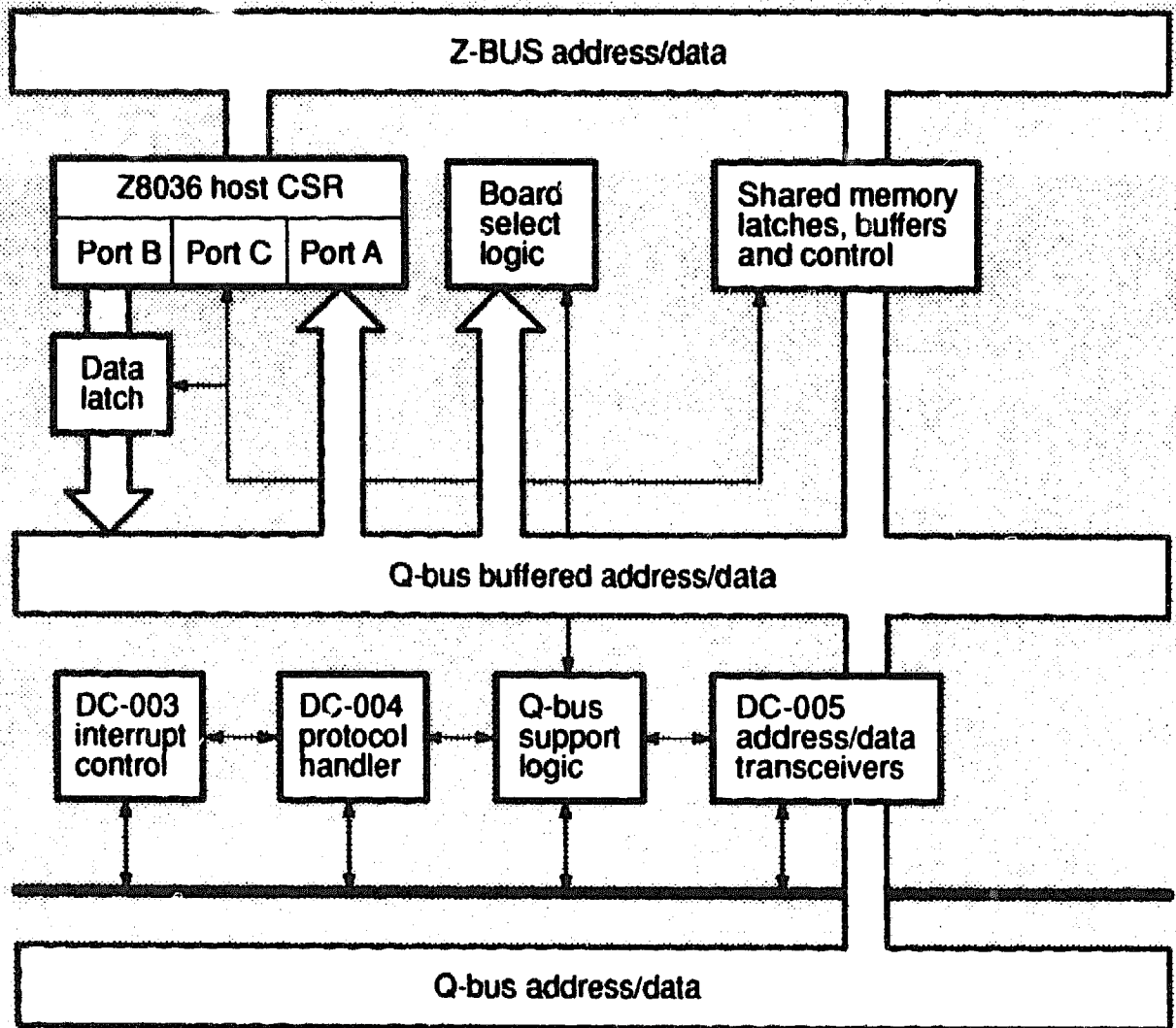
- Counters 1 and 2 may be linked to form a 32-bit counter under software control.
- The 5 MHz clock input provides for a maximum terminal count rate of 1.25 MHz with a timing resolution of 400 ns.

All timers are tested singly, and in linked mode, during the self-test diagnostics.

3.5 Host Interface

The DEQRA front-end processor is used with Digital Equipment Corporation's Q-bus-based VAX computers. Figure 3-11 shows the host interface block diagram.

Figure 3-11 Host Interface Block Diagram



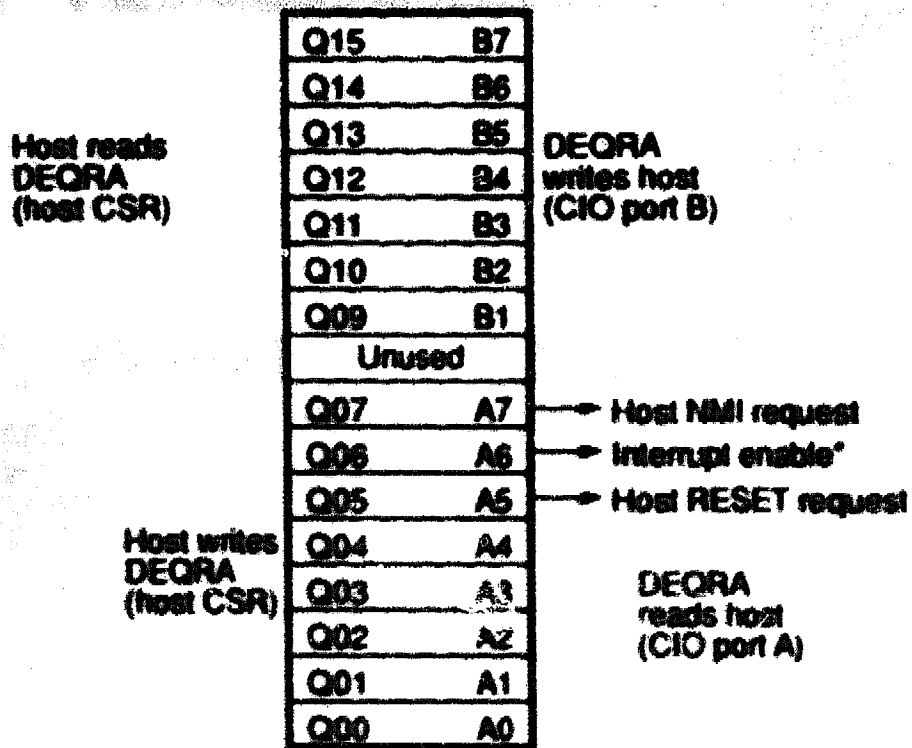
LKG-4952-911

The communication medium between the board and the host is provided by the combination of a shared memory and a command and status register. The use of shared memory reduces the complexity of the board interface to the host machine because the shared window is mapped into both the Q-bus and DEQRA memory space. The host driver simply fills a buffer in the shared window and notifies the DEQRA, through the CSR, that the buffer is ready for transmission. The DEQRA manipulates the data, as required, before transferring it to the token ring. Conversely, when a receive buffer has been filled and any data manipulation is completed, the DEQRA sends a message, through the CSR, informing the host of the availability of the completed buffers. Additional interrupts are generated to request and acknowledge download requests or initiate board resets and non-maskable interrupts (NMI) from the host side.

3.5.1 Command and Status Register

The parallel ports of the CIO are configured to be used as the host-interface command and status register. The CSR provides one byte in each direction for message transfer between the host and the board. Port A provides the eight-bit input port for host writes to the DEQRA, and port B provides the eight-bit output port for DEQRA writes to the host. The CSR bit definition is shown in Figure 3-12.

Figure 3-12 Command and Status Register



*A write to the 68020's CIO port B must occur before the host can see the IE bit previously posted.

LKG-4850-81

Port C, the four-bit port, is used to implement the hardware handshake required for proper latching and presentation of data between the DEQRA and the host system. Writes from the board to the host use an interlocked handshake. This means that additional writes by the DEQRA are not executed until the host has serviced the interrupt caused by the initial DEQRA write. Writes from the host to the DEQRA use a strobed handshake. This means that the host may overwrite the data in the CSR whether or not the DEQRA has

served the interrupt caused by the initial write. The DEQRA always reads the current data in the CSR; overwritten data is lost.

3.5.2 Interface Devices

The DEQRA uses Digital Equipment Corporation interface integrated circuit (IC) devices for direct connection to the host's Q-bus. These provide optimal bus-loading characteristics by decoupling the board-side logic from the bus. The interface integrated circuits are defined as follows:

- The DC-003 interrupt controller provides the interrupt request, acknowledge, and control functions for the Q-bus. The DEQRA uses level 4 interrupt requests on the Q-bus. Its interrupt priority is dependent on its position in the host system and is determined by the boards between it and the host CPU.
- The DC-004 protocol IC decodes Q-bus control signals and generates buffered control signals indicating read/write, byte/word, and CSR/memory selection that can be used by the DEQRA support logic.
- The DC-005 contains address-recognition logic, interrupt-vector drivers, and address/data bus transceivers in one package.

3.5.3 Support Logic

The Q-bus support logic is implemented in PAL devices. The buffered Q-bus signals from the Digital Equipment Corporation devices and the DEQRA-side control signals are used to assist in shared-memory access, block-mode transfer support, and board selection.

3.6 Shared Memory Base Address Selection

The Q-bus base address for the onboard 512k bytes of memory must be set before installation. If more than one DEQRA is installed in a system, or if another Q-bus module using a Q-bus address is installed, you must ensure that the addresses do not overlap. Each module must have a unique base address.

To select the shared memory base address, refer to the *DEC TRNcontroller 100 Hardware Installation* guide. The default address for the first DEQRA begins at address 12000000 and ends at address 13777777. The second DEQRA, if one exists, begins at address 14000000 and ends at address 15777777.

3.7 CSR Address Selection

The CSR switches allow the CSR address to be located anywhere in the Q-bus I/O floating address space. The default address for the first DEQRA is 761344 and 761346 for the second DEQRA.

The CSR base address may be obtained using VMS for both the MicroVAX 3000 series and VAX 4000 Model 300 systems. To select the CSR address, refer to the *TRNcontroller 100 Hardware Installation* guide.

3.8 Interrupt Vector Address Selection

The interrupt vector may be located anywhere in the 300 to 774 vector address space. The default vector address is 300. The correct vector may be obtained using VMS for both the MicroVAX 3000 series and VAX 4000 series systems. To select the interrupt vector address, refer to the *DEC TRNcontroller 100 Hardware Installation* guide.

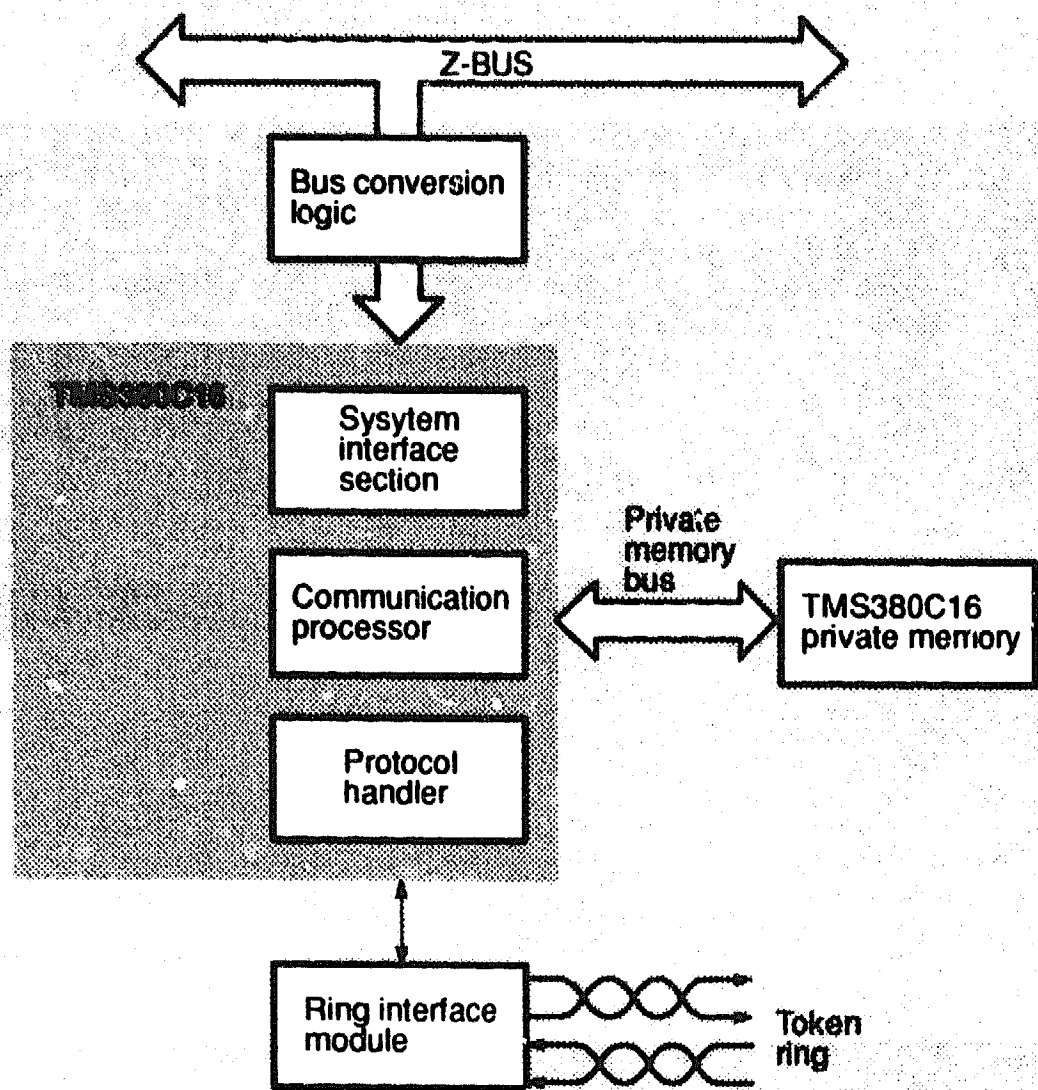
3.9 Shared Memory Jumper (JP1)

Jumper JP1 determines whether the DEQRA's shared memory is visible to the host at powerup. To make the shared memory visible to the host at powerup refer to the *DEC TRNcontroller 100 Hardware Installation* guide. Enable is the normal position of JP1.

3.10 Token Ring Interface

The token-ring interface is made up of the token-ring communications processor, token-ring private memory, the ring-interface module, and bus timing-conversion circuitry. The token-ring interface handles all data and control signals between the board and the token ring. Figure 3-13 shows a block diagram of the token-ring circuitry.

Figure 3-13 Token Ring Circuitry Block Diagram



LKG-49

3.10.1 TMS380C16

The TMS380C16 is Texas Instruments' token-ring communications processor. It is packaged in a single, 132 plastic quad flatpak. It has three major functions: a DMA transfer controller, a protocol handler, and a communications processor. The DMA transfer controller is the interface between the token circuitry and the rest of the DEQRA. It transfers data between the token-ring private memory and shared memory. It also notifies the CPU of any change in the token-ring status through interrupts. The communications processor runs source code provided by Texas Instruments to control the DMA transfer controller and monitor the token ring. The protocol handler performs hardware-based protocol functions. The protocol conforms to the IEEE 802.5 standard for a 4 or 16 Mbytes/s token-ring local area network. The CPU accesses the TMS380C16 through six registers. Two of these registers (SIFINT and SIFACTL) are used for controlling the TMS380C16, whereas the other four are used to access the TMS380C16's internal registers and private memory. Table 3-7 lists the registers with their addresses.

Table 3-7 TMS380C16 Register Addresses

TMS380C16 Registers	Address	Description
SIFDAT	13F7000	Data
SIFDATI	13F7002	Data
SIFADR	13F7004	Address
SIFINT	13F7006	Interrupt
SIFACTL	13F7008	Control
SIFADR2	13F700A	Address
SIFADRX	13F700C	Extended address

3.10.2 TMS380C16 Reset Operation

When the TMS380C16 is reset, the device goes into a halted state and waits for its communication processor's halt bit in its adaptor-control register to clear. This prevents the TMS380C16 from trying to execute code from power-up. Since the TMS380C16 fetches instructions from its private DRAM memory, the code needs to be downloaded after any hardware reset before the device can begin to execute instructions. After the code is downloaded and the halt bit is cleared, the TMS380C16 initializes its internal registers from private memory and runs its internal diagnostic tests. If the tests pass, the TMS380C16 begins normal operation.

3.10.3 Bus Timing Circuitry

The TMS380C16 is designed for standard 68000 bus architecture; therefore, conversion circuitry is needed to attach it to the Z-BUS. A PAL is used to synchronize and convert the control signals traveling between the Z-BUS and the TMS380C16. This conversion logic minimizes delays while providing deterministic transition cycles between these two asynchronously clocked sub-systems.

3.10.4 Token Ring Memory

The token-ring private memory consists of 512 Kbytes of DRAM with 100 ns access time. The DRAM is configured as 256k by 16 bits. The TMS380C16 uses the private memory to store instruction code as well as for packets of data received and transmitted onto the token ring. The TMS380C16 can be programmed for transparent mode allowing the CPU to have access to almost all of its private memory. (Because the TMS380C16 uses a specific area of memory as internal register space, the CPU is restricted from accessing this area.)

3.10.5 Ring Interface Module

The TMS380C16 is connected to the token ring by way of a Pulse Engineering analog interface module. This module handles the actual electrical interface to the ring and adheres to the token-ring standard IEEE 802.5. The DEQRA 4 or 16 Mbytes/s token-ring adapter card, uses the token ring optimized line interface (TROLI). The TROLI consists of the Texas Instruments' TMS38053 chip and some passive components. The physical ring is composed of a twisted wire pair for receiving data, and a separate twisted wire pair for transmitting data. The physical connection to the ring is a DSUB9 socket. Figure 4-1 shows the pin assignments for this connector.

3.11 Host-to-Board-to-Token Ring Transfers

The DEQRA interfaces to the host through shared memory and the CSR, and to the token ring by means of the TMS380C16. In normal operation, transferring a buffer from the host to the token ring begins by the host filling a buffer in shared memory and notifying the CPU, through the CSR, that a buffer is waiting to be transferred. The CPU performs any data manipulation that is needed before requesting the DMA transfer controller in the TMS380C16 to transfer the buffer. The DMA transfer controller transfers the buffer from shared memory to the TMS380C16's private memory. Once the buffer is in TMS380C16's private memory, the TMS380C16 notifies the CPU that the transfer is complete and sends the buffer out onto the token ring through the ring interface. If a buffer from the ring is destined for the

host, the TMS380C16 reads the buffer through the ring interface and stores it in its private memory. The TMS380C16 then notifies the CPU that a buffer is waiting to be transferred to the host. The CPU requests the TMS380C16 to transfer the buffer to shared memory. The DMA transfer controller on the TMS380C16 transfers the buffer, and the TMS380C16 notifies the CPU when the transfer is complete. The CPU first performs manipulation on the data, if needed, and then notifies the host, by way of the CSR, that a buffer is waiting in shared memory. The host reads the buffer from shared memory and notifies the CPU when the read is complete.

3.12 Details of Operation

The remainder of this chapter discusses the DEQRA's:

- Interrupt structure
- Board timing
- Z-BUS arbitration
- Resets
- Bus errors
- NMI functions

3.12.1 Interrupts

The DEQRA supports the 68020 interrupt level structure.

3.12.1.1 Interrupt Structure

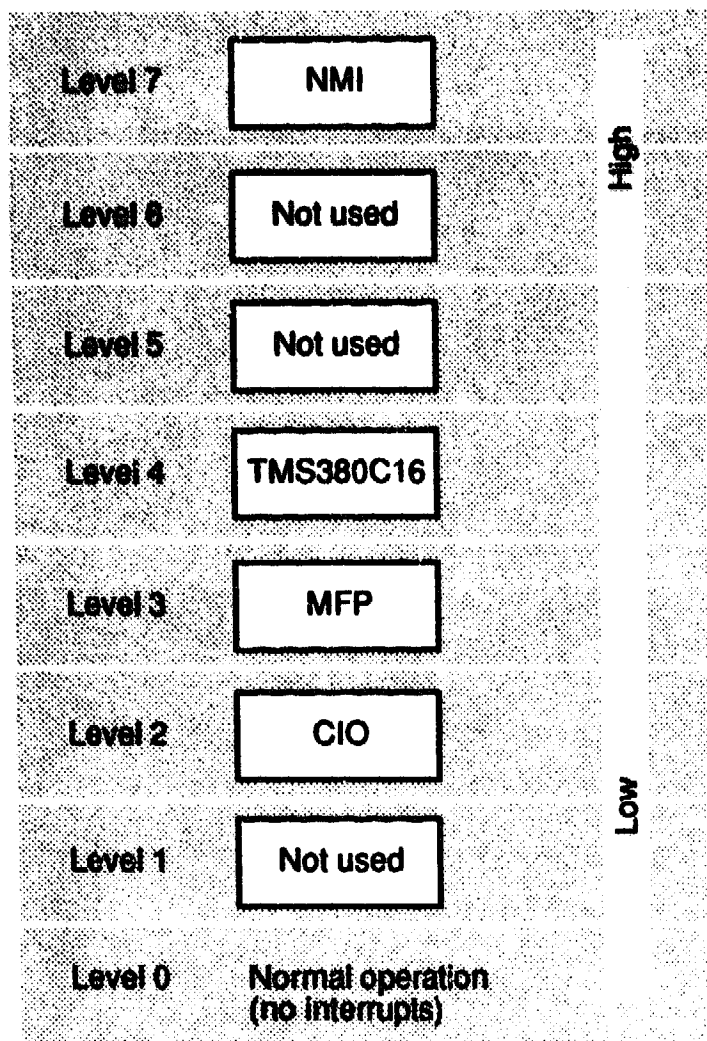
The DEQRA uses four of the seven 68020 processor-interrupt levels. The seven-level interrupt structure is implemented by encoding interrupt requests on three interrupt-priority-level signal pins. These input pins are sampled by the processor. If any of them are asserted, and if the encoded priority is greater than the current interrupt mask level, an interrupt request is made. The processor services the pending interrupt at the next instruction boundary. The 68020 supports both device-supplied vector and autovector peripherals, but the devices on the DEQRA are all DSV devices. When the processor is ready to initiate interrupt servicing, it begins an interrupt-acknowledge cycle. This cycle is similar to a normal read cycle. An \overline{IACK} cycle, at the pending interrupt level, enables the device requesting an interrupt to present a vector byte. The processor uses this byte as an offset to locate the peripheral's interrupt-service entry point.

3.12.1.2 Interrupt Levels

There are only three devices on the DEQRA that use interrupts. Therefore, there is only one device for each interrupt level. The following is a description of the interrupt levels shown in Figure 3-14:

- Level 7 is a non-maskable interrupt that causes the 68020 to stop the execution of the current program and break to the debugging tool. NMIs may be issued by pushing the NMI button on the front edge of the board, or by writing a 1 to bit 7 of the host interface CSR from the host side.
- Level 6 is not used.
- Level 5 is not used.
- Level 4 is used by the TMS380C16, the token ring communications processor.
- Level 3 is used by the MFP (operating system and status bits).
- Level 2 is used by the CIO for host CSR interface support.
- Level 1 is not used.

Figure 3-14 Interrupt Priority Structure



LKG-5027-911

3.12.2 Timing

The following sections describe the clock-generation logic of the DEQRA and the transaction timings that result from its implementation. Note that clock generation for token-ring circuitry is isolated from all other timing on the board and is discussed in Section 3.12.2.1.

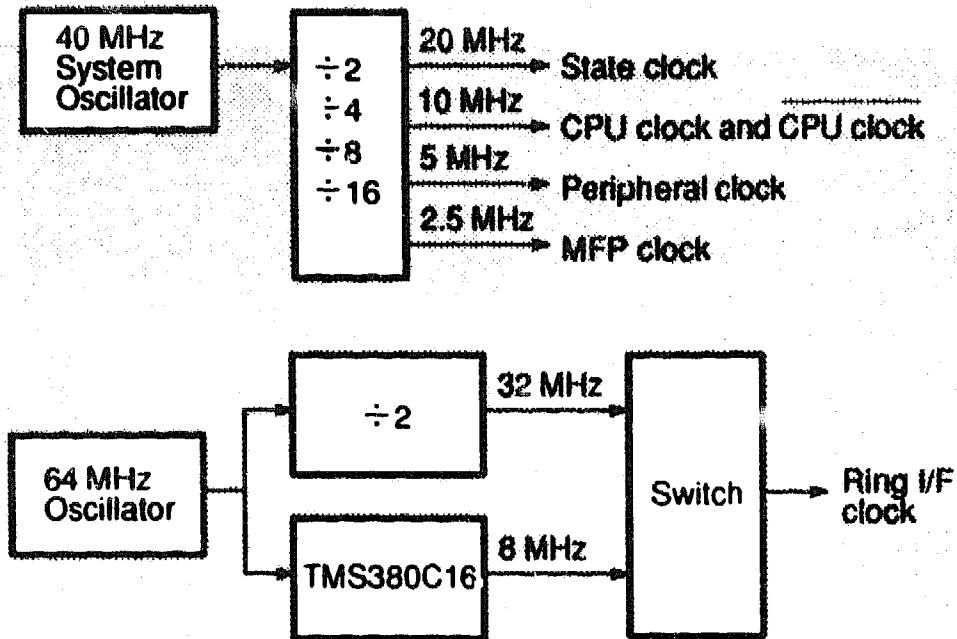
3.12.2.1 Clock Generation

The DEQRA uses synchronous state machine design techniques throughout the control logic to ensure accurate circuit performance and to increase circuit reliability. All clock signals, except the token-ring clock signals, are derived from a divider circuit that is clocked by the 40 MHz system oscillator as shown in Figure 3-15.

The 20 MHz state clock is used by the processor bus memory controller and DSACK generator state machines. The 10 MHz clock and its complement are used by the 68020, the Z-BUS conversion and arbitration state machines, and the shared memory controller. The 5 MHz clock (PCLK) is used by the CIO and the Z-BUS interrupt state machine. The 2.5 MHz clock is used by the MFP and is further divided for the very low resolution RESET, NMI, and BUS_ERROR state machines.

The token-ring circuitry uses a 64 MHz oscillator. The TMS380C16 divides the 64 MHz internally and provides an 8 MHz clock for the 4 Mbytes/s ring-interface selection. For the 16 Mbytes/s token-ring, the 64 MHz clock is divided by a simple D-flop to 32 MHz. The 32 MHz clock runs the ring-interface module. Switching circuitry is provided to select the 8 MHz clock or 32 MHz clock depending on the respective 4 Mbytes/s or 16 Mbytes/s ring-interface selection.

Figure 3-15 DEQRA System Timing



LKG-5030-911

3.12.2.2 Transaction Timing

The advanced architecture of the 68020 makes exact instruction-timing calculations difficult. The 256-byte instruction cache, the three-stage prefetch pipeline, the execution overlap capabilities, and the effects of operand misalignment complicate these calculations. Timing is also affected by memory system refreshes and Z-BUS arbitration delays. Refresh delays and Q-bus memory usage also affect Z-BUS device memory cycle times. Figure 3-16 shows transaction timing information for CPU and TMS380C16 transfers.

Figure 3-16 DEQRA Transaction Timing

	Processor bus				Z-BUS		
	MEM	EPROM	MFP	Regs	MEM	CIO	TMS
68020 cycle @10 MHz	333ns*	500ns	750ns	333ns	416ns**	583ns	583ns
TMS380C16 cycle @8MHz	N/A	N/A	N/A	N/A	500ns	N/A	N/A
Z8016 cycle @5 MHz	N/A	N/A	N/A	N/A	500ns	N/A	N/A

*For cycles that are not delayed by refresh.

**For cycles that are not delayed by refresh or shared memory cycles already in progress.

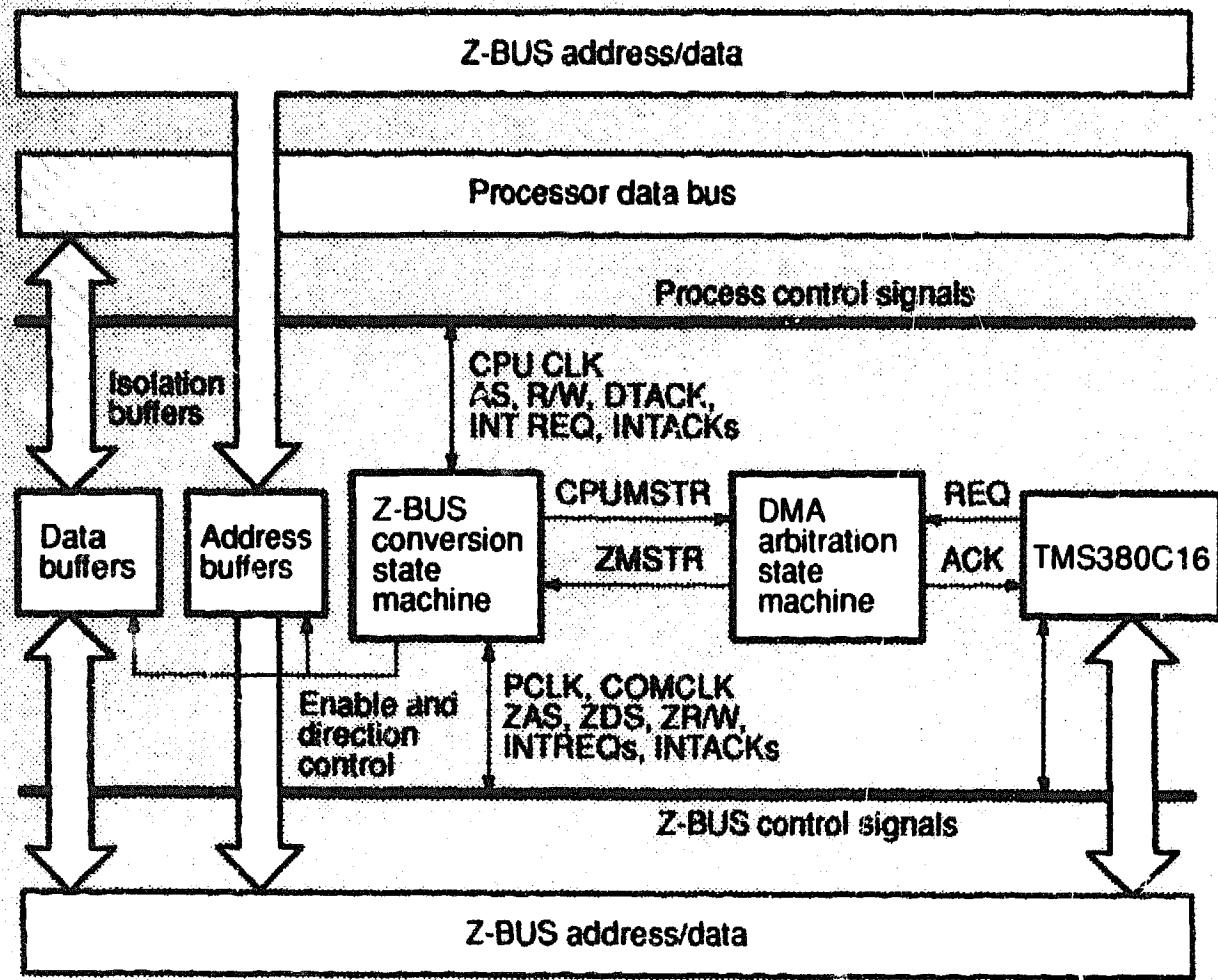
Typical instruction time **MOVL (Am)+, (An)+**
 worst case = 12 clocks @ 83.3 ns = 1.000 μ s
 cache case = 7 clocks @ 83.3 ns = .583 μ s

LKG-4957-91

3.12.3 Z-BUS Arbitration

The 68020 or the TMS380C16 may request use of the Z-BUS. The Z-conversion and the arbitration state machines work together to determine which of these will be the next Z-BUS master. See Figure 3-17.

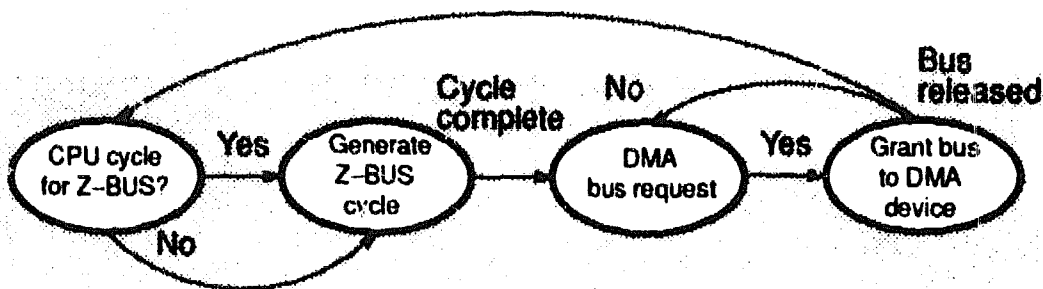
Figure 3-17 Z-BUS Arbitration Block Diagram



LKG-5031-911

These state machines are clocked on opposite phases of the 10 MHz clock, creating a 50 ns time slicing that effectively alternates Z-BUS access windows between the two state machines. This time slicing guarantees that the CPU and the TMS380C16 have equal access to the bus. When one of the state machines is master of the bus, it prevents the other from granting the bus. The stalled state machine generates appropriate signals to delay pending transactions until it gains control of the bus at the end of the current transaction. See Figure 3-18.

Figure 3-18 Z-BUS Arbitration State Diagram



LKG-4955-911

3.12.3.1 Z-Conversion State Machine

When the CPU begins a transaction in the Z-BUS address space, and the arbitration controller has not granted use of the bus to a TMS380C16, the Z-conversion state machine enables, disables, and sets the direction of the CPU-to-Z-BUS address and data buffers, generates Z-BUS address and data strobes, and returns the \overline{DTACK} signal to the processor support logic to indicate the end of the cycle. The state machine also controls the variable \overline{ZAS} to \overline{ZDS} delay timing and enables the vector buffer during interrupt-acknowledge cycles.

3.12.3.2 Arbitration State Machine

The arbitration state machine monitors the request lines from the TMS380C16 and issues grant acknowledgments to it.

3.12.4 Reset

In normal operation, the reset controller keeps the \overline{RESET} signal in a high-impedance state. This allows the 68020 to drive the line during the execution of a reset instruction. The reset controller PAL device generates an 85 μ s minimum reset pulse for the following conditions:

1. At power-up time, or at low power conditions, when the Q-bus signal \overline{DCLO} is asserted.
2. When the Q-bus signal \overline{BINTT} is asserted, if the INIT_DIS bit in the board-configuration register is not set.
3. When bit 5 of the host interface CSR is written by the host.
4. When the reset switch on the front edge of the board is pressed.

All of the devices, state machines, and configuration registers on the board are initialized to a known state when the \overline{RESET} signal is driven low. The Z-conversion state machine asserts \overline{ZAS} and \overline{ZDS} low simultaneously to reset the Z-BUS devices. The processor bus and Z-BUS memory controllers generate continuous refresh cycles during the reset time.

3.12.5 Bus Error

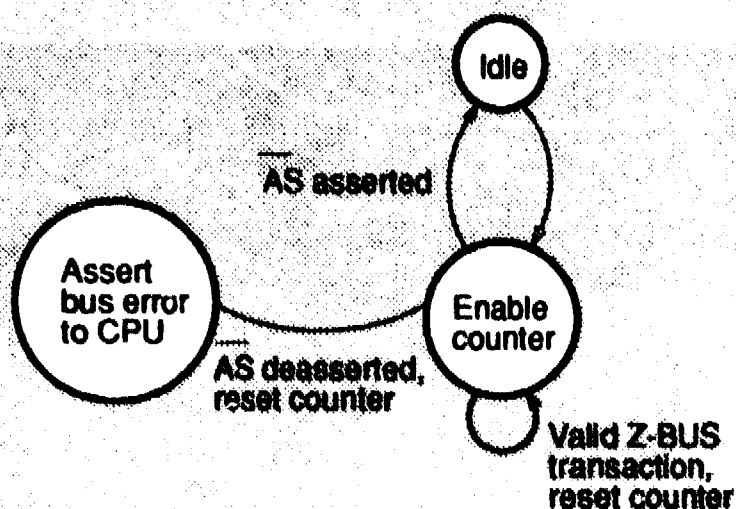
The bus exception controller monitors the bus activity and alerts the CPU when a transaction is not completed in a reasonable time. The bus-exception controller is implemented as a counter that is enabled during address strobe assertion.

3.12.5.1 Bus Exception Controller Operation

During normal transactions, the CPU ends a cycle by de-asserting the address strobe when it receives the \overline{DSACK} signals (See Figure 3-19). This de-assertion of the address strobe resets the bus exception counter. If the addressed device is defective, or an illegal address is attempted, no \overline{DSACK} s will be generated to end the cycle. The counter remains enabled, since the CPU keeps \overline{AS} asserted while it waits for \overline{DSACK} , and will reach its terminal count and assert the bus error \overline{BERR} signal to the CPU after 21 μ s. The CPU recognizes the \overline{BERR} signal, ends the cycle, breaks to the debugging tool, and then displays (on the console terminal) the transaction type, the address that was active at the time of the bus error, and the CPU register contents.

Processor bus control logic on the DEGRA causes bus errors to occur for write cycles in the EPROM memory space, or if an unassigned address is accessed.

Figure 3-19 Bus Exception State Diagram



LKG-5026-911

3.12.5.2 Delayed Z-BUS Transactions

The use of a dual-bus architecture means that a CPU transaction in the Z-BUS space could be delayed longer than the bus exception timeout period when the TMS380C16 is doing a block transfer. The bus-exception controller monitors the Z-BUS \overline{ZMSTR} and \overline{ZDS} signals and resets the counter, effectively restarting the count for each valid transaction on the Z-BUS. After the TMS380C16 finishes its transfer and releases the bus, the delayed CPU cycle begins its Z-BUS transaction and the bus-exception counter operates as described in Section 3.12.5.1.

There are no illegal address bus errors in the Z-BUS space. The Z-conversion state machine returns \overline{DTACK} signals for all CPU transactions in that space; therefore, a bus error reported in Z-BUS space means that either the TMS380C16 is the bus master but has stopped transferring data, or that one of the Z-BUS state machines is malfunctioning.

3.12.6 Non-Maskable Interrupt

The interrupt request controller issues a non-maskable interrupt request when the host has written bit 7 of the CSR register or when the NMI button on the board edge is pressed. If the NMI came from the button or if the host writes an 80_{16} to the CSR, the NMI routine displays the contents of the CPU registers, disassembles the instruction that was being executed at the time of the NMI, breaks to the debugging tool, and waits for keyboard commands. If the host writes an 81_{16} to the CSR, the NMI routine interprets this as a request from the host to run extended diagnostics. The complete self-test diagnostics are executed as if this were a power-up reset. This capability has been implemented to increase host-interface testing in a manufacturing environment and to give users the ability to request complete diagnostics from the host system.

Electrical Interfaces on the DEQRA

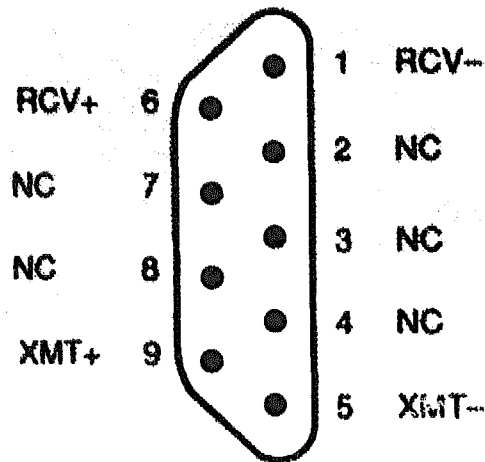
The DEQRA contains two different electrical interfaces. The primary interface is the token ring port that conforms to the IEEE 802.5 standard. The secondary interface is the console port.

4.1 Token Ring Port

The token-ring interface is compatible with an industry standard token ring local area network. Two twisted-pair wires, one pair for transmitting and one pair for receiving, comprise the physical token ring. This employs a differential Manchester-type coding scheme.

The token ring port uses the standard D-subminiature 9-pin connector. Figure 4-1 shows the pin assignments for the token ring port connector. The BN26P adapter cable must be used to connect the DEQRA to the token ring port.

Figure 4-1 Pin Assignments for the Token Ring Port Connector



LKG-4958-91

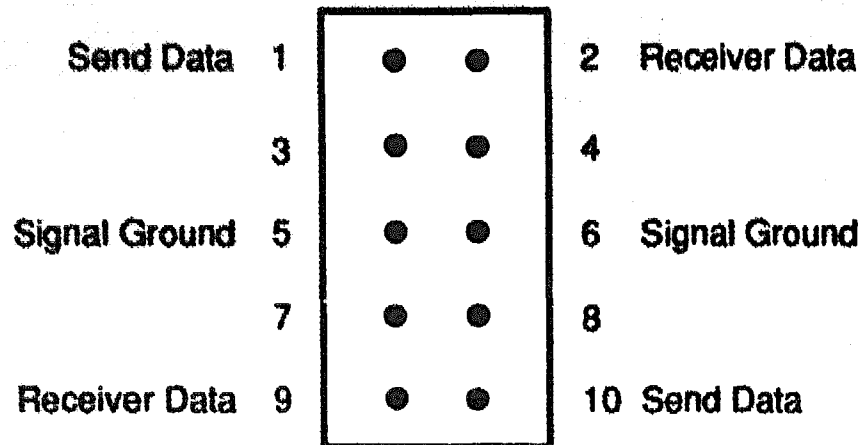
4.2 Console Port

The console port uses a 10-pin box connector. The console port is used for debugging Digital supplied microcode and for running specific diagnostic tests on the DEQRA board.

A BC29E-15 console cable is used to connect a console for direct RS-232 communication with the DEQRA. One end of the console cable has a 10-pin box connector and the other end has a D-subminiature 25-pin connector. The 10-pin box connector end of the console cable may be inserted in either orientation because the five signals are mirror-imaged at the 10-pin box connector on the DEQRA board.

Figure 4–2 shows the pin assignments for the console port connector.

Figure 4–2 Pin Assignments for the Console Port Connector



LKG-4959-91

Table 4–1 shows the pin assignments for both ends of the console cable.

Table 4-1 Console Cable Pin Assignment

10-Pin Box Connector	Signal	25-Pin DSUB Connector
1, 10	Send Data	3
2, 9	Receive Data	2
5, 6	Signal Ground	7

During normal operation the console port is not used. To prevent RFI emissions unplug the console cable.

NOTE

To provide ESD protection for the console interface ensure the console cover plate is in place.

Using the ODT68 Debugging Tool

The ODT68 debugging tool provides a tool for debugging VAX based application programs that use the DEC TRNcontroller 100 Q-Bus-to-Token Ring Adapter (DEQRA). This chapter describes the ODT68 debugging tool and defines the ODT68 command set.

NOTE

The ODT68 debugging tool is not used during normal maintenance.

5.1 Description of the ODT68

The available ODT68 commands are divided into a main command set and two command subsets. ODT68 displays three different command prompts depending on which command set you are accessing.

1. *program counter >*

The current contents of the program counter display in this prompt. This is the prompt that is seen when accessing the main debugging-tool commands. From here, you may examine or set memory, examine or set registers, set or clear breakpoints, trace code, and so on. The commands available to you when you see this prompt are described in Chapter 6. If you have defined an offset other than zero, the difference between the program counter and that offset displays as the prompt. For example, if you set no offset, the prompt may look like this:

```
810EE0::1>
```

If you set an offset, like 0, is,

```
810EE0::> 0 810000
```

the prompt changes to:

EE0:>

2. Aux >

This is the prompt that displays when accessing the commands from the auxiliary command subset. Access this set of commands by entering the **AUX** command at the program counter prompt. When the Aux prompt displays, you can then enter commands from the auxiliary command set. The commands available at the Aux prompt are described in Chapter 7.

3. Daig >

This is the prompt that displays when accessing the commands from the execute diagnostics command subset. Access this set of commands by entering the **DIAGS** command at the program counter prompt. When the Diag prompt displays, you can then enter commands for executing diagnostic tests for the system board. These commands are described in Chapter 8.

5.2 Required Equipment

To use the ODT68 to test applications for the DEQRA you need the following equipment:

- DEQRA installed in a VAX host system
- 15 foot console cable (BC29E-15)
- 9,600 bits/s RS-232 terminal for use as the console

5.3 Console and Terminal Installation

Use the console cable (BC29E-15) to connect a terminal to the DEQRA. Plug the 10-pin connector at one end of the console cable into the 10-pin header on the edge of the controller board. Plug the 25-pin D connector into the terminal, which must be set to a data rate of 9,600 bits/s.

5.4 Accessing ODT68

After the DEQRA board has completed the power-on/reset diagnostics, you can access the debugging tool by entering a **Ctrl/C** from the keyboard, by pressing the non-maskable interrupt (NMI) switch on the board, or by sending an NMI request through the host driver. You can also access the ODT68 by placing the 68020 assembly-language instruction **ILLEGAL** in the application. This will generate a breakpoint or "*panic trap*" when the processor encounters it, so

that the execution of the application stops and ODT68 starts. To return to the application program, use the **GOTO** command (see Chapter 6).

5.5 Overview of ODT68 Command Processing






The ODT68 commands and parameters are not case sensitive.





You must separate parameters from each other and from the command verb by at least one blank space or tab. If a parameter is a string containing either spaces or tabs, that parameter must be delimited with the double-quote character (").

If all parameters are optional, enter a **Return** following the command verb and all parameters are set to their default values. If parameters are included, you must enter them on the same line as the command verb. If you fail to enter any required parameters, ODT68 will prompt you for those parameters.

The ODT68 contains a line editor for making changes to the command line. Table 5-1 describes the keys that are available in the ODT68 line editor and their functions.

Table 5-1 Keyboard Functions for ODT68

Character	Function
	Moves the cursor to the right
	Moves the cursor to the left
	Scrolls up through earlier commands
	Scrolls down through later commands
	Deletes the character to the left of the cursor
Ctrl/A	Toggles between insert and overstrike mode
Ctrl/C	Terminates any command in progress
Ctrl/E	Moves the cursor to the end of the line
Ctrl/H	Moves the cursor to the beginning of the line
Ctrl/Q	Resumes output to the console
Ctrl/S	Halts output to the console
Ctrl/U	Deletes from the front of the line to the cursor
Ctrl/Y	Aborts a command

The command processor remembers the last 20 input lines. You may recall these by using the  and  keys. Pressing the  key moves "up" toward the earlier commands; pressing the  key moves "down" toward the later

commands. Commands may also be recalled by issuing the **RECALL** command.

In addition to the **RECALL** command, general system commands available in all command sets include a **HELP** command and an **EXIT** command.

The *underscore*, *plus*, and *pound-sign* characters have special meanings when used in ODT68 command descriptions.

- **Underscore _**

To aid in readability, the underscore character may be input any number of times within an address. For example, the address entered as 84__98_87 is equivalent to address 849887.

- **Plus +**

When the plus character is appended to an address, the value of the offset register is added to that address. Refer to the **OFFSET** command.

- **Pound sign #**

When a pound sign is appended to a number parameter, ODT68 will not use the parameter as an address, but will calculate an address by adding the parameter value to the last address entered. The pound sign used alone is equivalent to 0#. For example:

- This command dumps data starting at address 840000 and ending at address 840666:

D 84_0000 666#

- This command dumps data starting at address 840000 and ending at address 840023:

D # 23#

When the letter R is appended to an address, the next characters are expected to be a register name selected from Appendix B. The contents of the specified register are added to the entered address. When the letter R is not preceded by an address, the contents of the register are used as the address. For example, if A2 contains 820000:

- This command dumps four bytes of memory starting at address 820100.

D 100ra2 4#

- This command dumps memory starting at address 820000 and ending at address 8200FF.

D RA2 FF#

Unless otherwise noted, you must enter all numerical command parameters in hexadecimal. In general, only parameters specifying a count are entered in decimal.

Parameters are interpreted according to their positions on the command line. Therefore, you must enter them in the order shown.

EXIT

EXIT

The **EXIT** command exits the Aux or Conf command subset and returns to the main debugging menu. This command has no effect in the main debugging menu.

Format

prompt> E

HELP

The **HELP** command displays the commands available in the current menu. Each command displays with the parameters that apply to it. If **HELP** is entered with no parameters all of the command verbs available in the current command set are displayed. If a parameter is entered, then all of the commands that begin with the specified string display.

Format

prompt> H [*string*]

Where:

string Represents the beginning letter(s) of command verbs that are available in the command set.

Examples

1. After the following command is entered, the syntax for each command that starts with the letter p, **PEEK** and **POKE**, displays on the screen.

prompt> h p

2. After the following command is entered, the message "Sorry, no help on Qi." appears on the terminal.

prompt> H Qi

RECALL

RECALL

The **RECALL** command displays previous keyboard command entries. If an optional parameter is not specified, the last command is recalled. If a numeric parameter *n* is specified, the *n*-th command is recalled. If a non-numeric parameter string is specified, the most recently entered command that begins with the specified string is recalled. If the keyword *ALL* is specified, then all of the command lines stored in the buffer are displayed.

In order to optimize the use of the command recall buffer, the **RECALL** command itself is never stored in the buffer. Additionally, if an identical command is executed consecutively, the second command is not stored into the recall buffer.

Format

```
prompt> REC [number | string | ALL ]
```

Where:

<i>number</i>	Is a decimal number representing a specific numbered command to be recalled.
<i>string</i>	Is a string of characters indicating that the most recently entered command beginning with the specified string is to be recalled.
<i>ALL</i>	Causes all of the command lines stored in the buffer to display.

Examples

1. The following command recalls the last command entered on the command line:

```
prompt> REC
```

2. The following command recalls the third most recent command entered on the command line:

```
prompt> REC 3
```

3. The following command recalls the last disassemble-instructions command entered on the command line:

```
prompt> REC DI
```

4. The following command displays the last 20 commands entered on the command line. The commands are numbered with the most recent as number one and continues through number 20 as the oldest command in the buffer.

```
prompt> REC ALL
```

Using the ODT68 Debug Command Set

The ODT68 commands are divided into a main command set and two command subsets. This chapter describes the main command set. Refer to Chapter 7 for the auxiliary command subset and Chapter 8 for the execute diagnostics command subset.

6.1 Introduction

The ODT68 debug commands shown in Table 6–1 can be accessed at the program counter prompt. These commands allow you to:

- Set and clear breakpoints
- Display and change memory and registers
- Fill memory with a pattern or search for a pattern
- Execute or step through the application program

Table 6-1 Summary of Debug Commands

Command	Command Abbreviation	Description
EXIT	e	Disables at the debug command level. For a full description, see Chapter 5.
HELP	h	Gets help on command syntax. For a full description, see Chapter 5.
RECALL	rec	Recalls previously entered commands. For a full description, see Chapter 5.
ASCII	as	Displays memory as ASCII characters.
AUX	a	Accesses auxiliary command subset.
BREAKPT	b	Sets or clears breakpoint.
DIAGS	d	Accesses the diagnostics command subset.
DISA	dis	Disassembles memory.
DUMP	du	Dumps memory in hexadecimal and ASCII.
FILL	f	Fills memory with a constant pattern.
GOTO	g	Begins execution.
OFFSET	o	Sets offset.
PEEK	p	Peeks at the address.
POKE	po	Pokes at the address.
REGS	r	Displays or modifies registers.
SEARCH	s	Searches memory for value.
TRACE	t	Traces instructions.
TTB	tt	Traces to branch.

ASCII

This command displays the contents of a memory region in ASCII format. You must specify the starting address. If you do not specify an ending address, this command displays one line (64 bytes) and stops.

Format

PC> AS *starting_address* [*ending_address*]

Where:

<i>starting_address</i>	Is a hexadecimal number representing the first memory address whose contents are to be displayed in ASCII format
<i>ending_address</i>	Is a hexadecimal number representing the last memory address whose contents are to be displayed in ASCII format

Examples

1. The following command will display memory in ASCII, 64 bytes per line, starting at address 820000:

```
PC> AS 82_0000
```

2. The following command will display memory in ASCII, from address 810000 to address 810030:

```
PC> AS 81_0000 81_0030
```

AUX

AUX

This command allows access to the auxiliary debugging commands. After you enter this command, only the auxiliary debugging commands are accessible until the **EXIT** command is entered, which again allows access to the main debugging commands. The auxiliary debugging commands are described in Chapter 7.

Format

PC> A

BREAKPT

This command sets, displays, or clears instruction-tracing breakpoints. You can define up to 10 breakpoints, each referenced by its address. When the processor encounters a breakpoint, control is returned to ODT68 unless a pass-count value is specified. If you specify a pass-count value, ODT68 is re-entered only when the processor visits the breakpoint address the indicated number of times. Specify the pass count in decimal. If you precede an address with a hyphen, the breakpoint at that address clears. If *z* is the only parameter, all the breakpoints clear. If you do not specify a parameter, the entire list of current breakpoints is displayed. Breakpoints cannot be set in code that resides in ROM.

Format

PC> B [[-] *address* [:*pass_count*]]

or

PC> B *z*

Where:

<i>address</i>	Is a hexadecimal number that specifies the address at which a breakpoint is to be set.
<i>pass_count</i>	Is a decimal number that specifies how many times the address must be accessed before control is returned to ODT68

Examples

1. The following command displays the current breakpoint list:

PC> B

2. The following command creates a breakpoint at address 838D34:

PC> B 838D34

3. The following command creates a breakpoint at address 814326, but creates it so that the processor must reach that address three times before returning control to ODT68:

PC> B 814326:3

4. The following command removes the breakpoint at address 838D34:

PC> B -838d34

BREAKPT

5. The following command clears all of the breakpoints:

```
PC> B Z
```

DISA

This command disassembles memory contents into assembly code. The address of each instruction is displayed at the left margin. All operation codes and register names are displayed in lowercase, and all of the values are displayed in hexadecimal. PC-relative addressing modes (including the operands for BRA, BSR, and BCC) are disassembled using the value of the offset as the base address or zero if an offset has not been set (see the **OFFSET** command).

If the address provided is not the first word of an instruction or not an instruction at all, the results will be unpredictable. The disassembler does not recognize illegal addressing modes for particular instructions, although it does indicate an error if an illegal operation code is encountered. Coprocessor instructions are considered illegal.

ODT68 saves the last address entered. This address can be represented in future commands with the pound sign (#). For the **DISA** command only, the address saved is the address of the next instruction following those that were disassembled. For all other commands, the last entered address stored is the last address actually entered.

The number of instructions to disassemble is specified in decimal. If this parameter is omitted, 16 instructions will be disassembled. If the starting address is also omitted, the current value of the program counter is used as the default.

Format

PC> **DIS** [*starting_address*] [*number_of_instructions*]

Where:

starting_address

Is a hexadecimal number that specifies the address of the first instruction to be disassembled

number_of_instructions

Is a decimal number that specifies the number of instructions to be disassembled

Console Message

This message means that an unrecognized operation code was encountered.

DISA

Examples

1. The following command disassembles 16 instructions (the default) beginning at address 823456:

```
PC> DIS 82_3456
```

2. The following command disassembles 16 instructions beginning at the address currently stored in the program counter:

```
PC> DIS
```

3. The following command disassembles 16 instruction beginning at the address pointed to by register A3:

```
PC> DIS ra3
```

4. The following command disassembles three instructions starting at address 820000:

```
PC> DIS 820000 3
```

5. The following commands disassemble 16 instructions starting at the instruction following those previously disassembled (or starting at the last address entered, if the previous command was not **DIS**). In this example, the fourth instruction after address 820000 is the first instruction disassembled.

```
PC> DIS 0#
```

```
or
```

```
PC> DIS #
```

DUMP

This command displays the contents of a memory region. Each line displays 16 bytes and is divided into three sections: the starting address, the data starting at that address shown in hexadecimal format, and the same data displayed in ASCII format. If you do not specify an ending address, this command displays four lines (64 bytes) and stops. The following example shows the contents of a memory region:

```
0080_0400  00 81 0E 96 00 00 00 77-00 00 00 01 00 81 3C 5C |.....W.....<\|
0080_0410  56 31 2E 32 02 00 00 00-00 81 1A 6E 00 00 00 00 |V1.2.....n....|
0080_0420  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 |.....|
0080_0430  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 |.....|
```

Format

PC> DU *starting_address* [*ending_address*]

Where:

<i>starting_address</i>	Is a hexadecimal number that specifies the address of the first memory location that is to be displayed.
<i>ending_address</i>	Is a hexadecimal number that specifies the address of the last memory location to be displayed.

Examples

1. The following command displays 64 bytes on four lines starting at address 820000:

```
PC> DU 82_0000
```

2. The following command displays memory from address 810000 through address 810030:

```
PC> DU 810000 810030
```

FILL

FILL

This command fills a section of memory with a specified value.

Format

PC> F *size_code first_address last_address hexadecimal_value*

Where:

<i>size_code</i>	Must be the letter B, W, or L to specify that this value should be written to every byte, every word, or every longword.
<i>first_address</i>	Is a hexadecimal number that specifies the first address to be filled.
<i>last_address</i>	Is a hexadecimal number that specifies the address of the last byte to be filled (regardless of the size code).
<i>hexadecimal_value</i>	Is the hexadecimal value to be stored in the specified memory locations.

Examples

1. The following command places zeros in all of the byte locations from address 810000 through address 81FFFF:
PC> F B 810000 81FFFF 0
2. The following command places the pattern 00C4 from address 810000 through address 817FFF:
PC> F W 810000 817FFF C4
3. The following command places the pattern 1234ABCD from address 820000 through address 82FFFF:
PC> F L 820000 82ffff 1234abcd

GOTO

This command starts the execution of a program. If you do not specify an address, present value of the program counter is used as the starting point.

Format

PC> G [address]

Where:

address Is a hexadecimal number that specifies the address where program execution is to start.

Examples

1. The following command starts executing the program from the current value of the program counter:
PC> G
2. The following command sets the program counter to address 823456 and then begins executing the program from that address:
PC> G 823456

OFFSET

OFFSET

This command sets up a single relocation value. Once the offset has been set with this command, the offset value is added to any command parameter when a plus sign (+) is appended to the parameter value.

Format

PC> O *address*

Where:

address Is a hexadecimal number that is the offset value.

Example

The following command sets the offset value to 810000.

PC> O 810000

The plus sign can now be used to dump 64 bytes of memory starting at address 810020.

PC> DU 20+

PEEK

This command reads the location specified and displays the location and the value stored in that location on the console. You can then enter a new value to be stored at that location, or enter a line feed or a space character to leave the value unchanged and peek at the next location. This editing process automatically repeats until you enter a **[Ctrl/C]** or a **[Return]**, which terminates the peek environment and returns to the main debugging tool.

The **PEEK** command uses a special syntax to examine and edit the contents of each memory location. Table A-1 describes the active keys used in this special editor.

Format

PC> P *size_code address*

Where:

<i>size_code</i>	Must be the letter B, W, or L to specify the size (byte, word, or longword) of each displayed location.
<i>address</i>	Is a hexadecimal number that specifies the memory location that is to be displayed.

Examples

1. The following command peeks at the byte at address 840000:

PC> P B 840000

2. The following command peeks at the longword beginning at address 83FF00:

PC> P L 83FF00

POKE

POKE

This command complements the **PEEK** command. Use it to store a value in a location without reading it first. This is especially appropriate for sending data to write-only hardware device registers. As with the **PEEK** command, the line feed or space characters open consecutive locations until you enter a **Ctrl/C** or a **Return** to terminate the command.

The **POKE** command uses a special syntax to write the contents of each memory location. Table A-1 describes the active keys used with this special editor.

Format

PC> PO *size_code address*

Where:

<i>size_code</i>	Must be the letter B, W, or L to specify the size (byte, word, or longword) of each location.
<i>address</i>	Is a hexadecimal number that specifies the memory location that the value is to be stored in.

Examples

1. The following command allows writing data to word locations beginning at address 840000:

PC> PO W 840000

2. The following command allows writing data to longword locations beginning at address 840000:

PC> PO L 840000

REGS

This command displays the processor registers. When you do not specify parameters, the contents of all registers display, followed by a disassembly of the instruction at the current program counter. For example:

```

d0= 0000_0019    d1= 0000_0013    d2= 0000_2004    d3= 0082_9418
d4= 0000_0002    d5= 0000_E1DC    d6= 0000_E1DD    d7= 0000_0001
a0= 0081_3BB4    a1= 0081_3588    a2= 0082_D0B0    a3= 0081_0524
a4= 0082_CB90    a5= 0000_0000    a6= 008C_FFC0    a7= 008C_FFC0
usp= 008C_FFC0    isp= 0081_3348    pc= 0082_0024    sr= 2700_s-7-
sfc= 0000_0007    dfc= 0000_0007    vbr= 0081_0000    msp= 0000_0000
cacr= 0000_0001  caar= 0000_0000
00820024: 2044                move.l    d4, a0

```

If you specify a register name, only the contents of that specific register display. You may modify a register, without first looking at its contents, by typing a new value after the register name.

The register name abbreviations are listed below. See Table B-1 for the descriptions of these registers.

<i>DO</i>	<i>D4</i>	<i>AO</i>	<i>A4</i>	<i>USP</i>	<i>VBR</i>
<i>D1</i>	<i>D5</i>	<i>A1</i>	<i>A5</i>	<i>ISP</i>	<i>CACR</i>
<i>D2</i>	<i>D6</i>	<i>A2</i>	<i>A6</i>	<i>MSP</i>	<i>CAAR</i>
<i>D3</i>	<i>D7</i>	<i>A3</i>	<i>A7</i>	<i>SFC</i>	<i>SR</i>
				<i>DFC</i>	<i>PC</i>

You must specify a hexadecimal when you change the status register. Format it as follows:

```
sm-7-xnzvc
```

REGS

The characters shown correspond to the status bits, as follows:

Code	Description
s	supervisor state
m	master state
x	extend
n	negative
z	zero
v	overflow
c	carry

A character displays if the corresponding bit is set and blank if the bit is clear. The number in the center shows the interrupt priority mask value (0 through 7).

For example, s-3-nc displays when the processor is in the supervisor state, at interrupt mask level 3, with the negative and carry flags set.

Format

PC> R [register [new_register_value]]

Where:

<i>register</i>	Is 2 to 4 alphanumeric characters that specify the register that is to be displayed or modified.
<i>new_register_value</i>	Is a hexadecimal number that is to be entered into the specified register.

Examples

1. The following command displays the contents of all the registers:

```
PC> R
```

2. The following command displays the contents of register VBR only:

```
PC> R vbr
```

3. The following command does not display the contents of any register, it only changes the value in register A0 to 830F7C:

```
PC> R a0 830f7c
```

SEARCH

SEARCH

This command searches for a byte, word, or longword in a specified range of memory, from a *starting_address* to an *ending_address*. All of the searches examine every location within the specified range. The value being searched for does not need to be on a word or longword boundary even if a W or L is specified.

Format

PC> S *size_code starting_address ending_address hexadecimal_value*

Where:

<i>size_code</i>	Must be the letter B, W, or L to specify the size (byte, word, or longword) of the hexadecimal value being searched for.
<i>starting_address</i>	Is a hexadecimal number that specifies the starting address of the memory range that is to be searched.
<i>ending_address</i>	Is a hexadecimal number that specifies the ending address of the memory range that is to be searched.
<i>hexadecimal_value</i>	Is a hexadecimal number that specifies the value to be searched for. This length of this number must match the size code.

Examples

1. The following command does a byte-by-byte search for F1 between memory location 840000 and memory location 850000:

```
PC> S B 840000 850000 F1
```

2. The following command performs a longword search for 000004E2 between memory location 840000 and memory location 850000:

```
PC> S L 840000 850000 4e2
```

TRACE

This command is used to single step the processor. ODT68 sets the processor into trace mode and executes a single instruction. When the processor completes the instruction, control returns to ODT68.

This command continues to trace until all of the steps requested have been executed, at which time it will display all of the registers on the console and prompt the user for another ODT68 command.

It is possible to single step through code that resides in ROM, but the temporary breakpoint option cannot be used.

Format

PC> T [*number_of_steps*]

Where:

number_of_steps Is a decimal number that specifies the number of instructions to step through.

Examples

1. The following command traces one instruction; the one at the current program counter:
PC> T
2. The following command traces 15 instructions, beginning at the current program counter, and displays the registers when the last instruction executes:

PC> T 15

TTB

This command is similar to the **TRACE** command, except that it executes instructions until there is a change in the program flow (the program counter is updated in a non-sequential manner). This command uses a unique tracing feature of the 68020 that will not signal completion until a branch occurs in the execution of the program.

Format

PC> TT [*number_of_branches*]

Where:

number_of_branches Is a decimal number that specifies the number of branches that must occur before program execution will stop.

Examples

1. The following command traces the assembly program until the first branch in execution occurs:

PC> TT

2. The following command traces the assembly program until five branches have occurred and then it displays the contents of all the registers:

PC> TT 5

DIAGS

This command allows you to access the diagnostics commands. After you enter this command, only diagnostics commands can be accessed until you enter the **EXIT** command, which returns access to the standard debugging commands. The diagnostics commands are described in Chapter 8.

Format

PC> D

Using the ODT68 Auxiliary Command Subset

The auxiliary command subset contains a group of miscellaneous commands that provide information about the DEQRA hardware or set modes of operation for the ODT68 and confidence tests. Table 7-1 summarizes the auxiliary commands.

Table 7-1 Auxiliary Command Subset

Command	Command Abbreviation	Description
EXIT	e	Returns to the main ODT68 command menu. For a full description, see Chapter 5.
HELP	h	Gets help on command syntax. For a full description, see Chapter 5.
RECALL	rec	Recalls previously entered commands. For a full description, see Chapter 5.
BIA	bia	Displays board's burned-in address.
ENABLE	en	Performs confidence tests on reset.
INHIBIT	in	Prevents confidence tests on reset.
SD	<i>none</i>	For vendor internal use only.

BIA

BIA

Each token-ring product must have a unique burned-in address consisting of six bytes. The upper three bytes identify the company making the product and are assigned by the IEEE. The company address assigned to the DEQRA is 00 00 7C. All DEQRA products will have a unique address ranging from 0000 7C00 0000 to 0000 7CFF FFFF. The **BIA** command reads the boards burned-in address, checks its validity, and displays the address on the console.

Format

ALX > BIA

Example

The following command will report the burned-in address of the DEQRA by displaying on the screen " The BIA is 0000 7Cxx xxxx".

Aux> BIA

ENABLE

The **ENABLE** command clears a reserved memory location in the processor memory. This location is used by the five automatic test control programs to determine the required level of testing. When the flag clears, a series of device tests are enabled for execution. A user can execute this command only from the auxiliary menu.

The **ENABLE** command also configures ODT68 so that if the DEQRA is reset, the power-up diagnostics will execute. This is the default condition on power up; thereafter, confidence tests are disabled unless specifically enabled before a reset by using this command. If confidence tests are enabled when the board is reset, the diagnostics will overwrite any code and data that was previously in RAM.

Format

AUX > EN

INHIBIT

INHIBIT

The **INHIBIT** command configures ODT68 so that if the DEQRA board is reset, the power-up diagnostics will *not* be executed. This is the default condition after power-up, so it is not necessary to use this command unless the confidence tests have been enabled (since the board was last reset) with the **ENABLE** command. This command may only be executed from the auxiliary menu.

When the board is reset with confidence tests inhibited, RAM-resident data is preserved; however, ODT68 and its RAM area (addresses 800000 through 80FFFF) are re-initialized. Included in this re-initialization are the vector table, the breakpoints, and the command recall list.

If any breakpoints exist, and an application program is executing when the DEQRA is reset, the BKPT instruction will be left in the code without any record of the breakpoint retained by ODT68. In this case, it is best to redownload the original code.

Format

Aux > IN

SD

The **SD** command is intended only for internal vendor use.

Using the ODT68 Execute Diagnostics Command Subset

This chapter describes the execute diagnostic command subset.

Each command in this subset uses a count parameter. The count parameter defines how many times, in decimal, to repeat the diagnostic. If the count parameter is greater than zero, the test repeats until one of three things occur:

1. It has executed the selected number of times,
2. It fails, or
3. You enter a **Ctrl/C**.

If the count parameter is set to zero, the test continually repeats until you enter a **Ctrl/C**. This type of infinite count is useful for continually repeating a failing test and is helpful in determining the cause of the failure. The count defaults to one if the parameter is not specified. Table 8-1 summarizes the diagnostic command subset.

Table 8-1 Diagnostics Command Subset

Commar	Command Abbreviation	Description
EXIT	e	Returns to main ODT68 command menu. For a full description of this command see Chapter 5.
HELP	h	Gets help on command syntax. For a full description of this command see Chapter 5.
RECALL	rec	Recalls previously entered commands. For a full description of this command see Chapter 5.
CIO	cio	Performs CIO test
INT	int	Performs interrupt test
LOOP	l	Performs all tests
MEMTEST	memt	Performs the memory test
TMS	tms	Performs TMS380 test
TRM	trm	Performs token-ring RAM test
XBUSER	xb	Performs the bus error test
XEPROM	xe	Performs the EPROM test
XMFP	xm	Performs multi-function peripheral test
XRAM	xr	Performs CPU-bus RAM test
XZRAM	xz	Performs Z-BUS RAM test

CIO

The Motorola 68020 CPU clears an interrupt counter flag in memory for each timer, initializes the CIO timers, starts them, and starts the execution of a software timing loop. When the CIO timers expire, they interrupt the 68020. The interrupt service routines simply increment the interrupt counter flag for the appropriate timer. When the software timing loop has expired, the 68020 checks the interrupt counter flags to verify that the number of interrupts generated by each of the CIO's timers are consistent with an expected value.

The first phase tests all three CIO timers independently. The second phase checks two of the timers operating in linked mode. As with the first subtest, this one verifies that the linked timers generate the appropriate number of interrupts within the given time period.

This routine does not test the parallel I/O ports in the CIO. The parallel ports provide the read, write, and hardware handshake logic for the DEQRA's host interface command and status register. These ports require a host system for proper operation and undergo a hosted test procedure during the manufacturing cycle.

The successful completion of this test indicates that the following hardware is functioning properly:

- Processor level-2 interrupt request and acknowledge logic
- Z-BUS interrupt transaction state machine
- Z-BUS interrupt vector to 68020 data bus buffer

Format

Diag > CIO

Example

```
Diag > CIO
CIO test
  - all timers
  - 1 & 2 linked
CIO test 1 passed
CIO test: 1 passed, 0 failed
```

Console Messages

CIO test
- all timers
- 1 & 2 linked
CIO test passed

Indicates that the CIO test has run without errors during one of the automatic testing sequences, such as when the DEQRA is turned on, when it is reset, or when it is initialized.

INTERRUPT OVERRUN

Indicates that a second interrupt request from a timer occurred before the first one was serviced. Note that only timer 2 generates interrupts when the timers are run in linked mode.

TIMER *n* DID NOT INTERRUPT

Indicates that one of the timers failed to interrupt the processor.

TIMER *n* INCONSISTENT

Indicates that the number of interrupts received is not consistent with the number expected.

INT

The previous confidence tests indicate that the devices at each level are capable of proper interrupt transactions with the 68020 and its associated interrupt logic. The INT test verifies that the hardware can properly prioritize simultaneous interrupt requests on multiple levels.

The processor masks all interrupt requests by setting its interrupt-mask level to 7, by initializing a status flag in memory to all ones, and by commanding a device at each interrupt priority level to generate an interrupt request. At this point, an interrupt request should be asserted at each priority level. The processor enables interrupt processing by setting its interrupt-mask level to zero, and starts a software timing loop.

Each interrupt level is assigned a bit position equivalent to its priority number in the status flag. The interrupt service routine for each level first checks that all higher priority bits have been cleared, that no lower priority bits have been cleared, then clears its bit, transmits its level number to the console, and returns. In this manner each interrupt service routine verifies that all higher priority interrupts have already been serviced, and that no lower priority interrupts have been serviced. Interrupt priorities are as follows:

Interrupt Level	Interrupt
7	Non-maskable interrupt (not tested here)
4	TMS380
3	MFP
2	CIO
0	No

Format

Diag > INT

INT

Console Messages

```
Interrupt test
Interrupt Level Test
. . 4 3 2
Interrupt test passed
```

Indicates that the test has run without error when called by an automatic test control program.

```
Diag > INT
Interrupt test
Interrupt Level Test
. . 4 3 2
Interrupt test 1 passed
```

Indicates that the test has run successfully when executed from the Diag prompt

Error Messages

The priority levels display on the console as they are encountered. If an error occurs, each level which detects the error is followed by an exclamation point.

1. The following display indicates that the level 4 interrupt was serviced before the level 5 interrupt. Notice that level 4 and level 5 each recognize this as an error, but level 2 and level 3 cannot determine when the level 4 and level 5 bits were cleared.

```
6 4! 5! 3 2
```

2. The following display indicates that level 5 was not serviced. There is not an EXC connected, but level 3 and level 2 recognize that 5 was not serviced.

```
6 3! 2!
```

LOOP

The **LOOP** command starts an automatic test control program that operates very similar to the manufacturing loopback control program. This control program is used to troubleshoot extremely intermittent hardware failures in a laboratory environment. The program calls the confidence tests for execution and maintains pass-and-fail statistics for each of these tests. The basic check routines are not called. Statistics display on the console when the control program completes execution.

Each test's messages and error reports display on the console as it is executed. When running the **LOOP** program, the debugging routine is not started, regardless of the number of tests with errors or the number of errors in any test. A status table showing the number of times each test passed or failed displays when the specified number of iterations have completed or a **[Ctrl/C]** is entered at the console. Appendix D shows a sample **LOOP** command display.

Remember, a count of zero sets the **LOOP** command in an infinite loop. As always a **[Ctrl/C]** stops the looping, but not the loop currently in progress. A loop can take as long as 20 or 30 seconds to complete.

Format

Diag > L *[count]*

Where:

<i>count</i>	Is a decimal number that specifies the number of times the device tests are to be run before the test is terminated.
--------------	--

Example

The following command loops through the device tests 100 times. Appendix D shows an example of the loop command using the command L 2.

```
Diag > L 100
```

MEMTEST

MEMTEST

This command performs a sophisticated memory test on a specified region of RAM. This enables you to test a small section of either of the DEQRA memory systems with a rigorous set of data patterns. This test helps isolate extremely intermittent memory errors that may be data dependent.

The test writes a data pattern to the specified memory locations, delays, then reads each location and reports errors as they occur. This procedure is repeated for each data pattern in the test. Test results display on the console at the end of each iteration. The test executes repeatedly through the specified region until you enter a **Ctrl/C**. The **Ctrl/C** entry stops the test at the end of the current data pattern; it does not wait until the current pass is complete before aborting the command. When the test stops, the accumulated results are displayed.

NOTE

This is a destructive memory test that destroys code and data residing in the region of memory being tested. Testing RAM between 800000 and 80FFFF will destroy the processor's stack and vector table as well as ODT68's data area. Normal program execution ceases, and error reporting cannot be accomplished. Reset the DEQRA if the **MEMTEST** is run in this memory range.

Format

Diag > **MEMT** *starting_address ending_address*

Where:

<i>starting_address</i>	Is a hexadecimal number that specifies the beginning address for the block of memory to be tested.
<i>ending_address</i>	Is a hexadecimal number that specifies the ending address for the block of memory to be tested.

Console Message

```
Diag > MEMT 820000 821000
Memory Test Beginning ...
Pass 1
Pass 2
Pass 3
Pass 4 ^c
Memory Test ABORTED
at Pass # 4

Memory Test Statistics - Passed 3 Failed 0
```

Error Message

The following message indicates that a memory error has occurred. When an error occurs, the location is read twice and the data from both reads is displayed.

```
ptrn # loc xxxx_xxxx expected xx found xx,xx
```

TMS

The **TMS** command exercises the token-ring communications processor (TMS380C16), by executing a token-ring circuit loopback test. The test is accomplished by first having the CPU load a test packet into shared memory. The CPU then notifies the TMS380 that a transmit packet is ready. The TMS380 loads the transmit packet into token-ring memory. The packet loops back and is received by the TMS380, after which, the TMS380 notifies the CPU that a receive packet is waiting in token-ring memory. The CPU tells the TMS380 to transfer the received packet back to shared memory, where the CPU checks its contents against the contents of the transmitted packet.

The TMS380 must pass its own internal diagnostics tests, be initialized, and open onto the token-ring before it will announce it is ready to transmit packets. The **TMS** command begins testing the TMS380 by first resetting the TMS380 to put it into its known wait-for-download state. Next, the bring-up-diagnostics software (called BUDS) provided by Texas Instruments is downloaded to, and executed on, the TMS380. After that has successfully finished, the CPU initializes the TMS380 to operate in wrap mode. In wrap mode, packets sent from the TMS380 will be looped back in the ring-interface module. After a successful initialization, the TMS380 is ready to transmit and receive packets.

NOTE

In order for the TMS380 to run in wrap mode successfully, you must attach a token-ring cable to the DEQRA.

The circuitry connecting the CPU with the TMS380 is tested during the execution of the **TMS** command. The TMS380's DMA capability along with all of the token-ring interface circuitry is also tested.

Format

Diag > **TMS**

Example

```
Diag> TMS 
4/16 megabit ring speed? 16 
wrap mode (1), ring mode (0)? 1 
frame count (in hex)? 100 

Token Ring diagnostic program

Setting the ring speed
adr = 28, gpip = d0
Resetting TMS380...
Checking result of BUDS...
Initializing TMS380...
SIFINT = 0x0
  BIA is 0000 7C00 01C5
Opening TMS380...
SIFINT = 0x88, ssb.parm0 = 0x8000
Adapter open ok, ssb.parm0 = 8000

Frame Loopback Test:

256          Frames Looped Back

TMS380C16 test 1 passed
TMS380C16 test: 1 passed, 0 failed
```

Error Messages

main: reset failed.	Indicates that the TMS380 failed to reset properly, or the TMS380 never finished its reset. This indicates two possibilities: the ARESET bit could not be cleared, or there were invalid clock inputs. Either situation would cause this failure.
Invalid HW config inputs.	Indicates that one of the hardware-configurable bits in the TMS380's adapter control is improperly set.
main: eagle_load failed.	Indicates that the CPU was unable to download the BUDS to the TMS380.
main: execute failed.	Indicates that after downloading the BUDS, the TMS380 failed to execute.
main: BUDS failed.	Indicates that the TMS380 self-test diagnostics failed.
main: adapter_init failed.	Indicates that the initialization of the adapter failed.
main: adapter_open failed.	Indicates that the TMS380 failed to open onto the token-ring.
main: frame_loopback_test failed.	Indicates that the TMS380 failed to loop back a frame. This error is usually preceded by either the Receive_init failed or Miscompare: messages.
Receive_init failed.	Indicates that the TMS380 failed to initialize a receive buffer.
Miscompare: byte xx: w yy, r zz	Indicates that the transmit packet and the receive packet did not contain the same information. In this case, byte number xx was not the same. The transmit packet contained a yy while the receive packet contained a zz.

TRM

The TRM test is a memory test on the half megabyte of token-ring private memory. The RAM is divided into eight sections and each section must pass two different tests. First, a series of patterns are written into every location in a section and then verified. Next, an address test writes an incrementing pattern into each location in a section which is also verified. Token-ring private memory is only accessible through specified registers located on the TMS380C16 token-ring communications processor. These tests are destructive tests; any code or data stored in token-ring memory is wiped out if the TRM test executes.

Format

Diag > TRM

Example

```
Diag > trm Return
```

```
TRM test
```

```
Setting the ring speed
```

```
adr = 28, gpip = d8
```

```
Resetting TMS380...
```

```
Section 7
```

```
Section 6
```

```
Section 5
```

```
Section 4
```

```
Section 3
```

```
Section 2
```

```
Section 1
```

```
Section 0
```

```
Token ring memory test 1 passed
```

```
Token ring memory test: 1 passed, 0 failed
```

TRM

Console Message

TRM test
Section 7
Section 6
Section 5
Section 4
Section 3
Section 2
Section 1
Section 0
Token ring memory test passed

XBUSER

The bus error test exercises the hardware that detects long 68020 transactions. This hardware should issue a bus error signal (BERR) to the 68020 for any cycle that does not terminate within 15 microseconds.

In normal operation, bus exception processing will display the 68020 register set, the address that was being accessed at the time the bus error occurred, the disassembled instruction that was being executed, and then transfers control to the debugging tool.

Since this test forces a bus-error condition, the normal exception vector must be changed to call a special exception-processing routine. After initializing the vector table, the 68020 attempts to access a nonexistent device by reading a reserved location in the peripheral area of the memory map. Since there is no device to acknowledge the transaction, the monitoring logic should generate a bus exception after 15 microseconds. The test's exception vector calls a routine that handles the bus error as an expected event. It re-initializes the bus error exception vector to the default value, displays a test-passed status, and returns control to the calling program. If the hardware is defective, the expected bus error is not detected and the processor will wait forever for the bus cycle to complete.

The successful completion of this test indicates that the following hardware is functioning properly:

- The processor bus transaction timer
- BERR signal generation and recognition
- 68020 exception processing

Format

Diag > XB [*count*]

Where:

count

Is a decimal number that specifies the number of times the test is to cycle before terminating.

XBUSER

Console Messages

1. The following console message indicates the test has run successfully when called by an automatic test control program:

```
Bus error test          BUS ERROR TEST PASSED
```

2. The following console message indicates the test has run successfully twice when executed from the Diag prompt:

```
Diag > XL 2  
bus error test 1 passed  
bus error test 2 passed
```

Error Message

The following message appears when the bus error did not occur as expected:

```
Bus error test          ***** BUS ERROR TEST FAILED
```

XEPROM

The XEPROM test calculates an EPROM checksum for the EPROM address 0 through address 1FFFF. The purpose of this test is to ensure that no EPROM bits have changed since it was programmed.

This module first probes the EPROM to find the starting and ending EPROM addresses (which are usually 0 through 1FFFF). The EPROM is assumed to begin on a fixed boundary and to be contiguous.

Once this procedure identifies the extent of EPROM, it searches the EPROM address space for a unique pattern. Adjacent to this pattern is the correct checksum and information that identifies the range of EPROM addresses over which the checksum is to be calculated. The calculated checksum will be compared with the correct checksum stored in EPROM. If these checksums do not match, the information in the EPROM has changed and is probably incorrect. This algorithm is repeated until no more unique patterns are found.

Format

Diag > XE *[count]*

Where:

count

Is a decimal number that specifies the number of times the test is to cycle before terminating.

Example

```
Diag > XE 4
EPROM test 1 passed
EPROM test 2 passed
EPROM test 3 passed
EPROM test 4 passed
EPROM test: 4 passed, 0 failed
```


XEPROM

Console Messages

1. The following message appears when the EPROM test has run correctly during one of the automatic testing sequences, such as when the DEQRA is turned on, when it is reset, or when it is initialized:

EPROM test

EPROM TEST PASSED

2. The following message displays if the DEQRA processor encounters a bus error while probing for the EPROM:

***** EPROM PROBE FAILED

3. The following message displays if the checksum calculated by the test does not match the checksum stored in the EPROM:

***** EPROM TEST FAILED

XMFP

The multi-function peripheral device (MFP) provides the USART for the DEQRA's console, a parallel port for status input, and two counters that are used as system timers.

The first phase of this test checks each of the MFP registers by writing them with a test pattern. The registers are then read to determine if they contain the test pattern. The second phase of the test checks the MFP's USART functions with an internal loopback test. After loopback initialization, a character stream is transmitted to the MFP. The looped-back receive characters are compared to the transmitted character stream. The final phase of the test checks the MFP's counters and interrupt control registers. The counter registers are loaded and the counter expiration interrupts are enabled. The counters are started, and the 68020 begins a software timing loop. The MFP should interrupt when the counter expires. The interrupt service routine clears a flag indicating that it has interrupted. The 68020 checks the interrupt flag after the timing loop has expired.

The successful completion of this test indicates that the following hardware is functioning properly:

- MFP registers
- MFP counters, USART, and interrupt interface
- 68020 level-3 interrupt request and acknowledge logic

Format

Diag > XM

XMFP

Console Messages

1. The following console display indicates the test has run successfully when called by an automatic test control program:

Multi-function peripheral testX0123456789:;<=>?@ABCDEFGH IJKLMN...

2. The following console display indicates the test has run successfully when executed from the Diag prompt:

```
Diag > XM
probe for mfp passed
X0123456789:;<=>?@ABCDEFGH IJKLMNOPQRSTUVWXYZ ^_'abcdefghijkl...
MFP test 1 passed
```

Error Messages

1. The following message displays if a bus error occurs while the processor is probing for the MFP:

***** PROBE FOR MFP FAILED

2. The following message indicates that the MFP did not interrupt:

***** MFP TEST FAILED

XRAM and XZRAM

The DEQRA has two DRAM memory systems. The processor bus (CPU RAM) memory system is downloaded with application code that is executed by the 68020. The communications bus (Z-BUS RAM) memory system provides the shared memory with the host system and acts as the message transfer data buffer.

The CPU-bus and Z-BUS memory systems are tested with common test code. Only the address parameters passed to the test are different. The CPU RAM test is executed before the Z-BUS RAM test during the automatic tests. The tests may be executed in any order when entered at the console keyboard. The test consists of an immediate write-read phase, a time delay write-read phase, and an address phase. This test overwrites the current contents of the memory system being tested.

The three phases of testing provide extensive coverage of all of the bit cells in the DRAM devices. The various data patterns and test algorithms force worst-case conditions, including alternating on-and-off charges in adjacent cells of the devices. The read-write cycles test for stuck data bits. The write-an-array, delay a while, then read-the-array test checks for data errors caused by refresh errors or weak devices. The address test checks for improper address operation, that can be masked in the data testing phases, by writing and reading different data patterns at each address.

The successful completion of this test indicates that the following hardware is functioning properly.

- Processor bus memory system
- Z-BUS memory system
- Z-BUS conversion logic and state machines

Format

Diag > XR

or

Diag > XZ

XRAM and XZRAM

Console Messages

1. The following console display indicates the tests ran successfully when called by an automatic test control program:

```
Private RAM test: nMB      FRAM TEST PASSED
Z-BUS RAM test          FRAM TEST PASSED
```

2. The following console displays indicate the CPU-bus RAM or Z-BUS RAM test ran successfully when executed from the Diag prompt:

```
Diag > XR
FRAM probe passed -- executing RAM test
FRAM test 1 passed
```

or

```
Diag > XZ
FRAM probe passed -- executing RAM test
FRAM test 1 passed
```

Error Messages

1. The following message indicates that a bus error occurred while accessing the RAM:

```
***** FRAM PROBE FAILED
```

2. The following message displays specific address and data information for a memory failure:

```
ERROR - ram data test error - address = xxxxxxxx
data expected = xxxx, data in memory = xxxxxxxx
***** FRAM TEST FAILED
```


PEEK and POKE Command Edit Functions

Table A-1 lists the **PEEK** and **POKE** command edit characters and a description of each.

Table A-1 PEEK and POKE Command Edit Functions

Characters	Description
0-9, A-F, and a-f	Standard hexadecimal digits. ODT68 will beep if more than two digits are attempted to be input for a byte field. The maximum number of digits for word and longword fields is four and eight respectively.
<X	Deletes the previously entered hexadecimal digit.
Ctrl/U or Ctrl/X	Ignores any data input, re-displays the location and starts the input process over. This is useful when entering long numbers and you want to start over without using backspaces.
Return	Writes the entered data into the current memory location and exits to the command prompt.
space and line feed	Stores the data field into the current memory location and advances to the next memory location.
hyphen and caret (^)	Writes the entered data into the current memory location and decrements to the previous memory location.
Ctrl/C	Ignores any entered data and exits to the prompt.

The ODT68 discards all other characters, including underscore. If you enter any other character, the ODT68 will beep the console.

Register Names

Table B-1 contains a list of the DEQRA registers, the ODT68 recognized abbreviation for each register, and a description of each register.

Table B-1 DEQRA Registers

Name	Abbreviation	Description
D0 through D7	<i>none</i>	Data registers
A0 through A7	<i>none</i>	Address registers
USP	<i>u</i>	User stack pointer
ISP	<i>is</i>	Interrupt stack pointer
PC	<i>none</i>	Program counter
SR	<i>none</i>	Status register
SFC	<i>sf</i>	Source function code
DFC	<i>df</i>	Destination function code
VBR	<i>v</i>	Vector base register
MSP	<i>ms</i>	Master stack pointer
CACR	<i>ca</i>	Cache control register
CAAR	<i>caa</i>	Cache address register

Sample Power-Up Display

The following is a sample of what appears on the console display during a power-up routine:

```
@abcdefghijklmnopqrstuvwxyz
 ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789

Non-destructive RAM test
Confidence tests init
Initialize system
Bus error test          BUS ERROR TEST PASSED
EPROM test             EPROM TEST PASSED
Multi-function periph testX0123456789;;<=>?@ABCDEFGHIJKLMNO...
abcdefghijklmnopqrstuvwxyz MFP TEST PASSED

Multi-function periph init
Private RAM test: 1 Mb   FRAM TEST PASSED
Z-Bus RAM test          FRAM TEST PASSED
CIO (Z8036) test

CIO test
  - all timers
  - 1 & 2 linked
CIO test passed

Token ring memory test

TRM test

Resetting TMS380...
Section 7
Section 6
Section 5
Section 4
Section 3
Section 2
Section 1
Section 0
Token ring memory test passed
```

```

TMS380C16 test          skipped

Interrupt test
Interrupt Level Test
. . 4 3 2 .
Ok
Interrupt test passed

Self-Test complete
Inhibited confidence tests

    Digital Equipment Corporation
    DEQRA
    EPROM Linked: 07_SEP_1989_11:02

Waiting for host download ...
... ^C or NMI to start debugger.

```

CONFIDENCE TEST RESULTS

```

Test Not Started - NMI bypassed it
| Probe Failed - Bus timeout
| | Test Started - Not Completed
| | | Test Failed
| | | | Test Passed
| | | | |

```

```

-----
Bus Error | | | | *|
EPROM     | | | | *|
MFP (M68901) | | | | *|
Private RAM | | | | *|
Z-Bus RAM  | | | | *|
CIO (Z8036 ) | | | | *|
Token ring RAM | | | | *|
TMS380C16 |*| | | |
Interrupt  | | | | *|
-----

```

```

Motorola 68020 Debugger Version 1.1
0:: >

```

[illegible]

LOOP Command Display

The following is a sample display generated during the execution of the **LOOP** command. To get this display, the command entered from the Conf prompt was **L 2**.

```
Diag > L 2 Return
```

```
Beginning Diagnostic Test Loop : ^C to abort
```

```
Diagnostic Test Loop Pass 1
```

```
bus error test 1 passed
```

```
EPROM test 1 passed
```

```
X0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz...
```

```
MFP test 1 passed
```

```
Private Ram test 1 passed
```

```
Z-bus Ram test 1 passed
```

```
CIO test
```

- all timers
- 1 & 2 linked

```
CIO test 1 passed
```

```
TRM test
```

```
Resetting TMS380...
```

```
Section 7
```

```
Section 6
```

```
Section 5
```

```
Section 4
```

```
Section 3
```

```
Section 2
```

```
Section 1
```

```
Section 0
```

```
TRM test 1 passed
```

```
Coronado diagnostic program v1.0
```

```

Resetting TMS380...
Downloading TMS380...
Checking result of BUDS...
Opening TMS380...

Frame Loopback Test (Wrap Mode):
1000          Frames Looped Back

TMS test 1 passed
Interrupt Level Test
. . 4 3 2 .
Ok

Interrupt Level test 1 passed

Diagnostic Test Loop Pass 2

bus error test 2 passed
EPROM test 2 passed
X0123456789; ;<=>?@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_`abcdefghijklmnop...
MFP test 2 passed
Private Ram test 2 passed
Z-bus Ram test 2 passed
CIO test
    - all timers
    - 1 & 2 linked
CIO test 2 passed
TRM test
Resetting TMS380...
Section 7
Section 6
Section 5
Section 4
Section 3
Section 2
Section 1
Section 0

TRM test 2 passed

Coronado diagnostic program v1.0

Resetting TMS380...
Downloading TMS380...
Checking result of BUDS...
Opening TMS380...

Frame Loopback Test (Wrap Mode):

```

1000 Frames Looped Back

TMS test 2 passed

Interrupt Level Test

. . 4 3 2 .

Ok

Interrupt Level test 2 passed

	PASSED	FAILED
Bus Error	2	0
EPROM	2	0
Private Ram	2	0
Z-Bus Ram	2	0
MFP	2	0
CIO	2	0
Token ring Ram	2	0
TMS380C16	2	0
Int	2	0

Glossary

access control field

A bit in the token indicating to a receiving device that the token is ready to accept information.

ACK

See affirmative acknowledgment.

active monitor

A single station that exists to watch for lost tokens, circulating frames, and circulating priority tokens, to provide clock data and a latency buffer, and to initiate neighbor notification and error recovery procedures.

active monitor present (AMP)

A frame transmitted by a station to designate itself as the active monitor on the token ring. *See also active monitor.*

address

A unique locally administered or IEEE-designated code assigned to each device or workstation connected to a network.

address recognized indicator (ARI)

A bit of the frame status portion of the medium access control (MAC) frame that is set when a station recognizes its own address.

affirmative acknowledgment (ACK)

A reply indicating that the previous transmission block was accepted by the receiver and that the receiver is ready to accept the next block of transmission. The responses are returned in data-link escape (DLE) sequences in binary synchronous communications. Use of ACK0 and ACK1 alternately provides sequential checking control for a series of replies. ACK0 is also an affirmative (ready to receive) reply to an initialization sequence (line bid) in point-to-point operation.

alternate start (ALTSTART) interface

The component that initiates requests for activity on a device that can process two or more I/O requests simultaneously.

AMP

See **active monitor present**.

ARI

See **address recognized indicator**.

AST

See **Asynchronous system trap**.

ASTLM quota

Asynchronous system trap limit quota. A VMS term.

Asynchronous System Trap (AST)

A software-simulated interrupt for a user-defined service routine. ASTs asynchronously notify a user process that a specific event has occurred. An event must be predefined for an AST to interrupt the process, execute the routine, and resume the process at the interrupt point once the routine has completed.

auto vector peripheral

The software-assigned address of the device.

backplane

The area of the computer or other equipment where many different logic and control elements are connected.

BCC

Branch conditionally 68020 instruction. If a specific condition is met, then the program execution continues at a specific location.

beaconing

The token-ring adapters' error indicating function that searches for hard error problems on the token-ring network.

BRA

Branch always 68020 instruction. Program execution continues at a specific location.

BSR

Branch-to-subroutine 68020 instruction. The longword address of the instruction is pushed onto the system stack and then program execution continues at a specific location.

buffer

A routine or storage used to collect data when a difference in data transfer rate or timing of events occurs.

checksum

A mathematical procedure that produces a sum of digits or bits to determine if a message is received correctly without errors.

CIO

See **command I/O**.

circuit cost

An arbitrary positive integer assigned to a circuit by the DECnet network manager. Messages traveling over the network are routed over the path with the smallest total cost.

command I/O (CIO)

A register address used for input and output on the DEQRA.

communications executive block (CXB)

An area of memory that stores information for transmit requests.

confidence tests

The diagnostic tests performed when the DEQRA is reset.

control/status register (CSR)

A register for the control/status of a device or controller. CSR resides in the processor's I/O space.

COST parameter

A means of specifying the circuit cost with the SET CIRCUIT command.

counters

The performance and error statistics for a component (such as, line or circuit counters).

CSR

See control/status register.

CXB

See communications executive block.

data carrier

The selected medium used to transport or carry (communicate) data or information.

data terminal ready (DTR)

A control signal that enters a modem from the data terminal or communications device using the modem. When the signal is set, it informs the modem that the data terminal equipment is ready to transmit and receive data. When the signal is clear, the data terminal equipment is not ready.

DCL foreign command

See Digital Command Language foreign command.

DECnet

The Digital networking software that runs on server and client nodes in both local area and wide area networks. With DECnet, different types of computers with different operating systems can be connected and users can access information and services through a remote computer over the network connections. *See also Transport Control Protocol/Internet Protocol.*

DECnet-VAX

A Digital Phase IV network software product that allows a suitably configured VMS systems to participate as a routing or end node in DECnet computer networks.

DEC TRNcontroller 100 (DEQRA)

The Digital token-ring controller board that uses a dual-bus architecture to provide high-performance, front-end processing in a DECnet-VAX communications environment. This allows Q-bus MicroVAX systems to connect to 4 or 16 Mbps (megabytes per second) token ring networks and act as full-function DECnet Phase IV nodes and PATHWORKS for VMS servers.

DEQRA

See DEC TRNcontroller 100.

device driver

The set of instructions that the computer follows to reformat data for transfer to and from a particular peripheral device.

device supplied vector (DSV)

The device address configured on the board during installation.

diagnostic tests

The internal tests executed by firmware to detect and isolate malfunctions in the DEQRA hardware.

Digital Command Language (DCL) foreign command

A symbol that executes an image whose name is not recognized by the command interpreter as a DCL command, allowing the user to enter just the symbol name. A foreign command allows modification of the symbol terms using parameters.

direct memory access (DMA)

A device that permits I/O transfers directly into and out of CPU memory. This data transfer method is controlled either by the master CPU or by a dedicated DMA controller. With a dedicated DMA controller, information is transferred across the host bus with a minimum of intervention by the master CPU.

DMA

See **direct memory access**.

download mode

A TRDRIVER operating mode that occurs when the DEQRA has accepted the first block of download code or data. In this mode, the TRDRIVER accepts only reset, download write, and start execution I/O requests. *See also normal mode and reset mode.*

DSACK generator state machines

The logic which generates a signal upon the error-free completion of an address access.

DSV

See **device supplied vector**.

DTR

See **data terminal ready**.

EDI

See error detected indicator.

ending delimiter

A character that contains an error detection (E) bit and an intermediate frame (I) bit. The I bit is used to indicate that this is a frame other than the final one of a multiple frame transmission.

error detected Indicator (EDI)

An indicator bit of the medium access control (MAC) frame that is set if a station on the ring detects an frame checking sequence (FCS) error or nondata symbol in the frame.

Ethernet

A local area network that uses coaxial cable as a passive communications medium to interconnect computer systems, terminal server products, and office equipment at a local site. No switching logic or central computer is needed to establish or to control communications. *See also standard Ethernet and ThinWire Ethernet.*

extended interface

An enhanced set of functions added to the queued I/O (QIO) read function to support additional local area network (LAN) protocol stacks.

faceplate

A cabling system component used to join data and voice connectors. The faceplate can be wall-mounted or surface-mounted.

FCI

See frame copied indicator.

FCS

See frame checking sequence.

firmware

The programs kept in semipermanent storage, such as various types of read-only memory.

flag

A character or bit that identifies when a specific condition occurs, such as setting a switch, or the end of a word.

foreign command

See Digital Command Language foreign command.

frame

The standard transmission unit on a network and contains control information and data. Control information consists of delimiters, control characters, and checking characters, which provide addressing, sequencing, flow control, and error control on the respective protocol levels. Frames can be fixed or variable in length. On a token-ring network, a frame is created when a token has data appended to it.

frame checking sequence (FCS)

A 16-bit error check polynomial that monitors if the bit content of a frame is the same before and after transmission.

frame copied indicator (FCI)

A bit of the frame status portion of the medium access control (MAC) frame that is set when a station copies the frame from the ring.

full duplex mode

A TRDRIVER operating mode in which the TRDRIVER implements two queues of waiting I/O requests: one for read requests and one for write requests. *See also simplex mode.*

handshaking

The exchange of identifying or alerting signals between two data communication devices prior to any transfer of information.

hard error

A network error condition requiring that the source of the error be removed or that the network be reconfigured before reliable operation can resume.

hardware handshake

A set of questions and answers exchanged between pieces of hardware, usually for the purpose of indicating readiness to exchange information, data, or status. *See also interlocked handshake and strobed handshake.*

host computer

A node providing service for another node. A host computer also can be the controlling computer in a network.

IEEE

See Institute of Electrical and Electronics Engineers.

Inhibit diagnostics pattern

A unique string written into random access memory (RAM) after the basic powerup diagnostics complete so that the short reset diagnostics, instead of the longer basic powerup diagnostics, run when the chip is reset.

Input/output (I/O)

The data involved in an input process, an output process, or both, at the same time or separately, or that which relates to a functional unit or channel involved in such a process.

Installation Verification Procedure (IVP)

A procedure to verify that both the LAT-11 server and local area transport (LAT) node systems are working properly.

Institute of Electronic and Electrical Engineers (IEEE)

A leading United States group involved in facilitating information exchange and in coordinating, developing, and publishing standards. IEEE 488 is a popular standard for real-time data collection. IEEE 802 is the standard for various types of local area networks. 802.5 discusses the Ethernet and token access methods, and 802.2 deals with the Logical Link Control.

Interface

The boundary between two functional units, given by functional, common physical interconnection, and signal characteristics. Other characteristics are used if needed.

Interlocked handshake

A hardware handshake which is carried out as a series of step-by-step questions and answers. Each question must be answered before the next question can be asked; all questions must be answered for the handshake to be completed. *See also hardware handshake and strobed handshake.*

International Organization for Standardization (ISO)

A Geneva-based international agency that is responsible for developing international standards for information exchange and coordinating the work of the national standards bureaus. Primarily known for the seven-layer OSI reference model.

International Organization for Standardization (ISO) layered model

A communications protocol strategy which that views communication protocols as existing in seven layers, with each layer performing services for the layer above it. The seven layers (from lowest to highest) are: physical, data link, network, transport, session, presentation, and application.

Internet

A group of networks that includes regional and local networks at universities and commercial institutions. *See also* **DECnet and Transport Control Protocol/Internet Protocol**.

Internet Protocol (IP)

A connectionless technology where information is transferred in data units. Internet Protocol (IP) provides for the transmission of blocks of data (datagrams) between hosts identified by fixed length addresses and for the fragmentation and reassembly of long datagrams passed through a small packet network.

I/O

See **input/output**.

IOSB

See **I/O status block**.

I/O status block (IOSB)

A two-word array associated with a queued I/O (QIO) request. in which a code is returned on completion of the requested I/O operation. The QIO request system service, upon completion of the requested I/O operation, optionally returns a status code, the number of bytes transferred, and the device- and function-dependent information in an IOSB. An IOSB is not returned from the service call, but filled in when the I/O request completes.

IP

See **Internet Protocol**.

ISO

See **International Organization for Standardization**.

IVP

See **Installation Verification Procedure**.

jumper JP1

The DEQRA component that determines whether the DEQRA's shared memory is visible to the host at powerup.

LAD

See local area disk.

LAN

See local area network.

LAST

See Local Area System Transport.

LAT

See Local Area Transport.

LAVC

See Local Area VAXcluster.

layer

The group of related functions that constitute one level of hierarchy in open systems architecture. Each layer specifies its own role and assumes that lower level functions are provided. One of the seven levels of the Open Systems Interconnection (OSI) reference model.

LLC protocol

See Logical Link Control protocol.

lobe

The cable, possibly made up of several cable segments, linking an attaching device to an access unit in token-ring architecture. Lobe, node, and host refer to the same connection in most situations. *See also multistation access unit.*

lobe cable

An important parameter in IEEE 802.5 network design. A lobe cable, made up of a number of interconnected cables, is the distance from the wire center lobe port to the attached personal computer or host adapter. The lobe cable connects to the end of the adapter cable, usually at a wall plate, and terminates at the patch panel in the wiring closet. The adapter cable connects the lobe cable to the personal computer (host) adapter. Patch cables are used to connect wire center lobe ports to the patch panel.

lobe wire fault

A disruption in the signal path between the attaching device and the access unit.

local area disk (LAD)

The virtual disk software by Digital on a local area network.

local area network (LAN)

A privately owned data communications network that offers high-speed communications channels optimized for connecting information processing equipment. The geographical area of a local area network is usually limited to a section of a building, an entire building, or a group of buildings, such as a campus.

Local Area System Transport (LAST)

The network protocol used by the virtual disk server to send and receive data between two computers. LAST provides local area network services to local area disk (LAD) drives.

Local Area Transport (LAT)

A proprietary Digital architecture for terminal servers on Ethernet networks designed to conserve bandwidth and off-load processing from hosts.

Local Area VAXcluster (LAVC)

A type of VAXcluster configuration in which cluster communications is carried out over the Ethernet by software that emulates certain computer interconnect (IC) port functions. Hierarchical storage controllers are not used.

Logical Link Control protocol (LLC)

The sublayer of the Data Link Layer responsible for presenting a uniform interface to the user of the data link service, usually a network layer.

longword

A 32-bit word (a VAX word). Four contiguous bytes starting on an addressable byte boundary from right to left 0 through 31. The address of the longword is the address of the byte containing bit 0. When interpreted arithmetically, a longword is a two's complement integer with significance increasing from bit 0 to bit 230. When interpreted as a signed integer, bit 31 is the sign bit.

loop

A closed unidirectional signal path connecting input/output devices to a network.

MAC

See **medium access control**.

MAU

See **multistation access unit**.

Mbytes/s

Megabytes per second (M = 1,048,578).

medium

The electronic path or mechanism used to ship information between points.

medium access control (MAC)

A sublayer of the Data Link Layer responsible for insuring operation of the ring. The adapter services involved include scheduling and routing data transmissions and detection of and recovery from error conditions.

MFP

See **multi-function peripheral**.

module

A board, made of plastic, covered with an electrical conductor, on which logic devices (such as transistors, resistors, and memory chips) are mounted, and circuits connecting these devices are etched.

multi-function peripheral (MFP)

A bus device providing a full duplex asynchronous serial port, timers, and individually programmable I/O lines with interrupt capabilities.

multiplex

To simultaneously transmit two or more data streams on a single channel.

multistation access unit (MAU)

The wiring concentrator that attaches a device to the token ring LAN.

NAK

See **negative acknowledgment**.

NCP

See Network Control Program.

negative acknowledgment (NAK)

A BYSYNC protocol data-link character used to indicate that the previous transmission block was in error and the receiver is ready to accept a retransmission of the erroneous block. NAK is also the *not ready* reply to station selection (multipoint) or to an initialization sequence (line bid) in point-to-point operation.

NETBIOS

See Network Basis Input/Output System.

network

A group of interconnected computers or systems that communicate with each other to share resources and information.

Network Basis Input/Output System (NETBIOS)

A session layer interface between personal computer network applications and the underlying protocol software of the Transport and Network Layers; supports name resolution, datagram, and session services. A programming interface to provide an application program with local area network (LAN) communication without involving the data link control (DLC) interface is a part of NETBIOS.

Network Control Program (NCP)

The DECnet command interface used to configure, control, monitor, and test DECnet networks.

network device driver

A set of software or firmware instructions that performs most of the functions that involve direct interface with the operating system. These functions include buffer management, interprocess communication management, and interface resolution between device drivers. The network driver and the network process appear as a single device driver from the user perspective.

NMI switch

The non-maskable interrupt switch used to put the MC68020 processor on the DEQRA into ODT68 debugging mode.

normal mode

A TRDRIVER operating mode in which the TRDRIVER provides a communications channel between processes running in the VMS host system and tasks running on the DEQRA. *See also download mode and reset mode.*

ODT

See online debugging technique.

online debugging technique (ODT)

An interactive program linked to a user program for finding and correcting errors in the program. All addresses and data are communicated in octal notation.

Open Systems Interconnection (OSI)

A seven-layer model developed by the International Organization for Standardization (ISO) that covers all aspects of information exchange between systems and acts as the internationally accepted framework of standards for intersystem communication.

OSI

See Open System Interconnection.

PAL devices

See programmable array logic devices.

patch cable

A length of cable used to connect a product to the building cable or to connect two sections of building cable at a patch panel.

patch panel

A flat panel used to organize numerous cable terminations in order to make communication cable connections easier

PATHWORKS™

A Digital product that enables different client software to operate concurrently on the same personal computer through use of a single NDIS (network device interface specification) compliant Ethernet controller

PCSA

See Personal Computing System Architecture

Personal Computing System Architecture (PCSA™)

An extension of Digital Equipment Corporation's systems and networking architecture that merges the VAX and personal computer environments.

programmable array logic (PAL) devices

The components that contain control signals for arbitration, address multiplexing, and read/write control for memory.

protocol

An organized collection of meanings and usage rules that govern how communication is obtained by the functional units. For example, DECnet and Transport Control Protocol/Internet Protocol (TCP/IP) are network protocols.

pseudo driver

A logical entity treated as a I/O device by the user or the system. A pseudo driver can be redirected by an operator or within an application program to another physical device unit, but is not itself actually any particular physical device.

Q-bus

The peripheral bus used on MicroVAX and PDP computers.

Q-bus address switchpack

The eleven individual switches on the DEQRA's control/status register (CSR) switchpack determine the address of the board's CSR.

QIO

See **queued input/output**.

queued input/output (QIO)

The component that provides a set of interface calls for input and output over the DEC TRNcontroller 100.

queued input/output (QIO) interface

A VMS system service that prepares an I/O request for processing by the driver and performs device-independent preprocessing of the I/O request.

quota

The total amount of a system resource, such as CPU time, that a job is allowed to use in an accounting period, as specified by the system manager in the user authorization file.

Shared memory switchpack

The three shared memory switches determine one-half megabyte increments of shared Q-bus memory space.

simplex mode

An operating mode of the TRDRIVER in which the device is busy and other subsequent requests must wait in a queue until the TRDRIVER completes the current operation. *See also full-duplex mode.*

SMP

See standby monitor present.

soft error

A reoccurring network error necessitating several data transmissions for data to be received. A soft error affects the network's performance but only affects the network's overall reliability when the number becomes excessive.

speed switch (SPSW)

The bit of the multi-function peripheral (MFP) General Purpose Port register that indicates the ring speed.

SPSW

See speed switch.

stack pointer

The general register 14 (R14). The stack pointer contains the address of the top (lowest address) of the processor-defined stack. Reference to the stack pointer will access one of the five possible stack pointers, kernal, executive, user, or interrupt, depending on the value in the current mode and interrupt stack bits in the processor status longword.

standard Ethernet

A local area network that uses coaxial cable as a passive communications medium. Standard Ethernet is recommended for communications between floors and buildings. *See also Ethernet and ThinWire Ethernet.*

standby monitor

A station that can designate itself as the active monitor if the current active monitor is lost.

standby monitor present (SMP)

A frame transmitted by a station to designate itself as the standby monitor.

starting delimiter

A unique 8-bit pattern used to start each frame.

state

The total bit configuration; the functions that are currently valid for a given network component. States include line, circuit, local node, module, data terminal equipment (DTE), and logging.

state machines

A system where the output state at any time depends on the present input and the internal state.

strobe

The selection of a desired point or position in a recurring event or phenomenon, as in a wave, or a device used to make the selection or identification of the selected point.

strobed handshake

A hardware handshake which is qualified by a specific indication called a strobe. A strobe tells the hardware to perform the task now. Handshake information must be present and waiting for the strobe to appear and issue commands to the hardware. *See also hardware handshake and interlocked handshake.*

substate

An intermediate circuit state that is displayed for a circuit state display when the Network Control Program (NCP) commands SHOW or LIST are entered.

symbol

A name that represents a character value, a numeric value, or a logical value.

SYSGEN utility

A system management tool used to tailor a system for a specific hardware and software configuration.

TCP

See Transport Control Protocol.

TCP/IP

See Transport Control Protocol/Internet Protocol.

ThinWire Ethernet

A local area network that uses coaxial cable as a passive communications medium; recommended for communications between workstations, personal computers, and low-end systems in local work areas on a floor. *See also Ethernet and Standard Ethernet.*

throughput

A measure of the maximum quantity of useful information that a device or system can process or transfer in a given time.

TMS test

A ring test for the TMS380C16 chip, including diagnostics for the console to board mechanisms.

token

A sequence of bits, including a starting delimiter, an access control field, and an end delimiter, transferred from one device to another on the token-ring network. Possession of the token gives permission to transmit over the network. Data is transmitted by being appended to a token, turning the token into a frame.

token ring

A ring network that allows unidirectional data transmission between data stations by a token-passing procedure over one transmission medium so that the transmitted data returns to and is removed by the transmitting station. The Digital Token Ring product consists of the DEC TRNcontroller 100 (DEQRA); device driver software; and firmware containing diagnostics, installation, and IVP (Installation Verification Procedure). *See also DECnet and TCP/IP.*

Token ring lobe loopback test

A loopback test to check the operation of the token ring interface circuitry, the software, and the lobe cable.

Token Ring Network Device Driver for VMS (TRDRIVER)

The Digital token ring network device driver software that provides a communications channel from application software in the VAX system to the DEQRA. The TRDRIVER follows the same general sequence of programming operations found in all VAX device drivers and performs a basic set of functions.

Token Ring Optimized Liner Interface (TROLI)

The analog components that interface to the token ring.

Token Ring private memory

The memory reserved exclusively for the TMS380C16 chip to use to assemble a frame before transmission and to receive a frame after transmission. Access can only be gained through the chip's interface.

Token Ring RAM test

A test of the private memory of the TMS380C16 chip.

Transport Control Protocol (TCP)

The transport layer, connection-oriented, end-to-end protocol providing reliable, sequenced, and non-duplicated delivery of bytes to a remote or local user; provides reliable byte stream communication between pairs of processes in hosts attached to interconnected networks.

Transport Control Protocol/Internet Protocol (TCP/IP)

A connection-oriented, end-to-end protocol that provides reliable byte stream communications between pairs of processes in hosts attached to dissimilar, interconnected networks. An alternative to DECnet transport protocols. *See also DECnet.*

TRDRIVER

See Token Ring Network Device Driver for VMS.

TROLI

See Token Ring Optimized Line Interface.

UCB

See unit control block.

unit control block (UCB)

A structure in the I/O database that describes the current activity on and characteristics of a device unit; that which holds the fork block of the unit's device driver fork process; and that which provides a dynamic storage area for the device driver.

Vector address switchpack

The seven vector address switches select the 7xx vector address area or the 3xx vector address area.

Z-bus master

A bit of the multi-function peripheral (MFP) General Purpose Port register that indicates if the Z-bus is in use.

ZMSTR

See Z-bus master.

A

- Application environment, 2-4
 - communications traffic, 2-5
 - host performance, 2-5
- Arbitration state machine, 3-41
- Architecture, 3-1
 - concurrency, 3-5
 - dual-bus, 3-3
 - host interface, 3-5, 3-25
- ARESET bit, 8-12
- ASCII, 6-9
- ASCII command, 6-3
- Assembly language, 5-2, 6-7, 6-20
- Automatic tests, 7-3, 8-4, 8-6, 8-7
- AUX command, 6-4

B

- Basic tests, 8-7
- BERR signal, 8-15
- BIA command, 7-2
- BKPT instruction, 7-4
- Board configuration register, 3-17
- Board layout, 3-2
- Breakpoints, 5-1, 5-2, 6-5, 6-11, 6-19, 7-4
- BREAKPT command, 6-5
- BUDS software, 8-10, 8-12
- Burned-in address, 7-2
- Bus error, 3-42
- Bus exception controller, 3-42

C

- Carry flag, 6-16
- CIO command, 8-3
- Clock generation, 3-37
- Clock inputs, 8-12
- Command and status register, 3-27
- Commands
 - ASCII, 6-3
 - AUX, 6-4
 - BIA, 7-2
 - BREAKPT, 6-5
 - CIO, 8-3
 - DIAGS, 6-21
 - DISA, 6-7
 - DUMP, 6-9
 - ENABLE, 7-3
 - EXIT, 5-6, 6-4, 6-21
 - FILL, 6-10
 - GOTO, 6-11
 - HELP, 5-7
 - INHIBIT, 7-4
 - INT, 8-5
 - LOOP, 8-7
 - MEMTEST, 8-8
 - OFFSET, 6-12
 - PEEK, 6-13
 - POKE, 6-14
 - RECALL, 5-8
 - REGS, 6-15
 - SD, 7-5
 - SEARCH, 6-18
 - TMS, 8-10
 - TRACE, 6-19

Commands (cont'd)

- TRM**, 8-13
- TTB**, 6-20
- XBUSER**, 8-15
- XEPROM**, 8-17
- XMFP**, 8-19
- XRAM**, 8-21
- XZRAM**, 8-21

Communications bus

- description**, 3-19
- operation**, 3-19
- shared memory**, 3-21
- shared memory Q-bus support**, 3-24
- shared memory arbitration**, 3-23
- shared memory data organization**, 3-22
- shared memory refresh**, 3-24
- timers**, 3-25
- token ring communications processor**, 3-21

Communications traffic, 2-5

Concurrency, 3-5

Connectors

- console**, 4-2
- token ring**, 4-1

Console

- cable**, 3-14, 4-3, 5-2
- using**, 3-13

Console port

- connector**, 4-2

Conventions

- document**, xi

CPU-bus, 8-21

Ctrl/C, 8-7

D

D-connector, 5-2

Delayed Z-BUS Transaction, 3-43

DEQRA

- applications**, 1-2
- architecture**, 2-1, 2-2, 3-1
- block diagram**, 3-4
- bus error**, 3-42
- clock generation**, 3-37
- communications bus**, 3-19
- features**, 1-2

DEQRA (cont'd)

- general operation**, 2-2
- host interface**, 3-25
- interrupt levels**, 3-35
- interrupts**, 3-34
- interrupt structure**, 3-34
- memory map**, 3-5
- non-maskable interrupt**, 3-44
- purpose**, 1-1
- reset**, 3-42
- shared memory**, 3-21, 3-22, 3-23, 3-24
- timers**, 3-25
- timing**, 3-36
- token ring communications processor**, 3-21
- transaction timing**, 3-38
- transfers**, 3-33
- Z-BUS**, 3-19, 3-21, 3-22, 3-23, 3-24, 3-25
- Z-BUS arbitration**, 3-39

Diagnostics, 3-12

DIAGS command, 5-2, 6-21

DISA command, 6-7

Disassembler, 6-7

Display > memory, 6-9

Display > registers, 6-15

Document conventions, xi

Downloader, 3-12

DRAM devices, 8-21

DRAM memory systems, 8-21

Dual-bus architecture, 3-3

DUMP command, 6-9

E

ENABLE command, 7-3

EPROM, 8-17

- diagnostics**, 3-12

- downloader**, 3-12

- ODT68 debugging tool**, 3-13

EPROM checksum, 8-17, 8-18

EXC, 8-5

EXIT command, 5-4, 5-6, 6-4, 6-21

F

FILL command, 6-10

G

General operation, 2-2

 normal, 2-3

 reset, 2-3

 start-up, 2-3

General purpose port, 3-15

GOTO command, 6-11

H

Hardware Reset, 2-3

HELP command, 5-4, 5-7

Host driver, 5-2

Host interface, 3-5, 3-25

 command and status register, 3-27

 Q-bus interface devices, 3-29

 Q-bus support logic, 3-29

Host performance, 2-5

Host-to-ring transfer, 3-33

I

IEEE, 7-2

ILLEGAL instruction, 5-2

INHIBIT command, 7-4

INT command, 8-5

Interrupt counter flag, 8-3

Interrupt flag, 8-19

Interrupt levels, 3-35

Interrupt priorities, 8-5

Interrupt priority, 3-36

Interrupts, 3-34

Interrupt structure, 3-34

L

LED register, 3-17

LOOP command, 8-7

M

Memory map, 3-5

 non volatile memory space, 3-7

 peripheral space, 3-8

 processor bus, 3-6

 volatile memory space, 3-8

 Z-BUS, 3-7

 Z-BUS space, 3-8

MEMTEST command, 8-8

MFP device, 8-19

Microprocessor, 3-9

Modify> memory, 6-10, 6-14

Modify > memory, 6-13

Modify > registers, 6-15

Multifunction peripheral

 console, 3-13

 general purpose port, 3-15

 timers, 3-14

N

Negative flag, 6-16

NMI

 See Non-maskable interrupt

Non-maskable interrupt

 priority, 8-5

 request, 3-44, 5-2

 switch, 3-44, 5-2

NonVolatile memory space, 3-7

Normal operation, 2-3

O

ODT68

 debugging tool, 3-13

 line editor, 5-3

 required equipment, 5-2

OFFSET command, 6-12

P

PEEK command, 6-13

Peripheral memory space, 3-8

POKE command, 6-14

Processor bus

description, 3-9

EPROM, 3-12

memory, 3-12

memory map, 3-6

microprocessor, 3-9

multifunction peripheral, 3-13

operation, 3-9

registers, 3-16

signals, 3-9

Program Counter, 6-7, 6-11, 6-15, 6-19,
B-1

Prompt>

Aux, 7-1

Aux, 5-2

Daig, 5-2

Diag, 8-2

program counter, 5-1

Q

Q-bus

interface devices, 3-29

support logic, 3-29

R

RECALL command, 5-8

Registers, 6-15, 6-17, B-1

REGS command, 6-15

Reset controller, 3-42

S

SD command, 7-5

SEARCH command, 6-18

Shared memory, 3-21

arbitration, 3-23

data organization, 3-22

Shared memory (cont'd)

Q-bus support, 3-24

refresh, 3-24

visibility to host, 3-30

Shared memory Q-bus support, 3-24

Shared memory arbitration, 3-23

Shared memory data organization, 3-22

Shared memory refresh, 3-24

Start-up, 2-3

Status flag, 8-5

T

Terminal, 5-2

Timers, 3-14, 3-25

Timing, 3-36

clock generation, 3-37

transaction timing, 3-38

TMS380C16

bus timing circuits, 3-33

description, 3-32

processor, 3-21

register addresses, 3-32

reset operation, 3-32

TMS command, 8-10

token ring memory, 3-33

TRM command, 8-13

TMS command, 8-10

Token ring

bus timing circuits, 3-33

communications processor, 3-32

interface module, 3-33

memory, 3-33

Token ring communications processor, 3-21,
3-32, 3-33

Token ring interface, 3-30

bus timing circuits, 3-33

interface module, 3-33

TMS380C16, 3-32

TMS380C16 reset operation, 3-32

Token ring port connector, 4-1

TRACE command, 6-19

Transaction timing, 3-38

TRM command, 8-13

TTB command, 6-20

U

USART, 8-19

V

Volatile memory space, 3-8

X

XBUSER command, 8-15

XEPROM command, 8-17

XMFP command, 8-19

XRAM command, 8-21

XZRAM command, 8-21

Z

Z-BUS

arbitration, 3-39

hardware test, 8-3

memory map, 3-7

memory space, 3-8

memory test, 8-21

operation, 3-19

shared memory, 3-21

shared memory Q-bus support, 3-24

shared memory arbitration, 3-23

shared memory data organization, 3-22

shared memory refresh, 3-24

signals, 3-19

timers, 3-25

**token ring communications processor,
 3-21**

Z-BUS arbitration

arbitration state machine, 3-41

Z-conversion state machine, 3-41

Z-BUS memory space

memory section, 3-8

peripheral device section, 3-8

reserved section, 3-8

Z-conversion state machine, 3-41