
DRB32

EK-DRB32-TM-003

DRB32 Technical Manual

digital equipment corporation ● maynard, massachusetts

January 1989
July 1990

The DRB32 is a high-speed parallel port sold by Digital Equipment Corporation for use in qualified applications. A brief special agreement is required for purchase of the product. Contact your Digital sales representative for more information.

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.

Digital Equipment Corporation makes no representation that the interconnection of its products in the manner described herein will not infringe on existing or future patent rights, nor do the descriptions contained herein imply the granting of license to make, use, or sell equipment constructed in accordance with this description.

Book production by Educational Services Media Communications Group.

The following are trademarks of Digital Equipment Corporation:

DEC	PDP	VAXcluster
DECnet	ULTRIX	VMS
DECUS	UNIBUS	digital
DRB32	VAX	
DR11-W	VAXBI	

CONTENTS

FIGURES

TABLES

PREFACE

PURPOSE OF THIS MANUAL

The *DRB32 Technical Manual* (EK-DRB32-TM) describes the hardware components of the DRB32 adapter. This manual includes theory of operation, functional descriptions of components, VAXBI transactions for the DRB32, addressing and register descriptions, user interface signal descriptions, pin assignments, electrical and environmental specifications, and timing diagrams.

INTENDED AUDIENCE

This manual is for:

- Digital or customer personnel who install and repair this equipment in the field
- Customer engineers and programmers who incorporate this equipment into their own product or system.

The manual presumes familiarity with the VAXBI bus, VAX processors, and the VMS operating system.

STRUCTURE OF THIS MANUAL

Chapter 1 introduces the DRB32, describes its three subsystems, and illustrates typical DRB32 configurations.

Chapter 2 describes the theory of operation of the DRB32 adapter, including a block diagram of the basic module (T1022), functional descriptions and VAXBI transactions.

Chapter 3 describes the address space and registers of the DRB32, including BIIC registers and DRB32 module registers.

Chapter 4 describes the user interface signals and protocol of the DRB32 data path.

Appendix A includes electrical and environmental specifications, pin assignments, and timing diagrams.

Appendix B describes the DRB32-E module (T1024) for the DRB32-E option.

Appendix C describes both DRB32-W options: the T1023-00 and T1023-YA.

Appendix D describes the usage of the BCAI Access Register.

Appendix E is a glossary of DRB32-related terms used in this documentation set.

RELATED MANUALS

This is one of a family of processors, memories, and adapters that uses the 32-bit VAXBI bus. For a technical summary of all VAXBI modules, system components, and integrated circuits, see the *VAXBI Options Handbook*, EB-27271-46.

Other related technical manuals are:

DRB32 Introduction
Order No. EK-DRB32-OV

DRB32 Programmer's Manual
Order No. AA-HZ25C-TE

DRB32 Hardware Installation Guide
Order No. EK-DRB32-IN

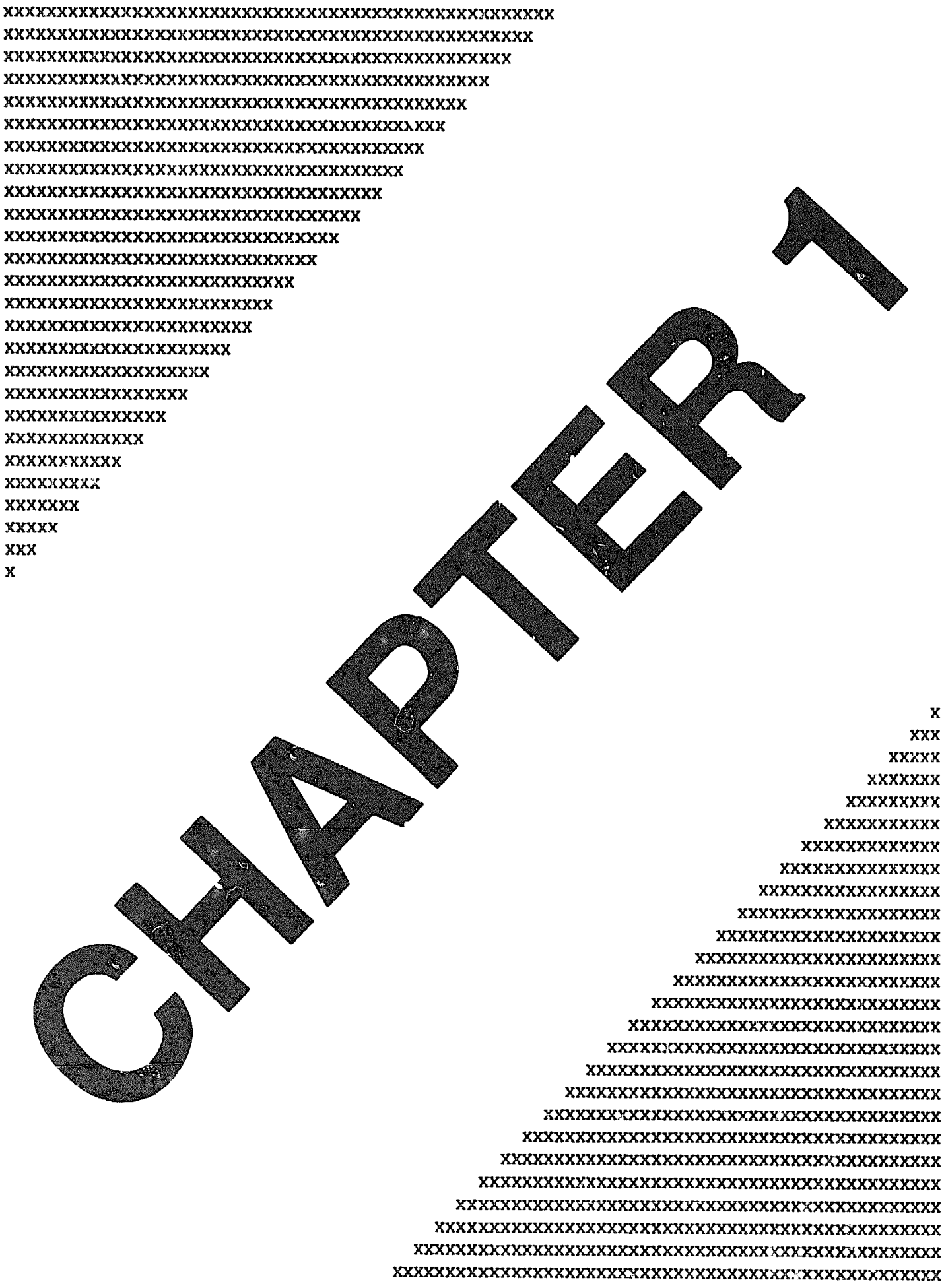
DR11-W User's Guide
Order No. ED-DR11W-UG-003

For further information on VAX/VMS, see the *VAX/VMS Documentation Set*.

CONVENTIONS USED

In this manual, the following conventions are used:

- The DRB32 adapter is referred to as the DRB32.
- The VAXBI bus is referred to as the VAXBI.


CHAPTER 1

CHAPTER 1
INTRODUCTION

The DRB32 is a high-performance gateway to the VAXBI. The adapter provides a half-duplex asynchronous interface between Digital's VAXBI bus and user-designed devices such as signal processors, FFT machines, array processors, high-speed A/D converters, high-speed instrumentation, and other specialized devices.

1.1 PRODUCT DESCRIPTION

The DRB32 is a VAXBI adapter that provides fast data transfers in longwords, words, or bytes, and connects external user devices to VAXBI systems. The DRB32 performs fast parallel I/O DMA transfers of up to 982,528 bytes. The hardware allows DMA transfers larger than 982,528 bytes with a double buffering capability. The DRB32 adapter can also be used in programmed I/O (data) mode, which requires a processor to intervene on each READ or WRITE cycle.

The DRB32 transfers data to the user device through a 32-bit data path and a pair of 8-bit control paths. Those user devices that do not accept the DRB32 parallel port I/O protocol require a DRB32 extension module, which converts the DRB32 I/O port protocol to a non-DRB32 protocol.

The DRB32 uses odd parity on its data path. Figure 1-1 shows the DRB32 adapter (T1022).

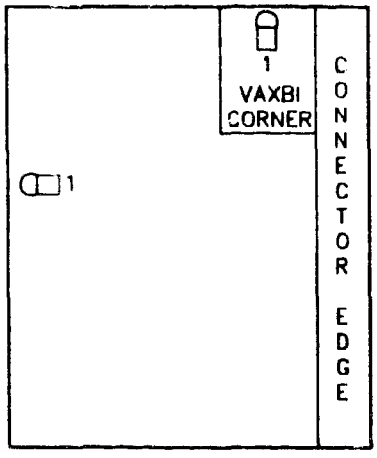


Figure 1-1 DRB32 Adapter (T1022)

1.2 DRB32 HARDWARE COMPONENTS

The DRB32 hardware has three subsystems:

- VAXBI interface
- 32-bit parallel I/O port
- T-11 subsystem

The DRB32 data path consists of the 32-bit parallel I/O port and the VAXBI interface. The T-11 is not part of the direct data transfer path. These components are shown in Figure 1-2.

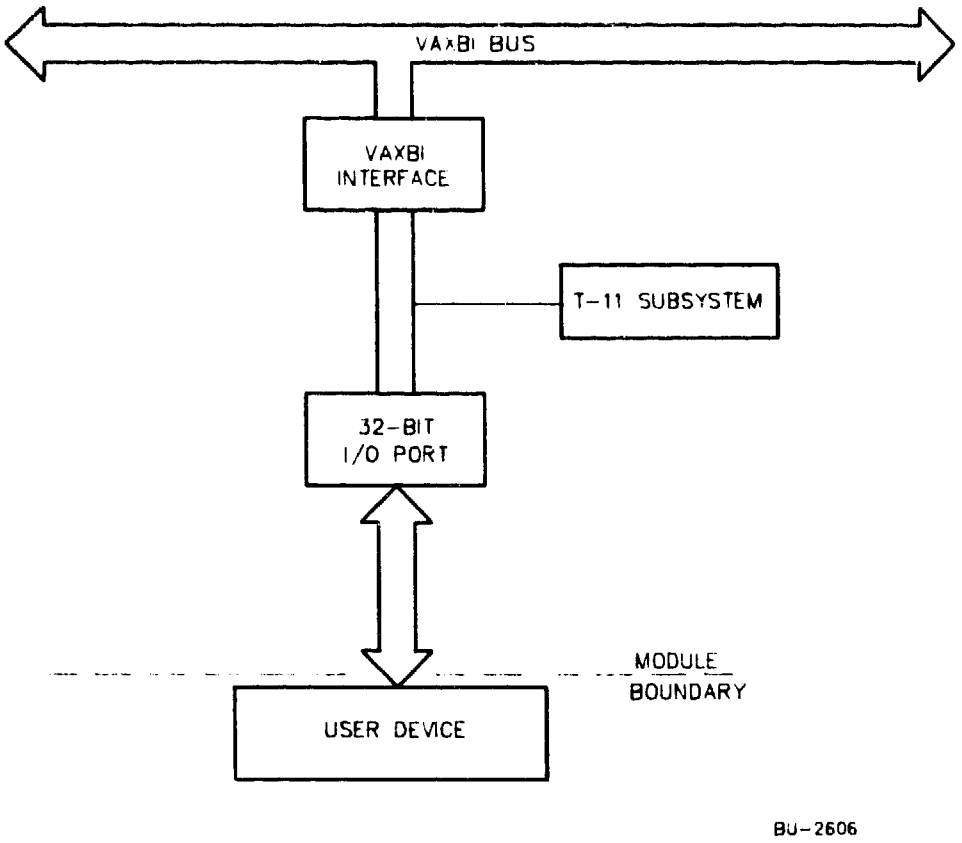


Figure 1-2 DRB32 Simplified Block Diagram

A VAXBI backplane cable connects the DRB32 I/O port to a device within the same system enclosure. If an extension module is needed, it is connected to the DRB32 I/O port by another VAXBI backplane cable. The extension module I/O port is connected to a connector panel, or another device in the same enclosure.

See Chapter 2 for a functional description of the DRB32.

1.3 DRB32 OPTIONS

The DRB32 is available in three options:

- DRB32-M
- DRB32-E
- DRB32-W
  - T1023-00
  - T1023-YA

These options are described in the following sections.

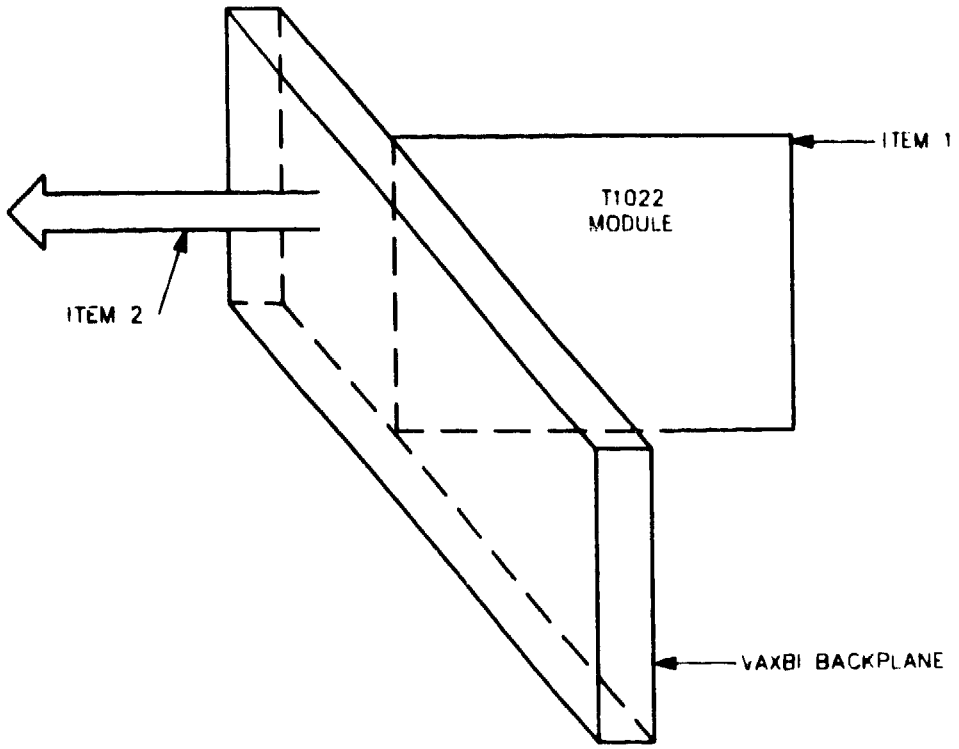
1.3.1 DRB32 Main Option (DRB32-M)

The DRB32-M is the basic DRB32 option and is designed for applications in which all cabling is kept within a single cabinet. Some examples are:

- Mount your equipment within a DEC STD FCC-compliant cabinet along with the BA32 box containing the DRB32.
- Connect the processor system cabinet to your equipment cabinet to form an FCC-compliant envelope.
- Design your own OEM box that can contain the DRB32 (your cabinet must also contain the VAXBI, as well as your own equipment).

The DRB32-M is the basic intracabinet DRB32 option. This option consists of the primary DRB module (T1022). A set of cables (CK-DRB32-L*) available in various lengths connects the DRB32 to the customer equipment within the same EMI/RFI tight enclosure. Each cable has a 30-pin Berg connector at both ends. The electrical interface consists of FAST (Fairchild Advanced Shottky TTL) logic.

This option is designed to be compliant with FCC regulations when fully contained within an FCC-compliant cabinet. It is not designed to be FCC compliant with the cables exiting from an RFI/EMI shielded cabinet. The DRB32-M option is shown in Figure 1-3.



ORDER NO.	ITEM	QUANTITY	DESCRIPTION
DRB32-M	1	1	BASE MODULE (T1022)
CK-DRB32-L•	2	1 SET (3)	INTERNAL CABLES (UP TO 15 FT)

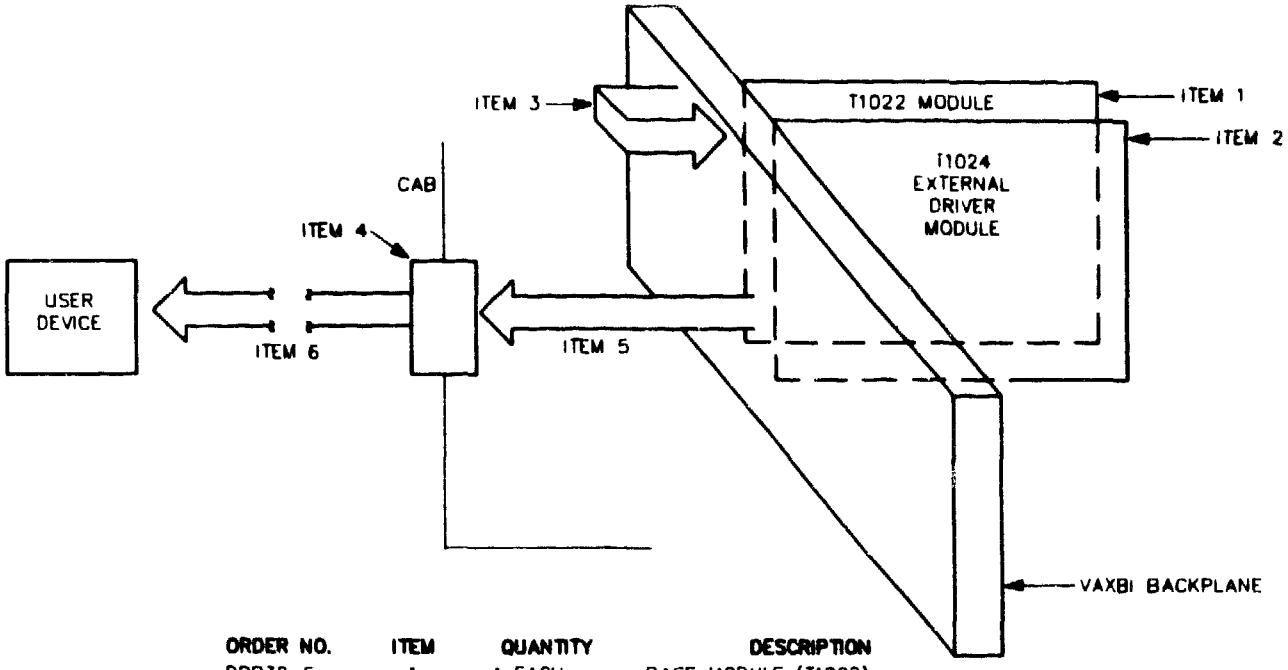
BG-2607

Figure 1-3 DRB32 Main Option (DRB32-M)

1.3.2 DRB32 External Option (DRB32-E)

This option is for situations in which user equipment must be separate from the cabinet containing the DRB32. Therefore, a cable must connect the two cabinets. A compatible bulkhead connector must be provided for the bulkhead of the customer's cabinet.

The DRB32-E external cable driving option is designed to drive an intercabinet cable that is up to 40 feet (12 meters) long. The external cable is designed to be RFI/EMI tight (to comply with FCC requirements). The electrical interface is a combination of differential-drive TTL and controlled-threshold TTL. The DRB32-E option is shown in Figure 1-4.



ORDER NO.	ITEM	QUANTITY	DESCRIPTION
DRB32-E	1	1 EACH	BASE MODULE (T1022)
	2	1 EACH	EXTERNAL DRIVER MODULE (T1024)
17-01092-02	3	1 SET (3)	INTER-MODULE CONNECTOR CABLES
70-23924-01	4	1 EACH	I/O CONNECTOR PANEL
CK-DRB32-L*	5	1 SET (3)	INTERNAL CABLES (UP TO 15 FT)
BS17Y-20 OR BS17Y-40	6	1 SET (3)	EXTERNAL CABLES (20 FT)
	6	1 SET (3)	EXTERNAL CABLES (40 FT)

BU-2608

Figure 1-4 DRB32 External Cable Driving Option (DRB32-E)

1.3.3 DRB32/DR11W Option (DRB32-W)

The DRB32-W option is provided for the customer with equipment currently designed to interface to the DR11-W. This option provides a means of connecting your current equipment with a VAXBI system without requiring a UNIBUS subsystem.

Since the DRB32-M option provides significantly higher performance than the DRB32-W option, redesigning the USER hardware to interface directly with the DRB32-M option would provide the benefits of increased throughput. Table 1-1 shows the data transfer rates for two of the DRB32 options.

NOTE

Please review Section 4.5 OPTIMIZING DATA TRANSFER SPEED, of this manual. This section supplies examples of how data transfer rates are calculated.

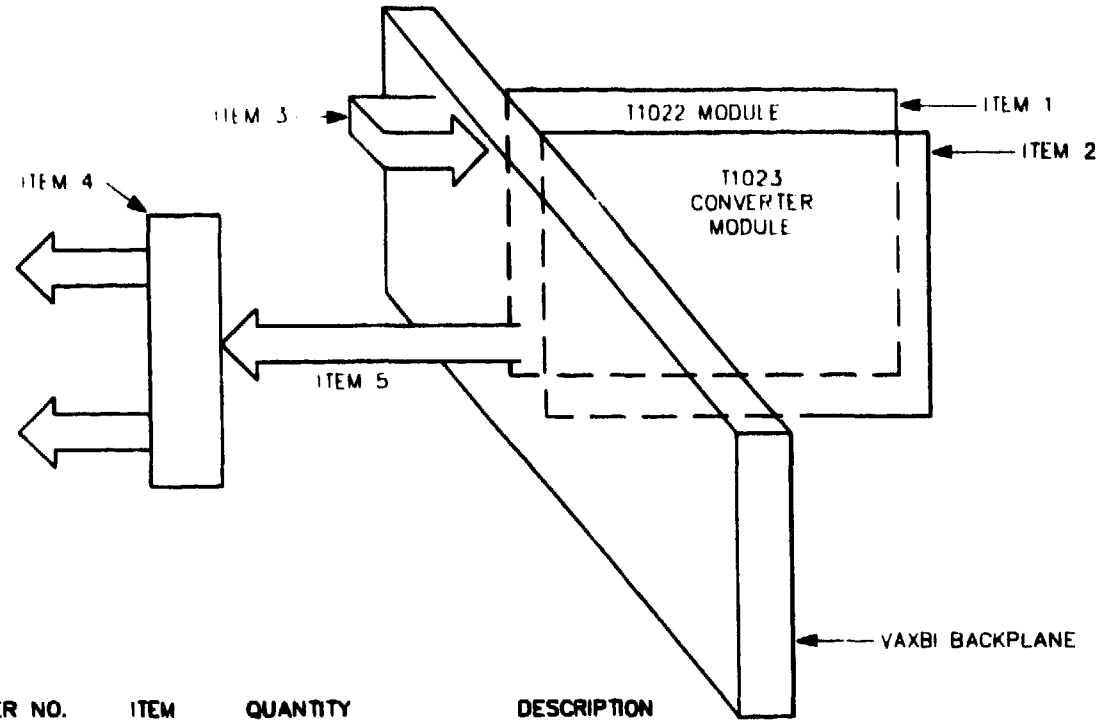
Table 1-1 Data Transfer Rates (MB/Second)

Adapter	READ from 8200,8300 VAXBI Memory	WRITE to 8200, 8300 VAXBI Memory	READ from 8500, 8550, 8700, 8800 Priv. Memory	WRITE to 8500, 8550, 8700, 8800 Priv. Memory
DRB32-M	6.0	6.7	3.3	4.7
DRB32-W	3.1	3.6	3.1	3.6

The DRB32-W option is an intracabinet connector (as is the DRB32-M). There are no specific provisions to make this device FCC compliant when used as an intercabinet connection. The DRB32-W option is shown in Figure 1-5.

NOTE

Two versions of the DRB32-W option exist. For a description of the differences between these two versions, please refer to Appendix C. When we talk about the DRB32-W option in this technical manual, we are talking about both versions unless we specifically mention both versions separately. The two versions of the DRB32-W module are the T1023-00 (produced before July 1990) and T1023-YA (produced after July 1990).



ORDER NO.	ITEM	QUANTITY	DESCRIPTION
DRB32-W	1	1 EA	BASE MODULE (T1022)
	2	1 EA	DR11-W CONVERTER MODULE (T1023)
17-01092-02	3	1 SET (3)	INTERMODULE CONNECTOR CABLES
70-23923-01	4	1 EA	BULKHEAD CONNECTOR
CK-DRB32-L*	5	1 SET (3)	INTERNAL CABLES (UP TO 15 FT)

BU-2609

Figure 1-5 DRB32-W Option

1.4 SOFTWARE

The DRB32 software is included in a VMS-layered product called DRB32VMSDRIVERS and must be ordered separately.

The following software is available for the DRB32-M and DRB32-E:

- VMS device driver (UQDRIVER) for the DRB32-M and DRB32-E
- Subroutine package for communications between two VAXBI systems through a back-to-back DRB32 link (DRB32$MESSAGE)
- Test code for drivers (DRB32$QIO)
- MACRO and object libraries
- Command procedures to load and build the driver and test programs.

The following software is available for the DRB32-W:

- VMS device driver to support DR11-W interfaces (UQWDRIVER)
- Test code for drivers (DRB32$WQIO)
- MACRO and object libraries
- Command procedures to load and build the driver and test programs.

The software and its installation are described in the *DRB32 Programmer's Manual.*

The diagnostics are available and are described in the *DRB32 Hardware Installation Guide.*

1.5 DRB32 CONFIGURATIONS

The DRB32 is available in two configurations:

- Intraenclosure DRB32
- Interenclosure DRB32

In the **intraenclosure** DRB32, both ends of the DRB32 data path and the cable connecting the DRB32 to the user device are completely enclosed in the same RFI-tight, FCC-compliant enclosure. This configuration is presented in the DRB32-M option.

In the **interenclosure** DRB32, the DRB32 adapter and the user device are in two different cabinets, with cable running between them. The length of the cable (a maximum of 40 feet or 12 meters) affects the data transfer rate. This configuration is presented in the DRB32-E option. These configurations are intended for different types of user applications. Two sets of electrical specifications are available for these configurations; see Appendix A.

1.6 STRATEGIES FOR CONNECTING TO THE VAXBI

The DRB32 is offered as the gateway to VAXBI systems. You can use the DRB32 to simplify the task of connecting your equipment to the VAXBI. You can choose from a number of ways to implement this connection. The following strategies range from the easiest implementation to the most time-consuming.

- Use the DRB32 hardware and software as supplied by Digital. Extensive testing of the DRB32 components means that this approach entails less risk.
- Use the DRB32 hardware and modify the supplied software. This could include extensive modification of the supplied device driver.
- Use the DRB32 hardware and write a device driver from scratch.
- Use the DRB32-M hardware and design a customer interface module. This may require modifications to the driver and other software. The DRB32-M supplies the customer with quick access to the VAXBI, no VAXBI license is needed if you use the DRB32-M module to interface with the VAXBI. The customer interface would be simpler to design than a full VAXBI option.

A customer interface module connects your user device to the DRB32 adapter. It fits into a VAXBI backplane and converts the DRB32 output signals and protocol into a different set of signals and protocol. These modules are not "VAXBI nodes" and, therefore, do not have a VAXBI corner, but they do occupy a slot in the VAXBI backplane. These modules draw power from the VAXBI backplane. An example of an interface module is the DR11-W converter module (T1023) that is part of the DRB32-W option.

- Design a new VAXBI option. This requires a VAXBI license and is expensive in development time, but may be a good solution for some specialized applications.

For many applications, the use of the standard DRB32 is the best solution.

1.7 TYPICAL VAXBI/DRB32 SYSTEM CONFIGURATION

A typical DRB32 system configuration, in which the DRB32 adapter is used to connect a user device to the VAXBI and a VAX processor, is shown in Figure 1-6.

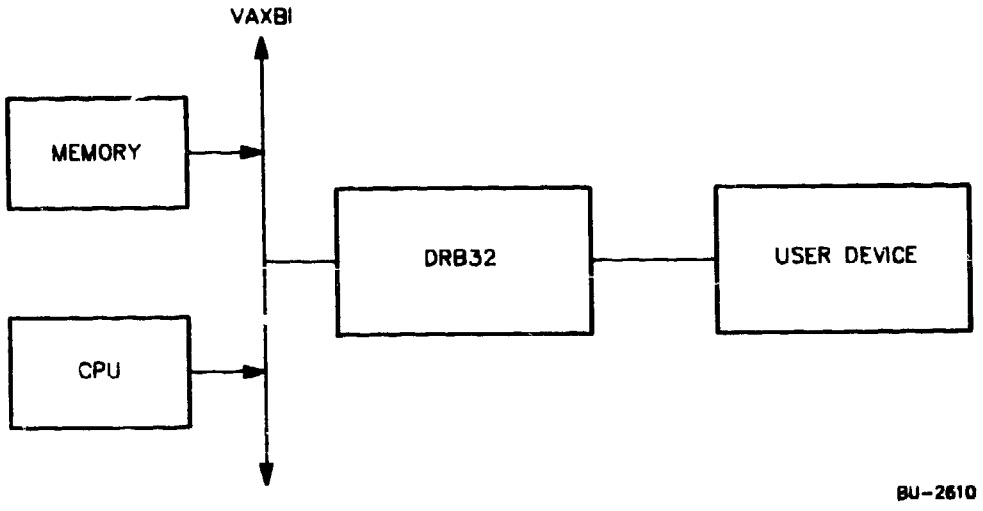


Figure 1-6 Typical DRB32 System Configuration

A DRB32-W system configuration (including the basic DRB32 module and the T1023 DRW-11 converter module) is shown in Figure 1-7.

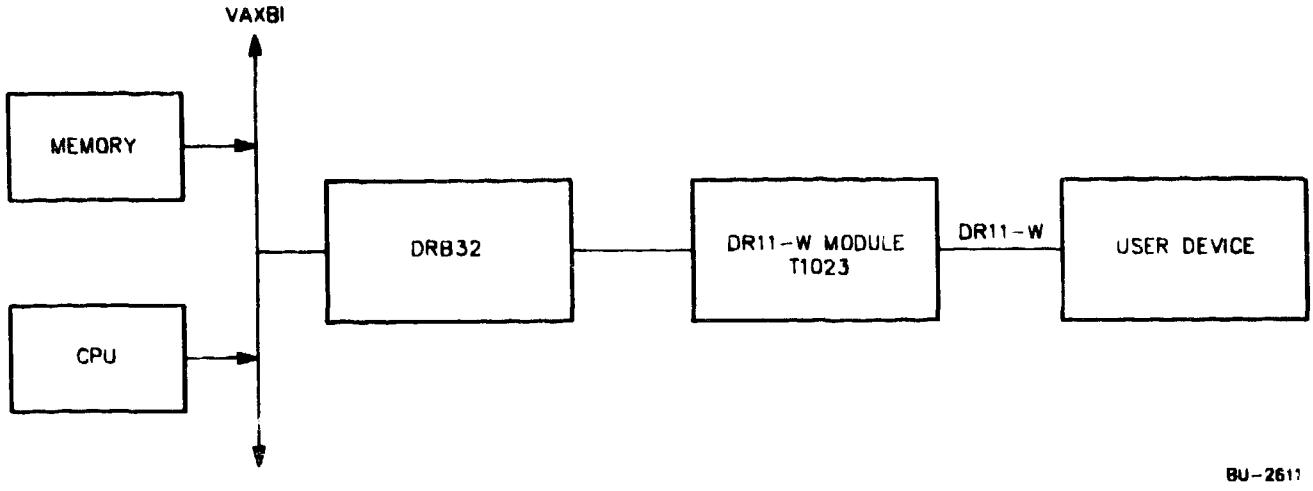


Figure 1-7 Typical DRB32-W Configuration

A DRB32 system configuration with a customer interface module is shown in Figure 1-8.

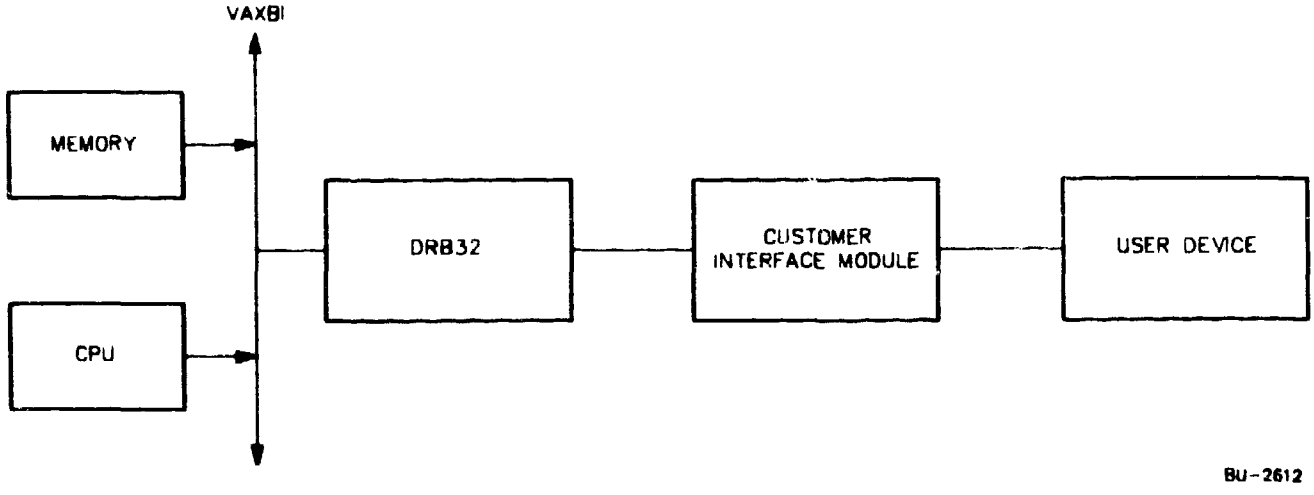
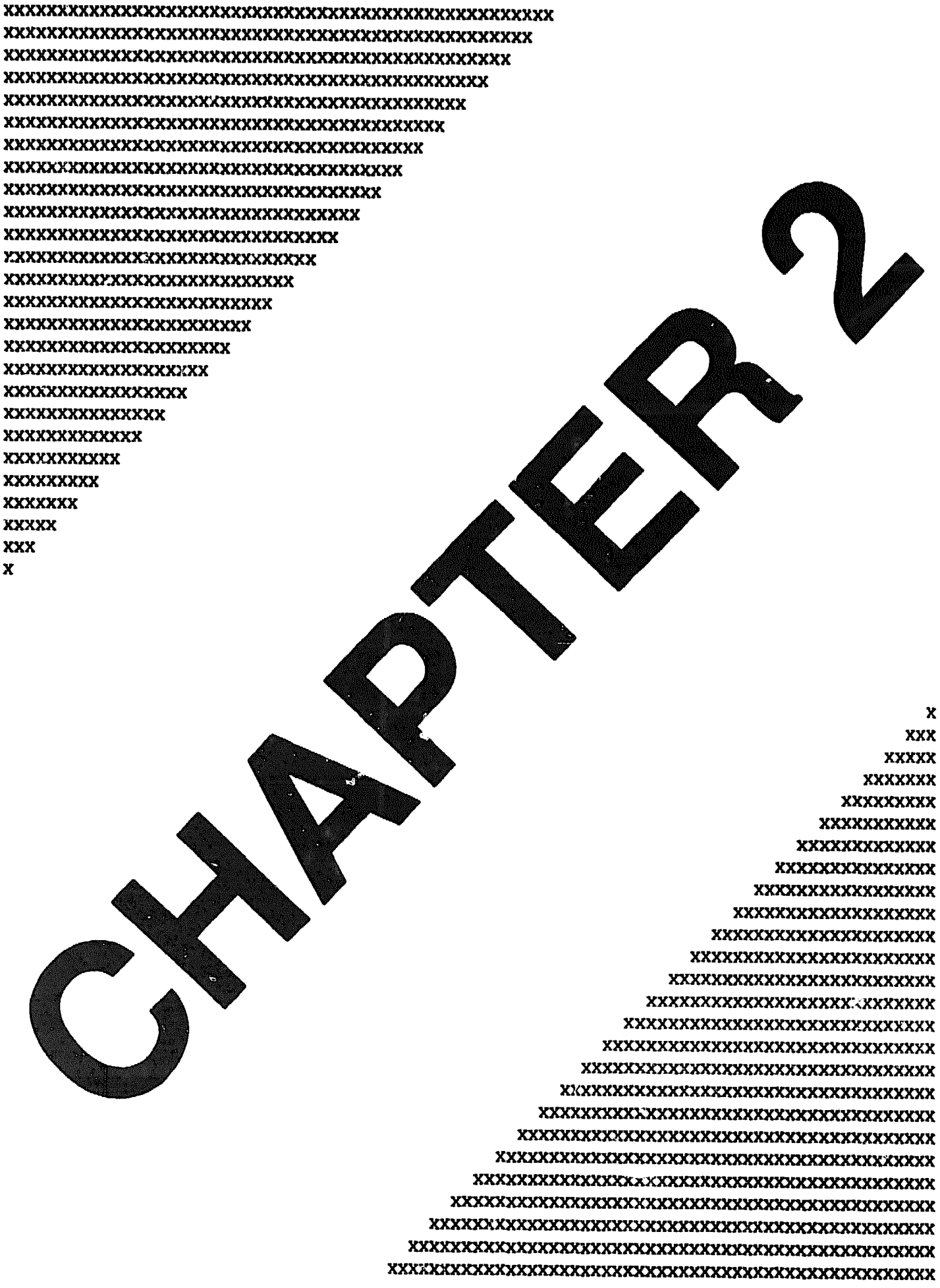


Figure 1-8 DRB32 System Configuration with Interface Module


CHAPTER 2

CHAPTER 2
FUNCTIONAL DESCRIPTION

This chapter provides a functional description of the DRB32 adapter. The following topics are covered in this chapter:

- Block-diagram description of DRB32 module
- VAXBI transactions for the DRB32
- DRB32 address space
- DRB32 power-up sequence.

The DRB32 is an adapter that provides fast data transfers between Digital's VAXBI systems and user devices such as array processors, A/D converters, and high-speed instrumentation. The DRB32 provides a simple connection for your devices to VAXBI systems. Large, fast data transfers are the major advantages of the DRB32.

The DRB32 is optimized for large DMA transfers in Block mode. Block mode allows for transfers up to 982,528 bytes without processor intervention. The DRB32 hardware has double buffering that allows DMA transfers that are larger than this 982,528 bytes limit. Data transfers can also be made in Data mode, in which a host processor on the VAXBI directly reads or writes the DRB32 register that sends and receives data from the user device.

The DRB32 uses map registers to translate virtual addresses to VAXBI physical addresses. The DRB32 map register area is 1919 longwords. Mapped adapters require a process running in a VAX system to load map registers into the adapter for virtual-to-physical address conversion. Transfer widths are selected by the user. The DRB32 can handle longword (32 bit), word (16 bit), or byte (8 bit) transfers. However, the selection of 16- or 8-bit widths results in lower performance. The DRB32 uses odd parity.

2.1 BLOCK-DIAGRAM DESCRIPTION OF THE DRB32 MODULE

The major blocks of the DRB32 are the following:

- Parallel I/O port
- VAXBI interface
- T-11 subsystem.

The DRB32 is a parallel I/O port (with a 32-bit, half-duplex data path and dual, 8-bit duplex control paths) that connects the VAXBI bus and a user device. The DRB32 parallel port has a user interface with signals for data, control, and device status. Attached to the parallel I/O port is a T-11 subsystem that performs module self-test. This section describes the three major parts of the DRB32 and some of the components of those parts. The user interface of the parallel I/O port is described in more detail in Chapter 4 of this manual. A block diagram of the DRB32 is shown in Figure 2-1.

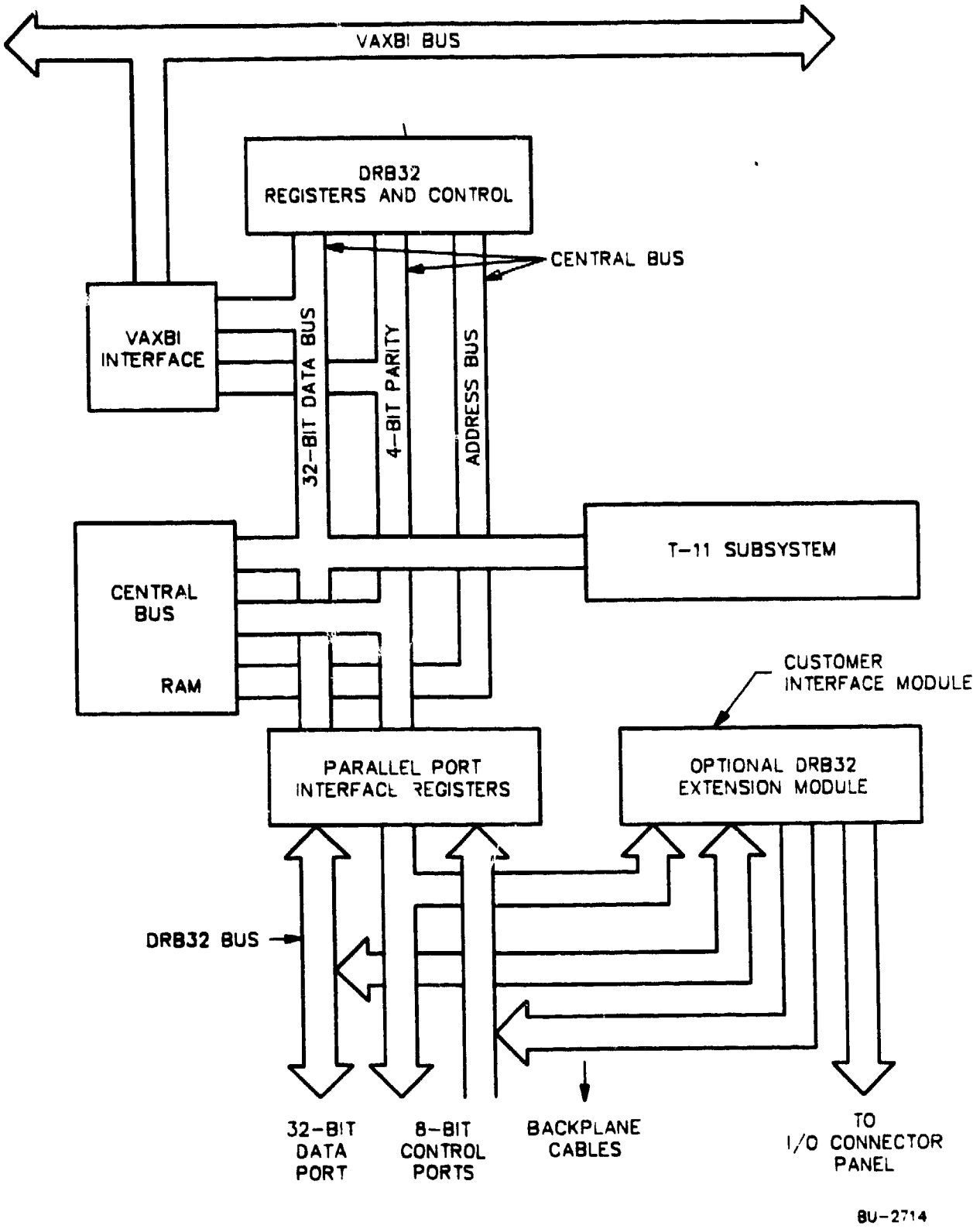


Figure 2-1 DRB32 Block Diagram

In Figure 2-1, the parallel I/O port includes the DRB32 microcontroller, the central bus, the CB (central bus) RAM (containing map registers) and the DRB32 user interface. The user interface has a 32-line (plus four parity bits) data path, two 8-bit (8-line plus two parity bits) control paths, as well as two device status signals.

The T-11 section of the DRB32 is a complete microcomputer subsystem, with a 10 MHz T-11 microprocessor, RAM, and ROM. The ROM contains self-test.

The VAXBI interface consists of the BIIC and BCAI chips. The BIIC handles VAXBI protocol, and the BCAI does byte alignment and provides some buffering.

The following sections describe the three major components in more detail.

2.1.1 Parallel I/O Port

The parallel I/O port of the DRB32 includes a microcontroller, the central bus, the CB (central bus) RAM, a 32-bit, half-duplex data port (with parity bits), and an 8-bit duplex control port. See Figure 2-1.

The DRB32 microcontroller times and controls all data transfers between the VAXBI and the user device.

The 32-bit data path carries the data between the VAXBI and the user device. It is a half-duplex data path, meaning that data flows in only one direction at a time and the path must be "turned around" for data to be transferred in the opposite direction.

The two 8-bit control paths carry control and protocol information. The central bus connects the microcontroller, the CB RAM, the VAXBI interface, the T-11, the data and parity buses, and the user device (through the user interface).

The CB RAM contains the map registers and provides a buffer area for communications between the VAXBI and the T-11.

The parallel port interface moves data between the user device and the DRB32 with positive confirmation. See Chapter 4 for more information on the parallel port interface.

2.1.1.1 Features of the 32-Bit Parallel I/O Port Data Path – The 32-bit parallel I/O port data path has the following features:

- Half duplex
- DMA (block) or Programmed I/O (Data) mode data transfers
- Large block transfers without processor intervention (up to 982,528 bytes)
- Fast data transfer rate
- Optional odd parity, one parity bit per data byte
- Two sets of transfer control and status registers for double buffering
- 32-, 16-, or 8-bit data path.

Since the DRB32 is a half-duplex device, transfers occur in one direction until the block transfer is finished. At that point, you can set the data flow to the opposite direction.

Block or Programmed I/O mode transfers can be selected by the user. To achieve high data transfer rates, see the suggestions in the section on "Optimizing Data Transfer Speed" in Chapter 4.

2.1.1.2 Features of the Two 8-Bit Control Paths - The two 8-bit control paths of the DRB32 parallel port have the following features:

- 8 bits in, 8 bits out
- Can be used as either 8-bit data paths or eight control lines
- Odd parity in each direction.

See section 4.3 of this manual for more information on the user interface control path.

2.1.1.3 Parity - The DRB32 can either accept odd parity on a per-byte basis for data from the parallel port, or internally generate parity for data before it is sent on the VAXBI. Parity is accepted from the VAXBI by the DRB32 and stored for map registers in the CB RAM. The parity is recalculated and checked when the map registers are combined with the Byte Offset Register to form the physical address.

2.1.1.4 Central Bus RAM - The CB RAM has 2K longwords (8K bytes) of storage that contain the map and control registers. The VAXBI uses the CB RAM to allow long continuous data transfers with Current and Next map register areas. The map register area in the CB RAM is segmented by software into two areas, a Current and Next set of map registers. The host processor first loads the Current map registers, and then loads the control and map registers for the Next segment transfer while the Current segment transfer is underway. When the Current registers are empty and that segment transfer is complete, the DRB32 checks to see if the Next registers are loaded. If they are, the DRB32 transfers their contents to the Current registers, and performs the indicated segment transfer.

2.1.2 T-11 Microprocessor Subsystem

The T-11 section of the DRB32 is a complete microcomputer subsystem. This subsystem contains:

- 16-bit PDP-11 microprocessor
- Diagnostic self-test in ROM
- 16-bit data bus
- An interface to the central bus of the DRB32.

The T-11 does not recognize or generate parity. Figure 2-2 is an illustration of the T-11 subsystem.

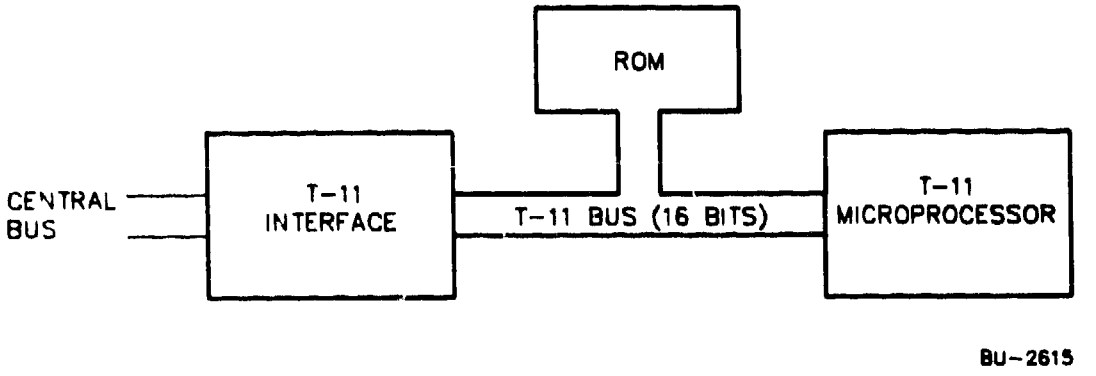


Figure 2-2 T-11 Subsystem

2.1.3 VAXBI Interface

The DRB32 responds to VAXBI master transactions as a VAXBI slave port. The DRB32 also can initiate master VAXBI transactions for data transfers, VAXBI interrupt transactions, and VAXBI loopback transactions. Figure 2-3 is an illustration of the VAXBI interface for the DRB32.

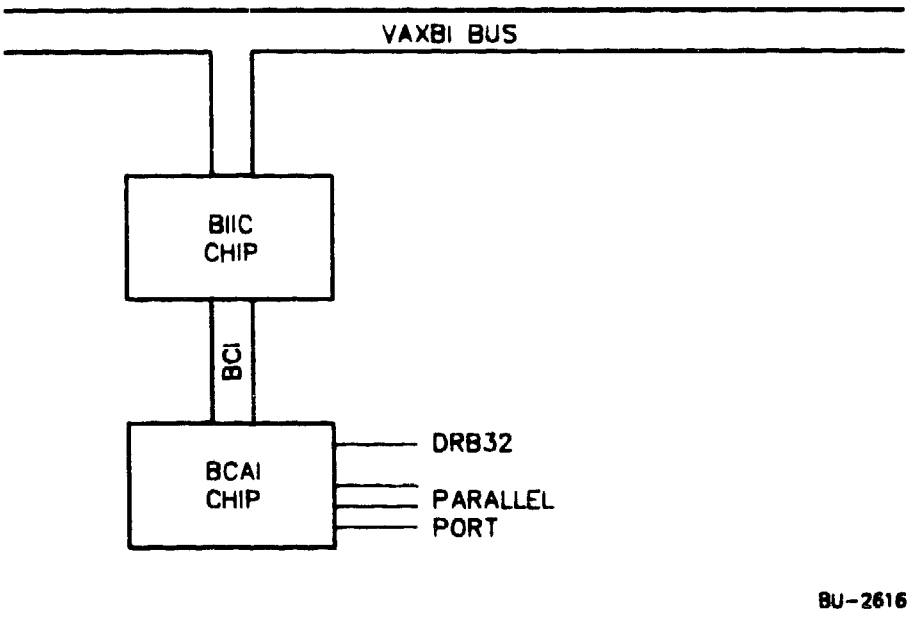


Figure 2-3 VAXBI Interface for the DRB32

The BIIC chip (Backplane Interconnect Interface chip) connects the DRB32 to the VAXBI. This chip handles all the bus protocol associated with data transfers on the VAXBI. It also handles data transfers and interrupt transactions.

The BCAI chip (BCI Adapter Interface Chip) provides high-speed data flow from the DRB32 to the BIIC. (BCI or VAXBI Chip Interface refers to the adapter side of the BIIC.) The BCAI generates and checks parity, and provides a dual-octaword buffer to smooth data transfers to the VAXBI. The BCAI also performs data alignment, therefore it allows user data buffers to start and end on any byte boundary in memory, while the data flows through the parallel port as aligned longwords.

2.2 VAXBI TRANSACTIONS

The DRB32 can act as bus master or respond as a slave node to VAXBI transactions. On the VAXBI, the node that gains control of the bus for a command transaction is the master. The node that responds is the slave. This section describes the following:

- VAXBI transactions initiated by the DRB32 as bus master
- DRB32 responses as slave node to VAXBI transactions.

2.2.1 DRB32 Master VAXBI Transactions

The DRB32 can initiate the following master VAXBI transactions:

- Octaword Read
- Octaword Write
- Masked Octaword Write with Cache Intent (WMCI)
- Interrupt (INTR).

2.2.1.1 READ – The DRB32 issues octaword READ transactions for all data transfers from the VAXBI to the user device.

2.2.1.2 WRITE – The DRB32 issues octaword WRITE transactions for all data transfers from the user device to the VAXBI. First and last segment transfers, however, may use a masked octaword WRITE (WMCI). If the first and last segments are complete octawords, a WRITE transaction is used. If the first and last segments are not complete octawords, a masked octaword WRITE (WMCI) is used to align the data to octaword boundaries.

2.2.1.3 Masked Octaword WRITE (WMCI) – The DRB32 issues masked octaword WRITE (WMCI) transactions for the first and last VAXBI transaction of a segment transfer from the user device to the VAXBI, when the segment transfer is not a complete octaword. The first masked octaword WRITE is used to align the user data to octaword boundaries on the VAXBI. The last masked octaword WRITE is used to transfer the exact number of bytes left in a data transfer.

2.2.1.4 Interrupt (INTR) – The DRB32 initiates a VAXBI interrupt (INTR) transaction for the following reasons:

- Current segment is complete
- I/O Control Register (IOCTL) change of state
- Errors.

The corresponding bit in the DRB32 Interrupt Enable Register (DRBIE) must be set for the DRB32 to issue the INTR transaction.

2.2.2 DRB32 Slave Response to VAXBI Commands

The DRB32 responds as a slave node to READ-type, WRITE-type, and data control commands issued by the VAXBI. The purpose of these commands is to allow the VAXBI node to read data from, or write data to, the DRB32's registers. The VAXBI node that originates these commands is the master node in the transaction.

READ-type commands are VAXBI transactions that the DRB32 responds to as if they were READ commands. READ-type commands include:

- READ To Node Space
- Interlock Read with Cache Intent (IRCI)
- Read with Cache Intent (RCI).

WRITE-type commands are VAXBI transactions that the DRB32 responds to as if they were WRITE commands. WRITE-type commands include:

- WRITE To Node Space
- WRITE with Cache Intent (WCI)
- Unlock Write Mask with Cache Intent (UWMCI)
- Write Mask with Cache Intent (WMCI).

The data control commands that the DRB32 responds to include the following:

- Identify (IDENT)
- STOP.

The DRB32 does not support the following VAXBI transactions, and therefore sends NO ACK if it receives them.

- READ To Adapter Window Space
- WRITE To Adapter Window Space
- Interrupt (INTR)
- Invalidate (INVAL)
- Broadcast (BDCST)
- Interprocessor Interrupt (IPINTR).

The DRB32 responses to each VAXBI transaction are described in the following sections.

2.2.2.1 READ-Type Transactions - The DRB32 responds in the same way to the following READ-type transactions: READ, IRCI, and RCI. The DRB32 only supports longword READ transactions to the device register space, and responds with NO ACK to all other lengths of READ transactions.

Up to eight stalls may be issued before the transaction is acknowledged, because of contention for the Central Bus on the DRB32.

2.2.2.2 WRITE-Type Transactions - The DRB32 responds in the same way to the WRITE-type transactions: WRITE, WCI, UWMCI, and WMCI. The DRB32 only supports longword WRITE transactions to the device register space, and responds with NO ACK to all other lengths of WRITE transactions.

Up to eight stalls may be issued before the transaction is acknowledged, because of contention for the Central Bus on the DRB32. The DRB32 ignores the mask bits in the WMCI and UWMCI transactions.

2.2.2.3 IDENT (Identify) - IDENT transactions occur after the DRB32 issues an interrupt to the host processor. The host processor issues an IDENT transaction to find out what device issued the interrupt. The DRB32 sends an interrupt vector.

The DRB32 has two vectors: the user interface vector and the error vector. The user interface vector is loaded into the BIIC User Interrupt Control Register by the operating system during the initial start-up sequence. The external vector (EX VECTOR) bit in the BIIC User Interface Interrupt Control Register is cleared during start-up. The error vector is loaded into the Error Interrupt Control Register by the operating system during system initialization. The error vector is sent in response to IDENT when the BIIC generates an interrupt because it detected an error. The DRB32 uses the user interface vector when it wishes to interrupt the host processor.

2.2.2.4 STOP - Host processors use the STOP transaction to temporarily halt a node from issuing VAXBI transactions. A STOP transaction received by the DRB32 does the following:

- Terminates any DRB32 segment transfer
- Clears the Go bit in the Parallel Port CSR Register
- Stops any future transfers until the Parallel Port Control Register is reloaded
- Completes any pending VAXBI master transactions before stopping the DRB32.

The Pending Segment bit is not cleared by a STOP transaction.

2.3 DRB32 ADDRESS SPACE

This section describes the following:

- VAXBI address space
- DRB32 address space as seen by the VAXBI.

The DRB32 does not implement window space.

2.3.1 VAXBI Address Space Description

The 1024MB system address space on the VAXBI is divided into two areas:

- Memory space (from address 0000 0000 through 1FFF FFFF hex)
- I/O space (from address 2000 0000 through 3FFF FFFF hex).

The VAXBI system I/O space is divided into several dedicated sections. One of these dedicated sections is the VAXBI node space. By assigning a section of node space to the DRB32, the VAXBI obtains access to the DRB32 address space.

VAXBI node space consists of 16 blocks of I/O addresses that are allocated to the 16 possible VAXBI nodes. The size of each block is 8KB. A VAXBI node's node space assignment is based on that node's ID (0 through 15). The starting address of node spaces for nodes 1 through 15 is 2000 0000 plus 8K times the node ID. See Table 2-1.

Table 2-1 Node Space Base Address Assignments

Node No.	Hex Address
0	2000 0000
1	2000 2000
2	2000 4000
3	2000 6000
4	2000 8000
5	2000 A000
6	2000 C000
7	2000 E000
8	2001 0000
9	2001 2000
A	2001 4000
B	2001 6000
C	2001 8000
D	2001 A000
E	2001 C000
F	2001 E000

The VAXBI system I/O space is shown in Figure 2-4.

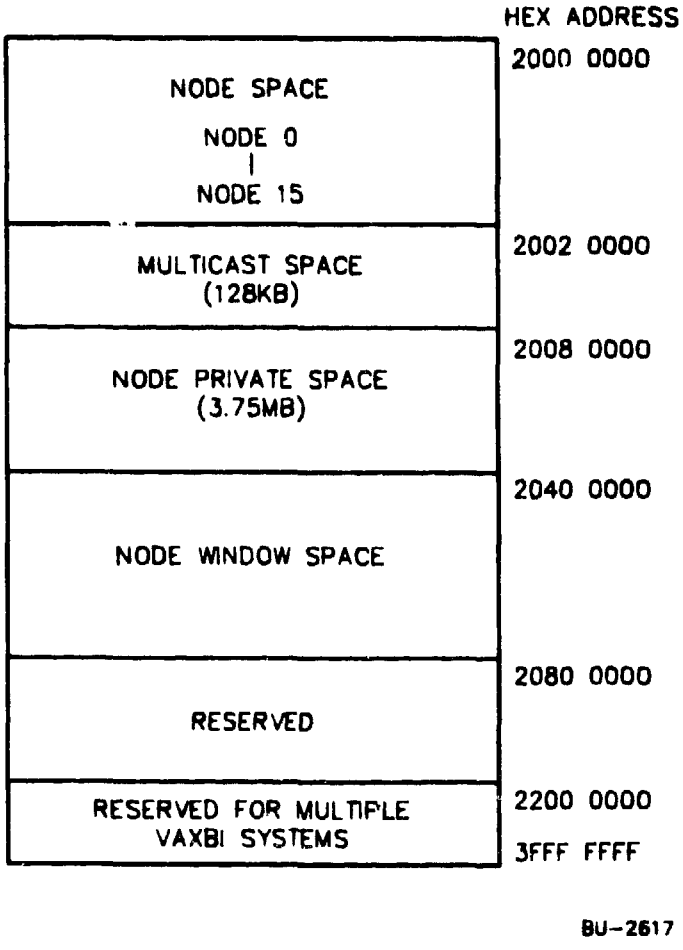


Figure 2-4 VAXBI I/O Address Space

2.3.2 DRB32 Address Space Description

To any of the other VAXBI nodes, the DRB32 address space appears as node space on the VAXBI. A VAXBI node can communicate with the DRB32 through this node space and can see the BIIC registers, parallel port registers, map registers, and the RXCD (Receive Console Data) Register.

2.3.3 DRB32 Address Space as Seen by the VAXBI

The DRB32 address space accessible to the VAXBI is shown in Figure 2-5.

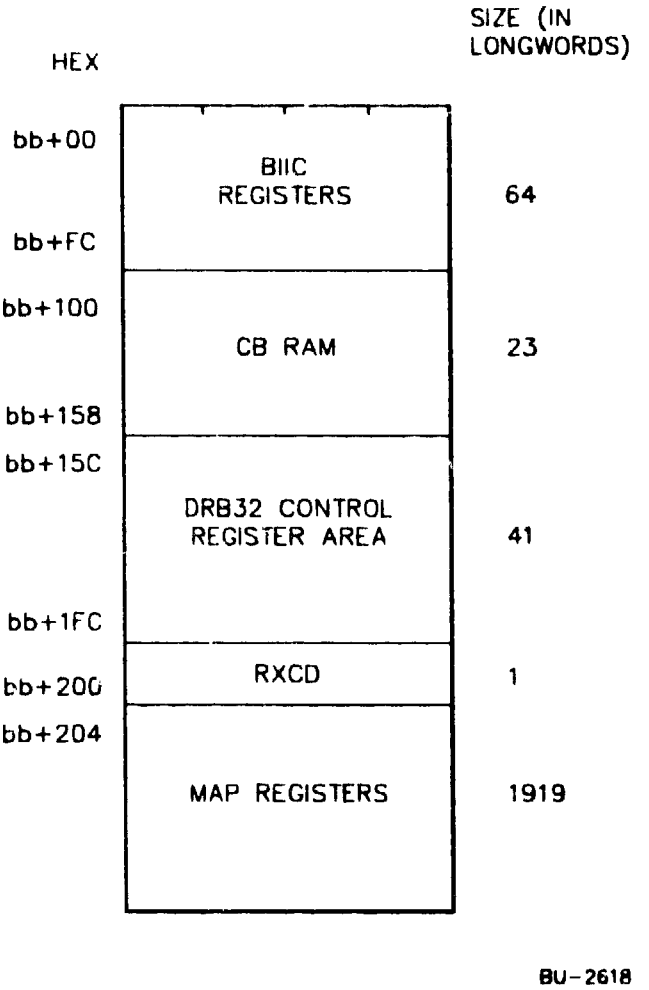


Figure 2-5 DRB32 Address Space

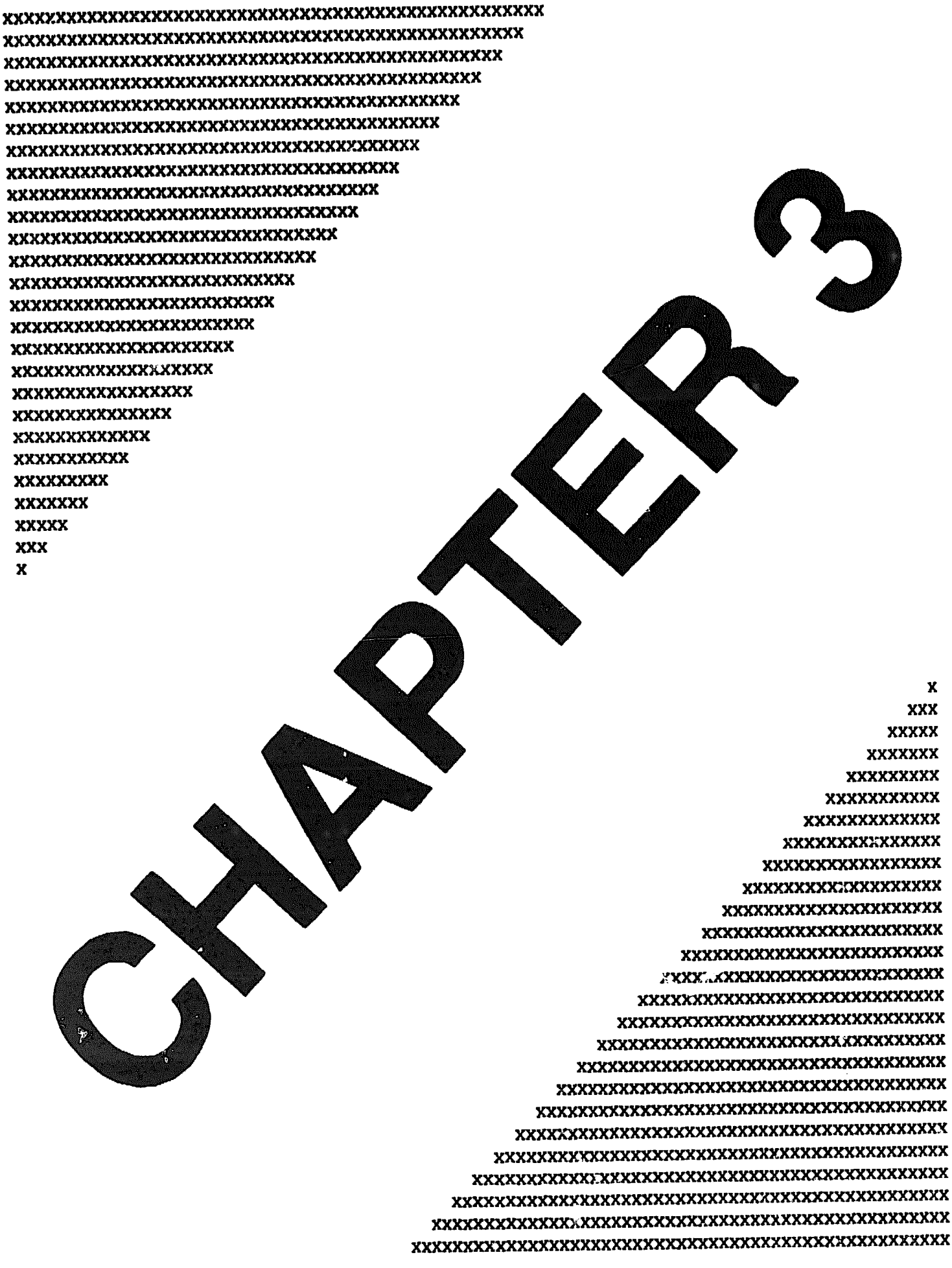
2.4 DRB32 POWER-UP SEQUENCE

The power-up sequence ensures that the DRB32 always starts up in a standard configuration. If necessary, your application can change these configurations while the application is running.

The DRB32 power-up sequence consists of the following:

- BIIC self-test executes
- DRB32 self-test executes
- The default configuration is set up
- The self-test LED lights and the Broke bit clears upon successful completion of the power-up sequence.

See Appendix A for a power-up timing diagram.


CHAPTER 3

CHAPTER 3
DRB32 REGISTER DESCRIPTIONS

This chapter describes the DRB32 registers. The DRB32 registers include BIIC registers and DRB32 module registers.

The BIIC registers are physically located in the BIIC chip, which is the interface between the VAXBI and the DRB32. The DRB32 module registers include the parallel port registers, map registers in the CB RAM, the BCAI registers, and the RXCD register. The DRB32 module registers control the parallel port and enable error notification and interrupt services.

3.1 BIIC REGISTERS

The BIIC registers are listed in Table 3-1.

Table 3-1 BIIC Registers

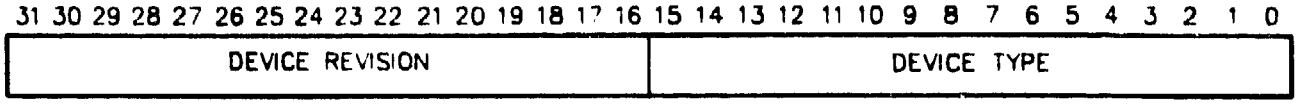
VAXBI Address	Register Name
bb+00	Device Reg (DTYPE)
bb+04	VAXBI Control and Status Reg (BICSR)
bb+08	Bus Error Reg (BER)
bb+0C	Error Interrupt Control Reg (EINTRCSR)
bb+10	Interrupt Destination Reg (INTRDES)
bb+14	IPINTR Mask Reg
bb+18	Force IPINTR/STOP Destination Reg
bb+1C	IPINTR Source Reg
bb+20	Starting Address Reg
bb+24	Ending Address Reg
bb+28	BCI Control Reg
bb+2C	Write Status Reg
bb+30	Force IPINTR/STOP Command Reg
bb+40	User Interface Interrupt Control Reg
bb+F0	GPR 0 (self-test error code)
bb+F4	GPR 1 (DRB32 internal code revision level)
bb+F8	General Purpose Reg 2
bb+FC	General Purpose Reg 3

The BIIC registers are described below.

3.1.1 DTYPE—Device Register

This register is written by self-test. It should not be written by application code.

VAXBI : bb+0000



BU-2619

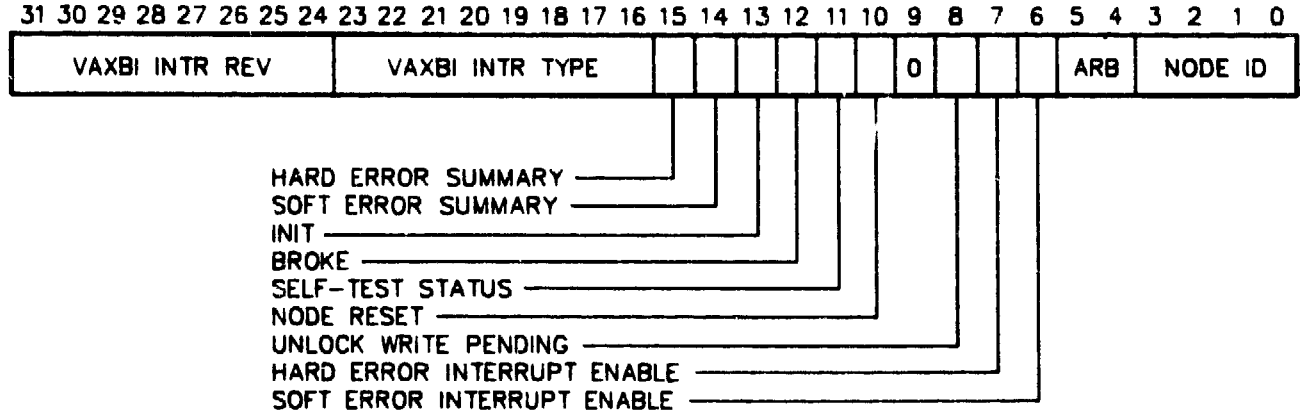
Device Revision <31:16>. This word indicates the current revision of the DRB32 hardware. It is written by self-test before the Broke bit in the VAXBI Control and Status Register is cleared.

Device Type <15:0>. This word is written by self-test before the Broke bit in the VAXBI Control and Status Register is cleared. It is loaded with the DRB32 device type (hex 0101).

3.1.2 BICSR—VAXBI Control and Status Register

The VAXBI Control and Status Register provides status information about the VAXBI and determines DRB32 Arbitration mode.

VAXBI : bb+0004



BU-2620

VAXBI Interface Revision - <31:24>. This field indicates the revision of the BIIC in this DRB32. This field is READ-only.

VAXBI Interface Type - <23:16>. This field indicates the type of VAXBI interface on the DRB32. It is always 00000001.

HES - <15> - Hard Error Summary. When set, this bit indicates that one or more of the hard error bits in the Bus Error Register is set. This bit is READ-only.

SES - <14> - Soft Error Summary. When set, this bit indicates that one or more of the soft error bits in the Bus Error Register is set. This bit is READ-only.

INIT - <13> - INIT bit. The DRB32 ignores this bit.

BROKE - <12>. The DRB32 clears this bit when the module has passed all internal self-tests. If this bit is set, the DRB32 is either in the process of running its internal self-test or has failed it. This bit can be cleared by writing a 1 to it, but should only be READ.

STS - <11> - Self-Test Status. This bit indicates the result of (only) the BIIC internal self-test. If the BIIC internal self-test is passed, the bit is a 1. If the BIIC internal self-test is failed, the bit is 0, the BIIC VAXBI drivers are disabled, and the chip cannot drive the bus. If the STS bit is reset, the WRITE transaction used to set the bus receives a NO-ACK response. This bit is READ/WRITE.

NRST - <10> - Node Reset. Writing a 1 to this bit initiates a complete DRB32 module self-test. When this bit is written (with a 1), the STS bit must also be written (with a 1) to ensure proper operation of the WRITE-type transaction. READs always return a 0. When the NRST bit is set, the DRB32 does not return ACK to slave transactions it receives, until DRB32 self-test successfully completes. Before completion of self-test, NO-ACKs are returned.

UWP - <08> - Unlock Write Pending. When set, this bit indicates that a Interlock Read with Cache Intent (IRCI) transaction has been successfully completed by this node and there has not yet been a subsequent Unlock Write Mask With Cache Intent (UWMCI) command.

HEIE - <07> - Hard Error Interrupt Enable. Determines whether an error interrupt is generated by the DRB32 when HES is asserted.

SEIE - <06> -Soft Error Interrupt Enable. Determines whether an error interrupt is generated by the DRB32 when SES is asserted.

ARB - <05:04> - Arbitration Control. The two ARB bits determine the DRB32 Arbitration mode. See Table 3-2 below.

Table 3-2 DRB32 Arbitration Modes

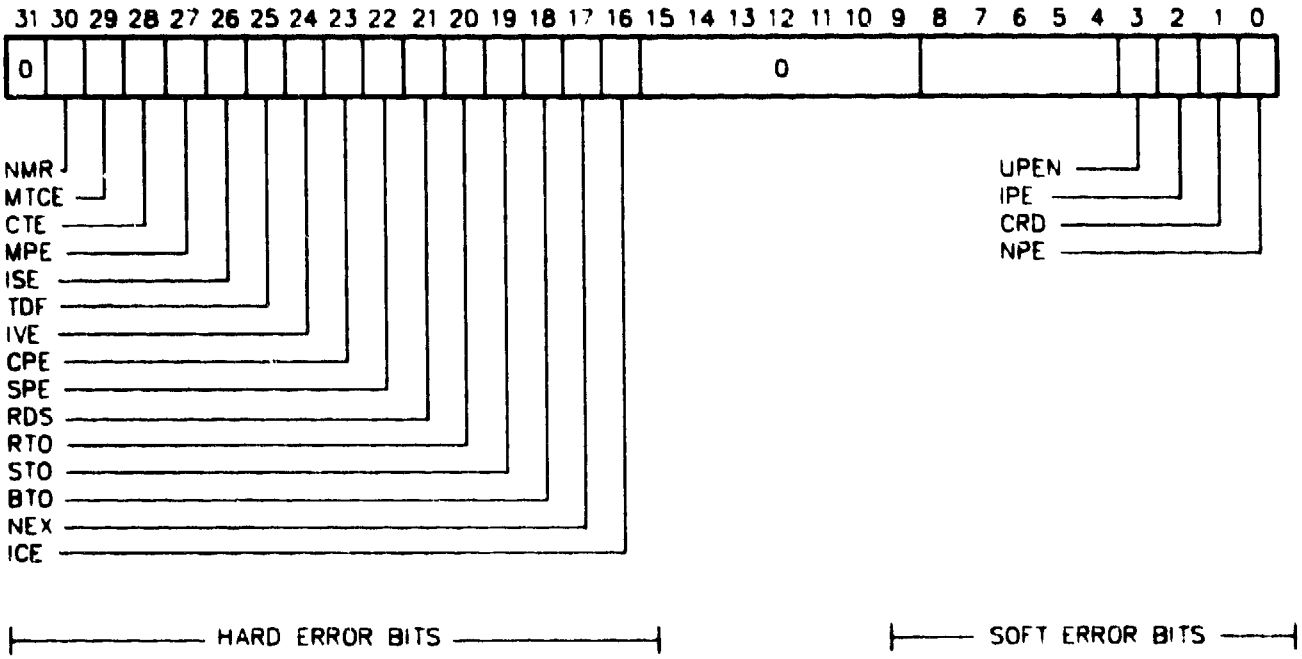
Code	Arbitration Mode
00	Dual round-robin arbitration
01	Fixed high priority
10	Fixed low priority
11	Disable arbitration

NODE ID - <03:00>. The DRB32 node ID is dependent upon the node ID plug placed on the VAXBI. The node ID is automatically loaded during the power-up sequence. This field is READ-only.

3.1.3 BER—Bus Error Register

This register indicates hard and soft errors detected on the bus.

VAXBI : bb+0008



BU-2621

NMR - <30> - NO-ACK to Multi-Responder Command Received. This bit is set by the bus master if it receives a NO-ACK command confirmation for an INVAL, STOP, INTR, IPINTR, BDCST, or RESERVED command.

MTCE - <29> - Master Transmit Check Error. This bit is set if transmitted data during a master transaction does not agree with received data, in cycles where the master is the only source of data.

CTE - <28> - Control Transmit Error. This bit is set if the BIIC detects a deasserted state on VAXBI NO ARB L, VAXBI BSY L and VAXBI CNF<2:0> L when it is trying to assert them.

MPE - <27> - Master Parity Error. This bit is set if the master detects a parity error on the bus during a READ-type or vector ACK data cycle. This bit can be set during BIIC self-test. It is cleared by the firmware initialization routine after the DRB32 self-test has passed.

ISE - <26> - Interlock Sequence Error. This bit is set if this node successfully completes a UWMCI transaction when the Unlock Write Pending (UWP) bit in the BCI Control Register is not set.

TDF – <25> – Transmitter During Fault. The TDF bit is set by either the master or the slave following the detection of a parity error during a cycle in which that node was responsible for transmitting proper parity on the VAXBI.

IVE – <24> – IDENT Vector Error. This bit is set by the slave if, after sending an interrupt vector to the master that issued an IDENT, anything but an ACK confirmation is received.

CPE – <23> – Command Parity Error. This bit is set when the BIIC detects a parity error in a command/address cycle.

SPE – <22> – Slave Parity Error. This bit is set by the slave if the BIIC detects a parity error during a WRITE-type ACK or STALL Data Cycle, or BDCST ACK Data Cycle.

RDS – <21> – Read Data Substitute. This bit is set if a Read Data Substitute or RESERVED status code is received during a READ-type or IDENT (for vector status) transaction.

RTO – <20> – Retry Timeout. This bit is set if the master receives 4096 consecutive RETRY responses from the slave for the same master port transaction.

STO – <19> – Stall Timeout. This bit is set if the slave port asserts the STALL code on the BCI RS<1:0> lines for 128 consecutive cycles.

BTO – <18> – Bus Timeout. This bit is set if the BIIC is unable to start at least one pending transaction before 4096 cycles have elapsed.

NEX – <17> – Non-Existent Address. This bit is set when a NO-ACK response is received for a READ-type or WRITE-type command sent by the BIIC.

ICE – <16> – Illegal Confirmation Error. This bit indicates that a RESERVED or illegal confirmation code has been received during a transaction in which the BIIC is involved.

UPEN – <03> – User Parity Enable. This is a READ-only bit that indicates the BIIC parity mode. A 1 indicates the BIIC is configured for user-generated parity, while a 0 indicates the BIIC will provide the parity generation. Since the BCAI always generates parity for the DRB32, this bit must always be set.

IPE – <02> – ID Parity Error. This bit indicates that a parity error was detected on the encoded master ID during an embedded ARB cycle.

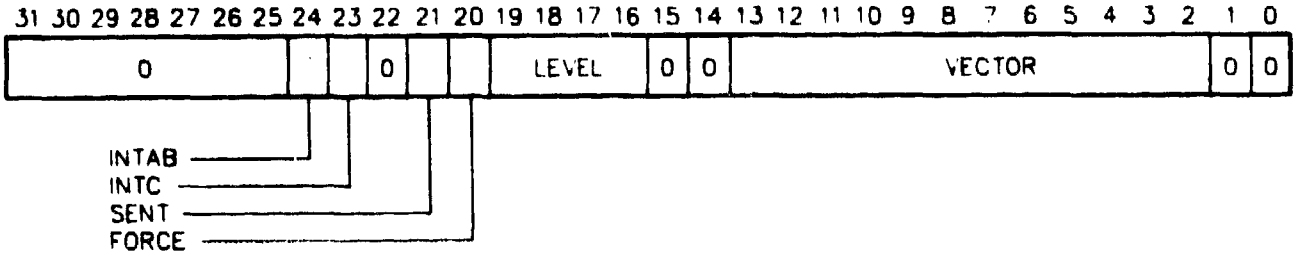
CRD – <01> – Corrected Read Data. This bit indicates that a corrected read data status code was received during a READ-type transaction initiated by the master port.

NPE – <00> – Null Bus Parity Error. This bit indicates that ODD parity was detected on the bus during the second cycle of a two cycle sequence during which VAXBI NO ARB L and VAXBI BSY L were unasserted.

3.1.4 EINTRCSR—Error Interrupt Control Register

The Error Interrupt Control Register controls the operation of interrupts initiated by a BIIC-detected bus error or by a Force bit set in this register.

VAXBI : bb+000C



BU-2622

INTAB – <24> – Interrupt Abort. The Interrupt Abort bit is set if an INTR command sent under control of this register is aborted. Reset this bit by writing a 1 to it.

INTC – <23> – Interrupt Complete. This bit is set when the vector for an error interrupt has been successfully transmitted or if an INTR command sent under the control of this register has aborted. This bit is reset when the interrupt request is removed.

SENT – <21>. The SENT bit indicates that an INTR command for this interrupt has been sent, and that an IDENT command is expected. This bit is cleared during an successful IDENT command matching the interrupt.

FORCE – <20>. This bit is set to initiate an error interrupt request.

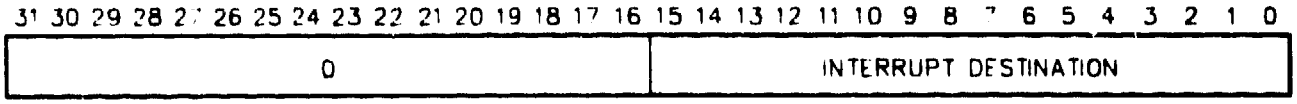
LEVEL – <19:16>. The LEVEL field determines the level(s) at which INTR commands are transmitted under the control of this register.

VECTOR – <13:2>. This field contains the vector used during error interrupt sequences.

3.1.5 INTRDES—Interrupt Destination Register

This register is loaded with the VAXBI nodes to be interrupted when a DRB32 sends an interrupt. If bit · 0 · is set, Node 0 is the target of the interrupt. If bit · 2 · is set, Node 2 is the target of the interrupt.

VAXBI : bb+0010 R/W

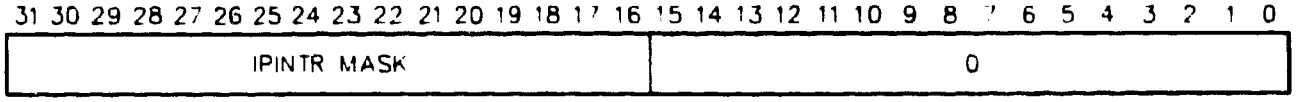


BU-2623

3.1.6 IPINTRMSK—Interprocessor Interrupt Mask Register

This register contains the mask that determines which nodes are permitted to send interprocessor interrupts to the DRB32. Since interprocessor interrupts are not supported on the DRB32, IPINTRMSK is set to (hex) 0000 by the VAXBI operating system at power-up, after the Broke bit is cleared by the DRB32 self-test.

VAXBI : bb+0014

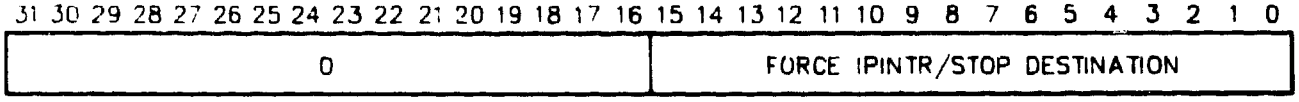


BU-2624

3.1.7 FIPSDES—Force Interprocessor Interrupt/Stop Destination Register

This register determines which nodes on the VAXBI are to be targeted by Force-bit Interprocessor Interrupt or STOP commands issued by this DRB32. Since interprocessor interrupts are not supported on the DRB32, FIPSDES is set to (hex) 0000 by the VAXBI operating system at power-up, after the Broke bit is cleared by the DRB32 self-test.

VAXBI : bb+0018 R/W

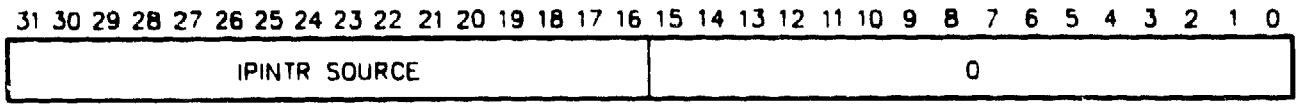


BU-2625

3.1.8 IPINTRSRC—Interprocessor Source Register

This register is loaded with the decoded ID of any node that sends an IPINTR command to the DRB32. This register should always be (hex) 0000, because IPINTR transactions to the DRB32 are not supported.

VAXBI : bb+001C

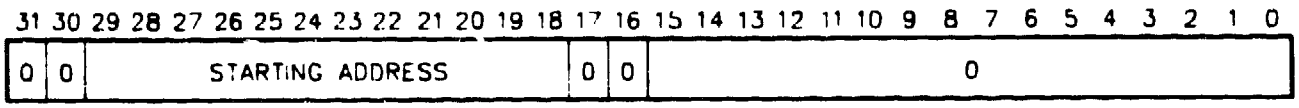


BU-2626

3.1.9 SADR—Starting Address Register

This register determines the starting address of the adapter window space of the DRB32. Since the DRB32 does not implement adapter window space, this register is cleared by the VAXBI operating system after self-test is completed.

VAXBI : bb+0020

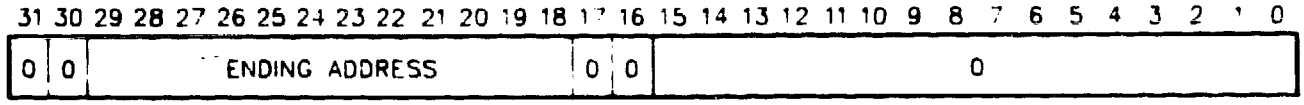


BU-2627

3.1.10 EADR—Ending Address Register

This register is loaded with the (last address + 1) of the adapter window space. Since the DRB32 does not implement adapter window space, this register is cleared by the VAXBI operating system after self-test is completed.

VAXBI : bb+0024

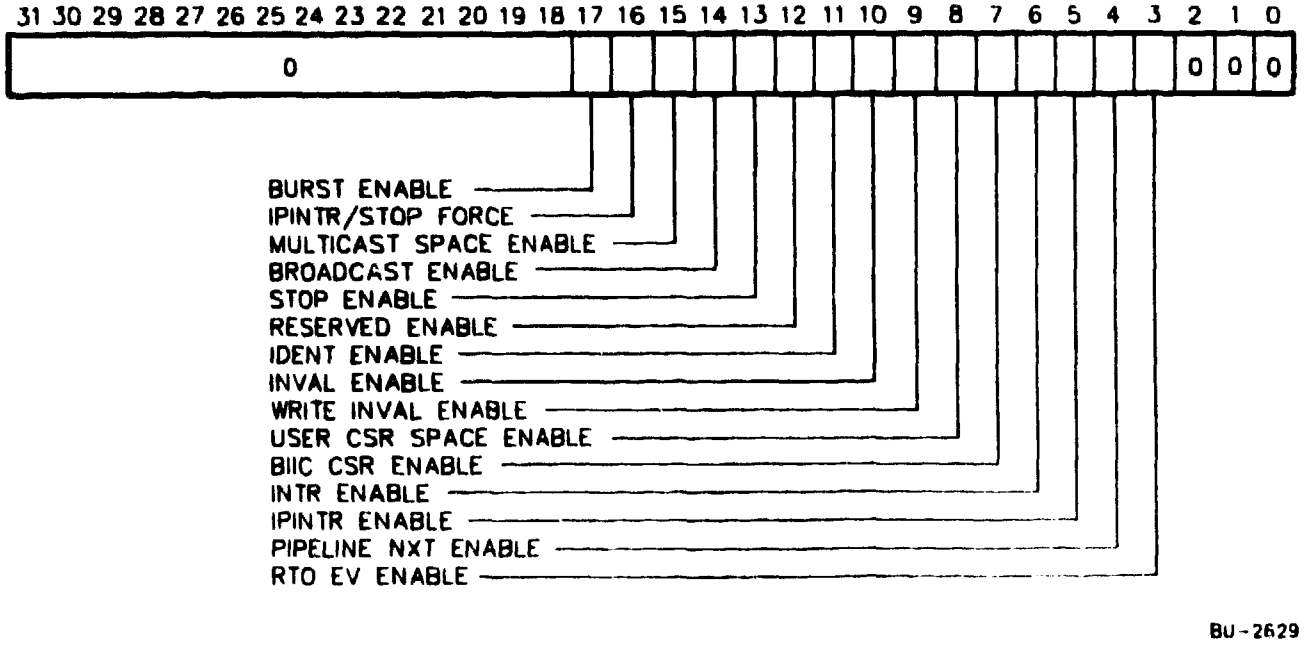


BU-2628

3.1.11 BCICSR—BCI Control Register

This register is used to control the BCI, enabling various VAXBI transactions and other control functions.

VAXBI : bb+0028



BU-2629

BURSTEN - <17> - BURST Enable. When set, this bit causes VAXBI NO ARB L to be asserted continuously after the next successful ARB by this node until the BURSTEN bit is reset.

IPINTR/STOP FORCE - <16> - Interprocessor Interrupt and Stop Force. When set, this bit causes the BIIC to arbitrate for the bus, transmitting an IPINTR or STOP command (depending on the command stored in the IPINTR/STOP Force Register), and using the IPINTR/STOP Destination Register for the destination field. The BIIC resets the IPINTR/STOP Force bit when the transaction is transmitted.

MSEN - <15> - Multicast Space Enable. This bit must always be 0 because the DRB32 hardware does not support multicast space.

BDCSTEN - <14> - BROADCAST Enable. This bit must always be 0 because the DRB32 hardware does not support broadcast transactions.

STOPEN - <13> - STOP Enable. When set, this bit allows the DRB32 to receive STOP commands from the VAXBI. This bit must always be 1, because the DRB32 must always be able to receive STOP transactions.

IDENTEN - <11> - IDENT Enable. When set, this bit causes the BIIC to issue hardware commands to the DRB32, which the DRB32 ignores. The BIIC automatically responds to IDENT commands to the DRB32 regardless of the state of this bit.

INVALEN - <10> - INVAL Enable. This bit must always be 0 because the DRB32 hardware does not support INVAL transactions.

WINVALEN - <09> - WRITE Invalidate Enable. This bit must always be 0 because the DRB32 hardware does not support WRITE invalidate transactions.

UCSREN - <08> - User CSR Space Enable. This bit allows the DRB32 to respond to slave transactions to the user CSR address space of the DRB32. This bit must always be 1 because the DRB32 must always respond to slave transactions.

BICSREN - <07> - BIIC CSR Space Enable. When set, this bit causes the BIIC to issue hardware commands to the DRB32, which the DRB32 ignores. The BIIC automatically responds to commands to the BIIC CSR Space regardless of the state of this bit.

INTREN - <06> - INTR Enable. This bit must always be 0 because the DRB32 hardware does not support Interrupt transactions to the DRB32.

IPINTR - <05> - IPINTR Enable. This bit must always be 0 because the DRB32 hardware does not support IPINTR transactions to the DRB32.

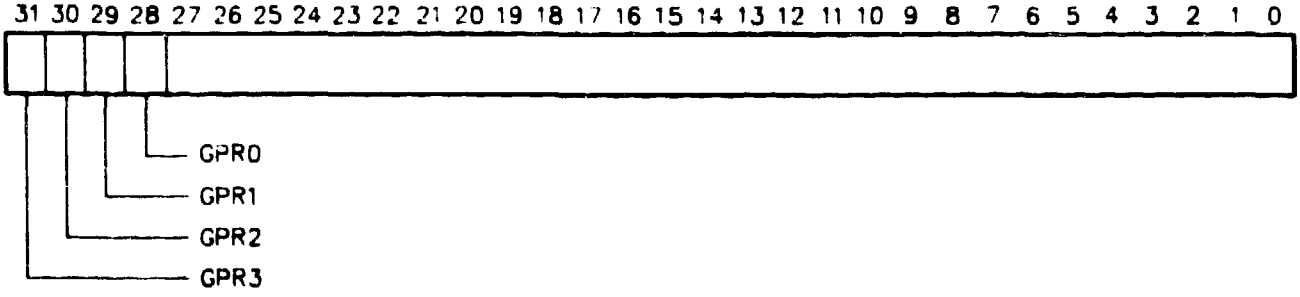
PNXTEN - <04> - Pipeline NXT Enable. This bit must always be 0 because the DRB32 hardware does not support the BIIC pipeline mode.

RTOEVEN - <03> - RTO EV Enable. When set, this bit enables the output of an RTO EV Code in place of the RCR EV Code following the occurrence of a retry timeout. This bit must always be 0, indicating that the DRB32 does not timeout after a fixed number of retries (it continues to retry forever).

3.1.12 WSTAT—Write Status Register

This register indicates which general purpose register(s) has been written to.

VAXBI : bb+002C

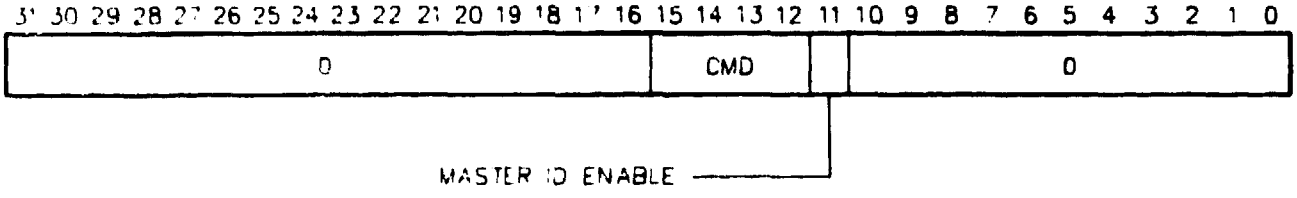


BU-2630

3.1.13 FIPSER—Force Interprocessor Interrupt/Stop Register

The FIPSER Register is not used by the DRB32.

VAXBI : bb+0030 R W



BU-2631

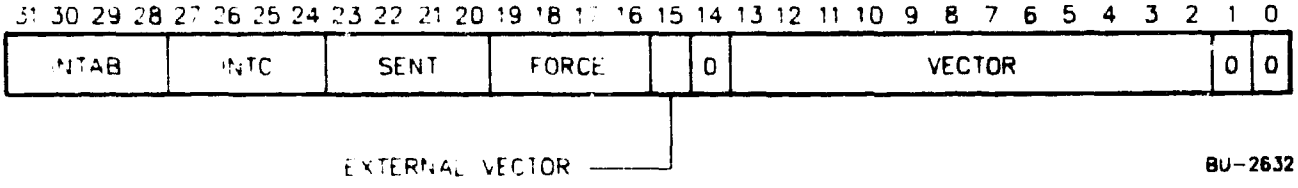
CMD – <15:12> – Command Code. This 4-bit field is transmitted during the command/address cycle of a transaction initiated by the IPINTR/STOP Force bit. Only IPINTR (1111) and STOP (1100) commands are allowed to be loaded into this field.

MIDEN – <11> – Master ID Enable. The setting of this bit allows the DRB32 master ID to be transmitted during the command/address cycle of a transaction initiated by the IPINTR/STOP Force bit.

3.1.14 UINTRCSR—User Interface Interrupt Control Register

This register controls the operation of the interrupts generated by the DRB32.

VAXB : bb+0040 R W



BU-2632

INTAB – <31:28> – Interrupt Abort. These bits are set if a DRB32-generated interrupt is aborted. These bits are reset by writing a 1 to the set bit.

INTC – <27:24> – Interrupt Complete. These bits (corresponding to the four interrupt levels) are set when the vector for an interrupt initiated by the DRB32 is successfully transmitted. The Interrupt Complete bits are reset when the cause of the interrupt is cleared.

SENT – <23:20>. These bits are set when an interrupt for the corresponding level has been sent. They are cleared by the VAXBI IDENT cycle corresponding to that level.

FORCE – <19:16>. These bits, when set, are the equivalent of asserting an interrupt at the corresponding level.

EX VECTOR – <15>. This bit indicates that an external vector will be supplied by the DRB32. The DRB32 hardware does not support external vectors, so this bit must always be 0.

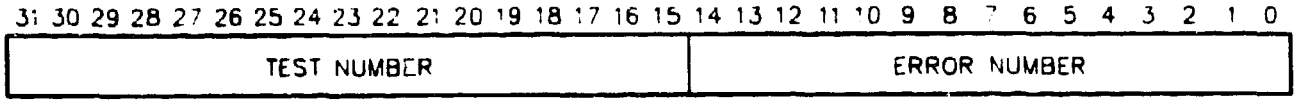
VECTOR – <13:2>. This section of the register is loaded by the VAXBI operating system with the interrupt vector transmitted during VAXBI IDENT transactions.

3.1.15 BIIC General Purpose Registers

The four BIIC general purpose registers are described below. GPR 2 and 3 are not used by the DRB32.

GPR0 – General Purpose Register 0. General Purpose Register 0 is loaded with the self-test error number if self-test fails, or with 0 if self-test successfully completes. See the *DRB32 Hardware Installation Guide* for more information on self-test and error numbers.

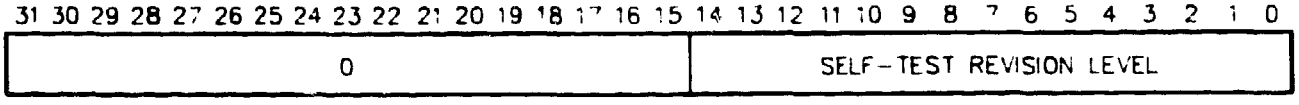
VAXBI : bb+00F0



BU-2633

GPR1 – General Purpose Register 1. (DRB32 internal code revision level) General Purpose Register 1 is loaded with the DRB32 self-test revision level.

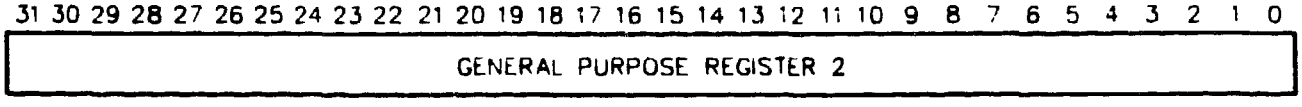
VAXBI : bb+00F4



BU-2634

GPR2 – General Purpose Register 2. General Purpose Register 2 is not used by the DRB32.

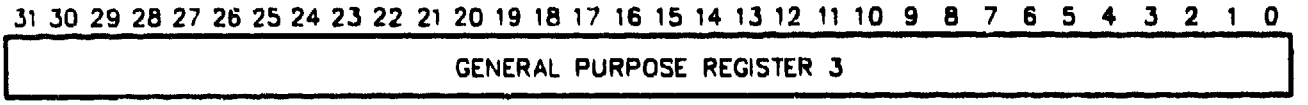
VAXBI : bb+00F8



BU-2635

GPR3 – General Purpose Register 3. General Purpose Register 3 is not used by the DRB32.

VAXBI : bb+00FC



BU-2636

3.2 DRB32 MODULE REGISTERS

The DRB32 module registers:

- Control the parallel I/O port
- Enable error notification and interrupt services
- Enable use of the map registers for data transfers.

3.2.1 Parallel I/O Port Registers

The parallel I/O port registers control the parallel I/O port, enable interrupt and error notification services, and enable the use of map registers. The parallel I/O port registers are shown in Table 3-3.

Table 3-3 Parallel I/O Port Registers

VAXBI Address	Register Name	
bb+015C	BAR	BCAI Access Register
bb+0160	PARCSR	Par Port Ctl & Status Reg
bb+0164	PARSTS	Par Port Setup & Test Reg
bb+0168	ERRREG	Error Reg
bb+016C	DRBIE	DRB32 Interrupt Enable Reg
bb+0170	DRBIRL	DRB32 Interrupt Req. Level Reg
bb+0174	DRBIFR	DRB32 Interrupt Flag Reg
bb+0178	CURMR	Current Map Reg Pointer
bb+017C	CURTOMR	Current Top Of Map Reg Area
bb+0180	CURBOFF	Current Address Byte Offset
bb+0184	CURBLFT	Current Number of Bytes Left in Segment

Table 3-3 Parallel I/O Port Registers (Continued)

VAXBI Address	Register Name	
bb+0188	PHYSADDR	Physical Address
bb+018C	IODAT	I/O Data Reg
bb+0190	IOCTL	I/O Control Reg
bb+0194	NXTMR	Next Map Reg Pointer
bb+0198	NXTTOMR	Next Top of Map Reg Area
bb+019C	NXTBOFF	Next Address Byte Offset
bb+01A0	NXTBLFT	Next Bytes Left in Segment
bb+01A4	T11ICR	RESERVED
bb+01A8	T11MDTR	RESERVED
bb+01AC	DPWR	Data Port Width Reg
bb+01B0	BOBP	RESERVED

The parallel I/O port registers are described below.

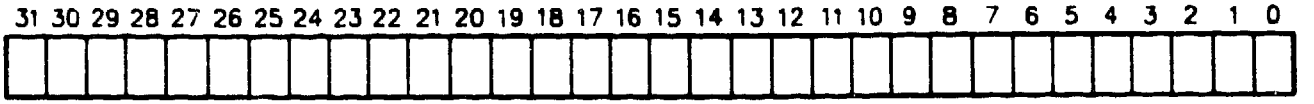
NOTE

In the register descriptions, an x indicates that the value of the bit is unpredictable. Use masking operations if performing tests on these registers.

3.2.2 BAR—BCAI Access Register

The BCAI Access Register is used to read the registers in the BCAI (which are not directly accessible to the VAXBI). This register is normally used for extracting stranded data from the dual-octaword register after a DRB32 error condition (or the setting of the Abort bit in the PARCSR) halts a DMA operation in which data is being read from a user device.

VAXBI : bb+15C R



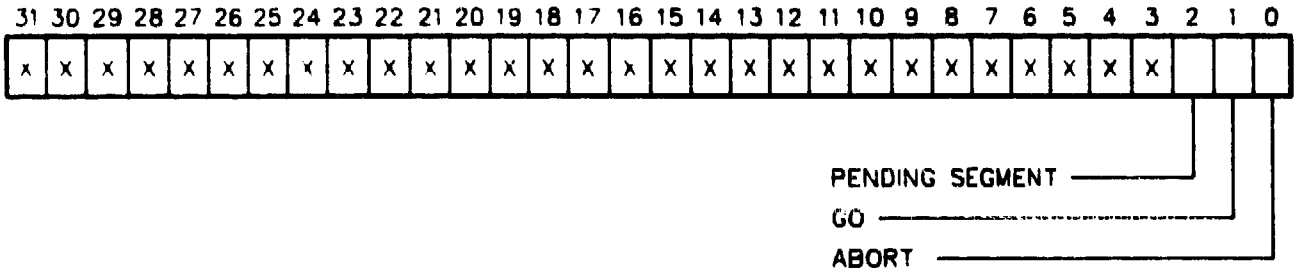
BU-2637

The lowest seven bits of the address of the BCAI register of interest is written into the BCAI Access Register. When the BAR is read after writing the address, the data in the selected BCAI register at the time of the address is read. Refer to Appendix D for detailed information on using this register.

3.2.3 PARCSR—Parallel Port Control and Status Register

The Parallel Port Control and Status Register is used by the device driver running in the host processor to start or abort a data transfer. The register also provides information about the current state of the transfer. All bits are set by the host processor and cleared by the DRB32 microcontroller.

VAXBI : bb+0160 R/W



BU-2638

ABORT – <0>. This bit is set by the host processor to tell the DRB32 microcontroller to terminate the current segment transfer. The bit is cleared by the DRB32 microcontroller when the Go bit for the next segment transfer is set. An 8-bit error code is put in the Error Register (refer to section 3.2.5.) to indicate that an Abort bit was the reason for the Idle state.

If the Abort bit is set while the DRB32 is sending data to a user device, the DRB32 completes the current VAXBI cycle and returns to its Wait state. Data obtained during that cycle is not sent to the user. If the DRB32 is transferring data from a user device to the VAXBI, setting the Abort bit terminates the transfer of data into the user port. Data from the user port that has already been accepted and loaded into the BCAI octaword buffer registers before the Abort bit is set is not transferred to memory when the Abort bit is set.

All registers and the contents of the CB RAM are left in their current state. The DRB32 responds to VAXBI transactions, but waits until the Go bit in this register is set again before proceeding with any segment transfer.

GO – <1>. This bit is set by the host processor to tell the DRB32 microcontroller to start a segment transfer. The bit is cleared by the DRB32 microcontroller to indicate that the microcontroller is not transferring data at this time.

PDGSEG – <2> – Pending Segment. This bit is set by the host processor to tell the DRB32 microcontroller that a segment transfer is queued up waiting for the current segment transfer to be done. This bit is cleared by the DRB32 microcontroller when the NEXT segment transfer is started, indicating to the host processor that a new set of NEXT registers can be loaded.

If PDGSEG is set at the same time that Go bit is off, the segment transfer is started using the NEXT Registers. When all the NEXT Registers have been put into the CURRENT Registers and the segment transfer has been started, the Go bit is set (to indicate that a transfer is in progress) and the PDGSEG is cleared (to indicate that the NEXT Registers can be loaded).

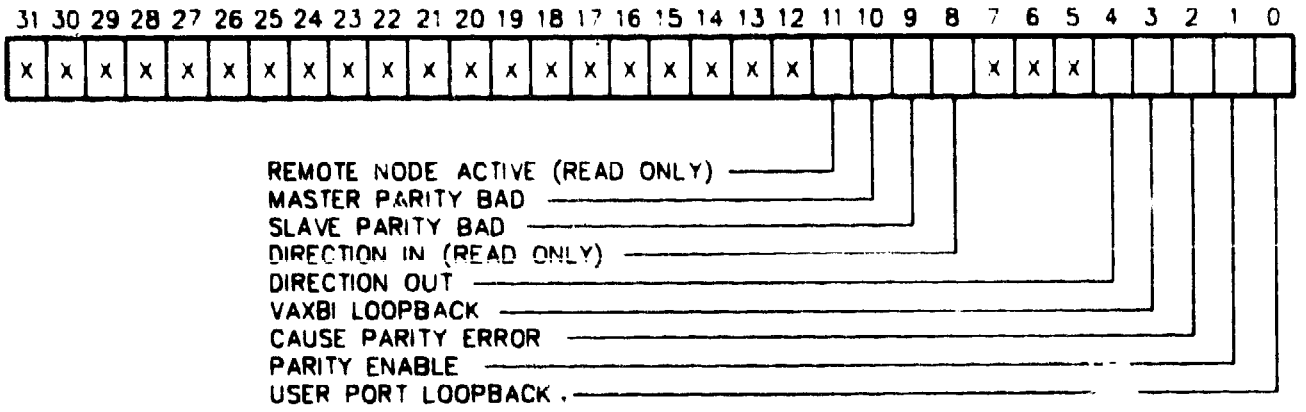
The host processor should not set up another segment transfer until this bit is cleared by the DRB32 microcontroller.

The PNDSEG bit remains set if an error condition occurs while the DRB32 is transferring data to or from the user device. The Go bit is automatically cleared under this condition, indicating that the data transfer has stopped. The PNDSEG bit remains in the state it was in when the error occurred, and this bit must be cleared by software before another data transfer is started.

3.2.4 PARSTR—Parallel Port Setup and Test Register

The Parallel Port Setup and Test Register is loaded by the host processor with information about the configuration and format of the current data transfer. This register also has diagnostic functions.

VAXBI : bb+0164 R/W



LB – <0> User Port Loopback. This bit is set by the host processor to indicate that the DRB32 is in **Userport Loopback mode**. In this mode, a longword of data written to either the I/O Data or Control Register can read back from the I/O Data or Control Register without disturbing the DRB32 output port. This bit is set to 0 by the power-up sequence, after the DRB32 module has passed power-up self-test.

When the User Port Loopback bit in the DRB32 Parallel Port Setup and Test Register is set, the DRB32 user port I/O lines are automatically put into tri-state float, so that no data is asserted on these lines.

NOTE

Loopback is used during self-test. For this reason if LB is set, expect the following to be true. If you write to the IODAT, you will find that the HIGH word (bits <31:16>) and the LOW word (bits <15:0>) have

been swapped. The OUTPUT <15:8> of the IOCTL register will be looped around to the INPUT <7:0> of the IOCTL.

PE - <1> Parity Enable. This bit is set and cleared by the host processor to set the parallel port parity mode. When the Parity Enable bit is set, the parallel port uses byte parity from the user port to generate VAXBI data parity, and byte parity from the map registers in the CB RAM to generate VAXBI address parity. When the Parity Enable bit is cleared, the BCAI internally generates VAXBI parity for data and address presented to it, without using the user or CB RAM parity bits. This bit is set to 0 by the power-up sequence after the DRB32 module has passed power-up self-test.

CPE - <2> Cause Parity Error. This bit is set by the host processor to deliberately create errors on the DRB32 internal parity bus. This is a diagnostic function that is only used by Digital diagnostic software. This bit is set to "0" by the power-up sequence after the DRB32 module has passed power-up self-test.

BILB - <3> VAXBI Loopback. This bit is set by the host processor to indicate that the DRB32 is in **VAXBI Loopback mode**. This mode is a hardware diagnostic mode and is not used in normal operation. In this mode, all VAXBI transactions issued by the DRB32 are VAXBI loopback transactions to the DRB32 slave port, if the address issued by the transaction is recognized by the DRB32. If the address is not recognized by the DRB32, the slave port returns a NO-ACK error code in the DRB32 Error Register. This mode is a hardware diagnostic mode and is not used in normal operation.

When the VAXBI loopback bit is set, block transfers, using the DRB32 master port (which are initiated by setting the Go bit in the PARCSR), transfer only the first longword in the first octaword buffer.

DIR O - <4>. This bit is set by the host processor to indicate the direction of data transfer to the user. When it is set, data is transferred TO the user; when cleared, data is transferred FROM the user.

DIR I - <8>. This bit indicates the direction of data transfer set by the user device. When this bit is set, the user device is transferring data TO the DRB32. When the bit is cleared, data is being transferred FROM the DRB32 to the user device.

Both ends of the DRB32 bus must agree on the direction of data transfer for the transfer to take place. Therefore, all user devices that interface to the DRB32 must only enable the tri-state drivers for D<31:0> and P<3:0>, if the DRB32 has the DIR OUT H signal asserted and the DIR IN H signal deasserted. See Table 3-4.

Table 3-4 Direction of Data Transfer

DIR OUT H	DIR IN H	Result	Note
L	L	Tri-state	Neither device transferring data
L	H	Tri-state	This device receiving data
H	L	Enable*	This device sending data
H	H	Tri-state	Both devices want to send, but do not agree on direction

* All D<31:0> and P<3:0> drivers are enabled in this state, no matter which one of the three possible data transfer widths is being used.

Slave Parity Bad – <9>. (READ only) This bit is set when either the I/O Data Register or the I/O Control Register is read, and the parity lines associated with the READ data indicate that parity is incorrect. This check is performed regardless of the state of the Parity Enable bit (bit <1>) in the PARSTR). If the user device is not generating parity information on either the data bus or the input control bus, this bit has no meaning and should be ignored. If the user device is generating parity information on the data bus or the control bus, this bit should be checked immediately after the IODAT or IOCTL register is read, to determine if a parity error has occurred. The bit is cleared when either of the IODAT or IOCTL registers is read, and the parity is correct.

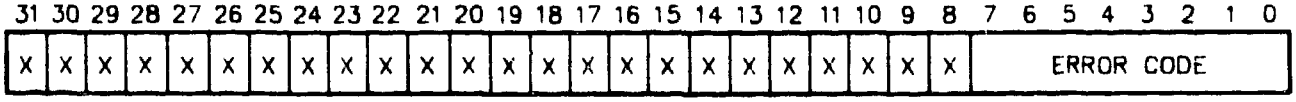
Master Parity Bad – <10>. (READ only) This bit is set when a parity error occurs (in data or address) during an octaword transaction initiated by a DMA transfer. An interrupt is not generated by setting this bit. However, the bad parity causes a bad EV code to be returned for the transaction, which sets the Error bit <3> in the DRBIFR, and that register generates an interrupt if the Error Enable bit is set in the DRBIE. This bit is cleared when a DMA transfer is initiated by setting the Go bit in PARCSR.

User Node Active – <11>. (READ only) This bit indicates the state of the user device by showing the state of the NODE ACTIVE IN H signal. When this bit is high, the user device is active and should be capable of transferring data. When this bit is low, the user device is inactive, and no attempt should be made to start a data transfer. Node inactivity can be the result of many conditions: two examples are power-down or failed self-test. The DRB32 should not enable any of its drivers until User Node Active is asserted.

3.2.5 ERRREG—Error Register

The DRB32 microcontroller loads this register when an error condition occurs. These bits remain until the next error condition occurs.

VAXBI : bb+0168 R



BU-2640

ERCODE – <7:0> Error Code. When an error occurs, the type of error is indicated by the contents of this byte. Error codes are shown in Table 3-5.

Table 3-5 ERRREG Error Codes

Error Code (HEX)	Meaning
1	Error during DMA transaction
9	Parity error on first map register or CURBOFF READ
A	Parity error on map register READ
B	Top of map table reached and byte count not equal to 0 (that is, you ran out of map registers)
11	Abort occurred during DMA transfer
12	VAXBI STOP command received
13	Illegal microcode address (microcode got lost)
14	VAXBI slave decode error (i.e., a VAXBI transaction that the microcode doesn't support was received)

The values are in hex. The values are valid only when the Error bit in the DRB32 Interrupt Flag Register is set. "Error during DMA transmission" (Error 1) means a VAXBI error occurred while the DRB32 microcode was performing a VAXBI transaction. To find more detail, look in the BIIC bus error register (bb+8) for both the master node (the DRB32) and the slave node (generally the memory). Some "Error during DMA transmission" errors may be caused by parity errors. Parity errors are indicated by two bits in the Parallel Port Setup and Test Register and might be caused by incorrect user-supplied parity.

The parity errors (Errors 9 and A) on first map register READ or subsequent map register READ mean either that data was stored in the map registers with bad parity, or the map register location is broken. There are two different sources of this error because the microcode fetches map registers in two places:

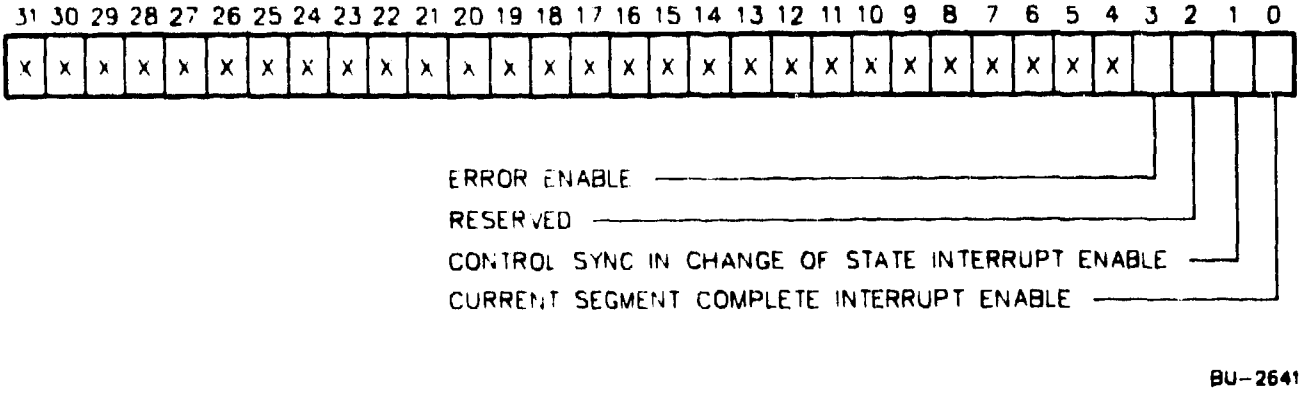
- When a DMA transaction starts, the microcode generates the first physical address from the map register and from the contents of CURBOFF
- The microcode routine that deals with page crossings fetches the next map register contents, and that becomes the next physical address.

Top of map register reached and byte count not equal 0 (Error B) is considered an error. This indicates that either the map register pointers in the hardware are broken, or a programming error was made (the programmer didn't allocate and set up enough map registers to map a buffer of the size indicated by the byte count register). See the *DRB32 Hardware Installation Guide* for more information on error codes and troubleshooting.

3.2.6 DRBIE—DRB32 Interrupt Enable Register

This register controls which events allow the DRB32 to send an interrupt to a VAXBI node.

VAXBI : bb+016C R/W



BU-2641

CSCIE - <0> - Current Segment Complete Interrupt Enable. When this bit is set, it allows the DRB32 to interrupt whenever an DRB32 segment transaction is stopped (that is, the Go bit in the PARCSR Register is cleared), either by completion or by error.

CINCOSIE - <1> - Control SYNC IN Change Of State Interrupt Enable. When this bit is set, the DRB32 issues an interrupt when the Control SYNC IN line changes state.

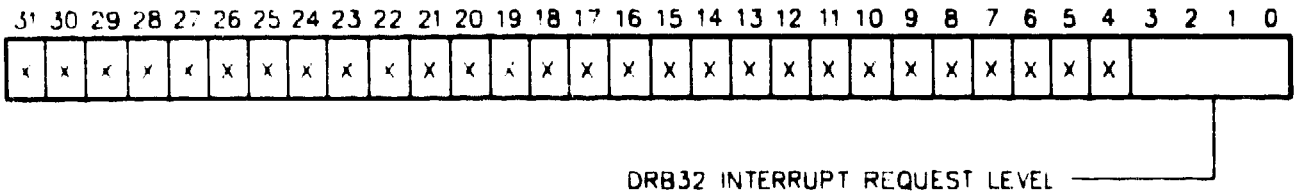
T11BINTIE - <2> - RESERVED.

ERRIE - <3> - Error Enable. If this bit is set, the DRB32 issues an interrupt when the DRB32 microcontroller detects an error in its own operation, or when a VAXBI error occurs on a DRB32-generated transaction.

3.2.7 DRBIRL—DRB32 Interrupt Request Level Register

This register is written with the DRB32 VAXBI interrupt level coded into bits <1:0>. When this register is read, the bit position of the interrupt level is read.

VAXBI + bb+0170 R W



BU-2642

Table 3-6 shows the DRB32 interrupt levels.

Table 3-6 DRB32 Interrupt Levels

DRBIRL	Interrupt Level	DRBIRL WRITE	DRBIRL READ
0	BR 4 (VAX IPL 14 hex)	0 (0000 bin)	1 (0001 bin)
1	BR 5 (VAX IPL 15 hex)	1 (0001 bin)	2 (0010 bin)
2	BR 6 (VAX IPL 16 hex)	2 (0010 bin)	4 (0100 bin)
3	BR 7 (VAX IPL 17 hex)	3 (0011 bin)	8 (1000 bin)

3.2.8 DRBIFR—DRB32 Interrupt Flag Register

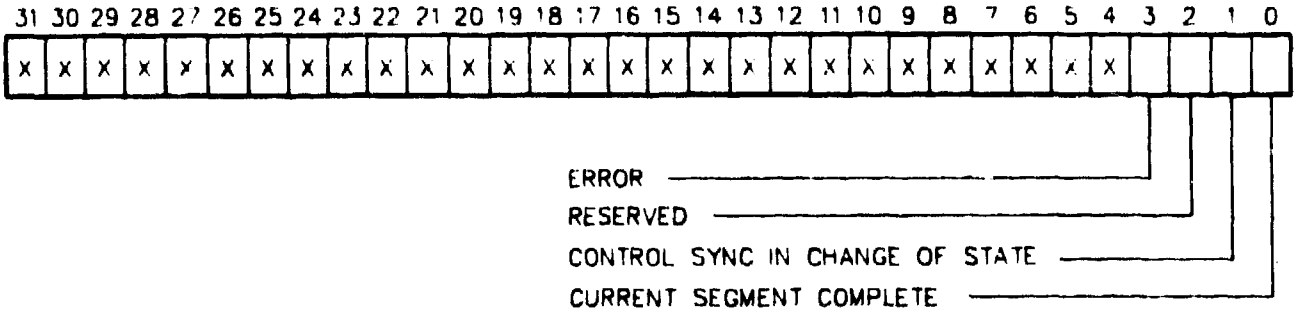
This register indicates which events caused the DRB32 to issue an interrupt transaction to a VAXBI node. Since VAXBI READ transactions are required to have no side effects, the following events must occur:

1. The DRBIFR register must be written to update its contents. The DRBIFR register only needs to be written if you wish to examine the present condition of the INTERRUPT FLAG REGISTER.

2. The DRBIFR has been written; this register now contains the correct information. You can read this register as often as needed. A read of the the DRBIFR register will not affect the contents of the DRBIFR register. The DRBIFR operation has no effect on the actual interrupt operation. Interrupts will generate VAXBI interrupt transactions as soon as they occur, if the appropriate Interrupt Enable bit is set.

In a multiprocessor system, the DRBIFR WRITE and READ must be an automatic operation to ensure that no interrupts are lost by consecutive WRITEs to the DRBIFR (by different processors) before a DRBIFR READ from either processor.

VAXBI : bb+0174 R/W WRITE TO FREEZE THEN READ



BU-2643

CSC – <0> – Current Segment Complete Interrupt. When this bit is set, the DRB32 segment transfer is done, and the NEXT register set can be loaded. If the CSCIE bit is set in the DRBIE Register, a VAXBI interrupt is generated.

CINCOS – <1> – Control SYNC IN Change Of State. The DRB32 sets this bit when the Control SYNC IN line changes state. If the CINCOSIE bit in the DRBIE Register is set, a VAXBI interrupt is generated. This bit is cleared after the DRB32 Interrupt Flag Register is read.

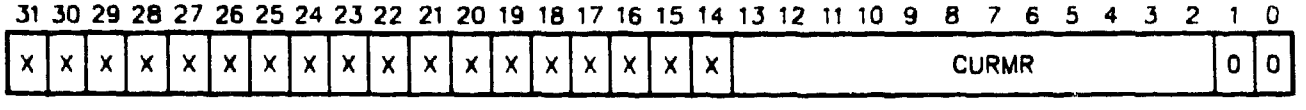
T11BINT – <2>. RESERVED

ERROR – <3>. This bit is set by the microcontroller to indicate that an error has occurred in the current segment transfer. The code in the Error Register indicates which error has occurred. The parallel port treats error conditions the same as ABORTs; all registers and the CB RAM are left intact, the parallel port returns to its Wait state and responds to VAXBI slave transactions after completing the current VAXBI master transaction. This bit is cleared after the DRB32 Interrupt Flag Register is read.

3.2.9 CURMR—Current Map Pointer Register

This register contains the CB RAM address of the map register being used for the physical address of the first page of the current data transfer.

VAXBI : bb+0178 R/W



BU-2644

This register is loaded at the start of a segment transfer with the address of the bottom of the map register table. The bottom two bits of this register (<1:0>) are always set to zero.

When a page boundary is crossed during a segment transfer, the DRB32 microcontroller compares the CURMR to the CURTOMR Register. If they are equal, the segment transfer is complete and the DRB32 stops transferring data. If they are not equal, the pointer is incremented, and then used to get the map register of the next page.

If the CURMR value reaches the top of the CB RAM address space and CURBLFT is not equal to zero, the following events occur.

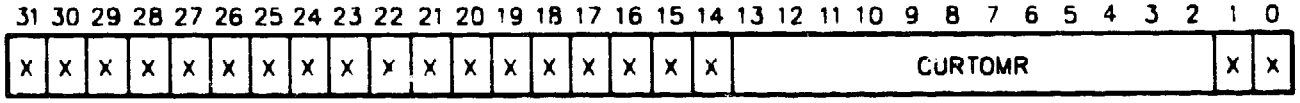
- The transaction is terminated.
- The BCAI octaword buffer pointer is loaded into the BCAI Octaword Buffer Pointer Register.
- An error code (B) is loaded into ERRREG.
- The Error bit is set in the DRBIFR Register This causes a VAXBI interrupt, if the Error Enable bit is set in DRBIE.

When data is being transferred from the user device to the DRB32, user data can be left in the BCAI octaword registers if the transaction is terminated as described above. The number of bytes left in the octaword registers agrees with the lower four bits of the BCAI Octaword Buffer Pointer Register.

3.2.10 CURTOMR—Current Top of Map Register Area Pointer

This register contains the location of the top of the current map register table in the CB RAM. The bottom two bits of this register (<1:0>) are always ignored by the microcontroller when it compares CURTOMR to CURMR. However, these bits can be written and read by the host processor.

VAXBI : bb+017C

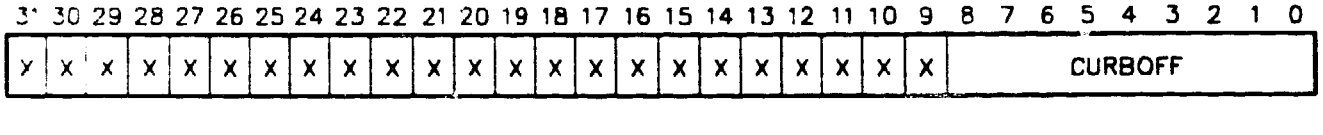


BU-2645

3.2.11 CURBOFF—Current Address Byte Offset Register

This register represents the offset from the start of the page to the start of the user's buffer in that page.

VAXBI : bb+0180 R/W



BU-2646

CURBOFF is concatenated on the bottom of the map register pointed to by the Current Map Register Pointer to form the physical address. The contents of this register can be read by reading the Physical Address Register.

3.2.12 CURBLFT—Current Number of Bytes Left in Segment

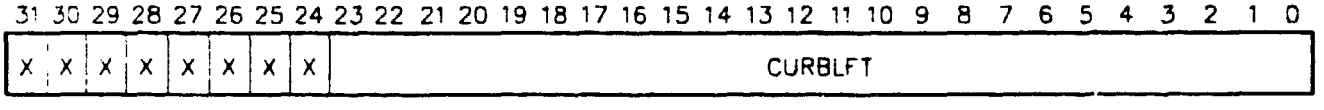
This register is loaded at the start of a data transfer with the number of bytes to be transferred. If the transfer aborts (or is stopped for any reason), CURBLFT contains the number of bytes NOT transferred.

(WRITE) CURBLFT is loaded by the host processor with the number of bytes to be transferred in the current segment transfer. For DRB32 WRITE transactions in which data is transferred to the user port, the CURBLFT Register is decremented by the number of bytes transferred each time a user port transaction is completed.

(READ) Reading this register indicates the number of bytes left (in the current segment) to be transferred to the user device or VAXBI. This lets you determine how far a segment transfer has progressed if an error condition occurs in the middle of that transfer. For DRB32 Read transactions in which data is transferred from the user port, the CURBLFT Register is decremented by the number of bytes transferred each time a VAXBI transaction is completed.

When this register is equal to zero, the segment has been completely transferred. If the Pending Segment bit in that register is set, the DRB32 loads the NEXT registers into the CURRENT registers, clears the Pending Segment bit in the Parallel Port Control and Status Register, and proceeds with the next segment transfer.

VAXBI : bb+0184 R/W



BU-2647

3.2.13 PHYSADDR—Physical Address Register

The Physical Address Register is the physical address (in VAXBI address space) of the start of the current page in the segment transfer. This register is READ-only.

This register is the result of the concatenation of the following:

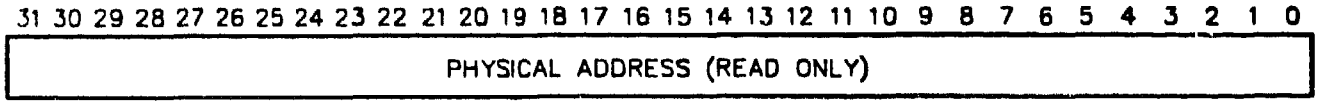
- The Transaction Length bits <31:30>, which are generated by the DRB32 microcontroller
- Bits <29:9> of the Map Register pointed to by the Current Map Register Pointer
- Bits <8:0> of the Current Byte Offset Register.

These three components of the physical address are available at different times because of the way the DRB32 microcontroller puts the physical address together.

1. The byte offset is available as soon as loaded by the user device.

2. The Current Map Register bits and the Transaction Length bits are available immediately after the transfer starts (the contents of bits <31:0> before a transfer starts are the last map register and the Transaction Length bits from the last DRB32 DMA transfer). The Byte Offset bits are set to zero after the transfer starts.

VAXBI : bb+0188 R

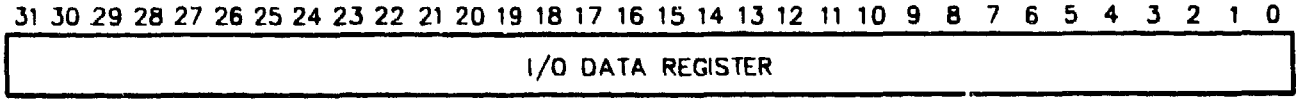


BU-2648

3.2.14 IODAT—I/O Data Register

This register contains the current state of the 32 parallel I/O port data lines. IODAT is used in programmed I/O. The microcontroller does not assert SYNC OUT H when the I/O Data Register is written. You can use a bit in the IOCTL Register as a flag bit to inform the DRB32 software that input data is stable (or has been read). However, this bit is not absolutely necessary, and its use is dependent on the application.

VAXBI : bb+018C R/W



BU-2649

When this register is read, its contents are the current state of the parallel I/O port data lines. The user device driving the data lines must be careful to present stable data while this register is being read.

The Slave Parity Bad bit (<9>) in the PARSTR is set if the parity on P<3:0> of the user data input lines does not agree with the data on D<31:0>. Ignore this bit if parity is not being used. To write data to a user device from the I/O Data Register, first enable the I/O data port output tri-state drivers. To do this, set the DIR OUT H bit in PARSTR. The user device must set the DIR IN H line in the same register.

Two things are indicated by the setting of these bits: 1) the DRB32 is going to send data to the user device and 2) the user device is not currently asserting data on the I/O data port. When these conditions are met, any WRITE to the I/O Data Register immediately asserts data on the I/O data lines.

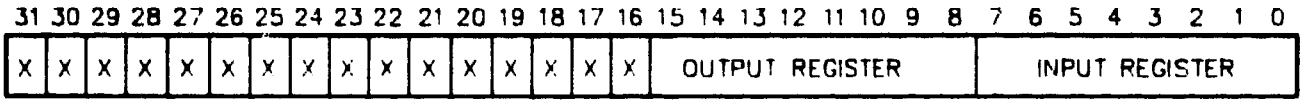
3.2.15 IOCTL—I/O Control Register

The contents of this register reflects the value of the eight inputs and outputs of the DRB32 user interface control lines. These lines can be used to communicate block transfer protocol to and from the user device. See Chapter 4 for further information on the user interface lines.

The IOCTL has eight READ-only input bits; each one reflects the state of the corresponding bit in the DRB32 control bus. The IOCTL has eight READ or WRITE output bits. They are set by writing into the appropriate byte in the I/O Control Register.

This register responds to parity errors by setting the Slave Parity Bad bit (<9>) in the PARSTR, if the parity on CTR P<0> from the user control input lines does not agree with the data on CTL D <7:0>. Ignore this bit if parity is not being used.

VAXBI : bb+0190 R/W

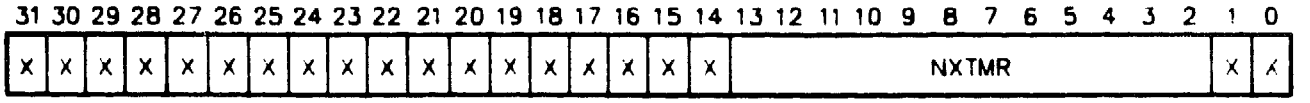


BU-2650

3.2.16 NXTMR—Next Map Register Pointer

The NXTMR contains the CB RAM address of the map register being used for the physical address of the first page of the Next data transfer.

VAXBI : bb+0194 R/W



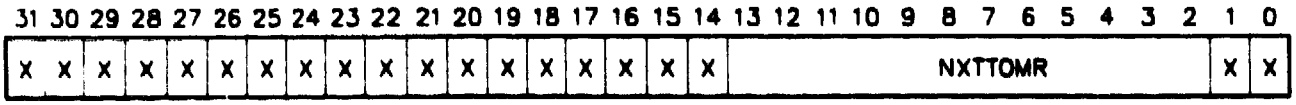
BU-2651

The contents of this register is loaded into the Current Map Register Pointer at the end of the current segment transfer if the Pending Segment bit in the Parallel Port Status and Control Register is set. The bottom two bits of this register (<1:0>) are always ignored by the microcontroller. However, the host processor can write and read them.

3.2.17 NXTTOMR—Next Top of Map Register Area Pointer

The NXTTOMR contains the location of the top of the Next map table in the CB RAM.

VAXBI : bb+0198 R/W



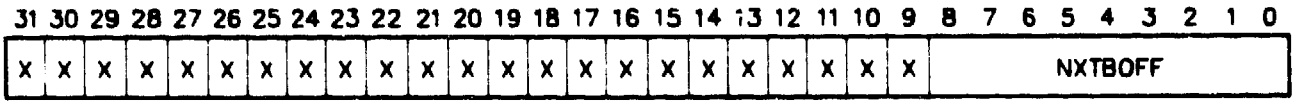
BU-2652

The contents of this register is loaded into the Top of Map Register Area Register at the end of the current segment transfer, if the Pending Segment bit in the Parallel Port Control and Status Register is set. The bottom two bits of this register (<1:0>) are always ignored by the microcontroller. However, the host processor can write and read these bits.

3.2.18 NXTBOFF—Next Address Byte Offset

The NXTBOFF is loaded with the byte offset of the next buffer.

VAXBI : bb+019C R/W



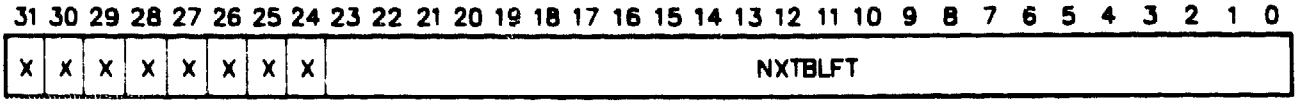
BU-2653

The contents of this register is loaded into the Current Byte Offset Register at the end of the current segment transfer if the Pending Segment bit in the Parallel Port Control and Status Register is set.

3.2.19 NXTBLFT—Next Number of Bytes Left in Segment

This register is loaded with the number of bytes in the Next buffer.

VAXBI : bb+01A0 R/W



BU-2654

The contents of this register is loaded into the Current Number of Bytes Left in Segment Register at the end of the current segment transfer if the Pending Segment bit in the Parallel Port Control and Status Register is set.

3.2.20 T11ICR

RESERVED

3.2.21 T11MDTR

RESERVED

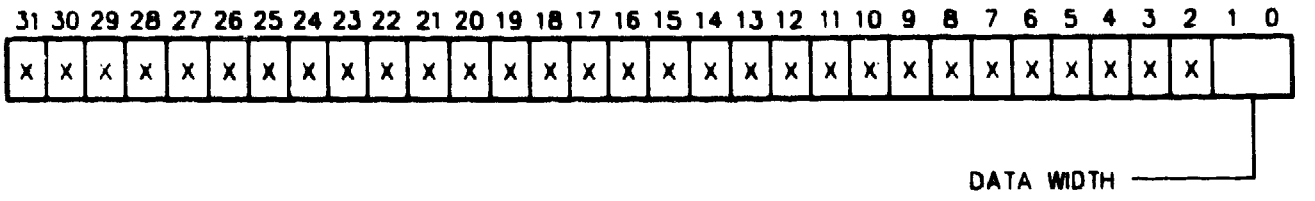
3.2.22 DPWR—Data Port Width Register

Use the Data Port Width Register to select the width of data presented and accepted at the user I/O data port. Data widths can be bytes (8 bits), words (16 bits), or longwords (32 bits).

When the DRB32 is reading or writing to or from the external device into VAXBI memory and the data path is under the control of the microcode while doing a block data transfer operation (DMA mode), the following results occur:

- In Word mode READs, the DRB32 packs two words from D<15:0> of the DRB bus into a single longword: the first word is placed in the low-order word of the longword.
- In Byte mode READs, the DRB32 packs four bytes from D<7:0> of the DRB bus into a single longword.
- When writing from VAXBI memory in Word or Byte mode, the DRB32 unpacks data and places the low-order word or byte onto the DRB32 bus before the high-order word or higher-order bytes.
- When a driver or similar program is writing or reading to or from the IODAT Register explicitly, every bit written to the IODAT Register goes out on the data bus. The user device must know that only the low 8 bits (or low 16 bits) are valid. The driver must also know to pack or unpack the data as it sends or receives it.

VAXBI : bb+01AC R/W



BU-2655

The codes for the data widths are shown in Table 3-7.

Table 3-7 Data Transfer Widths

Code <1:0>	Data Width	Data Lines
00	Longword (32 bits)	D<31:0>
01	Word (16 bits)	D<15:0>
10	Longword (32 bits)	D<31:0>
11	Byte (8 bits)	D<7:0>

3.2.23 BOBP

RESERVED

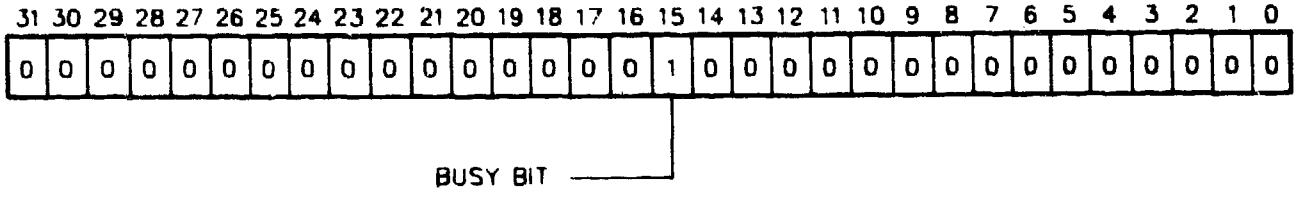
3.3 VAX CONSOLE REGISTER (RXCD)

The VAX Console Register (RXCD) is required for all VAXBI nodes. However, the RXCD Register is not used by the DRB32. This register is used in VAXBI systems as a console communications register. Since the DRB32 does not implement a VAXBI console, the self-test sets the Busy bit (<15>) in this register. See Table 3-8.

Table 3-8 VAX Console Register

VAXBI Address	Register Name
bb+0200	RXCD VAX Console Register

This register is R/W. The Busy bit is shown below.



BU-2656

3.4 USER-DEFINABLE REGISTERS (CB RAM)

The DRB32 supplies user-definable registers in the CB RAM. This area can be used for map registers and can be divided for setting up double buffering data transfers. They are shown in Table 3-9.

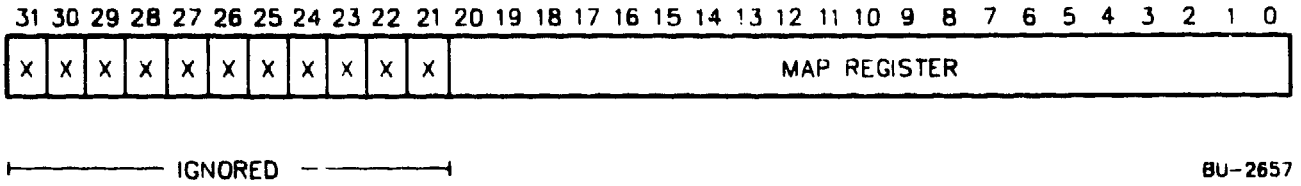
Table 3-9 User-Definable Registers

VAXBI Address	Register Name
bb+01D4	
	User-definable CB RAM area
bb+01FC	

3.4.1 Map Register

VAXBI : bb+0204 to bb+1FFC R/W

The locations in this area of the CB RAM normally include the map registers. When a page boundary is crossed during a segment transfer, the DRB32 microcontroller checks to see if the Current Map Register Pointer (CURMR) is equal to the Current Top of Map Register. If these registers do not agree, the microcontroller increments CURMR. The Map Register bits <20:0> are shifted up to <29:9> and zeros are put in <8:0> to generate the physical address in <29:0>. Two length bits are inserted in <31:30> before the physical address is sent out on the VAXBI.



The data to be put in this register is from system page tables and consists of VAX page frame numbers. The operating system or device driver must load page frame numbers into the map registers.


CHAPTER 4

CHAPTER 4
USER INTERFACE SIGNALS AND PROTOCOL

This chapter describes the connection of the DRB32 adapter to user devices and includes information on the following:

- Description of the DRB32 user interface
- Data path signals, timing, and protocol
- Control path signals, timing, and protocol
- Device status signals
- Optimizing data transfer speed.

See Appendix A in this manual for a list of the DRB32 module specifications.

4.1 USER INTERFACE DESCRIPTION

The DRB32 adapter transfers data to user devices through the **user interface,** which is the group of signals that comprises the data path between the DRB32 and a user device. The DRB32 user interface is designed to move data in large blocks from one system to another, with positive confirmation of data transmission and arrival.

The user interface has:

- A data path (32 data, 4 control, and 4 parity lines) for transmitting data from the DRB32 to the user device (or vice versa)
- A control path (eight protocol in each direction, and two control and two parity lines) for transmitting control signals and data
- Two device status signals.

The data path has data lines, control lines, and parity lines. The 32 data lines are half-duplex, with one parity bit per data byte. The control path also has data lines, control lines, and parity lines. The 20 (16 data, two control and two parity) lines of the control path allow you to generate a data transfer initiation and termination protocol appropriate to your application. Since the data lines are separate from the control lines, you can overlap control operations with data transfers.

The device status lines consist of two lines for transmitting status information about the readiness of the DRB32 or the user device to transmit data.

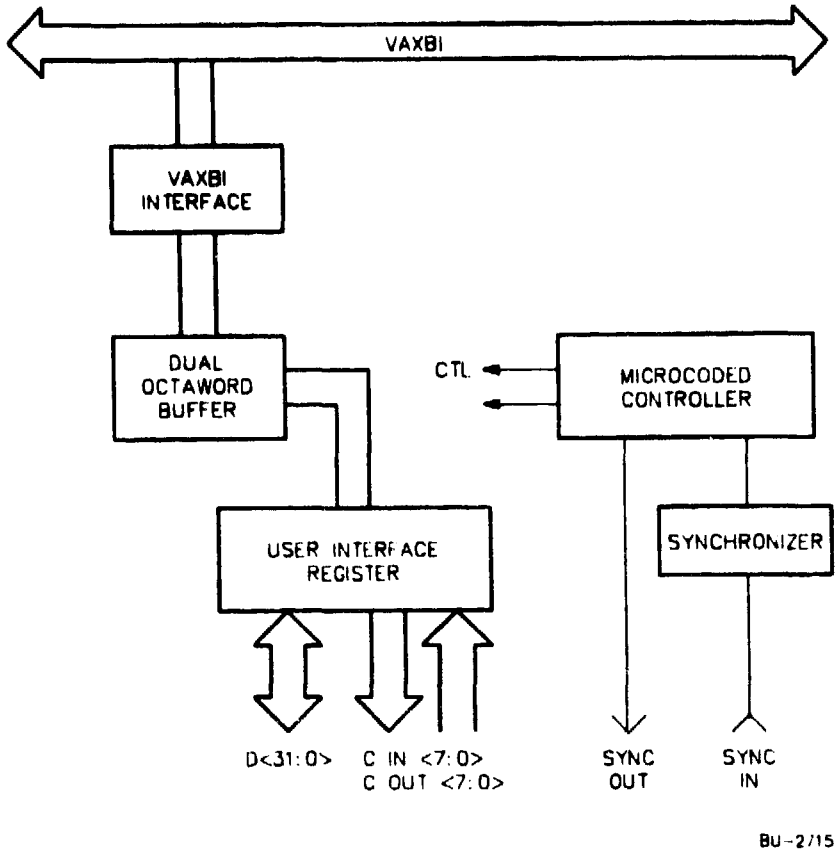


Figure 4-1 DRB32 Data Path Block Diagram

Figure 4-1 does not show the entire DRB32 module, but does show the essentials of the user interface data path. D<31:0> are the 32 lines of the user data path. There are also four parity bits associated with the data path and four control signals.

C IN <7:0> and C OUT <7:0> are the 16 lines (eight in each direction) of the control path. This path is separate from the data path and can be used as either an 8-bit, full-duplex bus, or as 16 separate control lines. The control path also has a parity line and a control line in each direction for a total of 20 lines.

Note that the SYNC IN signal has to be synchronized to the DRB32 clock before the microcontroller (which does all the work in the DRB32) can look at it. After synchronization, the SYNC IN signal goes directly to the microcontroller. The SYNC OUT signal is driven directly by the same controller.

The user interface IN lines on the DRB32 are connected to the OUT lines of the same type on the user device, as shown in Figure 4-2.

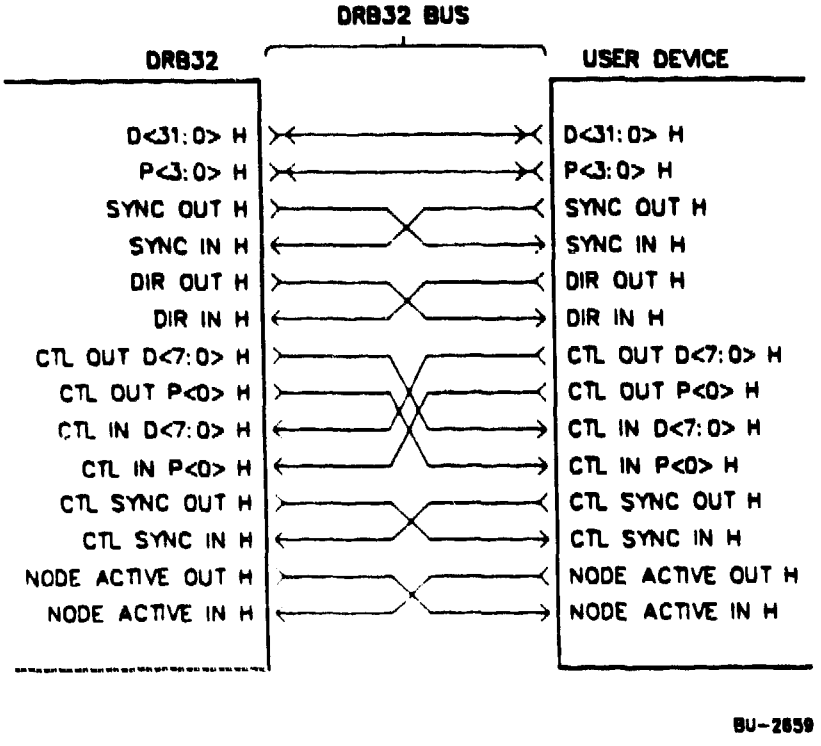


Figure 4-2 DRB32 Signal Connections

4.2 USER INTERFACE DATA PATH DESCRIPTION

The DRB32 user interface has a 32-bit, half-duplex path for transmitting data from the VAXBI to your user device, or vice versa. This data path consists of the following signals:

- Thirty-two data signals
- Four parity bits, one for each byte of data
- Four control signals.

These 40 signals are described in the sections below.

4.2.1 User Interface Data Path Data Signals

The data transfer path is 32 bits wide and has one parity bit associated with each byte of data, for a total of four parity bits. Data can be transferred in three widths:

- Longword (32 data bits plus 4 parity bits)
- Word (16 data bits plus 2 parity bits)
- Byte (8 data bits plus 1 parity bit)

There are no "data width" control lines. Data width information must either be communicated over the regular control lines, or is intrinsic in the application, with the use of the DPWR register.

The data and parity lines are half-duplex and tri-state, meaning that data transfer can occur in either direction over the same data lines, but only one direction at a time. Transfer direction is determined by the DIR IN/OUT lines.

The data path data signals are described in Table 4-1.

Table 4-1 Data Path Data Signals

Signal	Description
D<31:0>	Bidirectional lines used for all data transactions. Longword transfers use D<31:0>, word transfers use D<15:0>, and byte transfers use D<7:0>. The transmitting device must keep the appropriate D lines stable, starting before it asserts its SYNC OUT H line until the receiving device asserts the transmitting device's SYNC IN line.
P<3:0>	Bidirectional lines carrying parity information for D<31:0> lines. Timing for P<3:0> stability is the same as the timing for D<31:0>. Parity information is arranged on a parity-bit-per-data-byte basis. The sum of the data byte plus the parity bit associated with it is odd. Parity is only checked (and need only be generated) on used bytes, other than longword data transfer widths. See the table below:
Data	**Parity**
D<31:24>	P<3>
D<23:16>	P<2>
D<15:8>	P<1>
D<7:0>	P<0>

4.2.2 Data Path Control Signals

The DRB32 user path has two pairs of data path control signals:

- DIR IN and DIR OUT
- SYNC IN and SYNC OUT

The DIR IN/OUT signals determine data flow direction. The SYNC IN/OUT signals provide positive confirmation of data availability and acceptance.

4.2.2.1 DIR IN/OUT - The direction of data transfer must be determined before a transfer starts. The DIR IN and DIR OUT signals are used to determine data transfer direction. These bits appear in the PARSTR register.

The DRB32's assertion of DIR OUT H indicates that data is to be transferred from the DRB32 to the user device. The user device's assertion of DIR IN H to the DRB32 indicates that data is to be transferred from the user device to the DRB32.

Both ends of the user interface must agree on the direction of data transfer for the transfer to take place. Therefore, all devices that interface to the DRB32 must enable the tri-state drivers for the D<31:0> data signals and the P<3:0> parity signals only if DIR OUT H is asserted and DIR IN H is deasserted. See the Table 4-2.

Table 4-2 Direction of Data Transfer

DIR OUT H	DIR IN H	Result	Note
L	L	Tri-state	Neither device transferring data
L	H	Tri-state	This device receiving data
H	L	Enable*	This device sending data
H	H	Tri-state	Both devices want to send but do not agree on direction

* All D<31:0> and P<3:0> drivers are enabled in this state, no matter which one of the three possible data transfer widths is being used.

4.2.2.2 SYNC IN/OUT When Transmitting Data - SYNC OUT is asserted by the transmitting device after data is asserted on the data lines. SYNC OUT assertion by the transmitting device indicates to the receiving device that the data is good at this time, and can be clocked into a register.

SYNC OUT is deasserted by the transmitting device when SYNC IN assertion is received, which indicates that the receiving device has accepted the data on D<31:0> and is ready for new data.

SYNC OUT is normally connected to the SYNC IN receiver on the other device.

4.2.2.3 SYNC IN/OUT When Receiving Data - The receiving device uses the rising edge of SYNC IN to clock the data into input registers.

When the receiving device has accepted the data at its input, it asserts SYNC OUT to indicate to the transmitting device that data has been accepted and the receiving device is ready for new data.

SYNC OUT is deasserted by the receiving device when SYNC IN is deasserted by the transmitting device, indicating that it is getting ready to send a new data cycle.

The data path control signals are described in Table 4-3.

Table 4-3 Data Path Control Signals

Signal	Description
SYNC OUT H	Devices assert this signal. This signal has different meanings for the transmitting device and the receiving device. The receiving device asserts this signal to indicate to the transmitting device that it has accepted the data on D<31:0>, and the transmitting device can remove it from the bus. The transmitting device asserts this signal to indicate to the receiving device that data is stable on D<31:0>.
SYNC IN H	Devices receive the assertion of this signal. This signal has different meanings for the transmitting device and the receiving device. Assertion of this signal to the transmitting device indicates to it that the receiving device has accepted the data. Assertion of this signal to the receiving device indicates to it that the data is stable on D<31:0>. The rising edge of SYNC IN H can be used to clock data into an input register in the receiving device.
DIR OUT H	Devices assert this signal. It defines the direction of data transfer (along with DIR IN H) and is a level. DIR OUT H assertion by the local device indicates that data is to be transferred to the user device. DIR OUT H and DIR IN H determine which end of the DRB32 enables its tri-state drivers for the D<31:0> and P<3:0>. See table in the Parallel Port Setup And Test register description.
DIR IN H	Devices receive the assertion of this signal. This input defines data transfer direction along with DIR OUT H. DIR IN H assertion indicates that data is to be transferred to the DRB32 from the user device. DIR IN H and DIR OUT H determine which end of the DRB32 enables its tri-state drivers for D<31:0> and P<3:0>. See table in the Parallel Port Setup And Test register description.

4.2.3 Data Transfer Protocol and Timing

The DRB32 uses two signals to indicate data presence and acceptance: SYNC OUT and SYNC IN. These two signals are used for both sending and receiving data. SYNC OUT of the DRB32 is normally connected to SYNC IN of the user device and SYNC OUT of the user device is connected to SYNC IN of the DRB32 as shown in the following diagram:

```
DRB32                        USER DEVICE

SYNC OUT-------------> SYNC IN
SYNC IN <------------- SYNC OUT
```

4.2.3.1 Data Transfer from DRB32 to User Device - The following sequence of events occurs when data is transferred from the DRB32 to the user device. Assignment of interface mastership is not included in this description. Although this is written as a DRB32-to-user-device example, transfers in the opposite direction work similarly.

```
        DRB32                                   USER DEVICE
        -----                                   -----------
                                1.
Asserts DIR OUT H, requesting
permission to send data.        ------------>2.
                                             Receives DRB32's DIR OUT H
                                             as DIR IN H.  deasserts
                                             DIR OUT H, indicating it
                                             will receive.
                                3.<-----------
Receives User's DIR OUT H as
DIR IN H.

   a.  Enables drivers
   b.  Asserts data on D<31:0> lines
   c.  Waits 100 ns, asserts
       SYNC OUT H               ------------>4.
                                             Receives assertion of
                                             DRB32's SYNC OUT H as
                                             SYNC IN H.

                                               a. Clocks data into
                                                  register
                                               b. Asserts SYNC OUT H
                                5.<-----------
Receives assertion of User's
SYNC OUT H as SYNC IN H.

  a. Puts next data on D<31:0>
  b. Deasserts SYNC OUT H       ------------>6.
                                             Receives deassertion of
                                             DRB32's SYNC OUT H as
                                             deasserted SYNC IN H.
                                             deasserts SYNC OUT H.
                                7.<-----------
Receives deassertion of
user's SYNC OUT H as the
deassertion of SYNC IN H.
Asserts SYNC OUT H 100 ns
after D<31:0> asserted.         ------------>8.
                                             Return to Step 4 until all
                                             data is transferred.
```

4.2.3.2 Send Data Timing - Figure 4-3 is a timing diagram from the point of view of a device sending data at an end of the DRB32 user interface. Although the following description is presented as if the DRB32 is the sending device, the user device behaves in exactly the same way if it is the sending device.

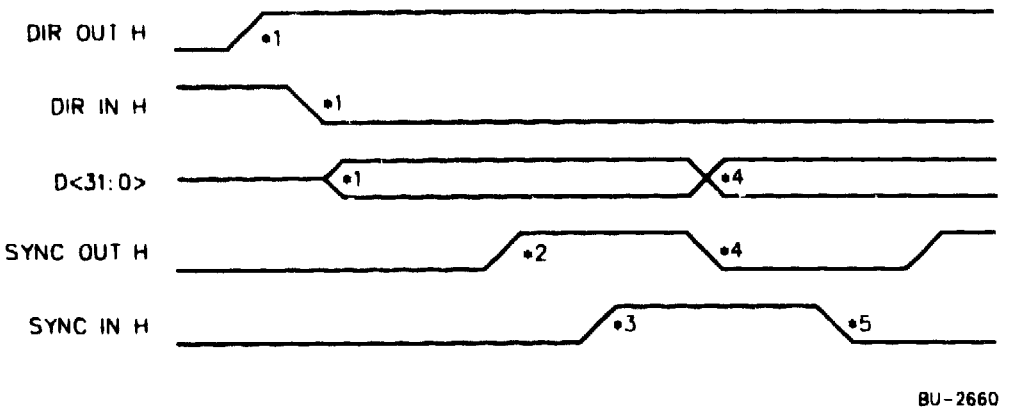


Figure 4-3 Send Data Timing Diagram

*1 Before the data transfer can start, the DRB32 and the user device must agree on the transfer direction. The DRB32 asserts DIR OUT H to inform the user device that DRB32 intends to send (assert data on the tri-state bus). The user device deasserts DIR IN H to inform the DRB32 that it is going to receive data. The D<31:0> lines are only enabled after both devices agree on data direction.

*2 After the longword of data is asserted on D<31:0> by the DRB32, the DRB32 asserts SYNC OUT H, informing the user device that data is valid.

*3 Some time (no maximum here) after the user device receives SYNC OUT H from the DRB32, it accepts the data on the D<31:0> lines. To indicate that it has captured data, the user device asserts the DRB32's SYNC IN H.

*4 When the DRB32 device receives the SYNC IN H assertion from the user device, it is free to put new data on D<31:0> and deassert SYNC OUT H in preparation for the next data cycle.

*5 When the user device receives the SYNC OUT H deassertion from the DRB32, the user device deasserts SYNC IN H to inform the DRB32 that it is ready for the next data cycle.

4.2.3.3 Receive Data Timing – Figure 4-4 is a timing diagram from the point of view of a device on one end of the DRB32 bus receiving data. Although the following description is presented as if the DRB32 is the receiving device, the user device behaves in exactly the same way if it is the receiving device.

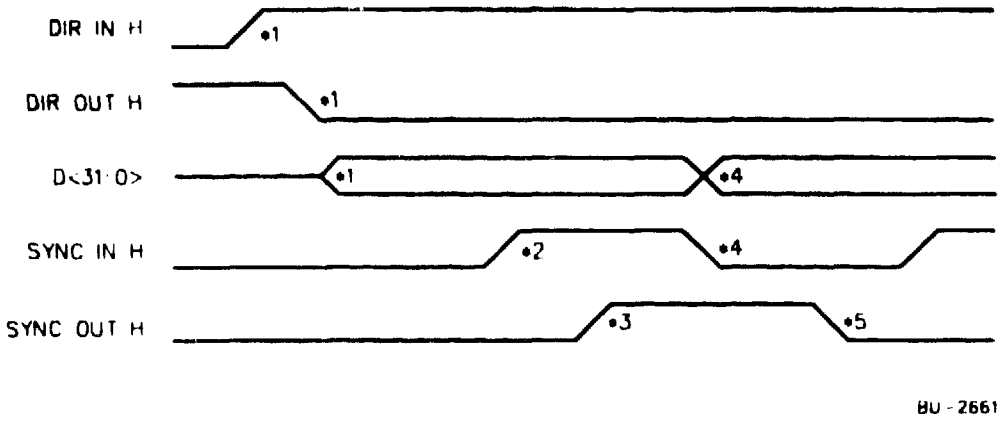


Figure 4-4 Receive Data Timing Diagram

*1 Before the data transfer can start, the DRB32 and user device have to agree on transfer direction. The DRB32 receives DIR IN H from the user device, indicating that user device intends to send (assert data on the tri-state bus). The DRB32 deasserts DIR OUT H to inform the user device that it will receive data. The DRB32's D<31:0> lines are tri-stated to allow the user device to drive the bus.

*2 Data is stable on D<31:0> at least 100 ns before the DRB32 receives SYNC IN H. The DRB32 uses the rising edge of SYNC IN H to clock D<31:0> into a register.

*3 When the DRB32 has accepted the data, it asserts SYNC OUT H, to inform the user device that it can assert new data on the bus.

*4 The user device deasserts SYNC IN H and asserts (if available) new data on D<31:0>. This indicates that the current cycle is over and that it is preparing to start a new cycle.

*5 When the user receives the deassertion of SYNC OUT H from the DRB32, the user device asserts SYNC IN H to start the next cycle. The assertion of SYNC IN H means that data is stable on the bus, and the DRB32 clocks it in.

4.3 USER INTERFACE CONTROL PATH DESCRIPTION

The DRB32 user interface control path has a total of 20 lines. These include 16 data lines, (eight in each direction), therefore allowing either end of the DRB32 to send a message to the other end without arbitrating for the control bus. Each 8-bit bus has 1 parity bit. There are also two control lines (CTL SYNC IN H and CTL SYNC OUT H) that are used to inform the other device that the control data lines have changed. The user interface control path hardware does not define protocol. Rather, it provides control lines and allows you to define those lines in the way most appropriate to your application.

4.3.1 User Interface Control Path Data Signals

The DRB32 interface control data signals can be used in two ways, depending on the application:

- Bus mode
- Line mode

In Bus mode, the control lines can be used as a control data path, passing 8-bit control messages in both directions between the two devices. In Line mode, the control lines are used as 16 control lines, 8 in and 8 out. The choice of independent lines or control data bus is user-determined.

The control data path consists of 18 lines: two 8-bit data/1-bit parity buses, one IN and one OUT. Odd parity is used. Because control data can be sent and received simultaneously, no arbitration is necessary to use the control data bus.

The control path data signals are described in Table 4-4.

Table 4-4 Control Path Data Signals

Signal	Description
CTL OUT D<7:0> H	These lines are direct outputs from the DRB32 to the user device. They are user-definable.
CTL OUT P<0> H	Parity for CTL OUT D<7:0> H. The sum of CTL OUT D<7:0> H plus the parity bit is odd. Timing is the same as CTL OUT D<7:0> H.
CTL IN D<7:0> H	These lines are direct inputs from the user device to the DRB32. They are user-definable.
CTL IN P<0> H	Parity for CTL IN D<7:0> H. The sum of CTL IN D<7:0> H plus the parity bit is odd. Timing is the same as CTL IN D<7:0> H.

4.3.2 User Interface Control Path Control Signals

The user interface control path has only one pair of control signals: CTL SYNC IN H and CTL SYNC OUT H. CTL SYNC OUT H is asserted by the DRB32 after any of the CTL OUT D<7:0> H lines are changed, to indicate to the user device that the change has occurred. CTL SYNC IN H is asserted by the user device to indicate to the DRB32 that the user device has changed CTL IN<7:0> signals.

The control path control signals are described Table 4-5.

Table 4-5 Control Path Control Signals

Signal	Description
CTL SYNC OUT H	Is asserted by the DRB32 for a minimum of 200 ns, whenever DRB32 changes CTL OUT<7:0>
	Normally a change in CTL SYNC OUT H interrupts the controlling process in the user device, so the user process can look at CTL OUT<7:0>
CTL SYNC IN H	Assertion of this signal for a minimum of 150 ns indicates to the DRB32 that the user device has changed CTL IN<7:0>
	This line is normally used to interrupt the controlling process in the DRB32 so that the DRB32 process can look at CTL IN<7:0>
	The CTL SYNC IN H signal must be asserted to update the IOCTL. Any changes to the IOCTL DATA lines (CTL IN<7:0> must be accompanied by a CTL SYNC IN H, to be latched into the IOCTL register.
	Change Of State interrupt must be enabled in the DRBIE register in order for CTL SYNC IN H to cause an interrupt.

4.3.3 Control Path Timing and Protocol

The following diagrams show the control path timing and protocol for the two control signals, CTL SYNC OUT H and CTL SYNC IN H.

4.3.3.1 CTL SYNC OUT H Timing – Figure 4-5 shows the timing for the CTL SYNC OUT H signal.

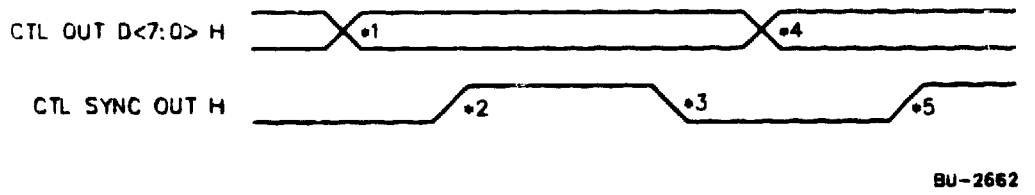


Figure 4-5 CTL SYNC OUT H Timing Diagram

*1 One or more of the CTL OUT D lines change state.

*2 After the CTL OUT D lines change, CTL SYNC OUT H is asserted. This informs the remote controlling process that one or more of the CTL OUT D lines have changed.

*3 After CTL SYNC OUT H is asserted, it is deasserted in order to be ready for the next change in the CTL OUT D lines.

*4 After CTL SYNC OUT H is deasserted, the CTL OUT D lines are permitted to change.

*5 CTL SYNC OUT H is asserted to inform the remote device that one or more lines in CTL OUT D have changed.

4.3.3.2 CTL SYNC IN H Timing – Figure 4-6 shows the timing for the CTL SYNC IN H signal.

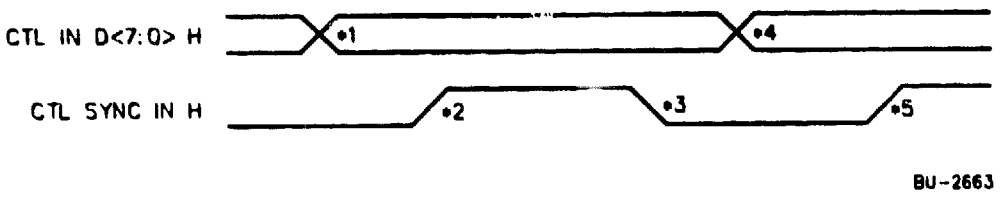


Figure 4-6 CTL SYNC IN H Timing Diagram

*1 One or more of the CTL IN D lines from the user device change state.

*2 After the CTL IN D lines assert they are stable, CTL SYNC IN H is asserted by the user device. This change interrupts the DRB32 controlling process, communicating the change in the CTL IN D<7:0> lines.

*3 After CTL SYNC IN H is asserted, it is de-asserted, to be ready for the next change in the CTL IN D lines.

*4 After CTL SYNC IN H is deasserted, the CTL IN D lines are permitted to change.

*5 CTL SYNC IN H is asserted by the user device to inform the DRB32 that one or more lines in CTL IN D have changed.

4.4 DRB32 DEVICE STATUS LINES DESCRIPTION

The DRB32 has two lines to send and receive the status of the user device's ability to send and receive data. These lines are:

- NODE ACTIVE OUT H
- NODE ACTIVE IN H

These lines are asserted high by a node if that node is available to send or receive data. Node availability means the node has successfully completed power-up and internal self-test.

User devices should assert NODE ACTIVE IN H to the DRB32 when they are powered up and ready to participate in data transfer transactions. Until it receives NODE ACTIVE OUT H assertion from the DRB32, a user device should not attempt to transfer data, control information to the DRB32, or assert its output drivers.

The NODE ACTIVE OUT H signal is not pulled up on the DRB32. The user device must pull up this signal.

4.4.1 Device Status Signal Descriptions

There are two device status signals:

- NODE ACTIVE OUT H
- NODE ACTIVE IN H

These signals indicate when the DRB32 and the user device are stable, and can transfer data. They are shown in Table 4-6.

NOTE
The user device must pull up NODE ACTIVE OUT H to +5 V with a 1KΩ resistor. The DRB32 pulls up NODE ACTIVE IN H in the same manner.

Table 4-6 Device Status Signal Descriptions

Signal	Description
NODE ACTIVE OUT H	This line is asserted by the DRB32 when its power is stable and it has run the tests necessary to determine that it is able to transfer data.
NODE ACTIVE IN H	This line is asserted by the user device when its power is stable and it has run the tests necessary to determine that it is able to transfer data. The DRB32 does not enable any of its output drivers until the user node asserts NODE ACTIVE IN H.

See Appendix A for a power-up timing diagram that includes timing for the NODE ACTIVE OUT H and NODE ACTIVE IN H signals.

4.5 OPTIMIZING DATA TRANSFER SPEED

The following discussion gives you some insights into the inner workings of the DRB32, and shows how the rates and rules for increasing speed are arrived at.

4.5.1 DRB32 Data Transfer Speeds

The data transfer rate for the DRB32 is 3.3MB/Second for two DRB32s connected back-to-back, 6.0MB/Second for READs from 8200 and 8300 VAXBI memory, and 6.7MB/Second for WRITEs to 8200 and 8300 VAXBI memory, if you turn SYNC OUT H around in less than 50 ns. To understand why there is a significant difference in the data transfer rates between the two configurations, the user needs to understand how the DRB32 works.

4.5.2 How the DRB32 Works

The DRB32 is a 32 bit data path with a two octaword buffer at the VAXBI end. A register at the user device end keeps data stable for the user while the internal data path does other work. See Figure 4-7.

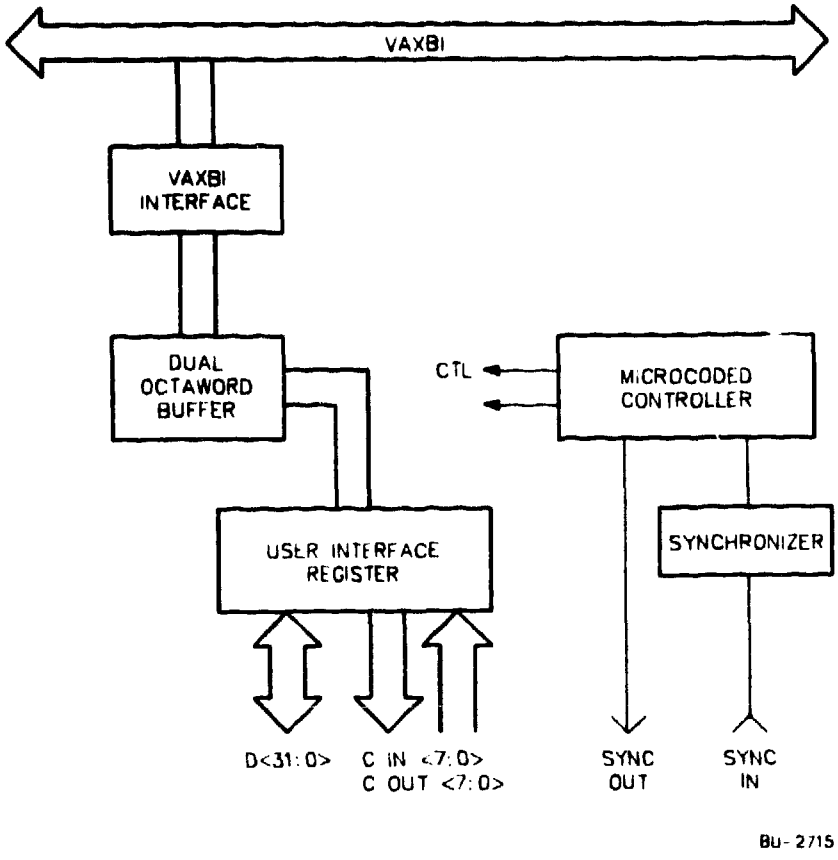


Figure 4-7 DRB32 Data Path

This simplified illustration shows:

1. There are only two sets of octaword data registers.
2. SYNC IN has to be synchronized to the DRB32 clock before it can be looked at by the microcontroller.
3. SYNC IN (after synchronization) goes directly to the microcontroller which does everything in the DRB32.
4. SYNC OUT is driven directly by the same microcontroller.

Item 1 is important because it limits the ability of the DRB32 to "pre-fetch" data. The DRB32 sends and receives data on the VAXBI in octawords to efficiently use the VAXBI bandwidth. Whenever the DRB32 collects an octaword of data in one of the octaword buffers, it requests a VAXBI octaword transaction. When the VAXBI grants the DRB32 its transaction, the DRB32 sends out the octaword of data.

If the data is being sent to the DRB32 faster than the DRB32 can send data to the VAXBI:

1. The DRB32 fills up the first octaword, then issues a request for the VAXBI octaword transaction.

2. While the DRB32 is waiting for the octaword transaction to finish, it fills up the other octaword buffer.

3. The DRB32 now has to wait until the VAXBI empties the first octaword buffer before it can accept any more data.

4. Once the VAXBI octaword transaction is completed, the DRB32 can fill the first octaword buffer and issue a request to the VAXBI to transfer the other buffer.

5. The DRB32 fills up the first octaword buffer and waits for the VAXBI to finish transferring the other octaword buffer... and so on.

This cycle means that if the data rate to or from the user is faster than the rate that the DRB32 transfers data to and from the VAXBI, the user sees:

- A very rapid data transfer for four longwords (the octaword buffer)
- Then a wait (until the next octaword is transferred to or from the VAXBI)
- Then another rapid data transfer.

To get maximum data transfer speed from the DRB32, the user device must have the data ready for the DRB32 when the DRB32 wants it. If the user device wants to transfer data at a steady rate (as opposed to the "bursty" way that the DRB32 does it), the user device should buffer the data at its end so that the user device can have an octaword "burst" of data ready when the DRB32 wants an octaword. In the case of WRITEs to the user device, the buffer should be able to accept an octaword "burst" of data.

4.5.3 The Synchronization Problem

The DRB32 user interconnect is asynchronous. The DRB32 internal circuitry is synchronous using double ranked 74F175s running at 20 MHz in order to synchronize the user interconnect to the DRB32. This results in a projected high MTBF, but also results in a 100 ns delay before the DRB32 microcontroller realizes that the user device has asserted SYNC IN H.

DRB32 speed is also related to the speed that the microcontroller can react to the synchronized (and delayed by 100 ns) SYNC IN H signal from the user. The sequence of events is:

1. The internal PC controller executes a branch to the address which says "The user has asserted SYNC IN H." The internal PC controller outputs this address either 50 or 100 ns later, depending on where SYNC IN occurred in the controller's cycle.

2. The ROM receives the new address, puts the new data in the user output buffer, and deasserts SYNC OUT H, 100 ns after the address is output from the internal PC controller.

3. The deasserted SYNC OUT H passes through a latch and a buffer and is presented to the user.

The total delays when SYNC IN H appears at the DRB32 input are:

1. 100 ns for the synchronizer

2. 50 to 100 ns for the internal PC to send out the address

3. 100 ns for the ROM to take the address and turn it into SYNC OUT H deassertion and assert the new data.

   Therefore, it takes 250 to 300 ns from SYNC IN H assertion to SYNC OUT H deassertion.

After SYNC IN H deassertion appears at the DRB32 input, the delays are:

1. 100 ns for the synchronizer

2. 50 to 100 ns for the internal PC to send out the address

3. 100 ns for the ROM to take the address and turn it into SYNC OUT H assertion.

Even if the user device "instantly" turns the SYNC OUT H signal around into the SYNC IN H signal, it takes 500 to 600 ns to transfer 4 bytes of data. If everything is working as well as it can (500 ns), the calculated data rate is:

8.0MB/Second

— assuming no time for VAXBI data cycles —

Since the data has to go to (or come from) the VAXBI, we must factor in the time it takes to send or receive data from the VAXBI. Under the best conditions, when the DRB32 is emptying one buffer to the user while the VAXBI is filling the other buffer, it has been measured that an octaword can be read from the VAXBI in 2.2 μs.

NOTE

The 2.2 μs number is derived from MS820-A/B memory stall cycles; memory cycles from the VAX 8800 to the VAXBI are somewhat slower, since 8800 memory isn't on the VAXBI and references to memory have to go through another interface. See the section later in this chapter for more information on VAX 8800 memory.

An octaword can be written slightly faster (an octaword every 2.0 μs). That calculates to:

7.2MB/Second for READs

8.0MB/Second for WRITEs

Since it takes the DRB32 400 ns to ensure that the previous VAXBI transaction completed without errors and to request the next one, it takes 2.6 μs for READs and 2.4 μs for WRITEs, which makes:

6.0MB/Second for READs

6.7MB/Second for WRITEs

In reality, it takes time for the SYNC OUT H signal to propagate down the wire to the user interface (approximately 2 ns/ft). It also takes time for the logic in the user interface to assert or deassert SYNC IN H once the SYNC OUT H arrives. Since user interfaces vary, examine two examples:

1. User interface turns SYNC OUT H around instantly (only consider wire delays).

2. User interface is another DRB32.

Case 1: A 25 foot (7.6 meter) cable connects the DRB32 to the user interface, which turns SYNC OUT H into SYNC IN H instantly. Fifty total feet (15.25 meters) of cable at 2 ns/ft equals 100 ns delay between the SYNC OUT H signal changes and the SYNC IN H signal changes. That adds 200 ns to a 4-byte transfer (800 ns to an octaword), which reduces the data throughput to:

4.7MB/Second for READs

5.0MB/Second for WRITEs

Since the synchronizer cycles in 50 ns increments, the DRB32 sees the delay time from the SYNC OUT H signal to the SYNC IN H signal, rounded up to the nearest 50 ns. If the cable is changed from 25 feet (7.6 meters) to 26 feet (7.9 meters), a 52 foot (15.8 meter) round trip, there will be a round trip delay time of 104 ns, which the DRB32 would round up to a 150 ns delay (300 ns/4B cycle, or 1.2 μs/octaword). Throughput would then become:

4.2MB/Second for READs

4.4MB/Second for WRITEs

Case 2: A 10 foot (3 meter) cable connects two DRB32s:

1. DRB32 No. 1 asserts data.

2. One hundred nanoseconds later, DRB32 No. 1 asserts the SYNC OUT H signal.

3. In 20 ns, the SYNC IN H signal of DRB32 No. 2 receives the SYNC OUT H signal of DRB32 No. 1.

4. Thirty nanoseconds (rounding up to 50 ns) later, the SYNC OUT H signal of DRB32 No. 1 is clocked out of the first synchronizer rank.

5. Fifty nanoseconds later, the SYNC OUT H signal of DRB32 No. 1 is out of the second synchronizer rank, and is at the input to the internal PC controller of DRB32 No. 2.

6. Fifty nanoseconds later, the internal PC controller outputs the new PC.

7. One hundred nanoseconds after that, the ROM of DRB32 No. 2 outputs the SYNC OUT H signal to DRB32 No. 1.

8. The SYNC OUT H signal of DRB32 No. 2 takes 20 ns to get to the SYNC IN H input of DRB32 No.1.

9. DRB32 No. 1 does steps 4, 5, 6 and 7 above (300 ns total) to deassert the SYNC OUT H signal and put new data on the interconnect.

10. The deassertion of the SYNC OUT H signal of DRB32 No. 1 takes 20 ns to get to DRB32 No. 2.

11. DRB32 No. 2 does steps 4, 5, 6 and 7 above (300 ns total) to deassert its SYNC OUT H signal (SYNC IN H of DRB32 No. 1).

12. DRB32 No. 1 does steps 4, 5, 6 and 7 above (300 ns total) to assert the SYNC OUT H signal for the second time, thus completing a cycle.

The resultant data rate is:

3.1MB/Second

4.5.4 DRB32 Data Transfer Speed with the VAX 8800

The VAXBI speed becomes the major component in DRB32 data transfer rates when other peripherals on the bus, or the interface to memory, is slower than the DRB32 microcontrolier speed.

For the VAX 8800, memory is not located on the VAXBI but on another bus that is connected to the VAXBI with a DB88 adapter. The VAXBI is used as an I/O bus and, although processor accesses to memory are speeded up, the VAXBI data transfer speed to and from memory is diminished. The VAX 8200 data transfer speed limit is the rate at which the DRB32 microcontroller can look at and react when the user device changes the SYNC IN H signal.

The VAXBI-to-memory adapter inserts five stall cycles in an octaword WRITE from the DRB32 to memory. Thirteen stalls are inserted in an octaword READ from memory to the DRB32. If there is no contention for either the VAXBI or the VAX 8800 memory bus (that is, optimum conditions) the data throughput is:

3.1MB/Second for READs

4.4MB/Second for WRITEs

Other system traffic reduces these numbers.

4.5.5 DRB32 Speed Matrix

The "speed matrix" for the VAX 8200 and VAX 8800 using the DRB32 is shown in Table 4-7.

Table 4-7 DRB32 Speed Matrix (MB/Second)

Adapter	READ from 8200, 8300 VAXBI Memory	WRITE to 8200, 8300 VAXBI Memory	READ from 8500, 8550, 8700, 8800 Priv. Memory	WRITE to 8500, 8550, 8700, 8800 Priv. Memory
DRB32-M	6.0	6.7	3.3	4.7
DRB32-W	3.1	3.6	3.1	3.6

4.5.6 Rules to Optimize Data Transfer Speed

To achieve maximum data transfer rates, take the following steps:

- Put an octaword buffer in the user device to smooth the data flow.
- Turn the SYNC OUT H signal around into the SYNC IN H signal as fast as possible (that is, acknowledge data receipt and readiness to accept new data as quickly as possible).
- Keep the cable lengths as short as possible to minimize wire delays.

An octaword buffer can ensure that the user device has data ready for the DRB32 when the DRB32 wants to transfer that data to the VAXBI, and thus prevent a transfer-wait-transfer cycle. Your user device must turn the DRB32 SYNC OUT H signal around in less than 50 ns to achieve the higher transfer rate. Delays in synchronizing signals or long cable lengths can also slow data transfers.


APPENDIX A

APPENDIX A
DRB32 SPECIFICATIONS

This appendix contains the following DRB32-M information:

- Timing specifications
- Electrical specifications
- Environmental specifications
- Pin assignments

A.1 TIMING SPECIFICATIONS

This section includes timing diagrams for the DRB32, including:

- Send data timing
- Receive data timing
- Control data OUT timing
- Control data IN timing
- Power-up timing

The transmit waveforms and times are measured at the input to the bus transmitter. Receive waveforms and times are measured at the output of the receiver. Therefore, all times in this section include delay times of the transmitter and receiver circuitry, as well as the propagation times in the cables between them.

A.1.1 Send Data Timing

Figure A-1 shows the send data timing for the DRB32 adapter and Table A-1 describes the send data timing.

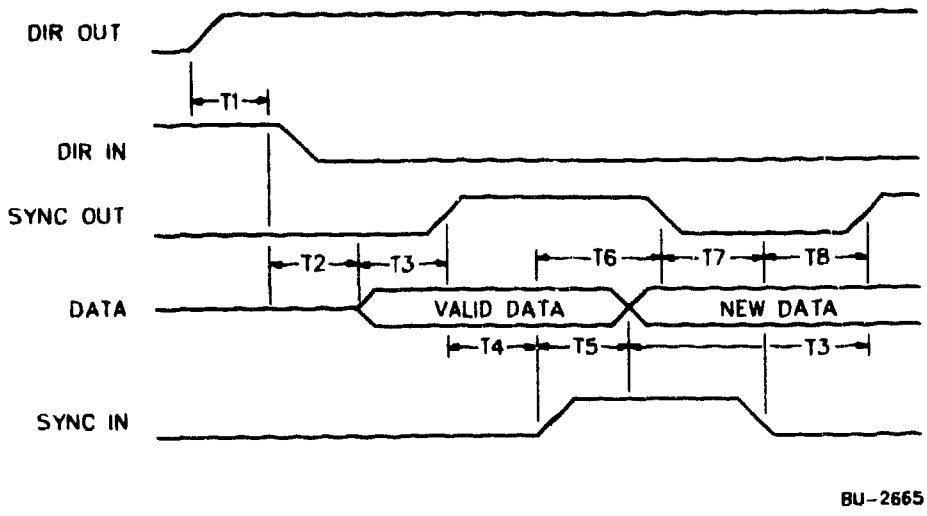


Figure A-1 Send Data Timing Diagram

Table A-1 Send Data Timing Description

No.	Meaning	Min	Max
T1	Time from DIR OUT H assertion to DIR IN H deassertion	No min	No max
T2	Time from DIR agreement to data enable	Min = 0 ns	No max
T3	Time from valid data assertion to SYNC OUT H assertion	Min = 100 ns	No max
T4	Time from SYNC OUT H assertion to SYNC IN H assertion	Min = 0 ns	No max
T5	Time from SYNC IN H assertion to new data assertion	Min = 0 ns	No max
T6	Time from SYNC IN H assertion to SYNC OUT H deassertion	Min = 0 ns	No max
T7	Time from SYNC OUT H deassertion to SYNC IN H deassertion	Min = 0 ns	No max
T8	Time from SYNC IN H deassertion to SYNC OUT H assertion.	Min = 0 ns	No max

A.1.2 Receive Data Timing

Figure A-2 shows the receive data timing for the DRB32 adapter and Table A-2 describes the receive data timing.

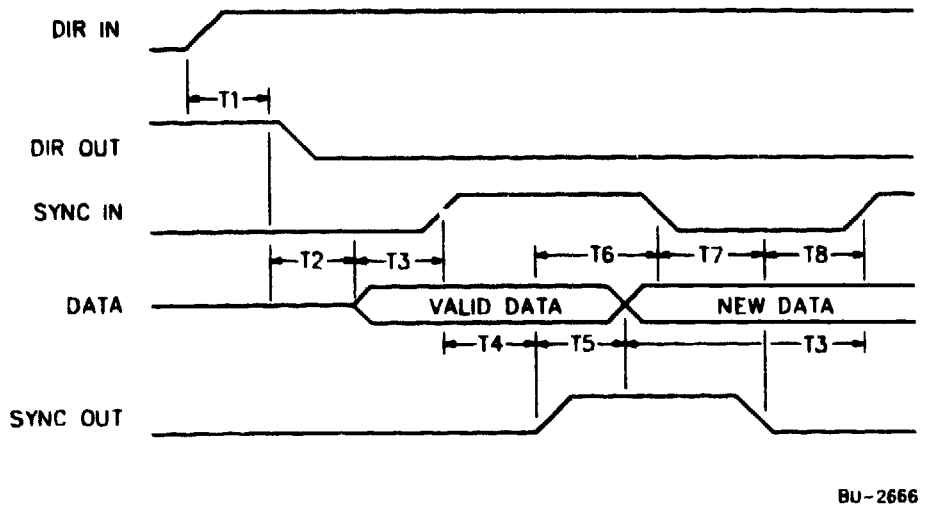


Figure A-2 Receive Data Timing Diagram

Table A-2 Receive Data Timing Description

No.	Meaning	Min	Max
T1	Time from DIR IN H assertion to DIR OUT H deassertion	No min	No max
T2	Time from DIR agreement to data enable	Min = 0 ns	No max
T3	Time from valid data assertion to SYNC IN H assertion	Min = 100 ns	No max
T4	Time from SYNC IN H assertion to SYNC OUT H assertion	Min = 0 ns	No max
T5	Time from SYNC OUT H assertion to new data assertion	Min = 0 ns	No max
T6	Time from SYNC OUT H assertion to SYNC IN H deassertion	Min = 0 ns	No max
T7	Time from SYNC IN H deassertion to SYNC OUT H deassertion	Min = 0 ns	No max
T8	Time from SYNC OUT H deassertion to SYNC IN H assertion	Min = 0 ns	No max

A.1.3 Control Data OUT Timing

Figure A-3 shows the timing for the CTL DATA OUT and CTL SYNC OUT signals and Table A-3 describes the control data OUT timing.

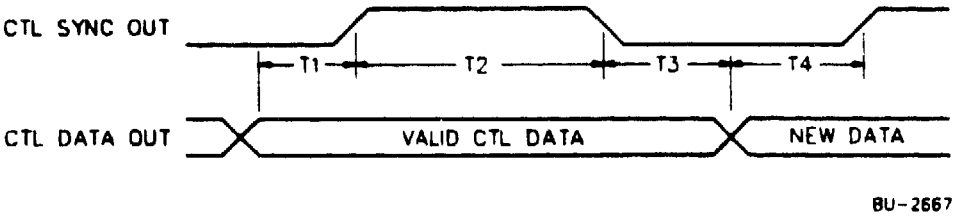


Figure A-3 Control Data OUT Timing

Table A-3 Control Data OUT Timing Description

No.	Meaning	Min	Max
T1	Time from valid data assertion to CTL SYNC OUT H assertion	Min = 100 ns	No max
T2	Time from CTL SYNC OUT H assertion to CTL SYNC OUT H deassertion	Min = 200 ns	No max
T3	Time from CTL SYNC OUT H deassertion to valid data removal	Min = 100 ns	No max
T4	Time from new valid data assertion to CTL SYNC OUT H assertion	Min = 100 ns	No max

A.1.4 Control Data IN Timing

Figure A-4 shows the timing for the CTL DATA IN and CTL SYNC IN signals and Table A-4 describes the control data IN timing.

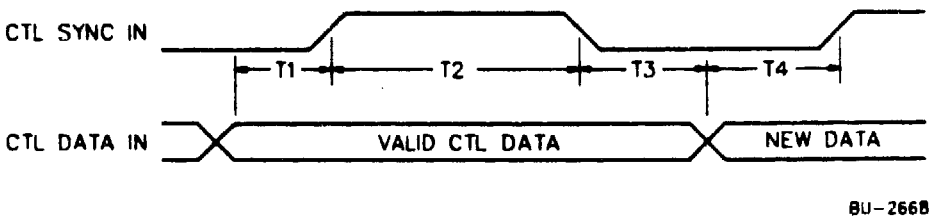


Figure A-4 Control Data IN Timing

Table A-4 Control Data IN Timing Description

No.	Meaning	Min	Max
T1	Time from valid data assertion to CTL SYNC IN H assertion	Min = 100 ns	No max
T2	Time from CTL SYNC IN H assertion to CTL SYNC IN H deassertion	Min = 200 ns	No max
T3	Time from CTL SYNC IN H deassertion to valid data removal	Min = 100 ns	No max
T4	Time from new valid data assertion to CTL SYNC IN H assertion	Min = 100 ns	No max

A.1.5 Power-Up Timing

Figure A-5 shows the power-up timing for the DRB32 adapter, and for the NODE ACTIVE OUT and NODE ACTIVE IN signals. Table A-5 describes power-up timing.

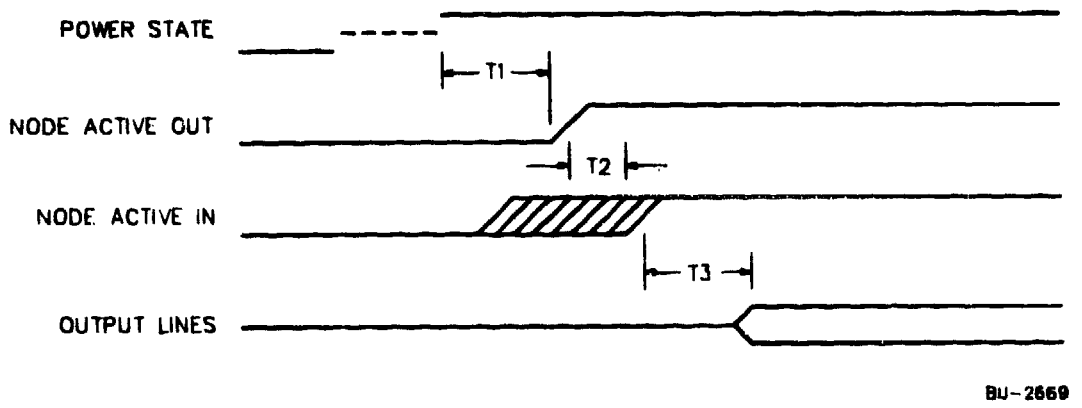


Figure A-5 Power-Up Timing Diagram

Table A-5 Power-Up Timing Description

No.	Meaning	Min	Max
T1	Time from DCLO L deasserts to NODE ACTIVE OUT assertion	1 ms	5 s
T2	Time from NODE ACTIVE OUT H assertion to NODE ACTIVE IN H assertion	No min	No max
T3	Time from NODE ACTIVE IN/OUT assertion to outputs active	Min = 0 ns	No max

A.2 DRB32 ELECTRICAL SPECIFICATIONS

This section specifies the parameters of the electrical connections to the DRB32 bus. The components, values, and tolerances stated in this section must be used to ensure proper operation. Since the requirements of the intraenclosure (DRB32-M) and interenclosure (DRB32-E) configurations are different, two sets of electrical requirements are given.

A.2.1 Intraenclosure DRB32-M Configuration

The **intraenclosure** DRB32-M requires that both ends of the bus and the cable connecting the devices are completely enclosed in the same RFI-tight, FCC-compliant enclosure. Because the power grounds are common and the DRB32 cable is not subject to noisy environment outside the enclosure, an unshielded cable can connect the two ends and normal TTL parts can be used to drive the intraenclosure DRB32.

A.2.1.1 Intraenclosure Drivers and Receivers – The drivers and receivers for the intraenclosure DRB32 bus are single-ended drive and receive devices. Two parts from the FAST logic family are used:

- 74F543 - Bidirectional octal latch/driver/receiver
- 74F244 - Bidirectional octal driver/receiver

These driver/receivers are the **only** parts that can be used to ensure proper operation. The DC specifications are given in Table A-6.

Table A-6 DC Specifications for the Intraenclosure DRB32-M

Parameter	Min	Max	Conditions
Input High Voltage	2.0 V	5.25 V	
Input Low Voltage	0.0 V	0.8 V	
Output High Voltage	2.5 V		Output current = −12 mA
Output Low Voltage		0.55 V	Output current = 64 mA
Output-Off Input Current			
Input High		70 μA	Vcc = 5.25 V, Vout = 2.4 V
Input Low		0.6 mA	Vcc = 5.25 V, Vout = 0.5 V
Output Short-Circuit Current	−100 mA	−225 mA	Vcc = 5.25 V

The drivers can sustain indefinitely (without damage) a short to ground through the termination resistors (100 Ω).

A.2.1.2 Cables – The propagation time for the cables for the intraenclosure DRB32-M adapter is given in Table A-7.

Table A-7 Propagation Time for Intraenclosure DRB32-M

Parameter	Min	Max
Propagation time	None	0.13 ns/inch

A.2.1.3 Terminators – Termination for the intraenclosure DRB32 bus is a 100-Ω resistor between the bus line and the driver/receiver at each end of the bus on every bidirectional signal. On unidirectional signals, the 100-Ω termination resistor is placed between the transmitter and the bus. This essentially provides the benefit of termination at one end, and significantly reduces the current requirements of both the drivers and the DRB32 device as a whole. The series termination also reduces the di/dt for the output of the driver, providing a smoother transition at the output of the driver.

The disadvantage of the series terminator is that the receiving end of the bus is not terminated because the high-impedance receiver provides no virtual ground. This causes a reflection at the receiving end and doubles the time required for the signal to settle. The timing specifications for the DRB32 signals take this time into account.

A.2.2 Interenclosure DRB32-E

Refer to Appendix B for the DRB32-E electrical specifications.

A.2.3 Intraenclosure DRB32-W

Refer to Appendix C for the DRB32-W electrical specifications.

A.3 ENVIRONMENTAL SPECIFICATIONS

The DRB32-M is designed to operate within a Class B cabinet-based system and within a Class C OEM cabinet environment. The following tables show the operating and storage specifications.

Table A-8 DRB32 Operating Environmental Specifications

Parameter	Specification
Temperature	+5°C to 50°C with a maximum rise across the module of 10°C. This is based on operation at sea level (760 mm Hg). Maximum operating temperature is reduced by a factor of 1.8°C per 1000 m of altitude above sea level.
Air Flow	Minimum of 200 linear feet per minute
Humidity	10% to 95% with maximum wet bulb 32°C, and minimum dew point 2°C
Shock	Half sine shock pulse of 10 Gpk and 10 ± 3 ms duration
Vibration	5–30 Hz sinusoidal vibration displacement .010/inch (peak to peak)
	30–500 Hz sinusoidal vibration amplitude acceleration = 0.5 Gpk (1 Gpk = 9.8 m/s/s)
EMI	No specification since EMI/RFI shielding is provided by the system cabinet

Table A-9 DRB32 Storage Environmental Specifications

Parameter	Specification
Temperature	−40°C to 66°C
Humidity	95% relative humidity at 66°C
Shock	Half sine shock pulse of 40 Gpk and 30 ±10 ms duration (1 Gpk = 9.8 m/s/s)

The DRB32-E and DRB32-W options have the same environmental specifications.

A.4 DRB32 PARALLEL I/O PORT PIN ASSIGNMENTS

Figure A-6 shows the 80 DRB32 parallel port pin assignments.

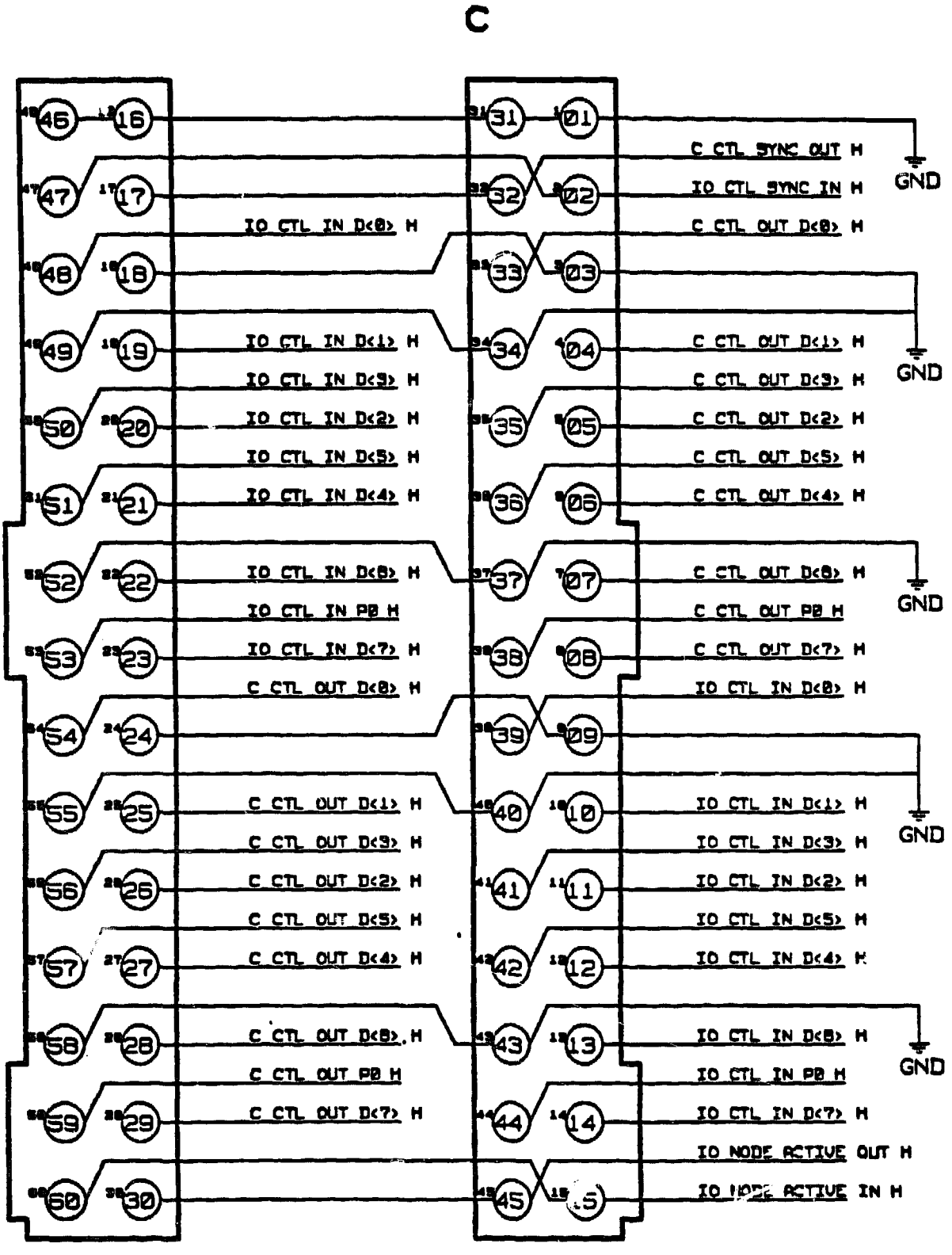


Figure A-6 DRB32 Parallel Port Pin Assignments (Part 1 of 3)

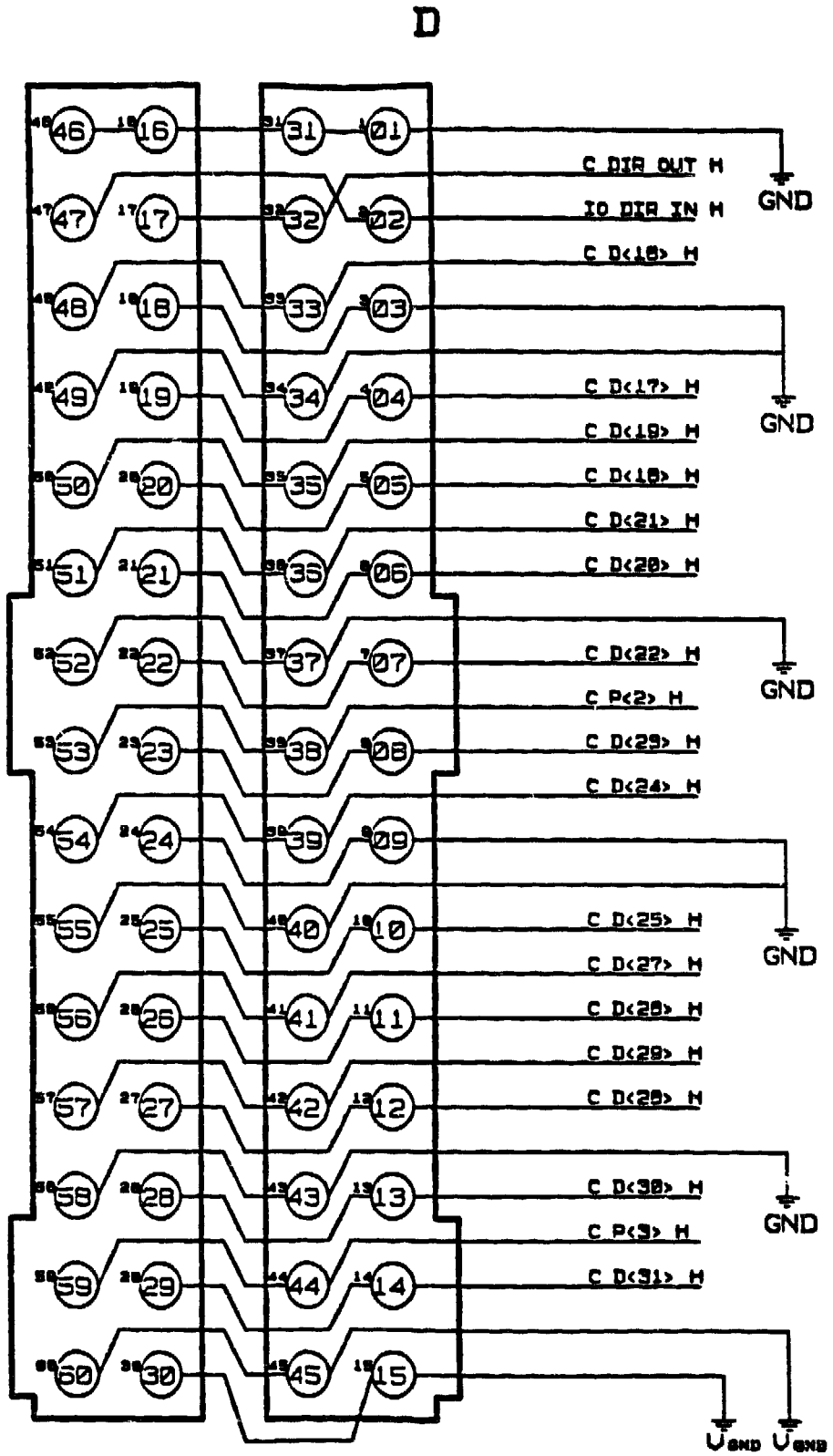


Figure A-6 DRB32 Parallel Port Pin Assignments (Part 2 of 3)

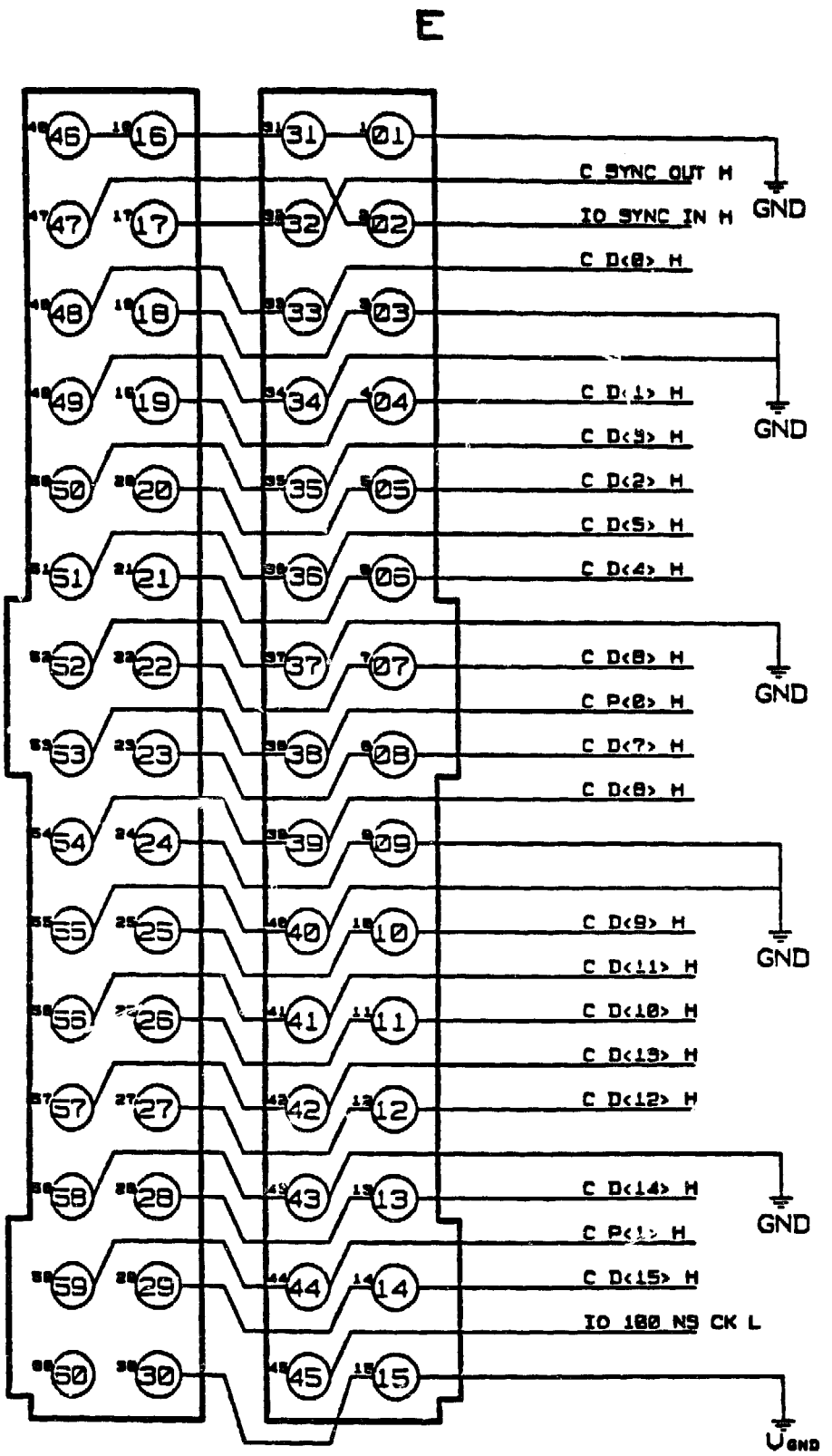
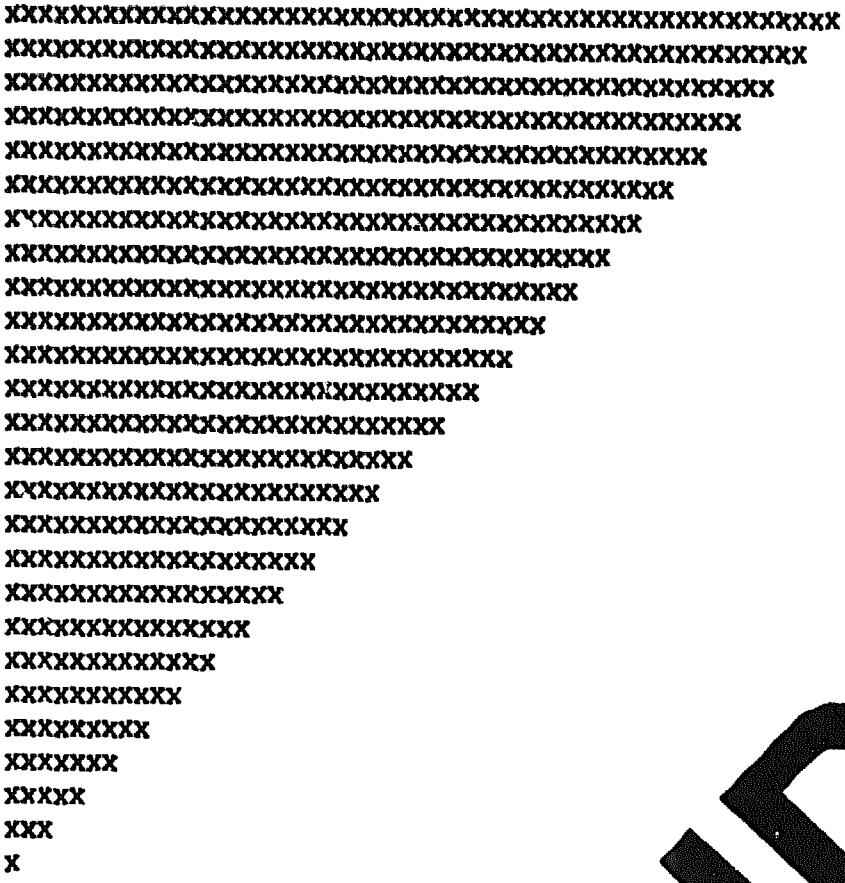
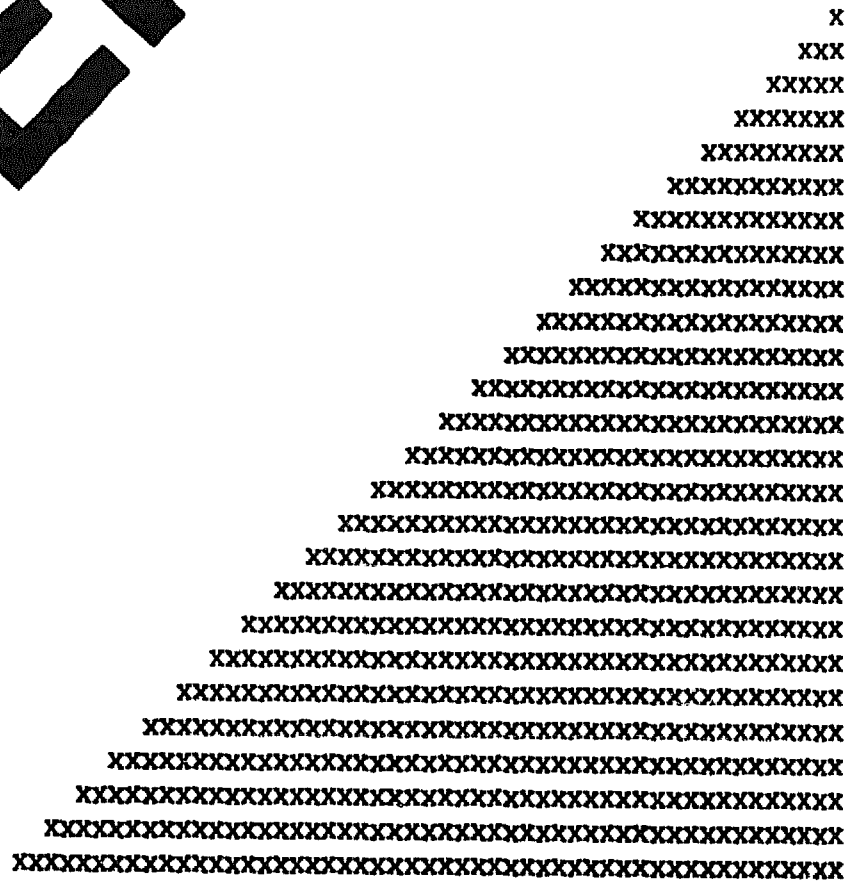


Figure A-6 DRB32 Parallel Port Pin Assignments (Part 3 of 3)



APPENDIX B



APPENDIX B
DRB32-E OPTION

B.1 OVERVIEW

The DRB32-E option consists of the basic DRB32-M adapter module, the T1024 external driver module that allows for FCC-compliant intercabinet connections of up to 40 feet (12 meters) to connect two VAXBI systems, and an I/O connector panel. All DRB32-M signals can be driven in either direction through the T1024, if both VAXBI systems have the DRB32-E option installed.

The module itself consists of high-speed, low-noise transceivers on the data lines and differential line drivers and receivers on high-speed critical lines. The worst-case propagation delay of data transfers through the module is 40 ns. The basic cabinet configuration is shown in Figure B-1.

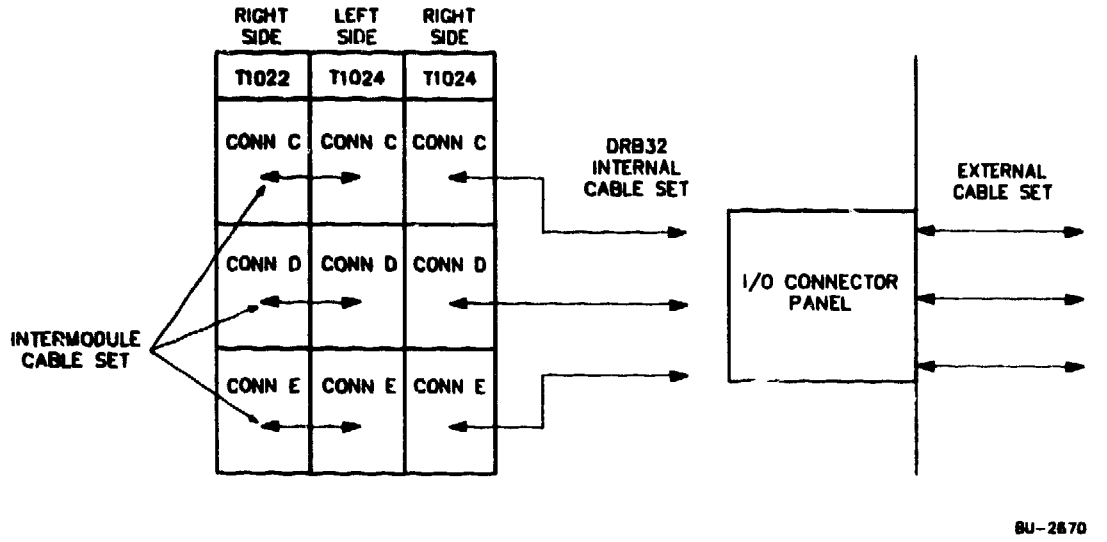


Figure B-1 DRB32-E Option Basic Cabinet

A typical DRB32-E configuration is shown in Figure B-2. The signal pin assignments at the I/O connector panel are shown in Figure B-3.

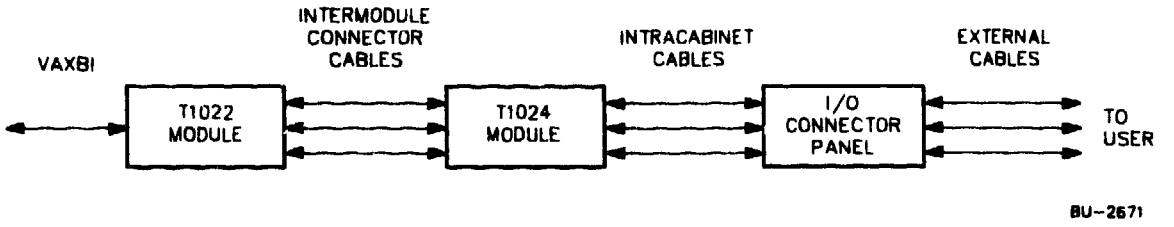


Figure B-2 Typical DRB32-E Configuration

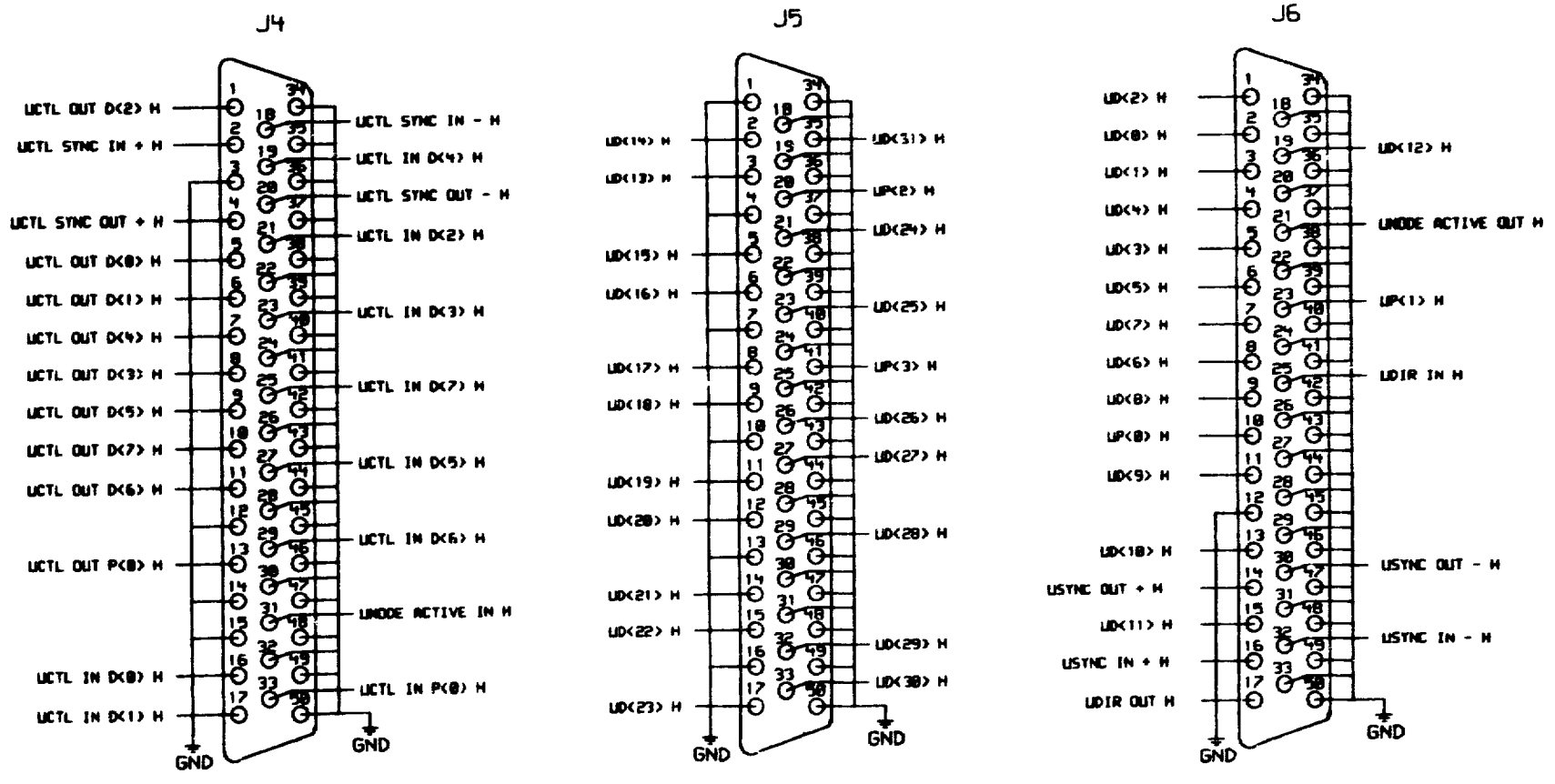


Figure B-3 DRB32-E Signal Pin Assignments

B.2 I/O INTERCONNECTIONS BETWEEN CABINETS

The T1024 must be able to redrive the DRB32-M's I/O signals over long cable lengths with a minimum of noise and signal degradation. When interfacing to the T1024, the user must adhere to the following guidelines.

Two different families of drivers and receivers are used in the DRB32-E intercabinet configuration. The following signals use differential drivers and receivers:

- USYNC IN +H
- USYNC IN –H
- UCTL SYNC IN +H
- UCTL SYNC IN –H
- USYNC OUT +H
- USYNC OUT –H
- UCTL SYNC OUT +H
- UCTL SYNC OUT –H.

The following DRB32-E signals use single-ended drivers/receivers:

- DATA <31:0>
- PARITY <3:0>
- CTL IN DATA <7:0>
- CTL OUT DATA <7:0>
- CTL IN PARITY <0>
- CTL OUT PARITY <0>
- DIR IN
- DIR OUT
- NODE ACTIVE IN
- NODE ACTIVE OUT.

B.2.1 Differential Drivers (75110A)

These line drivers are single-input, differential-output devices. Their input characteristics are those of TTL-integrated circuits. The output current is nominally 12 mA for these drivers.

The propagation delay times are given below.

- tPLH from input A or B to output Y or Z = 15 ns, maximum
- tPHL from input A or B to output Y or Z = 15 ns, maximum

When driving USYNC IN +H, USYNC IN –H and UCTL SYNC IN +H, UCTL SYNC IN –H, the user MUST use the 75110 differential line driver, since these signals are received on the T1024 by a 75107 differential line receiver.

B.2.2 Differential Receivers (75107)

The differential line receiver is the DEC 75107B. This line receiver is a differential-input, single-output device, with an input sensitivity of 25 mV and maximum propagation delay time of 25 ns. The 75107B has totem pole outputs.

Since the T1024 drives the signals USYNC OUT +H, USYNC OUT –H and UCTL SYNC IN +H, UCTL SYNC IN –H with a 75110, the user must use a 75107 differential line receiver when receiving these signals.

B.2.3 Single-Ended Bus Transceivers (26S10)

The single-ended driver/receiver is the DEC 26S10, a 100-Ω, controlled-threshold (2.0-V switching), 4-bit transceiver, having the following characteristics:

- VOL = .8 V @ 100 mA
- VTH (HIGH) = 2.25 V
- VTH (LOW) = 1.75 V
- Data input to bus = 15 ns
- Enable input to bus = 18 ns
- Bus to receiver out = 15 ns.

The T1024 uses these high-speed, controlled-threshold transceivers for driving all other signals to the user. It has low-impedance inputs. The user must use these devices when interfacing with the T1024 on the following signals:

- Data <31:0>
- Parity <3:0>
- CTL IN DATA <7:0>
- CTL OUT DATA <7:0>
- CTL IN PARITY <0>
- CTL OUT PARITY <0>
- DIR IN
- DIR OUT
- NODE ACTIVE IN
- NODE ACTIVE OUT

B.2.4 Termination

ALL signals must be terminated at the user end of the cables. Since the cable and drivers have a characteristic impedance of 100 Ω, the cables must be terminated to 100 Ω to minimize reflections. A Thevenin equivalent has been derived to accommodate this:

FOR TRANSCEIVERS

150 Ω ± 5% 1/4 W to +5 V

330 Ω ± 5% 1/4 W to 0 V (GROUND)

FOR DRIVERS AND RECEIVERS

330 Ω ± 5% 1/4 W to +5 V

140 Ω ± 5% 1/4 W to 0 V (GROUND)

The user must adhere to these requirements and terminate his end of the cable with these values.

The differential receivers (75107) must be placed as close to the cable end as possible. Both + and − receiver inputs must be terminated as shown above.

The differential drivers (75110) must be placed as close to the cable end as possible. Both + and − driver outputs must be terminated as shown above.

B.3 CABLE ASSEMBLIES

Three cable configurations are required with the DRB32-E option. Three VAXBI I/O interconnect cables are used to connect the T1024 to the T1022 on the backplane. The second cable set is used to connect the T1024 to the I/O connector panel, and the third set of three cables is used for intercabinet connection. The last cable set is referenced with the cable assembly number of either:

- BS17Y-20

  or

- BS17Y-40

The BS17Y-20 is a 20 foot (6 meter) user cable set and the BS17Y-40 is a 40 foot (12 meter) user cable set. Forty feet (12 meters)is the maximum cable length for the DRB32-E option.

B.4 I/O INTERFACING RULES FOR THE DRB32-E OPTION

You must adhere to the following rules for the proper operation of the DRB32-E:

1. No stubs in excess of six inches are allowed at the user end on the I/O signals.
2. No multiple stubs are acceptable on the I/O signals.
3. Only one connection to a driver, receiver, or transceiver is permitted.
4. Pins 1 and 8 of the 26S10 must be connected to a low-impedance ground.
   a. Fifteen mil ETCH connections are not acceptable.
   b. Fifty mil ETCH should be used on double-sided PC boards or multilayer PC boards that make this connection on the outer layers.
5. Ground continuity must be maintained throughout the cable system. The flat cable and twisted pair cable drain wires must be connected to each other and to logic ground.
6. Both ends of the external cable must be attached to the cabinet frame.
7. Only one connection from logic to frame ground is permitted across the entire system.

B.5 PROPAGATION DELAY THROUGH THE T1024 MODULE

The T1024 module was designed for high-speed data transfers between VAXBI systems, with minimum delay from input to output. The timing diagrams below (Figures B-4, B-5, and B-6) show the worst-case delay through the T1024. A 1 ns/ft delay time for the BC17Y-xx cable must be added to the overall delay time.

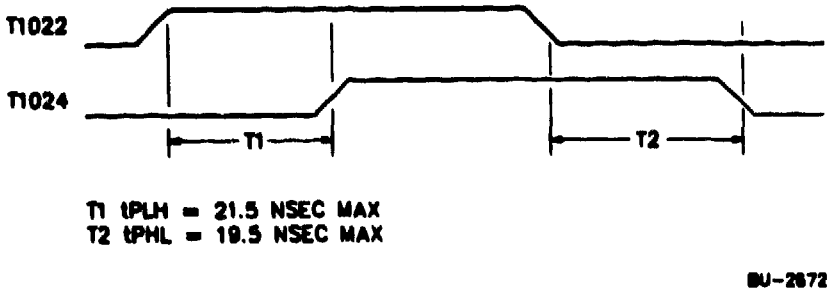


Figure B-4 Propagation Delay Through the 74F240 and the 26S10

tPLH 75107A (25 NSEC MAX) 74F240 (6.5 NSEC MAX)
tPHL 75107A (25 NSEC MAX) 74F240 (4.5 NSEC MAX)

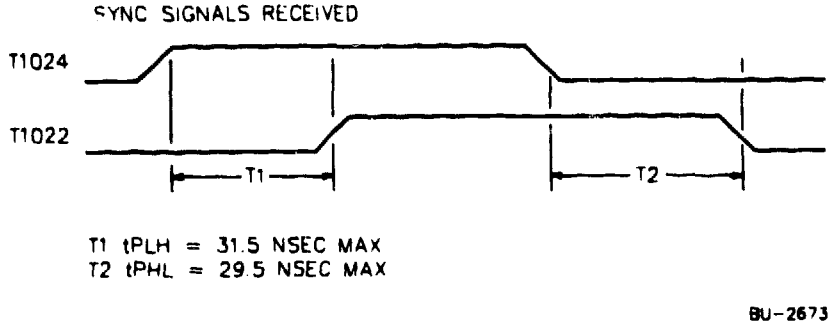


Figure B-5 Delay for All Signals That Are Driven Differentially

tPLH 75110A 15 NSEC MAX
tPHL 75110A 15 NSEC MAX

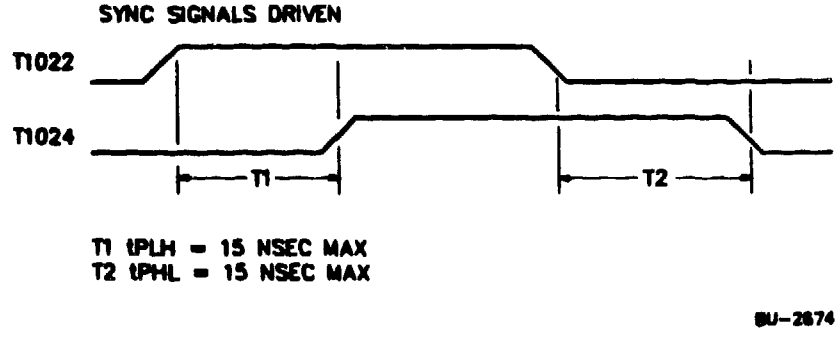


Figure B-6 Delay for Signals That Are Driven Through the 75110

B.6 I/O Signal-Name Interpretation

Signals driven by the DRB32 are referred to as "OUT" and, when received by the user, "IN." The signals received by the DRB32 are referred to as "IN," but are outputs from the user. Figure B-7 is a representation of how the signal names change from the T1022 to the T1024 and finally to the user.

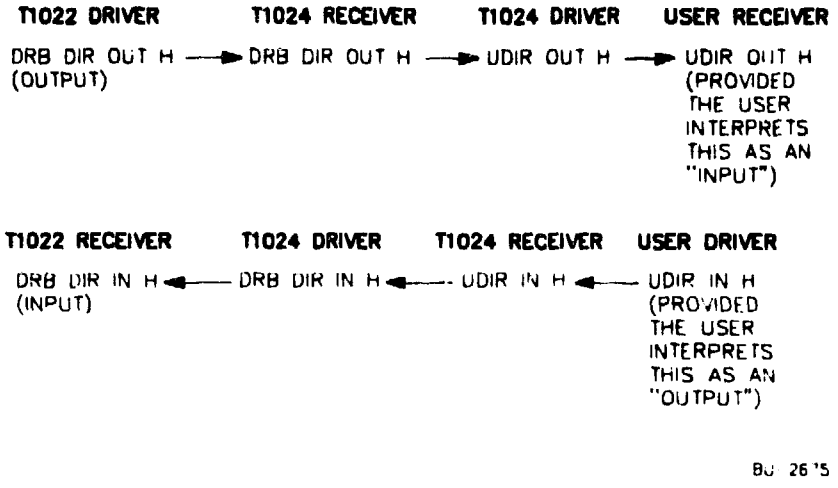


Figure B-7 Signal Transfer Name Interpretation

To simplify the interpretation of "IN" and "OUT," the user should name the user signal with "OUT," if it is the user's OUTput and the DRB32-E's INput. Likewise, the user should name the user signal with "IN," if it is the user's INput and the DRB32-E's OUTput.

APPLICATION NOTE

Please note, when using the T1024 module, that the following signals are tied to the following pins where Y is the output and Z is the inverted output.

USYNCOUT +H tied to Z
USYNCOUT –H tied to Y
UCTLSYNCOUT +H tied to Y
UCTLSYNCOUT –H tied to Z

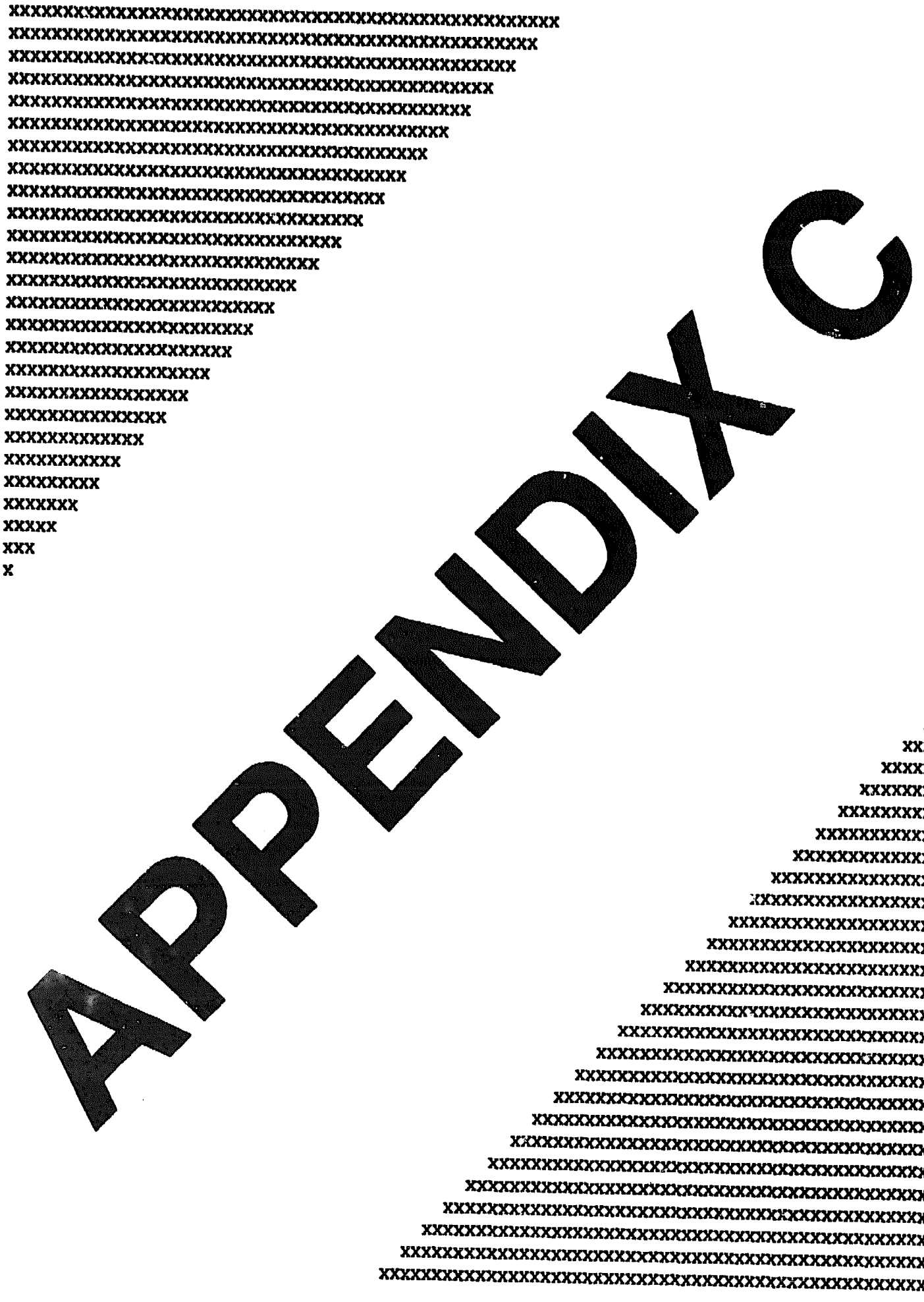
B.7 POWER REQUIREMENTS

The power requirements for the T1024 module are shown in Table B-1.

Table B-1 T1024 Power Requirements

Voltage	Tolerance	I Max
+5.00 V	±250 mV	4.0 A
−5.00 V	±250 mV	100 mA

Since the user MUST use the same differential drivers and receivers as the T1024, a −5.00 V power source must be provided for the user interface circuitry.


APPENDIX C

APPENDIX C
DRB32-W OPTION DESCRIPTION AND SPECIFICATION

The DRB32-W option consists of the basic DRB32-M adapter board, the T1023 DR11-W converter module that enables direct connection to DR11-W type devices, and a bulkhead connector. If your current equipment connects to a DR11-W, this option provides a means of connecting those devices to a VAXBI system (without requiring a UNIBUS subsystem).

The DRB32-W option is an intracabinet interface (as is the DRB32-M). There are no specific provisions to make this device FCC compliant when used as an intercabinet connection.

Since the DRB32 provides significantly higher performance than the DR11-W, redesigning your hardware to interface directly to the DRB32 will provide the benefit of increased throughput. The DRB32-W provides about twice the throughput of a DR11-W. The DRB32-M provides anywhere from three to five times the throughput of the DRB32-W. (See Table C-1.)

Table C-1 Comparison of Adapter Transfer Rates (MB/Second)

Adapter	READ from 8200, 8300 VAXBI Memory	WRITE to 8200, 8300 VAXBI Memory	READ from 8500, 8550, 8700, 8800 Priv. Memory	WRITE to 8500, 8550, 8700, 8800 Priv. Memory
DRB32-M	6.0	6.7	3.3	4.7
DRB32-W	3.1	3.6	3.1	3.6

NOTE

Please review Section 4.5 OPTIMIZING DATA TRANSFER SPEED, of this manual. This section supplies examples of how data transfer rates are calculated.

C.1 T1023 DESCRIPTION

The T1023 converter module is a DRB32 interface adapter to the DR11-W bus. The T1023 emulates the functions of the DR11-W UNIBUS module. The T1023 data path packs and unpacks the half-duplex DRB32 32-bit data bus READs and WRITEs to the 16-bit DR11-W full-duplex data buses. Two DR11-W interface transactions are required for every one DRB32 transaction. The T1023 passes non-pertinent control signals to the DRB32 control port.

C.1.1 DR11-W and DRB32-W Differences

NOTE

Please look at Section C.9 HARDWARE FEATURES T1023-YA, if you have a T1023-YA module. The "READY/BUSY" and "Each DR11-W transaction" differences, differ from the T1023 and the T1023-YA module.

The differences between the DR11-W and the DRB32 include:

- Each DR11-W transaction must consist of two (two READs or two WRITEs) 16-bit words or four bytes, while a DRB32 transaction consists of one 32-bit word. If an odd number of words are transferred, the DRB32 software driver must assert (set) the Purge bit during WRITEs to the DRB32. This allows data stored in the T1023 buffer to be transferred to the DRB32. During READs with an odd number of words, the software driver must assert the Purge or Ready bit to terminate the transfer.
- BURST RQ L is ignored by DRB32-W hardware and software. This bit has no meaning in the VAXBI context.
- A 00 H is ignored by the DRB32-W hardware and is not passed to the DRB32 software.
- WC INC H is ignored by the DRB32-W hardware and is not passed to the DRB32 software.
- BA INC EN H is ignored by the DRB32-W hardware and is not passed to the software.
- The GO H pulse is 200 ns for the DRB32-W, and 160 ns (± 15%) for the DR11-W.
- The DRB32 INIT H pulse is generated via a software set/clear of bit 4 (INIT bit) of the I/O Control Register. The DR11-W INIT H pulse is 360 ns (± 15%).
- A user device connected to the DRB32-W must not change C1 CNTL H or C0 CNTL H during a DMA transaction. Any transition of the control bits is ignored by the software driver until the complete block is transferred. Changing C1 CNTL H may result in a bus conflict error, which stops the transaction and reports an error. A DR11-W device can change C1 CNTL H every cycle (even in diagnostic mode).
- DATIP transfers are not allowed, because read-modify-write requires changing control bits every DR11-W cycle.
- Byte transfers by the T1023 cannot begin with an odd byte (A00 H = 1). The T1023 longword buffer is loaded from the low byte to the high byte.
- READY/BUSY protocol is different. On the DR11-W, READY H is set before BUSY H deasserts at the end of the transfer. On the DRB32-W, READY H is also set before BUSY H deasserts at the end of a write to user transfer, but READY H is set after BUSY H deasserts at the end of a DRB32-W read from user device transfer. Keying off of the deassertion of READY H and BUSY H at the end of a read from user device transfer may cause problems for a DR11-W device.

The following sections describe the T1023, which is the DRB32 converter module for DR11-W.

C.2 BASIC FUNCTIONS OF THE T1023

The T1023 performs the following functions:

- Transfer data from the DRB32 to the DR11-W port
- Transfer data from the DR11-W to the DRB32 user interface port

- Pass I/O control data to the DRB32 control port.

The T1023 is connected to the data lines (D lines) on the DRB32 user port, and has an INPUT and OUTPUT path to the DR11-W. See Figure C-1.

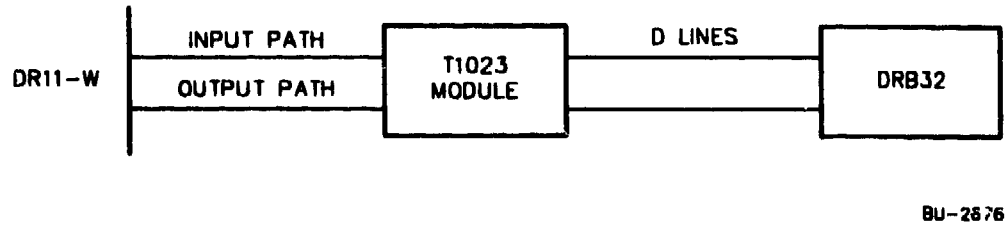


Figure C-1 T1023 Module Connection to DRB32 and DR11-W

C.3 DR11-W INTERFACE SIGNALS

The T1023 passes data to the DR11-W through the **DR11-W interface**. The DR11-W interface signals include both input and output signals, which are described in the following sections. Figure C-2 shows the signal pin connections at the bulkhead connector.

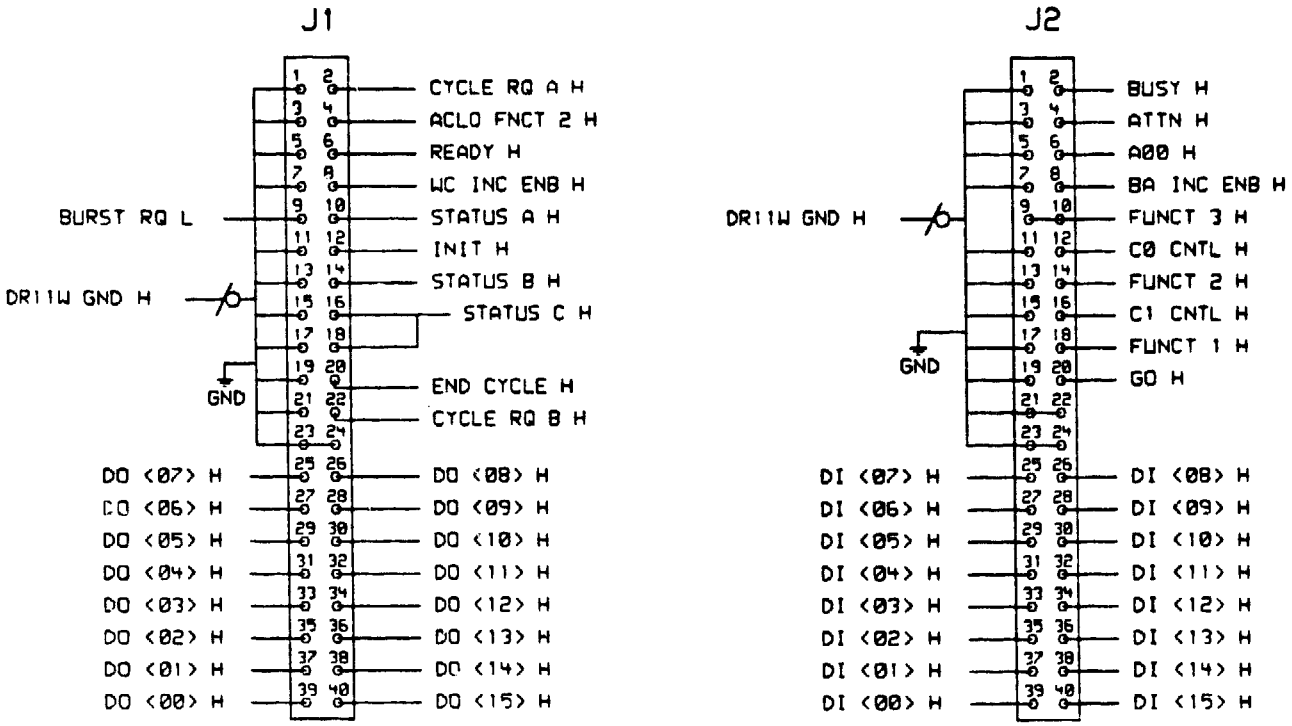


Figure C-2 DRB32-W Signal Pin Assignments

C.3.1 Input Signals

The DR11-W interface input signals include:

- Data inputs
- Status inputs
- Bus control signals
- Interrupt signal

C.3.1.1 Data Inputs - The DR11-W interface data inputs use the 16 DI H lines.

DI <15:0> H DR11-W Data Inputs

C.3.1.2 Status Inputs - The status inputs include:

- STATUS A H - User Device Status bit <1>
- STATUS B H - User Device Status bit <2>
- STATUS C H - User Device Status bit <3>

These bits are intended to be used for transfer of set up and protocol information together with the FNCT <2:0> bits in the I/O Control Register (IOCTL).

C.3.1.3 Bus Control - The bus control input signals include:

- CYCLE RQ A H - DMA cycle start pulse A
- CYCLE RQ B H - DMA cycle start pulse B
- BURST RQ L - Set burst mode (received and ignored)
- C1 CNTL H - Cycle type selection
- C0 CNTL H - Cycle type selection
- A00 H - Low order address bit for byte transfers (ignored)
- BA INC ENB H - Allow Bus Address Register to increment (ignored)
- WC INC ENB H - Allow Word Count Register to increment (ignored)

There are also the following signals:

C1 CNTL H	C0 CNTL H	Operation
0	0	DRB32-W PORT READ
0	1	DRB32-W PORT READ FROM MEMORY WITH WRITE INTENT
1	0	DRB32-W PORT WRITE TO MEMORY
1	1	DRB32-W PORT WRITE BYTE TO MEMORY

C1 CNTL H	C0 CNTL H		DIROUT H	DIRIN H
0	X	DRB32-W PORT ← DATA FLOW ← DRB-32	1	0
1	X	DRB32-W PORT → DATA FLOW → DRB-32	0	1

C.3.1.4 Interrupt - The interrupt signal for the DBR32-W interface is ATTN H. ATTN H - interrupt to the processor. ATTN H must be deasserted during a transaction and must be deasserted for a transaction to begin.

C.3.2 Output Signals

The DBR32-W interface output signals include:

- Data outputs
- Function outputs
- Bus control signals
- Miscellaneous outputs

These signals are described in the following sections.

C.3.2.1 Data Outputs - The 16 DO lines are data outputs from the T1023 module.

DO <15:0> H Data outputs from the T1023

C.3.2.2 Function Outputs - The function outputs from the T1023 include:

- FNCT 1 H - User Device Command bit <1>
- FNCT 2 H - User Device Command bit <2>
- FNCT 3 H - User Device Command bit <3>

These bits are intended to be used for transfer of set up and protocol information together with the STATUS <C,B,A> bits in the I/O Control Register (IOCTL).

C.3.2.3 Bus Control Outputs – The bus control output signals are:

- END CYCLE H – DMA is about to end (100 ns pulse)
- BUSY – DMA is in progress (assertion level selected via switch on bulkhead PCB)

C.3.2.4 Miscellaneous Outputs – The miscellaneous outputs include:

- ACLO FNCT 2 H
- INIT H
- READY H
- GO H

The miscellaneous output signals are described in Table C-2.

Table C-2 Miscellaneous Output Signals

Signal	Use
ACLO FNCT 2 H	FUNCT 2 ORed with NODE ACTIVE OUT H
INIT H	Initialize the user device
READY H	The T1023/DRB32-M is ready
GO H	The DMA transaction has just begun (200 ns pulse)

C.4 T1023 REGISTER SET

The T1023 registers include:

- I/O Control Register (IOCTL)
- I/O Data Register (IODAT)

These registers control the T1023. They are described below.

C.4.1 IOCTL – I/O Control Register

The I/O Control Register contains the value of the inputs and outputs of the DRB32 control bus. These inputs and outputs are used to communicate block transfer protocol to and from the T1023.

```
VAXBI:  bb+0190  R/W
```

This register has eight input bits and eight output bits. The eight READ-only input bits reflect the state of the corresponding bit in the DRB32 control bus. The eight output bits (READ or WRITE) are set by writing into the appropriate byte in the I/O Control Register. (See Figure C-3.)

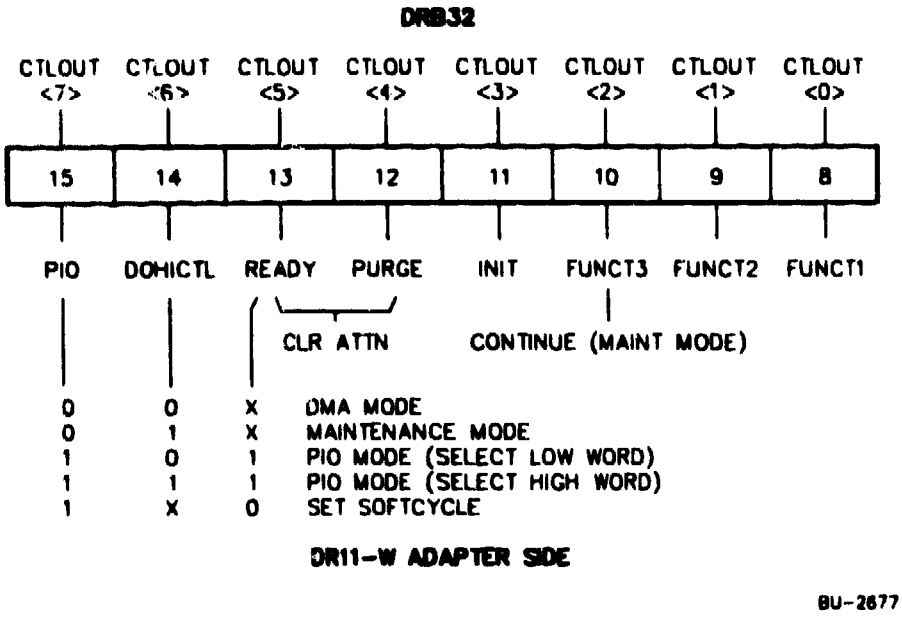


Figure C-3 I/O Control Register - Output Bytes

Figure C-4 shows the eight input bytes.

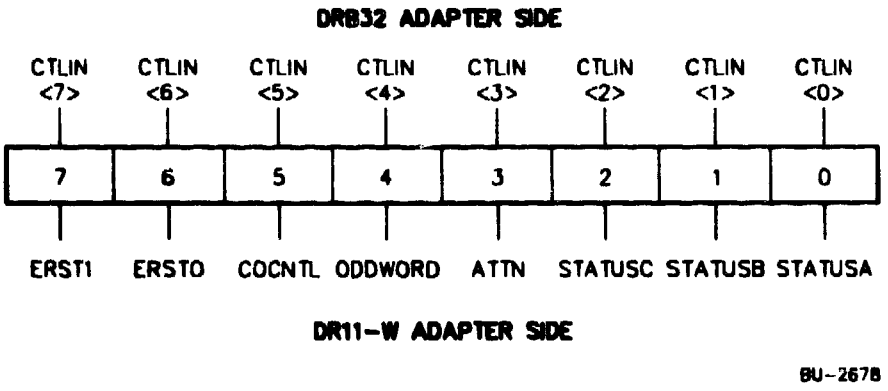


Figure C-4 I/O Control Register - Input Bytes

STATUS C, STATUS B, STATUS A - CTL IN <2:0>. The Status bits are general purpose control bits that can pass protocol from the user device via the DR11-W port to the DRB32 software driver by means of the DRB32 control port. These signals have no effect on the T1023.

ATTN - CTL IN <3>. The ATTN bit, when asserted, causes the DRB32 control port signal CTL SYNC IN to be asserted. This causes the DRB32 to issue a VAXBI Interrupt (if the appropriate Interrupt Enable bit is set). The ATTN flip-flop can be cleared by setting and then clearing PURGE (CTL OUT<4>) while READY (CTL OUT<5>) bit is set, or by setting the INIT bit (CTL OUT<3>).

ODD WORD - CTL IN <4>. When set, this bit indicates that the low word is valid on D<31:0> during transfers to the DRB32 from the user device. When this bit is set at the end of a transaction, a "dangling word" is stored in the T1023 buffer. This last word can be retrieved by setting and then clearing the PURGE (CTL OUT<4>) bit while the READY (CTL OUT<5>) bit is clear.

C0 CNTL - CTL IN <5>. C0 CNTL is a DR11-W signal used to request a byte transaction from the user device to the DRB32-W.

ERST0 - CTL IN <6>. Error Status bit 0. See Table C-3 below.

ERST1 - CTL IN <7>. Error Status bit 1. See Table C-3 below.

Table C-3 Error Status Bits

ERST1	ERST0	Error Condition
0	0	None
0	1	DATIP
1	0	MULTI CYC REQ
1	1	BUS DIR Conflict

FNCT 3, FNCT 2, FNCT 1 - CTL OUT <2:0>. The FNCT bits are general purpose control bits that can pass protocol from the DRB32 software driver to the DR11-W port, by means of the DRB32 control port and the T1023.

INIT - CTL OUT <3>. This bit is driven directly by the DRB32 software, and is set when the DR11-W user device is to be initialized. INIT clears the ATTN flip-flop.

PURGE - CTL OUT <4>. This bit has three possible functions:

1. The 32-bit data is transferred (purged) from the DRB32-W buffer to the DRB32-M by setting and then clearing the PURGE bit while the READY bit is clear. This is used by the software driver to transfer a "dangling word" from the DRB32-W to the DRB32-M module, if ODD WORD was set at the end of a transaction.

2. The ATTN flip-flop is cleared by setting and then clearing the PURGE bit while the READY bit is set.

3. The module is caused to enter Internal Loopback mode by setting and then clearing the PURGE bit while the READY bit is set and in Maint mode (PIO=0; DOHICTL=1). The Internal Loopback mode is cleared by exiting Maint mode.

READY - CTL OUT <5>. When set, this bit indicates that the DRB32 is ready to perform a DMA transaction. When this bit is clear, the DRB32 is setting up a DMA transfer or doing a DMA transfer. This bit is driven directly by the software. By clearing this bit, a transaction is allowed to begin. By setting this bit, a hung state in the T1023 is cleared by forcing the T1023 controller to return to its idle state. The READY bit must be set during Programmed I/O (PIO) mode and to enter or exit Internal Loopback mode.

DOHICTL - CTL OUT <6>. This bit controls the DRB32-W DO<15:0> H drivers in Programmed I/O mode and must be deasserted (0) during DMA operation. When this bit is asserted, the high word of the D<31:0> H bus (bits 16-31) is driven onto DO<15:0> H. When this bit is deasserted, the low word of the D<31:0> H bus (bits 0-15) is driven onto DO<15:0> H. When receiving data in Programmed I/O (PIO) mode, the DI<15:0> word appears simultaneously on D<31:16> and D<15:0>.

PIO - CTL OUT <7>. The bit has two functions, depending on whether the Ready bit is set or clear:

1. The DRB32-W enters Programmed I/O (PIO) mode by setting PIO while the Ready bit is set. When the PIO bit is set, the DIR IN H signal is DIR OUT H inverted. When PIO is clear, the DIR IN H signal is controlled by CI CNTL H.

2. A SOFT CYCLE pulse is generated by setting and then clearing PIO while the Ready bit is clear. The SOFT CYCLE pulse is used to begin a DMA WRITE to the user device or a DMA READ from the user device.

C.4.2 IODAT - I/O Data Register

The I/O Data Register (see Figure C-5) contains the current value of the DRB32 parallel port data lines.

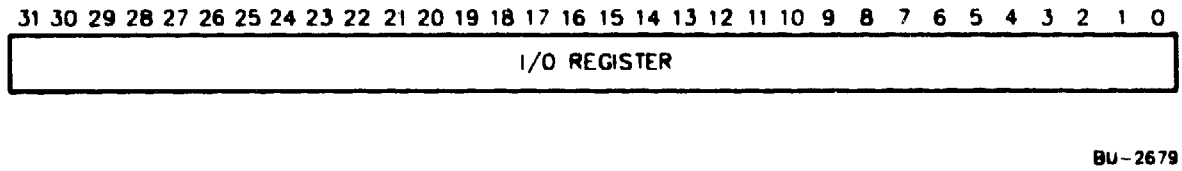


Figure C-5 T1023 I/O Data Register

During Programmed I/O mode, the DO CTL bit of the IODAT Register controls the value of the data lines.

To write data to a T1023 via the I/O Register, the I/O data port output tri-state drivers must first be enabled. To enable the tri-state output drivers, the DIR O bit in the I/O Set Up and Test Register must be set, and the T1023 must set the DIR I which is driven by the C1 CNTL signal of the DR11-W interface. This indicates both that the DRB32 is going to send data to the T1023 and that the T1023 is not asserting data on the I/O data port. When this condition is met, any WRITE to the I/O Register immediately asserts D<15:0> on the I/O data lines.

C.5 DMA TRANSACTIONS AND TIMING DIAGRAMS

Since the DRB32 is optimized for large DMA transfers, some examples and timing diagrams of DMA transfers are shown here. This section assumes that all necessary set-up information required by the software drivers at both ends is passed by means of the STATUS <C,B,A> and FUNCT<3,2,1> lines, which are connected to each other for this purpose. It is also assumed that all registers appropriate to the transaction are correctly set up in the DRB32.

C.5.1 DRB32 WRITE to the DR11-W Interface

The DRB32 uses the following procedure when it issues a WRITE transaction to the DR11-W interface. To initiate a transaction, the user device asserts ATTN H, which in turn asserts CTL SYNC IN H at the DRB32 interface. This assertion interrupts the VAXBI processor.

The DRB32 software driver must read the DRBIFR Register (in the DRB32) to determine the nature of the interrupt. The driver then reads the ATTN H bit at bit position CTLIO<3> to determine that the T1023 has initiated the interrupt. When ready to proceed, the DRB32 software driver clears the READY H bit in CTLIO<13>.

At this point, the T1023 asserts GO H for 200 ns and waits for the user device to assert CYCLE RQ A H or CYCLE RQ B H. Before asserting CYCLE RQ A H or CYCLE RQ B H, the user device drives C1 CNTL H and C0 CNTL H appropriately. The C1 CNTL H line must be deasserted for data to flow from the DRB32 to the DR11-W interface.

When CYCLE RQ A H or CYCLE RQ B H is received, the T1023 asserts BUSY H. At this time the DRB32 low word is driven onto DO <15:0>. Note that the data from the DRB32 might not yet be stable.

The T1023 waits for the DRB32 to assert SYNC OUT H, signifying that data is stable on the D<31:0> H lines. The T1023 then latches the 32-bit DATA, pulses END CYCLE H, and deasserts BUSY H.

The T1023 now waits for the next CYCLE RQ A H or CYCLE RQ B H. When these signals are received, the T1023 asserts BUSY H, drives the DRB32 high word onto the DO <15:0> lines, and asserts SYNC IN.

The T1023 then pulses END CYCLE H, and deasserts BUSY H. The T1023 then waits for the DRB32 to deassert SYNC OUT H.

When the T1023 receives the deassertion of SYNC OUT H, it deasserts SYNC IN and returns to its Start state. The cycle can then be repeated until the transfer is complete, at which point the DRB32 software driver sets READY H in CTLIO <13>.

If a DMA transfer is to contain an odd number of words, the last word can only be transferred when the DRB32 software driver asserts (sets) PURGE in CTLIO <12>.

C.5.2 DRB32 READ of the DR11-W Interface

The DRB32 uses the following procedure when it issues a READ transaction to the DR11-W interface.

To initiate a transaction, the user device asserts ATTN H, which in turn asserts CTL SYNC IN H at the DRB32 interface. This causes an interrupt. The DRB32 can read the ATTN H bit at bit position CTLIO<3>.

When ready to proceed, the DRB32 sets up the direction control bits, and clears the READY H bit in CTLIO <13>. At this point, the T1023 asserts GO H for 200 ns, and waits for the user device to assert CYCLE RQ A H or CYCLE RQ B H and C1 CNTL H and C0 CNTL H. C1 CNTL H must be asserted (must be 1) for data to flow to the DRB32 from the DR11-W interface.

When CYCLE RQ A H or CYCLE RQ B H is received, the T1023 asserts BUSY H. The T1023 waits 200 ns, then closes the low T1023 latch, then BUSY H deasserts.

The T1023 now waits for the next CYCLE RQ A H or CYCLE RQ B H. When one is received, the T1023 waits 200 ns, then closes the high T1023 latch, pulses END CYCLE H, asserts SYNC IN H and deasserts BUSY H. The T1023 then waits for SYNC OUT H to be asserted. The assertion of SYNC OUT H from the DRB32 indicates to the T1023 that the longword of data has been received.

When SYNC OUT H is received, the T1023 deasserts SYNC IN H, opens low and high latches, and returns to its Start state and waits for more DMA transactions, until the DMA transfer is complete. At this time the DRB32 software driver sets READY H in CTLIO <13>.

If a DMA transfer contains an odd number of words, the last word can only be transferred when the DRB32 software driver asserts (sets) the PURGE bit in CTLIO <12>.

C.5.3 DMA Timing for a DRB32 WRITE to the DR11-W User Port

Figure C-6 and Table C-4 show the DMA timing for a DRB32 WRITE transaction to the DR11-W user port.

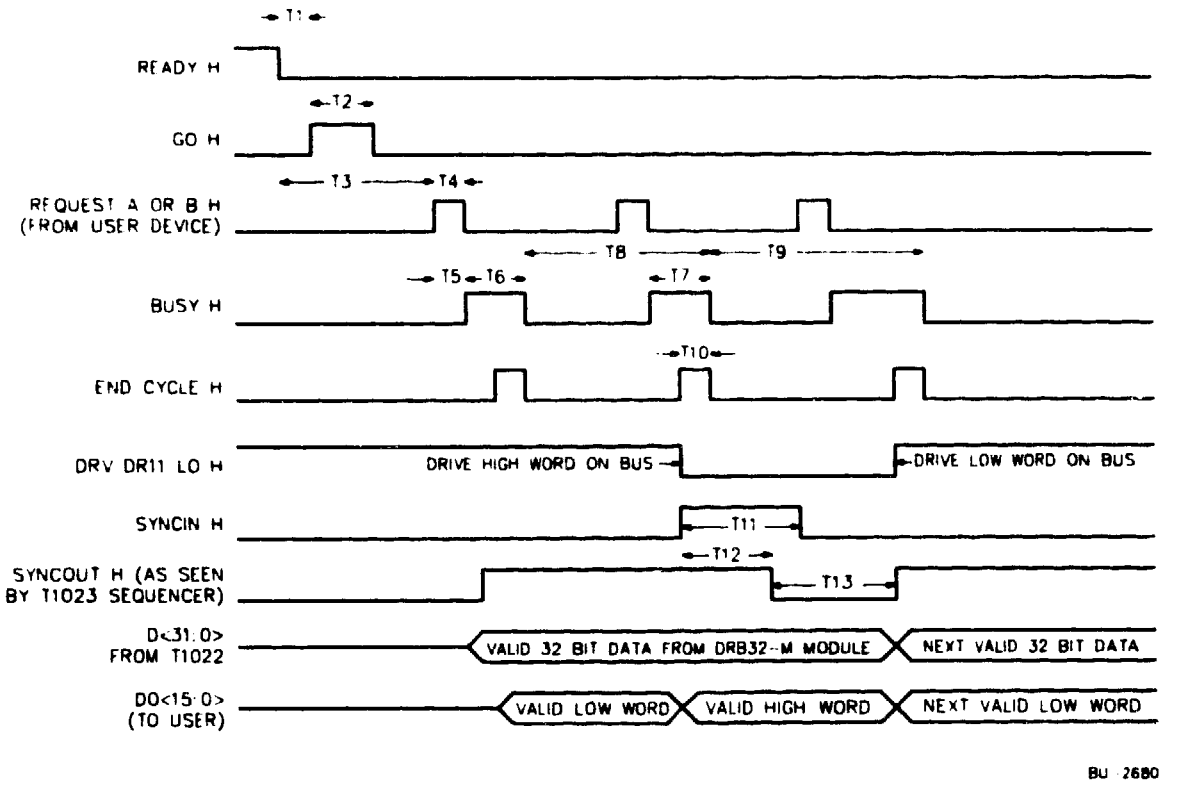


Figure C-6 Timing for DRB32 WRITE (Transfer from DRB32-W to User Device)

Table C-4 Timing for DRB32-W WRITE (Transfer from DRB32-W to User Device)

Time	Min	Max	Description
T1	100 ns	NA	From READY to GO
T2	200 ns	NA	GO pulse width
T3	0 ns	No Max	From READY to REQUEST A or B
T4	120 ns	No Max	REQUEST A or B pulse width
T5	100 ns	NA	From REQUEST A or B to BUSY
T6	200 ns	No Max	BUSY pulse width - first word
T7	200 ns	NA	BUSY pulse width - second word
T8	400 ns	No Max	From BUSY to BUSY - first word
T9	800 ns	No Max	From BUSY to BUSY - second word
T10	100 ns	NA	END CYCLE pulse width
T11	400 ns	No Max	SYNC IN pulse width
T12	300 ns	No Max	From SYNC IN to SYNC OUT
T13	400 ns	No Max	From SYNC OUT deassertion to SYNC OUT reassertion

C.5.4 DMA Timing for a DRB32 READ of the DR11-W User Port

Figure C-7 and Table C-5 show the DMA timing for a DRB32 READ transaction to the DR11-W user port.

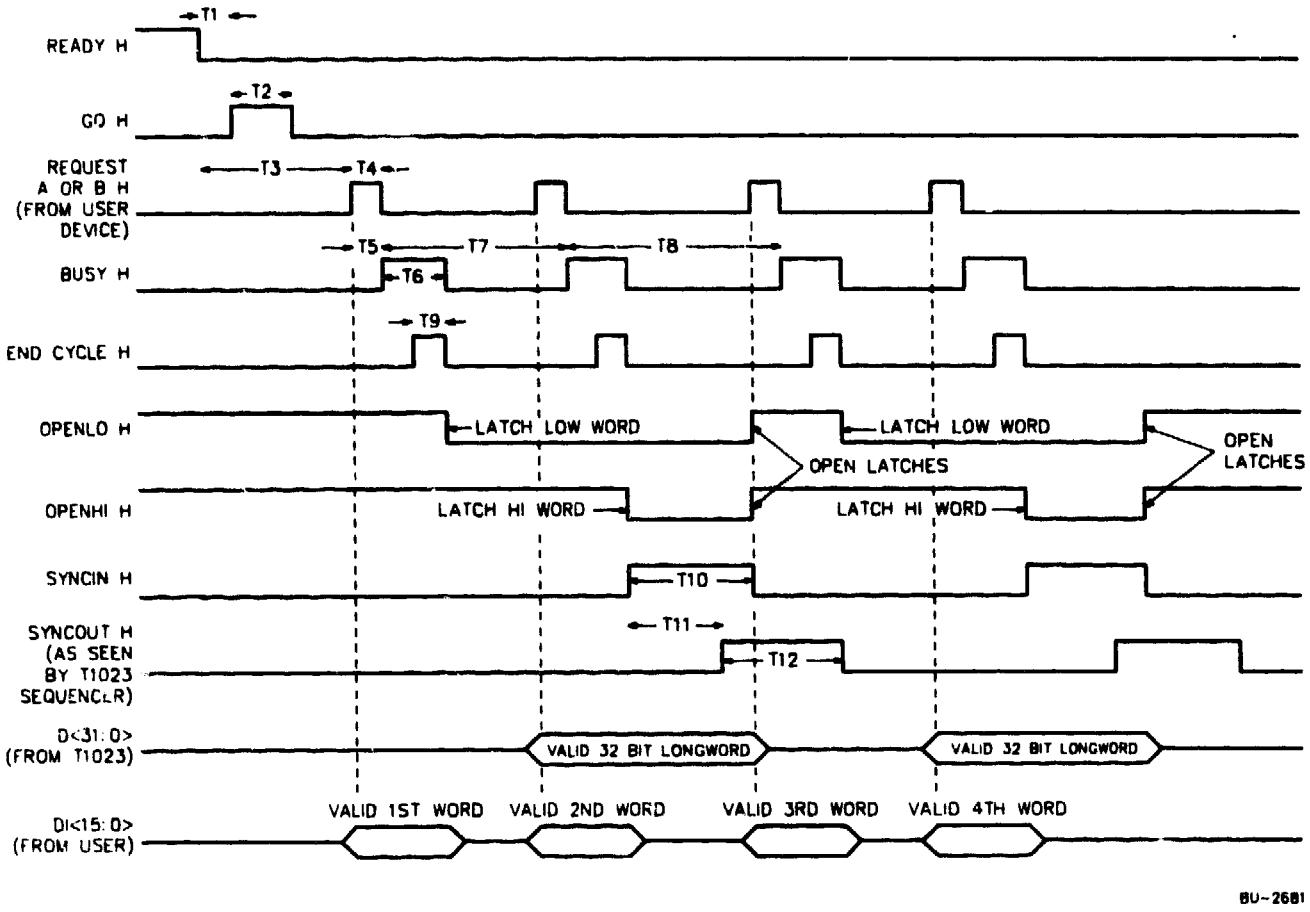


Figure C-7 Timing for DRB32-W READ (Transfer from User Device to DRB32-W)

Table C-5 Timing for DRB32-W READ (Transfer from User Device to DRB32-W)

Time	Min	Max	Description
T1	100 ns	NA	From READY to GO
T2	200 ns	NA	GO pulse width
T3	0 ns	No Max	From READY to REQUEST A or B
T4	120 ns	No Max	REQUEST A or B pulse width
T5	100 ns	NA	From REQUEST A or B to BUSY

Table C-5 Timing for DRB32-W READ (Transfer from User Device to DRB32-W) (Continued)

Time	Min	Max	Description
T6	200 ns	NA	BUSY pulse width
T7	500 ns	No Max	From BUSY to BUSY - first word
T8	600 ns	No Max	From BUSY to BUSY - second word
T9	100 ns	NA	END CYCLE pulse width
T10	400 ns	No Max	SYNC IN pulse width
T11	300 ns	No Max	From SYNC IN to SYNC OUT
T12	400 ns	No Max	SYNC OUT pulse width

C.6 PROGRAMMED I/O MODE

Programmed I/O is a data transfer mode that requires the software drivers to intervene and be active in each data transfer cycle. During programmed I/O mode, all signals used for DMA are silent. Programmed I/O mode is also called "data mode" by the VMS device drivers. Protocol information passes from the user software driver to the DRB32 software driver by means of the Funct and Status bits which are connected to the DRB32 by means of the CTLIO port.

To end Programmed I/O mode, both the PIO bit CTLIO<11> and the Ready bit (CTLIO<13>) must be set. The Ready bit must never be cleared during Programmed I/O mode. This allows the direction of the DRB32 DATA<31:0> H drivers to be set by the DIR OUT H signals.

DO CTL H CTLIO<14> controls which word of the longword on the D<31:0> H line is driven onto DO<15:0> H. The high word is driven when this bit is set, and the low word is driven when this bit is clear.

C.6.1 Programmed I/O WRITEs to the DR11-W Port

Programmed I/O WRITE transactions to the DR11-W port follow the procedure below.

1. The DRB32 software driver writes the I/O data register and the DIR OUT H bit. This allows the data to appear on the D<31:0> lines.

2. The DRB32 software driver writes the DO HI CTL H (CTLIO<14>) bit to select either D<31:16> (CTLIO<14> set) or D<15:0> (CTLIO<14> clear).

3. The DRB32 software driver then writes the appropriate Funct bits to notify the user software driver that data is available.

4. When the user software driver has accepted the data on the DO<15:0> lines it must notify the DRB32 software driver by writing the Status bits appropriately and asserting ATTN H.

C.6.2 Programmed I/O READs to the DR11-W Port

Programmmed I/O READ transactions follow the procedure below.

1. When the user device presents data to the T1023 on the DI<15:0> lines, this data appears on the DRB32 D<31:0> H lines, if the DRB32 software driver has cleared the DIR OUT H bit. Note that the DI<15:0> data appears simultaneously on D<31:16> and D<15:0>.

2. The user software driver then notifies the DRB32 software driver that data is available by setting the Status bits appropriately and asserting the ATTN H signal.

3. When the data is accepted, the DRB32 software driver responds by writing the Funct bits appropriately.

C.7 ELECTROMECHANICAL SPECIFICATIONS

The electromechanical specifications for the T1023 include:

- Power requirements
- Environmental requirements
- Electrical specifications

C.7.1 Power Requirements

The power requirements for the T1023 are shown in Table C-6.

Table C-6 T1023 Power Requirements

Voltage	Tolerance	I(Amps)
5.00	± 250mV	2.0

C.7.2 Environmental Requirements

The operating environment requirements for the T1023 are shown in Table C-7.

Table C-7 T1023 Operating Environment

Parameter	Condition
Temperature	5°C to 50°C (41°F to 122°F)
Humidity	10% to 95% with maximum wet bulb of 32°C (90°F) and minimum dew point of 2°C (36°F)
Altitude	To 2.4 km (8,000 ft) non-condensing

Table C-8 shows the T1023 storage environment.

Table C-8 T1023 Storage Specifications

Parameter	Condition
Temperature	−40°C to 66°C (−40°F to 151°F)
Humidity	To 95% non-condensing
Altitude	To 9.0 km (30,000 ft)

C.7.3 T1023 Electrical Specifications

Table C-9 shows the electrical specifications for the T1023.

Table C-9 8881 Driver Output DC Specifications

Parameter	Min	Max	Conditions
Output Low Voltage (VOL)	NA	.80 V	Output current = 70 MA
High Leakage Current (IOH)	NA	25 μA	Output voltage = 3.5 V
Output High Voltage (VOH)	2.5 V 3.5 V	180/390 (± 5%) Ω	Terminators

Table C-10 shows the receiver input DC specifications.

Table C-10 8640 Receiver Input DC Specifications

Parameter	Min	Max	Conditions
Input High Voltage (VIH)	1.7 V	NA	
Input Low Voltage (VIL)	NA	1.3 V	
Input Low Current (IIL)	NA	10 μA	Input voltage = 0.0 V
Input High Current (IIH)	NA	80 μA	Input voltage = 2.5 V

C.7.3.1 Cables – The interface cables needed to connect to the T1023 are the BCO6R-XX 40-conductor, 120-Ω cables. The maximum length is 50 feet (15.25 meters) of cabling between the VAXBI backplane and the user device. For example, if the internal cables from the backplane to the bulkhead connector are 15 feet (4.5 meters), then the BCO6R-XX cables from the bulkhead connector to the user device can be up to 35 feet (10.6 meters) long, for the maximum total of 50 feet (15.25 meters).

C.7.3.2 DR11-W Terminators – All DR11-W input and output signals are terminated in a 180/390 center-tapped terminator VCC to GND. This presents a Thevenin impedance of 120 Ω and a Thevenin voltage of 3.3 V. The user should terminate all signals in the same fashion.

C.7.3.3 User Drivers and Receivers – User devices should use the same drivers and receivers used on the T1023. Figure C-8 shows the user device driver configuration.

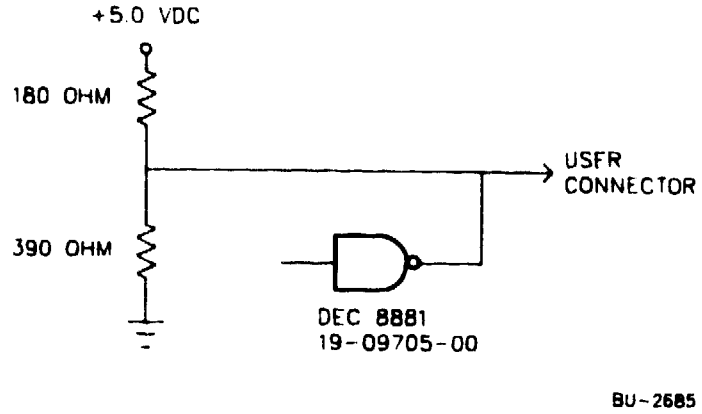


Figure C-8 User Driver Configuration

Figure C-9 shows the user device receiver configuration.

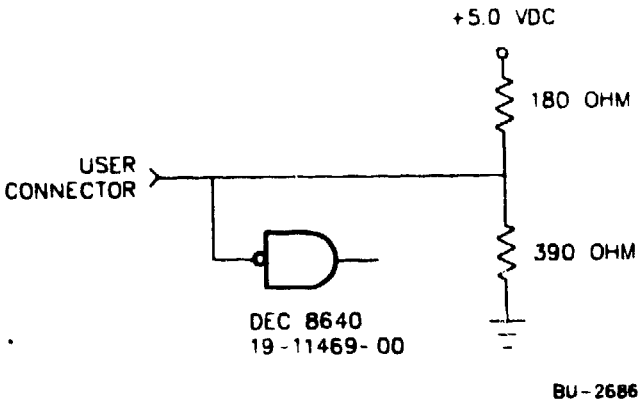


Figure C-9 User Receiver Configuration

C.8 INTRODUCTION TO T1023-YA

The purpose of this section is to provide a description of the new DRB32-W (T1023-YA) module. This section provides the hardware and software information needed to support the use of the new DRB32-W (T1023-YA). This section discusses the features, limitations, and implementation of these new features. The goal of this section is to provide users of the DRB32-W (T1023-YA) with an easy path in which to utilize the new DRB32-W module.

The following descriptions are referenced in this document.

- MODULE MARKING - The top of the T1023-YA module is stamped with T1023-YA.
- OLD BOARD MODE - In this mode the T1023-YA runs as the T1023-00 module. None of the added features is implemented.
- NEW BOARD MODE - In this mode the T1023-YA uses its on-board counter, and other new features that are not available with the T1023-00 module.
- NEW DRB32-W - This refers to the T1023-YA module.
- OLD DRB32-W - This refers to the T1023-00 module.

C.9 HARDWARE FEATURES OF THE T1023-YA

The new DRB32-W module is an enhancement to the current (OLD) DRB32-W module, these enhancements are discussed in detail below.

Design changes include:

With the addition of a readable word counter, user bus data transactions terminate in a normal controlled manner.

- READY H is asserted at the end of a transfer via hardware, not software. When all data is transferred, the counter sets a DONE bit. This DONE signal causes the user ready to be asserted.

NOTE
READY H is now set before BUSY H deasserts at the end of a transfer.

- READY H is asserted if IATTN, or error during DMA transfer occurs. If an error condition is received or IATTN is set, the user bus ready is set via hardware; the DONE signal is also set.
- FPLS modifications allow busy assertion/deassertion to match DR11-W hand shaking more closely.
- ODD WORD transfers are supported with the new DRB32-W module. ODD WORD READ and ODD WORD WRITE transfers are supported by the new DRB32-W.
- Purge is no longer needed to transfer the last word of an ODD WORD transfer. The purge capability still exists; it simply is no longer necessary in this instance.

UQWDRIVER VMS software can determine if a module is a new DRB32-W or an old DRB32-W. This is accomplished by using the software to test the module for new board logic. The software also has the ability to put a new DRB32-W into new or old board mode.

The T1023-YA module is completely backward compatible with the "SAMPLE" UQWDRIVER supplied with the driver kit. In old board mode, the T1023-YA runs as an old DRB32-W.

Data integrity of user control bus is maintained when writing to or reading from the DRB32-W word counter. The DRB32-W keeps the user control lines at the state to which they were previously set before the command to read or write the counter was issued. This prevents any control lines from toggling or being changed in any way while reading/writing the counter.

The DRB32-W word counter allows correct, accurate data to be retrieved by a user (via the driver) during abnormal completion of data transfers. Software is able to restart from the last valid word transferred. Software accomplishes this by reading the word counter, which has the correct word transfer count.

The new DRB32-W module's DONE bit is readable by the driver whenever the IODAT lines are not being utilized. The DONE bit does not have to be checked. The TRANSFER completes properly without checking the DONE bit, by using the DRB32-M and DRB32-W protocol.

C.9.1 Timing Considerations for READ and WRITE transactions

READ and WRITE transactions can be started with the CYCLE modifier in the UQWDRIVER. This qualifier enables the DRB32-W to start a transaction for the user device. If the user device has no way of setting a cycle request A or B, then the CYCLE modifier can be used.

If the user device can supply a cycle request A or B, then the CYCLE modifier should not be used. When not using the CYCLE modifier, refer to the timing diagrams on pages C-11 and C-13 of this manual.

The timing when using the CYCLE modifier is shown in the following timing diagrams. Please note the CYCLE modifier is listed as SOFT CYCLE in these diagrams. SOFT CYCLE supplies BUSY H to the user device. The user device has to supply CYCLE request A or B after the first BUSY is set by the CYCLE modifier.

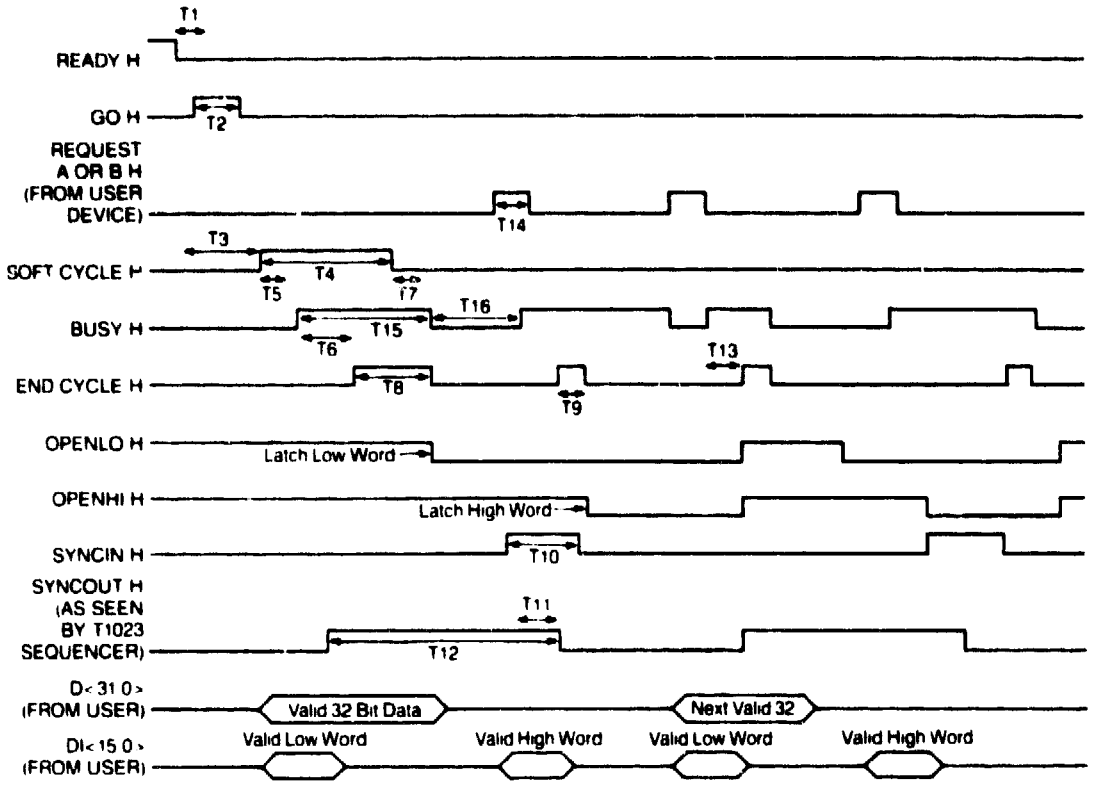


Figure C-10 Timing for DRB32-W WRITE (Transfer from DRB32-W to User Device) Transfer Started by SOFT CYCLE

Table C-11 Timing for DRB32-W WRITE (Transfer from DRB32-W to User Device) Transfer Started ' SOFT CYCLE

Time	Min	Max	Description
T1	100 ns	N/A	READY deassertion to GO assertion
T2	200 ns	N/A	GO pulse width
T3	0 ns	No Max	From READY assertion to SOFT CYCLE assertion
T4	200 ns	No Max	SOFT CYCLE pulse width
T5	100 ns	N/A	From SOFT CYCLE assertion to BUSY assertion
T6	300 ns	N/A	From BUSY assertion to END CYCLE assertion
T7	100 ns	N/A	From SOFT CYCLE deassertion to BUSY deassertion
T8	100 ns	No Max	END CYCLE first pulse width
T9	100 ns	N/A	All additional END CYCLE pulse width
T10	400 ns	No Max	SYNC IN pulse width
T11	300 ns	No Max	From SYNC IN to assertion of SYNC OUT deassertion
T12	400 ns	No Max	SYNC OUT pulse width

Table C-11 Timing for DRB32-W WRITE (Transfer from DRB32-W to User Device) Transfer Started by SOFT CYCLE (Continued)

Time	Min	Max	Description
T13	200 ns	N/A	From BUSY assertion to END CYCLE assertion, other than the first word of transfer
T14	120 ns	No Max	Request pulse width
T15	300 ns	No Max	BUSY pulse width
T16	100 ns	No Max	From BUSY deassertion to BUSY assertion

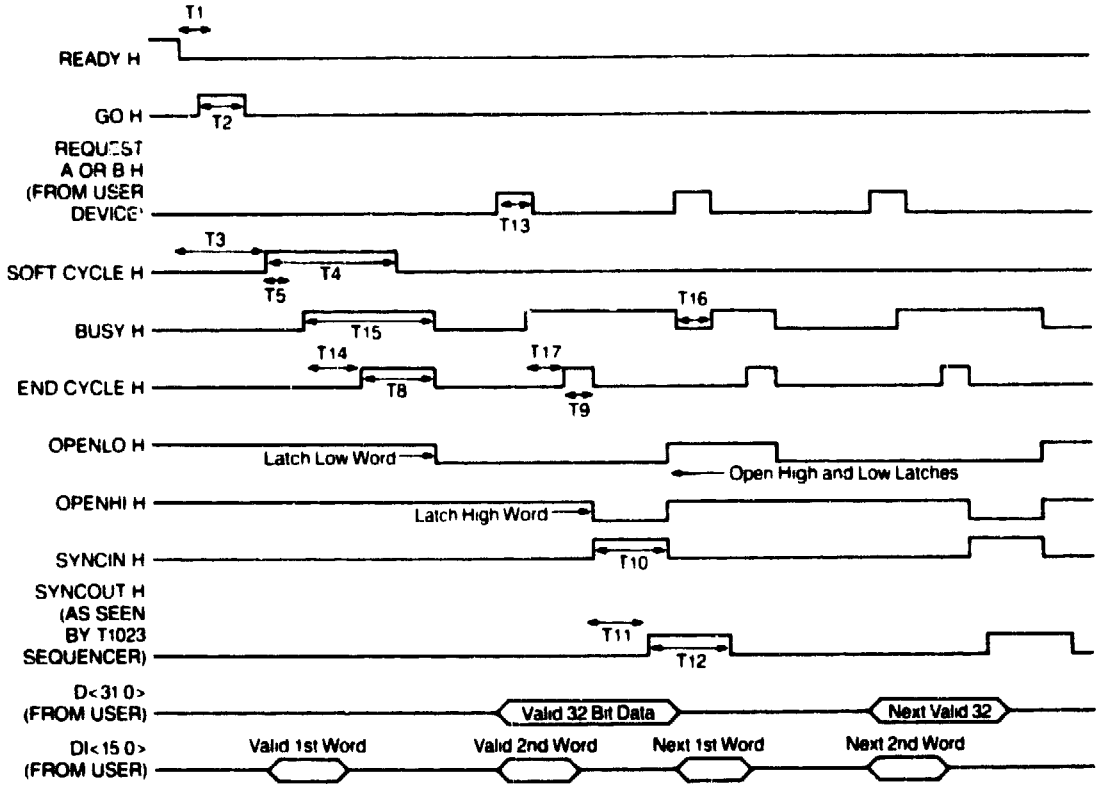


Figure C-11 Timing for DRB32-W READ from User Device Transfer Started by SOFT CYCLE

Table C-12 Timing for DRB32-W READ from User Device Transfer Started by SOFT CYCLE

Time	Min	Max	Description
T1	100 ns	N/A	READY deassertion to GO assertion
T2	200 ns	N/A	GO pulse width
T3	0 ns	No Max	From READY assertion to SOFT CYCLE assertion
T4	200 ns	No Max	SOFT CYCLE pulse width

Table C-12 Timing for DRB32-W READ from User Device Transfer Started by SOFT CYCLE (Continued)

Time	Min	Max	Description
T5	100 ns	N/A	From SOFT CYCLE assertion to BUSY assertion
T6	300 ns	N/A	From BUSY assertion to END CYCLE assertion
T7	100 ns	N/A	From SOFT CYCLE deassertion to BUSY deassertion
T8	100 ns	No Max	END CYCLE first pulse width
T9	100 ns	N/A	All additional END CYCLE pulse width
T10	400 ns	No Max	SYNC IN pulse width
T11	300 ns	No Max	From SYNC IN assertion to SYNC OUT assertion
T12	400 ns	No Max	SYNC OUT pulse width
T13	120 ns	No Max	Request pulse width
T14	300 ns	N/A	From BUSY assertion to END CYCLE assertion, for the first word of a DMA transfer
T15	300 ns	No Max	BUSY pulse width
T16	100 ns	No Max	From BUSY deassertion to BUSY assertion
T17	200 ns	N/A	From BUSY assertion to END CYCLE assertion, for all but first word of DMA transfer

C.10 SOFTWARE

C.10.1 Driver Modifications to allow Use of the Word Counter Functionality

In order to invoke the on-board word counter functionality of the DRB32-W, three sections of code were added. All code is clearly marked with comments beginning with ";." A brief description of the changes is given below. For a more detailed look at these changes, please refer to the driver source code, and programmer's manual included in this kit.

To incorporate the instructions that put the DRB32-W module in the mode that utilizes the word counter functionality, place a semicolon at the start of the line immediately preceding the label 0$: in the CONTROLLER_INIT routine. This line is clearly marked within the source code.

The changes to UQWDRIVER are described below:

In routine CONTROLLER_INIT (the driver's controller initialization routine), code was added that invokes the word counter functionality in the DRB32-W module. This code utilizes a sequence of function codes written to the IOCTL to achieve this result.

- In routine Block_Mode, several lines of code have been added. These instructions load the word count value into the hardware's on-board word counter. This is accomplished by issuing a series of function codes that cause data in the IODAT Register to be loaded into the word counter.

- In routine Set_up_r0_r1, function code reads the word counter value in the hardware's on-board word counter when an incomplete transaction occurs. This allows the accurate transfer byte count to be returned by the IOSB.

NOTE

To return the module to OLD board mode, remove the semicolon at the start of the line immediately preceding the label 0$: in the CONTROLLER_INIT routine. This line is clearly marked within the source code. Failure to remove the semicolon causes incorrect transfer completion, when trying to use old board mode.

Please refer to the hardware section for more details about the function codes used in turning on, loading, and reading the word counter.

C.10.2 Diagnostics

Table C-13 Diagnostic Tests in EVDRI

Test Number	Name	Function
1	DRB32-W Internal Loopback Longword DMA Test	Tests the ability of the DRB32-W to transfer longwords via DMA transactions
2	DRB32-W Internal Loopback Byte DMA Test	Tests the ability of the DRB32-W to transfer bytes via DMA transactions
3	DRB32-W Internal Loopback ATTN H Interrupt Test	Tests the ability of the DRB32-W to respond to and clear an ATTN H interrupt
4	DRB32-W Internal Loopback Direction Error Interrupt Test	Tests the ability of the DRB32-W to respond to a direction error interrupt and clear the error condition
5	DRB32-W Internal Loopback DATAIP Error Interrupt Test	Tests the ability of the DRB32-W to respond to a Data In Pause error interrupt and clear the error condition
6	DRB32-W External Loopback Longword DMA Test	Tests the ability of the DRB32-W to transfer longwords via DMA transactions
7	DRB32-W External Loopback Byte DMA Test	Tests the ability of the DRB32-W to transfer bytes via DMA transactions
8	DRB32-W External Loopback EVEN Word Transfer Test	Tests reliability of the DRB32-W word counter functionality during EVEN word transfers
9	DRB32-W External Loopback ODD WORD Transfer Test	Tests reliability of the DRB32-W word counter functionality during ODD WORD transfers

Table C-13 Diagnostic Tests in EVDRI (Continued)

Test Number	Name	Function
10	DRB32-W External Loopback ATTN H Interrupt Test	Tests the ability of the DRB32-W to respond to and clear ATTN H interrupt
11	DRB32-W External Loopback Direction Error Interrupt Test	Tests the ability of the DRB32-W to respond to a direction error interrupt and clear the error condition
12	DRB32-W External Loopback DATAIP Error Interrupt Test	Tests the ability of the DRB32-W to respond to a Data In Pause error interrupt and clear the error condition
13	DRB32-W Single-System LINK Test T1023-YA	Tests the ability of two DRB32-W modules (T1023-YA) to pass messages in PIO and DMA
14	DRB32-W Single-System LINK Test T1023-00	Tests the ability of two DRB32-W modules (T1023-00) to pass messages in PIO and DMA

C.11 LIMITATIONS OF THE T1023-YA

The new DRB32-W runs the SAMPLE UQWDRIVER supplied with the software kit. When modifying the driver, all the implementation rules and warnings must be followed for correct use of all the new DRB32-W features. Please refer to the *DRB32 Programmer's Manual* for information on the use of this sample driver.

ODD BYTE transfers are not supported. (Nor were they supported with the DR11-WA.) The user must supply the DRB32-W counter with the correct number of words to transfer. ODD WORD READ and WRITE transfers are supported with the new DRB32-W module. Failure to load the correct count into the new DRB32-W word counter results in unpredictable data transfers.

C.12 IMPLEMENTATION OF T1023-YA FEATURES

The new features on the T1023-YA are implemented by using the IOCTL Register. Please refer to the programmer's guide to make sure that the correct changes have been made to the driver. To access the T1023-YA features, the following process is implemented in the software driver.

NOTE

This process is only supplied to better explain how the new T1023-YA features are implemented. The DRB32-W driver (UQWDRIVER) uses this process to access all the T1023-YA features. This process must be implemented if any driver other than UQWDRIVER is used.

C.12.1 Process

Please note that when writing the IOCTL a Change of State Interrupt is created; For this reason the user should have Interrupts disabled (DRBIE Register), before writing to the IOCTL Register, unless the user wants this interrupt. After the manipulation of the IOCTL is complete, any pending interrupts (DRBIFR Register) must be cleared. When implementing the new board mode features, make sure interrupts are disabled and pending interrupts are cleared following the method mentioned above.

The following features must be implemented while in LOOPBACK mode. LOOPBACK mode is entered by setting (asserting) READY, PURGE, and MAINTENANCE bits in the IOCTL. If the user deposits a 7000 into the IOCTL Register, the user enters LOOPBACK mode.

Once the desired feature or features have been implemented, the module must be taken out of LOOPBACK mode. To exit LOOPBACK mode, clear (deassert) MAINTENANCE and PURGE, but keep READY asserted.

C.12.2 READ Implementation

To read the contents of the new DRB32-W counter, follow the procedure below.

1. Clear DRBIE.
2. Save PARSTR Register.
3. Clear bit 4 in the PARSTR Register.
   This sets the direction of information. Direction of transfer is input from the user to the DRB32.
4. Write the IOCTL with a 7500.
   This tells the logic to put the DRB32-W word counter bits <0:15> onto the IODAT lines.
5. Read IODAT Register.
6. IODAT bits < 0:15 >
   These bits are the counter information, in two's complement form. The whole counter is not readable, however bits <0:14> supply the user with more than enough bits to verify the counter.
7. IODAT bit < 15 >
   This bit is the DONE bit. If this bit is SET, the new module counter has completed all its requested transfers. If this bit is CLEAR, the module has not completed its requested transfer.
8. Restore PARSTR to its original state.
9. Write IOCTL with 2000.
   This takes the module out of LOOPBACK mode. This command must be issued to take the module out of LOOPBACK mode; failure to do this could cause many problems.
10. Clear DRBIFR.
11. Restore DRBIE.

C.12.3 LOAD Implementation

To load the new DRB32-W counter with the TRANSFER size, follow the procedure below. The counter keeps track of words transferred.

1. Use the CURRENT BYTES LEFT Register.
2. Convert the current bytes left into a word count.
3. Do a two's complement of the word count.
4. Load this count into the DRB32-W counter, as follows.
   a. Clear DRBIE.
   b. Save PARSTR Register.
   c. Write the IOCTL with a 7200.
      This prepares the counter to receive data.
   d. SET bit 4 in the PARSTR Register.
      This sets the direction of information. Direction of transfer is output to the user from the DRB32.
   e. Write the IODAT Register with the complemented word count.
   f. Write the IOCTL with a 7600.
      This loads the word counter value into the word counter on the new DRB32-W.
   g. Write the IOCTL with a 7000.
      Just clearing LOOPBACK mode after setting 7600 could leave the counter in an unknown state. By writing the IOCTL with a 7000, the counter is put into a known state, which is ready to start counting.
   h. Restore PARSTR to its original state.
   i. Write IOCTL with 2000.
      This takes the module out of LOOPBACK mode. This command must be issued to take the module out of LOOPBACK mode; failure to do this could cause many problems.
   j. Clear DRBIFR.
   k. Restore DRBIE.

C.12.4 New DRB32-W Counter Clock

The clock for the new DRB32-W counter works in the following way.

1. The counter clocks every time a word is read or written.
2. The counter clocks every word or 2 bytes, no matter what mode of data transfer has been chosen.

C.12.5 T1023-YA Verification (Make sure the DRB32-W is a T1023-YA)

If the top of the module has a T1023-YA stamped on it, the module is a new board. If it is impossible to look at the module, new board/old board status can be determined by the following steps.

1. Clear DRBIE.
2. Write the IOCTL with a 7800.
   This sets up a condition in the logic that can determine if the module is a new or old board.
3. Read IOCTL.
   If ODD WORD (bit 4) of IOCTL is set, a new module (T1023-YA) is present. If ODD WORD (bit 4) of IOCTL is not set, an old module (T1023-00) is present.
4. Write IOCTL with 2000.
   This takes the module out of LOOPBACK mode. This command must be issued to take the module out of LOOPBACK mode; failure to do this could cause many problems.
5. Clear DRBIFR.
6. Restore DRBIE.

C.12.6 Setting New Board Mode

To set the module to new board mode, follow the procedure below.

1. Clear DRBIE.
2. Write IOCTL with a 7F00.
   This enables all the new board logic, on the new DRB32-W module. If this command is not issued, none of the new board mode features will work.
3. Write IOCTL with 2000.
   This takes the module out of LOOPBACK mode. This command must be issued to take the module out of LOOPBACK mode; failure to do this could cause many problems.
4. Clear DRBIFR.
5. Restore DRBIE.

C.12.7 Setting Old Board Mode

To put the new DRB32-W board in old board mode, issue the following commands.

NOTE

The module powers up in OLD board mode. If you put the new module (T1023-YA) into new board mode, you must restart the software driver with the needed corrections to set up old board mode. Just follow the step explained in section C.10.

1. Clear DRBIE.

2. Write the IOCTL with a 7100.
   This clears and disables all new board logic. Until the module is put in new board mode, this module will run as an old DRB32-W module.

3. Write IOCTL with 2000.
   This takes the module out of LOOPBACK mode. This command must be issued to take the module out of LOOPBACK mode; failure to do this could cause many problems.

4. Clear DRBIFR.

5. Restore DRBIE.

C.13 OTHER T1023-YA CONSIDERATIONS

C.13.1 Installation Instructions

To install the T1023-YA module, follow the instructions in the Installation Manual.

C.13.2 Timing Information

Other than the READY H signal, the timing diagrams are similar to those in the technical manual. NOTE: READY H is now set before BUSY H deasserts at the end of a transfer.

C.13.3 Specifications

Power requirements for the new T1023-YA are 3.2 Amps. All other specifications listed in this manual are unchanged.

C.13.4 Diagnostics

The method for running the diagnostics has not changed. Please refer to the *DRB32 Installation Guide.*

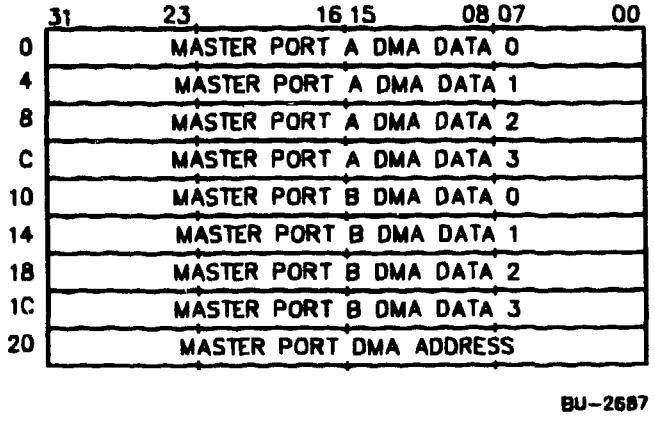

APPENDIX D

APPENDIX D
USING THE BCAI ACCESS REGISTER (BAR)

The DRB32-M contains a dual-octaword register file that is used to buffer data received from the user device. The register file is physically contained inside the BCI Adapter Interface Chip, a custom IC on the DRB32-M. This register allows the DRB32-M to pack four longwords into one octaword VAXBI transaction, making efficient use of VAXBI bandwidth. Because the DRB32-M buffers data received from the user device, it is possible that when the DRB32-M's Abort bit is set, some data will be left in the dual-octaword register file. The DRB32-M does not write this data to memory when an abort occurs. If your application requires that this data be retrieved, the BCAI Access Register, or BAR, must be used. The use of this register in retrieving stranded data is described below.

D.1 WHAT THE DUAL-OCTAWORD REGISTER FILE LOOKS LIKE

The diagram below shows the dual-octaword register file. The numbers on the left are the hexadecimal offset from the start of the register file.



The first eight longwords are the data registers. The last longword is the physical address in memory where the NEXT VAXBI transaction will be directed.

These registers are used by the DRB32-M to align data from any byte to a full longword and vice versa. The register file is byte-addressable. These registers can only be accessed as full longwords by a device driver using the BAR, which means you will obtain one full longword of data. Because the register file is byte-addressable, the longword you read can start on any byte boundary in the register file. The data registers wrap around. That is, after MASTER PORT B DMA DATA 3 is written, the next register written is MASTER PORT A DMA DATA 0.

To get data from these registers, write the address of a byte in the register file (a number from 0 to 20, hex, inclusive) to the BAR. The DRB32-M retrieves the longword of data starting at that byte from the BCAI and stores it in a temporary location. Then read from the BAR and get the data in the register.

D.2 HOW THE DRB32-M USES THE DUAL-OCTAWORD REGISTER FILE

When receiving data from the user port, the DRB32-M uses the dual-octaword register file. When a longword of data is obtained from the user device, the DRB32-M loads it into the next sequential four bytes of the register file. The first longword is written into the first octaword, and the starting byte is determined by the low four bits of the byte offset of the buffer. For example, if the byte offset is 17 hex, the first byte of data is loaded into the register file at byte 7 (not 17!).

Subsequent longwords of data are written into sequential locations in the register file. When the data crosses an octaword boundary, the DRB32-M starts a VAXBI transaction to write the filled octaword to memory (with the mask bits set as appropriate), and the DRB32-M continues to fill the second octaword. When the next longword to be written to the register file overwrites locations in the previous octaword, the DRB32-M stalls the user device until the previous VAXBI transaction has completed. Then, the DRB32-M writes the longword into the register file, starts another VAXBI transaction, and continues receiving data from the user port. Note that the second octaword wraps around into the first octaword.

The DRB32-M's CURBLFT register keeps track of the number of longwords that have been obtained from the user device. This value is decremented by the same microinstruction that loads the data into the BCAI register file. Thus, the value in CURBLFT always reflects the amount of data received from the user device, and, if a transfer successfully completes, it also reflects the amount of data written to memory. If an abort occurs in the middle of a parallel port data transfer, the value in CURBLFT does not accurately reflect the amount of data written to memory.

The MASTER PORT DMA ADDRESS register is incremented by 16 (that is, by one octaword) after it is read out of the BCAI as part of a VAXBI transaction. This increment operation is done automatically by the BCAI itself; it is not under the direct control of the DRB32-M microcode. Thus, the contents of the MASTER PORT DMA ADDRESS register always indicates the physical address in memory where the NEXT VAXBI write transaction will go.

The DRB32-M always writes full octawords to memory. That is, the next physical address used is always an octaword address, regardless of whether or not the full octaword is to be written, as the VAXBI mask bits take care of screening out unused bytes when a full octaword is not desired. The physical address generated by the DRB32-M for a data transfer to or from memory always has the low three bits clear. A special case exists for the situation in which a transfer is aborted before the DRB32-M has a chance to do any WRITEs to memory.

The DRB32-M cannot automatically flush the BCAI registers because of the DRB32-M micromachine. The micromachine writes the longword of data from the user device into the BCAI and decrements CURBLFT in the same microinstruction. In the next microinstruction, it increments the pointer into the BCAI register file. For the micromachine to be able to flush the BCAI on an abort, these three operations must be uninterruptable. However, in order not to violate the VAXBI specification, the DRB32-M must allow interrupts (to serve VAXBI transactions directed at the DRB32-M) to occur within these two microinstructions. If a device driver happens to set the Abort bit between these two microinstructions, the DRB32-M loses track of how much data is in the BCAI and where it ends.

D.3 HOW TO GET THE DATA FROM THE BCAI

Three pieces of information are required to successfully obtain data when a READ from the user port is aborted:

1. Quantity of data
2. Location of data in the BCAI dual-octaword register file
3. Destination of data in memory.

This section describes the algorithm to use to obtain data left in the BCAI when a READ from the user port is aborted.

1. The contents of CURBLFT contains the number of bytes remaining to be transferred from the user port. If you subtract this from the transfer size loaded into CURBLFT at the start of the transfer, you know how much data the DRB32-M received from the user port. If this value is zero, no data was transferred from the user port. If you then add this value to the starting address of the buffer, you know the address of the byte just after the ending address of the buffer, if all the data was written to the buffer. Let's call this address the red address.

2. Determine if the DRB32 has written anything to memory by using this formula:

   ((START_BYTE_COUNT – CURBLFT) + CURBOFF<4:0>) > 15

   START_BYTE_COUNT is the original transfer size, CURBLFT is the contents of the CURBLFT register after the Abort bit has been set, and CURBOFF<4:0> is the low 5 bits of the buffer's byte offset (that is, the offset from the start of a page to the start of the buffer's address in that page). If this relation is false (that is, it is less than or equal to 15), then the DRB32-M has not issued a WRITE transaction to memory. The number of bytes left in the BCAI is:

   (START_BYTE_COUNT – CURBLFT)

   and the starting byte in the BCAI is CURBOFF<4:0>. In this case, go to Step 8.

   Note that CURBOFF, above, is the value the driver originally wrote to the DRB32-M's CURBOFF Register when the transfer was started. You cannot get this value from the DRB32-M's CURBOFF Register, because the DRB32-M clears this register when you set the Go bit. You must keep a copy of CURBOFF in a driver data structure and use that copy in the calculation described above.

3. Isolate the byte offset from the red address because the red address is a virtual address. This value of byte offset is called the RED BOFF. The blue address (which we obtain in the next step) is a physical address. The two addresses cannot be compared directly.

4. Get the MASTER PORT DMA ADDRESS from the BCAI by writing the value 20 hex to the BAR and then reading the BAR. Let's call this the blue address.

5. Isolate the byte offset from the blue address and call it the blue boff.

6. The red address represents the number of bytes obtained from the user device, but the blue address represents the number of bytes actually written to memory; therefore, the red address must be higher than the blue address. If the red boff is smaller than the blue boff, a page crossing occurred, and we have to add 512 to the red boff.

7. The number of bytes left in the BCAI is now:

   (RED BOFF – BLUE BOFF)

   If this value (BYTES_IN_BCAI) is subtracted from the red address, you have the virtual address in the memory buffer where the data must be written. However, you must be careful with this address because it may not do you any good, depending on what operating system you are using. If you are using VMS, for example, this address is a process virtual address. Since you do not have direct access to that address from a device driver, you have to use some operating system routines to copy the data to the user's buffer.

8. Calculate where in the BCAI the data starts. Since the pointer into the BCAI register file cannot be trusted (and you don't have any access to it anyway), this has to be calculated from the initial byte offset, the total number of bytes received from the user port, and the BYTES_IN_BCAI.

   The low four bits of the original byte offset indicate the byte in the first octaword in the register file where the first byte of data was written. Add this to the total number of bytes received from the user port, and then take this number modulo 32 (that is, take the remainder when you divide by 32). This value is the byte in the register file where the next byte of data would have been written if the transfer hadn't been aborted. Subtract BYTES_IN_BCAI from this. If the result is less than zero, add 32 (to account for the fact that the register file wraps around). This value is the starting byte of the data left in the BCAI.

9. Copy the data out of the BCAI. Take the value obtained in the previous step and write it to the BAR. Then read the longword of data from the BAR. Increment the starting byte by 4, decrement BYTES_IN_BCAI by 4, and repeat until all the data has been retrieved.

10. Copy the data to the user's memory buffer, depending on the rules of the operating system you are using.


APPENDIX E

APPENDIX E
Glossary

Adapter A node that interfaces other buses, communication lines, or peripheral devices to the VAXBI bus.

Arbitration Cycle A cycle during which nodes arbitrate for control of the VAXBI bus.

Assert To cause a signal to take the "true" or asserted state.

Asserted To be in the "true" state.

Assertion The transition of a signal from deasserted to asserted.

Atomic Pertaining to an indivisible operation.

Bandwidth The data transfer rate measured in information units transferred per unit of time (for example, megabytes per second). All bandwidth figures quoted in this manual take into account command/address and embedded ARB cycle overhead.

BCI VAXBI chip interface; synchronous interface bus that provides for all communication between the BIIC and the user interface.

BIIC Backplane interconnect interface chip; chip that serves as a general purpose interface to the VAXBI bus.

BIIC CSR Space The first 256 bytes of the 8-KB node space, which is allocated to the BIIC's internal registers. See also Node Space.

BIIC-Generated Request A transaction request generated by the BIIC rather than by the user interface. The BIIC can request only error interrupts.

BIIC-Generated Transaction A transaction performed solely by the BIIC with no assistance from the master port interface. Only INTR and IPINTR transactions can be independently generated by the BIIC. The user interface initiates BIIC-generated transactions by using the IPINTR/STOP force bit, the user interface or error interrupt force bits, or by asserting one of the BCI INT<7:4> L lines. A BIIC-generated transaction can also result from a BIIC-generated request, which results from a bus error that sets a bit in the Bus Error Register.

Block Mode DMA data transfer mode.

Bus Access Latency The delay from the time a node desires to perform a transaction on the VAXBI bus until it becomes master.

Bus Adapter A node that interfaces the VAXBI bus to another bus.

Busy Extension Cycle A bus cycle during which a VAXBI node not necessarily the master or slave of the transaction, asserts the VAXBI BSY L line to delay the start of the next transaction.

CB RAM Central Bus RAM. Part of the DRB32 parallel I/O port; contains map registers and 8-KB storage.

Client User program that links with DRB32$MESSAGE subroutine package to communicate with another user program.

Command/Address Cycle The first cycle of a VAXBI transaction. The information transmitted in this cycle is used to determine slave selection. In some cases, the data on the VAXBI D<31:0> lines is not an actual address, but it serves the same purpose: to select the desired slave nodes(s).

Command Confirmation The response sent by the slave(s) to the bus master to confirm participation in the transaction.

Command Confirmation Cycle The third cycle in a VAXBI transaction during which slave(s) confirm participation in the transaction.

Configuration Data Data loaded into the BIIC on power-up that includes the device type and revision code, the parity mode, and the node ID.

Cycle The basic bus cycle of 200 nanoseconds (nominal), which is the time it takes to transfer the smallest piece of information on the VAXBI bus. A cycle begins at the leading edge of T0/50 and continues until the leading edge of the next T0/50.

Data Cycle A cycle in which the VAXBI data path is dedicated to transferring data (such as read or write data, as opposed to command/address or arbitration information) between the master and slave(s).

Data Mode Programmed I/O data transfer mode.

Data Transfer Commands VAXBI commands that involve the transfer of data as well as command/address information: read-type, write-type, IDENT, and BDCST commands.

Deassert To cause a signal to be in the "false" or deasserted state.

Deassertion The transition of a signal from asserted to deasserted.

Decoded ID The node ID expressed as a single bit in a 16-bit field.

Device Type A 16-bit code that identifies the node type. This code is contained in the BIIC's Device Register.

DMA Adapter An adapter that directly performs block transfers of data to and from memory.

DMA Mode Data transfer mode directly to and from memory; block mode.

DRB32-E The external driver option of the DRB32 adapter, consisting of the T1022 module, the T1024 module, and connecting cabling.

DRB32-M The basic DRB32 module option (T1022).

DRB32-W The DR11-W interface option of the DRB32 adapter, consisting of the T1022 and T1023 boards and connecting cabling.

DRB32$MESSAGE Program consisting of subroutines that enable communication between two DRB32 adapters.

DRB32$QIO Program for the DRB32-M/-E, consisting of subroutines that check UQDRIVER; they are examples of how to access device driver from specific program environments.

DRB32$WQIO Program for the DRB32-W, consisting of subroutines that check UQWDRIVER; they are examples of how to access device driver from specific program environments.

Encoded ID The node ID expressed as a 4-bit binary number. The encoded ID is used for the master's ID transmitted during an embedded ARB cycle.

EVDRH The DRB32 level 3 diagnostic.

H Designates a high-voltage logic level (that is, the logic level closest to Vcc). Contrast with L.

Interlock Commands The two commands, IRCI (Interlock Read with Cache Intent) and UWMCI (Unlock Write Mask with Cache Intent), that are used to implement atomic read-modify-write operations.

Internode Transfer A VAXBI transaction in which the master and slave(s) are in different VAXBI nodes. Contrast with Intranode Transfer.

Interrupt Port Those BCI signals that are used in generating INTR transactions.

Interrupt Port Interface That portion of user logic used to interface to the interrupt port of the BIIC.

Interrupt Vector In VAX/VMS systems, an unsigned binary number used as an offset into the system control block. The system control block entry pointed to by the VAXBI interrupt vector contains the starting address of an interrupt handling routine. (The system control block is defined in the *VAX-11 Architecture Reference Manual.*)

Intranode Transfer A transaction in which the master and slave are in the same node. Loopback transactions are intranode transfers. Contrast with Internode Transfer.

L Designates a low-voltage logic level (that is, the logic level closest to ground). Contrast with H.

Local Memory VAXBI memory that can be accessed without using VAXBI transactions; for example, VAXBI-accessible memory on a single board computer.

Loopback Extension Cycle A cycle of a loopback transactions during which a node asserts both BI BSY L and BI NO ARB L to delay the start of the next transaction.

Loopback Mode Diagnostic mode in which DRB32 does loopback transactions instead of VAXBI transactions.

Loopback Request A request from the master port interface asserted on the BCI RQ<1:0> L lines that permits intranode transfers to be performed without using the VAXBI bus.

Loopback Transaction A transaction in which information is transferred within a given node without use of the VAXBI data path. Contrast with VAXBI Transaction.

Mapped Adapter A DMA adapter that performs data transfers between a system with a contiguous memory space and VAXBI address space (in which memory need not be contiguous). The mapping is done by using a set of map registers located in the adapter.

Master The node that gains control of the VAXBI bus and initiates a VAXBI or loopback transaction. See also Pending Master.

Master Port Those BCI signals used to generate VAXBI or loopback transactions.

Master Port Interface That portion of user logic that interfaces to the master port of the BIIC.

Master Port Request A request (either VAXBI or loopback) generated by the master port interface through the use of the BCI RQ<1:0> L lines.

Master Port Transaction Any transaction initiated as a result of a master port request.

Module A single VAXBI card that attaches to a single VAXBI connector.

Multiresponder Commands VAXBI commands that allow for more than one responder. These include the INTR, IPINTR, STOP, INVAL, and BDCST commands.

Node A VAXBI interface that occupies one of 16 logical locations on a VAXBI bus. A VAXBI node consists of one or more VAXBI modules.

Node ID A number that identifies a VAXBI node. The source of the node ID is an ID plug attached to the backplane.

Node Reset A sequence that causes an individual node to be initialized; it is initiated by the setting of the Node Reset bit in the VAXBI Control and Status Register.

Node Space An 8-KB block of I/O addresses that is allocated to each node. Each node has a unique node space based on its node ID.

Odd Parity The parity line is asserted if the number of asserted lines in the data field is an even number. The DRB32 uses odd parity.

Parity Mode Specifies whether parity is generated by the BIIC or by the User interface.

Pending Master A node that has won an arbitration but has not yet begun a transaction.

Pending Request A request of any type, whether from the master port or a BIIC-generated request, that has not yet resulted in a transaction.

Pipeline Request A request from the master port that is asserted prior to the deassertion of BCI RAK L for the present master port transaction; that is, a new request is posted prior to the completion of the previous transaction.

Power-Down/Power-Up Sequence The sequencing of the BI AC LO L and BI DC LO L lines upon the loss and restoration of power to a VAXBI system. See also System Reset.

Private Memory Memory that cannot be accessed from the VAXBI bus.

Programmed I/O (PIO) Adapter An adapter that does not access memory on the VAXBI bus but interacts only with a host processor.

Programmed I/O Mode Data transfer mode in which the processor must intervene on each Read or Write transaction; data mode.

Read Access Time The delay from the time a node requests read data until it receives that data from the VAXBI bus.

RCLK (Receive Clock) The clock phase during which information is received from the VAXBI bus; equivalent to T100/150.

Read Data Cycle A data cycle in which data is transmitted from a slave to a master.

Read-Type Commands Any of the various VAXBI read commands, including READ, RCI (Read With Cache Intent), and IRCI (Interlock Read With Cache Intent).

RESERVED Code A code reserved for use by Digital.

RESERVED Field A field reserved for use by Digital. The node driving the bus must ensure that all VAXBI lines in the RESERVED field are deasserted.

Reset Module In a VAXBI system, the logic that monitors the BI RESET L line and any battery back-up voltages and that initiate the system reset sequence.

Resetting Node The node that asserts the BI RESET L line.

Retry State A state that the BIIC enters upon receipt of a RETRY confirmation code from a slave. If the master reasserts the transaction request, the BIIC resends the transaction without having the user interface provide the transaction information again. The command/address information and the first data longword, if a write transaction, are stored in buffers in the BIIC.

Single-Responder Commands VAXBI commands that allow for only one responder. These include read- and write-type commands and the IDENT command. Although multiple nodes can be selected by an IDENT, only one returns a vector.

Slave A node that responds to a transaction initiated by a node that has gained control of the VAXBI bus (the master).

Slave Port Those BCI signals used to respond to VAXBI and loopback transactions.

Slave Port Interface That portion of user logic that interfaces to the slave port of the BIIC.

STALL Data Cycle A data cycle of a read- or write-type transaction during which the slave asserts the STALL CNF code to delay the transmission of the next data word.

System Reset An emulation of the power-down/power-up sequence that causes all nodes to initialize themselves; initiated by the assertion of the BI RESET L line.

T-11 16-Bit PDP-11 microprocessor.

Target Bus The bus that a VAXBI node interfaces to the VAXBI bus.

Transaction The execution of a VAXBI command. The term "transaction" includes both VAXBI and loopback transactions.

UQDRIVER The VMS device driver for the DRB32-M/-E that is supplied with the DRB32 software kit.

UQWDRIVER The VMS device driver for the DRB32-W that is supplied with the DRB32 software kit.

User Interface All node logic exclusive of the BIIC.

User Interface CSR Space That portion of each node space allocated for user interface registers. The user interface CSR space is the 8-KB node space minus the lowest 256 bytes, which compose the BIIC CSR space.

User Interface Request A transaction request from the user interface, which can take the form of a master port request, an assertion of a BCI INT<7:4> L line, or the setting of a force bit.

User Port Loopback Mode DRB32 diagnostic mode in which a longword of data written to either IODAT or IOCTL registers can be read back from those registers, without sending data out on the bus. This mode is entered by setting the LB bit in the PARSTR.

VAX Interrupt Priority Level (IPL) In VAX/VMS systems, any number between 0 and 31 that indicates the priority level of an interrupt with 31 being the highest priority. When a processor is executing at a particular level, it accepts only interrupts at a higher level; and on acceptance starts executing at that higher level.

VAXBI Corner The physical corner of a board that connects to the VAXBI; must include the BIIC chip and other circuitry.

VAXBI Loopback Mode DRB32 diagnostic mode in which all VAXBI transactions issued by the DRB32 are VAXBI loopback transactions to the DRB32 slave port. This mode is entered by the VAXBI or the T-11 setting the BILB bit in the PARSTR.

VAXBI Primary Interface The portion of a node that provides the electrical connection to the VAXBI signal lines and implements the VAXBI protocol; for the DRB32, this is the BIIC.

VAXBI Request A request for a VAXBI transaction from the master port interface that is asserted on the BCI RQ<1:0> L lines.

VAXBI System All VAXBI cages, VAXBI modules, reset modules, and power supplies that are required to operate a VAXBI bus. A VAXBI system can be a subsystem of a larger computer system.

VAXBI Transaction A transaction in which information is transmitted on the VAXBI signal lines. Contrast with Loopback Transaction.

Vector Data Cycle A data cycle in which an interrupt vector is transmitted from a slave to a master.

Window Adapter A bus adapter that maps addresses that fall within one contiguous region (a "window") of a bus's address space into addresses in a window (possibly in a different region) in another bus's address space.

Window Space A 256-KB block of I/O addresses allocated to each node based on node ID and used by bus adapters to map VAXBI transactions to other buses.

Write-Type Commands Any of the various VAXBI write commands, including WRITE, WCI (write with cache intent), WMCI (write mask with cache intent), and UWMCI (unlock write mask with cache intent).

Write Data Cycle A data cycle in which data is transmitted from a master to a slave.

XADRIVER DR11-W device driver supplied with VMS.


INDEX

INDEX