

DIO II Accessory Development Guide

**HP 9000 Series 300 Computers
Models 330/350**

HP Part Number 98562-90010



Hewlett-Packard Company

3404 East Harmony Road, Fort Collins, Colorado 80525

NOTICE

The information contained in this document is subject to change without notice

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MANUAL, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE Hewlett-Packard shall not be liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material

WARRANTY

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office

Copyright 1987 Hewlett-Packard Company

This document contains proprietary information which is protected by copyright All rights are reserved No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company The information contained in this document is subject to change without notice

Restricted Rights Legend

Use, duplication or disclosure by the Government is subject to restrictions as set forth in paragraph (b)(3)(B) of the Rights in Technical Data and Software clause in DAR 7-104 9(a)

Printing History

New editions of this manual will incorporate all material updated since the previous edition. Update packages may be issued between editions and contain replacement and additional pages to be merged into the manual by the user. Each updated page will be indicated by a revision date at the bottom of the page. A vertical bar in the margin indicates the changes on each page. Note that pages which are rearranged due to changes on a previous page are not considered revised.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

February 1987...Edition 1

Table of Contents

Chapter 1: Introduction

Objectives of This Manual	1-1
Liability and Support	1-2
What Is Not Covered In This Document	1-3
Recommended Design Methodology	1-4

Chapter 2: DIO II Bus Overview

Introduction	2-1
DIO II Bus Block Diagram	2-2
Terminology	2-3
Signal Names	2-4
Interface System Elements	2-9
Bus Subsystems	2-12
General Bus Timing Background	2-13

Chapter 3: DIO II Memory Map

Introduction	3-1
DIO II Memory Map and Card Registers	3-3
DIO II Memory Map	3-3
Cached Region	3-4
Uncached Region	3-5
Uncached Region Card Registers	3-6
DIO Memory Map and Card Registers	3-7
DIO Memory Map	3-7
DIO External I/O Memory Map	3-8
DIO I/O Card Registers	3-9

Chapter 4: Data Transfers

Introduction	4-1
Data Transfer Signal Names	4-2
Data Transfer Overview	4-4
Address and Data Alignment	4-5
Data Transfer Timing	4-10
Bus Skew	4-10
Address Only Cycles	4-10
SINGLE READ	4-12
Single Write	4-17
Block Transfers	4-21
Read Modify Write Cycles	4-28
ENDT Overview	4-32
Read Cycle Using ENDT	4-33
Write Cycle Using ENDT	4-35

Chapter 5: Bus Error	
Introduction.....	5-1
Bus Error Signals	5-1
Bus Error Timing	5-2
Bus Timeout	5-2
Auto-Locate of Processor RAM	5-3
BERR for Page Faulting	5-3
Guidelines for Utilizing BERR	5-4
Chapter 6: Interrupt Operation	
Introduction.....	6-1
Interrupt Signals	6-2
External Vectored Interrupt Cycle	6-4
Autovectored Interrupt Cycle.....	6-6
Chapter 7: Bus Arbitration	
Bus Arbitration Signals	7-1
Bus Arbitration Overview	7-3
Bus Arbitration Sequence	7-5
Chapter 8: DMA Operation	
Introduction.....	8-1
DMA Signals	8-2
DMA Overview	8-3
HP 98620 DMA Controller	8-4
DMA Output Cycle	8-5
DMA Input Cycle.....	8-8
DMA Speed Considerations	8-11
Terminating DMA Transfers.....	8-11
Software Interface	8-12
HP 98620B Compatible Registers	8-13
General Control Register.....	8-17
CHANNEL SPECIFIC Registers.....	8-19
Software Performance	8-23
Use of Interrupts Versus Fast Handshake.....	8-23
Programmable BUS Bandwidth.....	8-24
Chapter 9: DIO Bus Utilities	
Bus Drive Disable.....	9-1
Reset Operation	9-2
Halt Operation	9-3
Function Code Signals	9-3

Chapter 10: Electrical Specifications	
Power Distribution and Grounding	10-1
Power Supply Tolerances	10-2
Current Requirements	10-2
DIO II Current Requirements	10-2
DIO I/O Card Current Requirements	10-3
On-Card Fuse Specification	10-3
Signal Loading	10-4
DIO II Signal Loading	10-4
DIO Signal Loading	10-5
Chapter 11: Mechanical Specifications	
Introduction	11-1
DIO II Boards and DIO Card Specifications	11-2
DIO II (System) Boards	11-2
DIO Cards	11-6
Cardcage Specifications	11-9
DIO II (System) Cardcage Specifications	11-9
DIO Cardcage Specifications	11-10
Minimizing RFI	11-11
PC Layout Rules	11-12
DIO II Pin Assignment	11-12
DIO II (System) Boards	11-12
DIO Cards	11-15
Chapter 12: Operation In The HP 9888A Bus Expander	
Features Of The Bus Expander	12-1
Operating Limitations In The Bus Expander	12-2
Chapter 13: Bus Slave Design Summary	
External I/O Card Design Guidelines	13-1
External DIO II Design Example	13-2
Chapter 14: DIO II and DIO Card Qualification	
Software Qualification	14-2
Hardware Qualification	14-2
Configuration Testing	14-2
Environmental Testing	14-2
Safety Compliance	14-4

Introduction

Objectives of This Manual

The *DIO II Accessory Development Guide for HP 9000 Models 330/350 Computers* is a specification type of manual which is meant to be used by experienced design engineers in the development of interface cards for the DIO II Bus. It is not a replacement for the previous *HP 9000 Series 200/300 Computers Accessory Development Guide (Manual Reorder No. 09800-90011)* which is used with the DIO Bus of the Series 200 computers (and Series 300 Models 310/320 computers). The *DIO II Accessory Development Guide* (this manual) is used with the HP 9000 Series 300 Models 330 and 350 computers.

For ease of communication in this manual:

- *DIO Bus* will refer to the bus used in the HP 9000 Series 200 computers (and the Series 300 Models 310 and 320). It will also refer to the bus in the DIO II machine that has the same DIO signals as the Series 200 DIO Bus (96-pin connector), or the 100-pin connector used for small DIO cards.
- *DIO II Bus* will refer to the bus used in the HP 9000 Series 300 Models 330 and 350 Computers that connect to the larger systems cards. It will also refer to the added 48-pin bus that extends the previous DIO bus functions and makes the computer a DIO II device.
- *DIO Accessory Development Guide* will refer to the *HP 9000 Series 200/300 Computers Accessory Development Guide (Manual Reorder No. 09800-90011)*. This is the manual for the DIO Bus.
- *DIO II Accessory Development Guide* will be the reference for this manual.

The *DIO II Accessory Development Guide* was designed to meet the following objectives:

1. The primary objective is to provide sufficient information to design a DIO II Bus device, such as an I/O card, processor board, or RAM cards.
2. A secondary objective is to support the design of bus masters. While bus master design is not covered explicitly, the specifications in this manual must be followed by bus master designers to guarantee that bus slave timing requirements are met. Additional bus master guidelines for Series 200 (including Series 300 Models 310 and 320) devices are available in the *HP 9000 Series 200/300 Computers Accessory Development Guide* (Reorder Number 09800-90011).

Liability and Support

The information provided in this manual is believed to be correct and accurate. However, Hewlett-Packard Company does not assume any liability for inaccuracies or omissions. Responsibility for any devices designed through direct or indirect use of all or part of the contents of this manual lie wholly with the designer and manufacturer of the device(s). This includes but is not limited to: electrical and personnel safety, radio frequency interference and electromagnetic compatibility, compatibility with existing or future hardware, and all legal and safety issues arising from implementation of the design.

Hewlett-Packard does not provide field service support for devices not developed and/or marketed by HP. Field support by HP is limited to the unmodified computer, peripherals, and accessories, and only in combinations that are listed in supported peripherals/device publications in effect at the time the computer and related equipment was sold by HP.

Hewlett-Packard Company does not warrant HP computers, accessories, and peripherals against damage due to improper design or use of devices designed and manufactured by outside parties when such devices are installed and used in HP-supplied computer systems (unless such warranty support has been specifically agreed to in writing by Hewlett-Packard Company). However, use of accessories in HP-supplied equipment does not effect normal warranty against defects in material and workmanship for equipment manufactured by HP. Damage to HP-supplied equipment that is caused by defective or improperly designed devices manufactured by other parties will be repaired upon request by the customer or other responsible parties on a time and material basis.

What Is Not Covered In This Document

The following information is not contained within this document:

1. As stated previously, bus master design is not explicitly covered in this document although enough information is available to adequately do the job. If further information is necessary the designer may want to refer to the following documents:
 - HP 9000 Series 200/300 Computers Accessory Development Guide (Reorder Number 09800-90011)
 - Motorola MC68020 32-Bit Microprocessor User's Manual
 - Model 330/350 Service Manual (Chapter 3, Functional Description)
2. Several of the DIO II BUS timing and electrical specifications were derived from characteristics of the 74LS244 and 74LS245 drivers. Characterization of the ALS logic families has been done to some extent and is currently being used in many designs.

Recommended Design Methodology

Listed below is a recommended methodology for designing a card for the DIO Bus.

1. Prior to beginning a project that uses DIO External I/O space or the DIO II “uncached” space, the designer should review the select code assignments to ensure that no two devices are trying to use the same select codes.
2. Designers should review the design of existing DIO II Bus products. The products listed in the Default Select Code Assignment Table provide a good reference of existing DIO and DIO II BUS products. Designers should also reference the *HP 9000 Series 200/300 Computers Accessory Development Guide* (Manual Reorder No. 09800-90011).
3. Even though a designer follows this document rigorously, testing of the product in its intended environments is **ABSOLUTELY ESSENTIAL**. Chapter 14, *DIO II and DIO Card Qualification* gives guidelines for this testing.

DIO II Bus Overview

2

Introduction

This chapter provides basic background information on the DIO II Bus. Included in the chapter are: a block diagram, list of signal names, terminology, functional interface system elements, defined bus subsystems, and a discussion of bus timing.

Before discussing the DIO II Bus in detail, it should be pointed out that it is a superset of the DIO Bus. Therefore, some of the restrictions associated with the DIO are inherent in the DIO II Bus. However, because of the bus structure, both DIO and DIO II cards are supported.

However, the DIO II Bus has the additional capabilities of:

- supporting a 4 Gigabyte address space
- 4 byte transfer ability (called long words).
- supporting both 16 and 32 bit bus masters.
- supporting block transfers (for future implementation).

DIO II Bus Block Diagram

Figure 2-1 shows the relationship between the DIO II Bus and the cards or boards that connect to it. The interfaces are subject to change as new cards and boards are developed.

Note that the DIO Interface cards are 16 data bit cards that were developed for the Series 200 Computers, or for the Series 300 Models 310 and 320 computers. They use only one connector on the bus.

Note also that the DIO II Interface cards are 32 data bit cards. They use two bus connectors (one that has the same signals as the DIO Interface connector, and an additional connector that carries the additional signals needed to provide 32-bit operation).

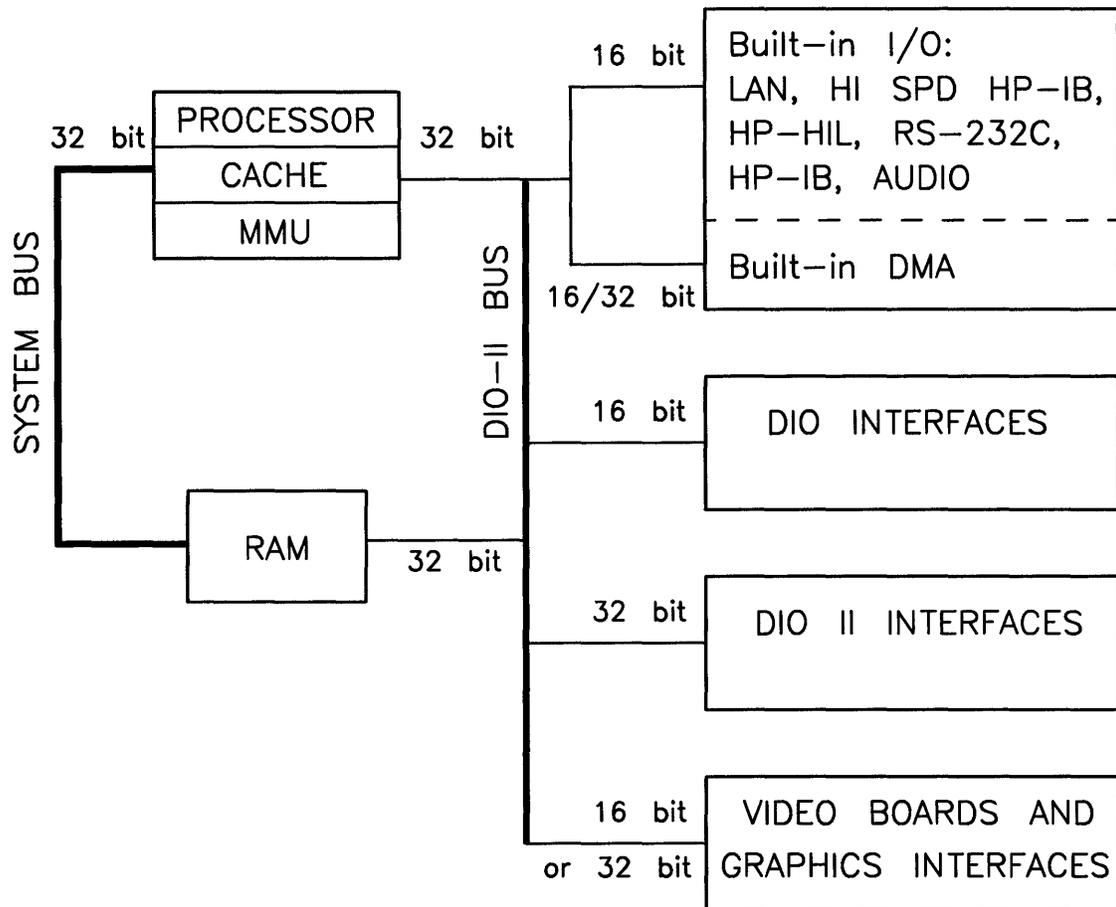


Figure 2-1. DIO II Bus Architecture

Terminology

The following terminology is used throughout this manual:

- A bar over a name denotes an active low signal (example: $\overline{\text{BAS}}$).
- When a signal is referenced as asserted (true, false, etc.), it is relative to the signal function. For example: to say that BAS is asserted means the Buffered Address Strobe is active (i.e. performing its function of strobing the address). Whether it exists as BAS, or $\overline{\text{BAS}}$ on the backplane is irrelevant.
- References to high and low refer directly to TTL voltage levels. When referring to high and low signals, the actual name of the signal is used. For example, when the signal $\overline{\text{BAS}}$ is described as being low, the signal entitled $\overline{\text{BAS}}$ has a TTL logic low level. The TTL levels are defined as follows:

High	Equal to or greater than 2.0 Volts
Low	Equal to or less than 0.8 Volts

- The composite signal $\text{BR}/\overline{\text{W}}$ high indicates a read operation; $\text{BR}/\overline{\text{W}}$ low indicates a write operation.

Note

There are two address strobes defined in DIO II specifications. They are $\overline{\text{BAS24}}$ and $\overline{\text{BAS32}}$. When discussing address strobes, where either $\overline{\text{BAS24}}$ or $\overline{\text{BAS32}}$ can be used, the generic term of $\overline{\text{BAS}}$ will be used.

There are three data transfer acknowledge signals defined in DIO II specifications. They are $\overline{\text{DTACK16}}$ ($\overline{\text{DTACK}}$ in the DIO Accessory Development Guide), $\overline{\text{DSACK16}}$, and $\overline{\text{DSACK32}}$. When discussing data transfer acknowledge signals, where any one of the three could be asserted, the generic term of $\overline{\text{DTACK}}$ will be used.

- The definition of the term byte is 8 bits. A word is defined to be 16 bits. A long word is defined to be 32 bits. The least significant bit of a byte, word, or long word is defined as Bit 0.
- The terms “cycle” and “state” are defined, as these terms are sometimes used erroneously. Cycle refers to a **complete** clock cycle, e.g. 125 nsec for an 8 MHz clock. State refers to **one half** of a clock cycle and is based on Motorola’s nomenclature. States are numbered from S0 to S7, representing 8 states, or 4 cycles.

Signal Names

The bus signals that are used in computer operation are defined in Tables 2-1 through 2-5. These tables are also available in the chapters that explain signal operation.

Table 2-1. Data Transfer Signal Names

Signal Name	Definitions
$\overline{\text{ADACK}}$	ADdress ACKnowledge is an output from a card that is addressed. This is used by the bus expander to reverse its data bus buffers when a card in the Bus Expander is addressed. All DIO II slaves must generate an $\overline{\text{ADACK}}$ signal.
$\overline{\text{AS}}$	Address Strobe. Not a bus signal. Refers to $\overline{\text{BAS24}}$ or $\overline{\text{BAS32}}$.
BA1	Buffered Address 1 is used to determine if the upper or lower word has the valid data.
BA1-BA31	This is the 31 bit address bus. BA0 is not on the bus.
$\overline{\text{BAS}}$	Buffered Address Strobe is not a bus signal but is used to generically refer to either $\overline{\text{BAS24}}$ or $\overline{\text{BAS32}}$.
$\overline{\text{BAS24}}$	Buffered Address Strobe 24 defines when the 23 bit address space (A1-A23) is valid. This signal is the old DIO $\overline{\text{BAS}}$.
$\overline{\text{BAS32}}$	Buffered Address Strobe 32 defines when the 31 address lines are valid. This line is asserted over the complete 32 bit address space. This means that $\overline{\text{BAS32}}$ will be asserted when talking to DIO space with a 32 bit master.
BD0-BD15	Buffered Data 0 - 15 is the DIO data bus. This data port maps to the most significant bits in DIO II.
$\overline{\text{BDS}}$	Buffered Data Strobe is not a bus signal but is used generically to refer to either $\overline{\text{BUDS}}$ and/or $\overline{\text{BLDS}}$.
BFC0-BFC2	Buffered Function Codes 0 - 2 defines the type of transaction occurring on the bus. Refer to the MC68020 manual for correct decoding.
$\overline{\text{BLK}}$	BLOCK indicates that a block transfer is occurring on the bus.
BR/ $\overline{\text{W}}$	Buffered Read/ $\overline{\text{Write}}$. High for read and low for write.
$\overline{\text{BUDS}}$, $\overline{\text{BLDS}}$	Buffered Upper Data Strobe and Buffered Lower Data Strobe. $\overline{\text{BUDS}}$ indicates that the upper byte of either the upper or lower word is valid. $\overline{\text{BLDS}}$ indicates that the lower byte of either the upper or lower word is valid.
$\overline{\text{DSACK}}$	$\overline{\text{DSACK}}$ is a generic term meaning $\overline{\text{DSACK32}}$ or $\overline{\text{DSACK16}}$ but not $\overline{\text{DTACK16}}$.
$\overline{\text{DSACK16}}$	Data transfer and Size ACKnowledge 16 is asserted by a slave with a 16 bit data port in response to the assertion of $\overline{\text{BAS32}}$. It also informs the bus master that it has completed the bus cycle. Note that $\overline{\text{DSACK16}}$ informs the master as to the size of the data port and contains no information as to number of bytes transferred.

Table 2-1. Signal Names (cont'd)

Signal Name	Definitions
$\overline{DSACK32}$	Data transfer and Size ACKnowledge 32 is asserted by a slave in response to $\overline{BAS32}$. It informs the master it has a 32 bit data port and that it has completed the bus cycle. Note that $\overline{DSACK32}$ informs the master as to the size of the data port and contains no information as to the number of bytes transferred.
\overline{DTACK}	Data Transfer ACKnowledge is not a bus signal but is used to generically refer to either $\overline{DTACK16}$, $\overline{DSACK16}$, or $\overline{DSACK32}$. This signal is used by the accessed slave to inform the bus master that it has completed a bus cycle. During a read cycle, it indicates that the bus slaves data is valid on the bus. During a write cycle it indicates that the bus slave has accepted the data. $\overline{DTACK16}$ is the old DIO \overline{DTACK} and is used by those devices that are running a cycle in response to the assertion of $\overline{BAS24}$. The two new \overline{DSACK} signals determine the size of the data port and each is asserted in response to a bus cycle beginning with the assertion of $\overline{BAS32}$.
$\overline{DTACK16}$	Data Transfer ACKnowledge 16 is asserted by a slave to end a bus cycle started by the assertion of $\overline{BAS24}$. It must be asserted when responding to $\overline{BAS24}$ since the master only has a 16 bit port and the slave must emulate the same size port. This signal is the old DIO \overline{DTACK} .
\overline{ENDT}	ENable DTack is defined for DIO only. It is generated by masters and used by bus slaves to control generation of \overline{DTACK} . Slaves must be able to respond with a \overline{DTACK} if this signal is not generated.
\overline{LWORD}	Long WORD indicates that a long word transfer is being initiated on the bus. It is defined with respect to $\overline{BAS32}$ and must be driven when $\overline{BAS32}$ is asserted.
\overline{RMC}	Read Modify write Cycle indicates that a read modify write transfer is being initiated on the bus.
XD0-XD15	eXtended Data 0 - 15, this is the lower 16 bits of the 32 bit DIO II data bus. These signals are not defined for DIO cycles.

Table 2-2. Bus Error Signals

SIGNAL	DEFINITION
$\overline{\text{BERR}}$	Bus ERRor, when asserted, causes the processor to terminate the bus cycle. When BERR is negated, the processor begins its exception processing.

Table 2-3. Interrupt Signals

SIGNAL	DEFINITION
$\overline{\text{IR1}}$	Interrupt 1 – Keyboard/real time clock LOWEST LEVEL
$\overline{\text{IR2}}$	Interrupt 2 – 9826/36 internal floppy controller -
$\overline{\text{IR3}}$	Interrupt 3 – External I/O -
$\overline{\text{IR4}}$	Interrupt 4 – External I/O -
$\overline{\text{IR5}}$	Interrupt 5 – External I/O -
$\overline{\text{IR6}}$	Interrupt 6 – External I/O -
$\overline{\text{IR7}}$	Interrupt 7 – Reset key, powerfail HIGHEST LEVEL
$\overline{\text{IACK}}$	Interrupt ACKnowledge – output from a DIO Bus Master, not supported on Series 300 Models 330 and 350 CPU's.
$\overline{\text{IACK32}}$	Interrupt ACKnowledge 32 – output from the Bus Master and is supported by DIO II processors.
$\overline{\text{VECTOR}}$	Output of interrupting device if it has an interrupt vector to put on BD0-BD7. This signal is not supported on Series 300 CPU's
$\overline{\text{VECTOR32}}$	Output of interrupting devices if it has an interrupt vector to put on BD0-BD7 or XD0-XD7 depending on which DSACK is asserted. This signal is supported by DIO II processors.

Table 2-4. Bus Arbitration Signals

SIGNALS	DEFINITION
\overline{BR}	Bus Request. Asserted by the device(s) requesting control of the bus. It is wire-ORed among all bus masters.
$\overline{XBG3}$	Extended Bus Grant 3. This is output from a master controller only. It goes to the bus grant input of bus master A.
$\overline{XBG2}$	Extended Bus Grant 2. It may be a master controller output (if $\overline{XBG3}$ is not used), or it is a bus grant output from bus master A. It is connected to the bus grant input of bus master B.
$\overline{XBG1}$	Extended Bus Grant 1. It may be a master controller output (if $\overline{XBG3}$ and $\overline{XBG2}$ are not used), or it is a bus grant output from bus master B. It is connected to the bus grant input of bus master C.
\overline{BG}	Bus Grant. It may be a master controller output (if $\overline{XBG3}$, $\overline{XBG2}$ and $\overline{XBG1}$ are not used), or it is a bus grant output from bus master C. It is connected to the bus grant input of bus master D. This is the default shipping condition for all master controllers currently being produced as of January 1987.
$\overline{BG1}$	Bus Grant 1. Bus grant output from bus master D. It is connected to the bus grant input of bus master E.
$\overline{BG2}$	Bus Grant 2. Bus grant output from bus master E. It is connected to the bus grant input of bus master F.
$\overline{BG3}$	Bus Grant 3. Bus grant output from bus master F. It is connected to the bus grant input of bus master G. Note that bus master G's bus grant output is not connected to another device. It should also be noted that $\overline{BG3}$ is not defined for DIO II. It is not a valid pin on the connector. The only way it can be used is if bus master F and G are in the DIO I/O or option slots and not in the DIO II system slots. Refer to the pinouts of the DIO II and DIO connectors in Chapter 11.
\overline{BGACK}	Bus Grant ACKnowledge. A tri-state signal generated by the bus master accepting the bus, to verify that the bus is now controlled by the new bus master.

Table 2-5. DMA Signals

SIGNAL	DEFINITION
$\overline{\text{DMAR0}}, \overline{\text{DMAR1}}$	DMA Request. Asserted by an I/O card to request a DMA transfer on DMA Channel 0 or DMA Channel 1.
$\overline{\text{DMACK0}}, \overline{\text{DMACK1}}$	DMA ACKnowledge. Response from the DMA Controller, acknowledges DMA request on Channel 0 or Channel 1.
$\overline{\text{DMARDY}}$	DMA ReaDY. Indicates that the I/O card has provided the data (DMA input) or accepted the data (DMA output).
$\overline{\text{DONE}}$	DONE. An output from the DMA Controller to flag the last DMA transfer. DONE can be used at the option of the I/O card designer to determine when DMA is done (For example: to assert EOI on the last byte in an HP-IB transfer).
$\overline{\text{FOLD}}$	FOLD. An output from the DMA Controller to indicate that the data byte needs to be folded from the upper byte of the data bus (from memory) to the lower byte (to an I/O card) or from the lower byte (from an I/O card) to the upper byte (to memory). Fold is not asserted on the DIO or DIO II backplane in DIO II products.

Interface System Elements

The functional modules of the DIO II Bus are shown in this section. Where signals go specifically from one functional module to another functional module, the two modules are shown side-by-side. Modules that drive many other modules (for example: RESET) are shown as stand-alone.

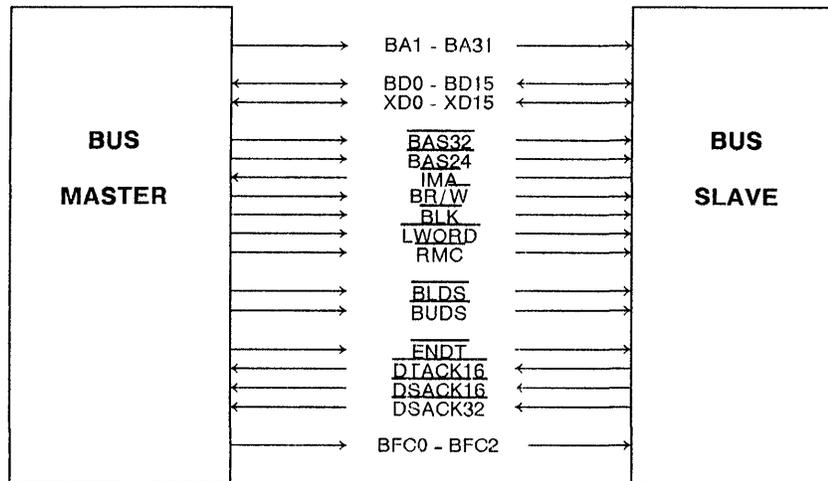


Figure 2-2. Data Transfer Module

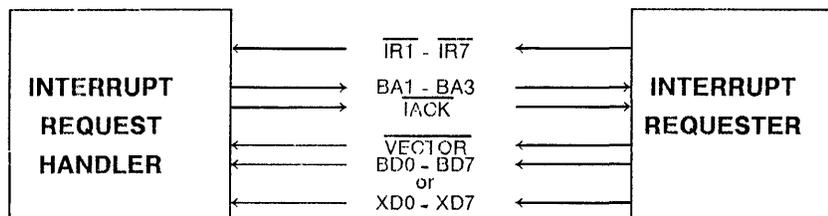


Figure 2-3. Interrupt Request Module

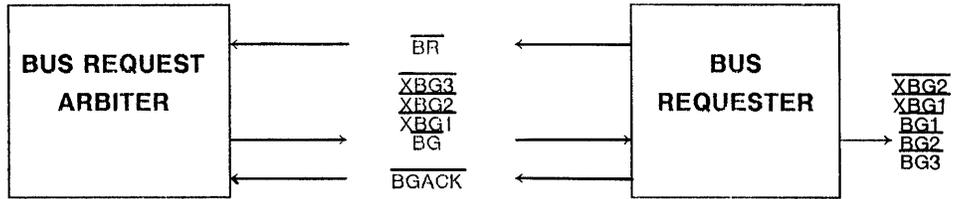


Figure 2-4. Bus Arbitration Modules

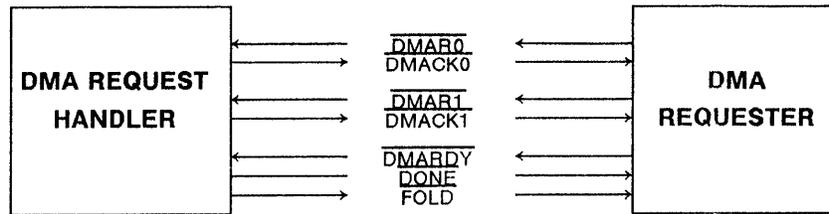


Figure 2-5. DMA Modules

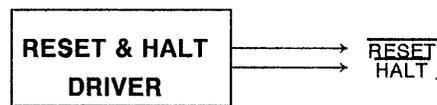


Figure 2-6. Bus RESET and HALT Module

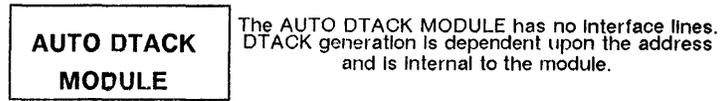


Figure 2-7. Auto DTACK Module

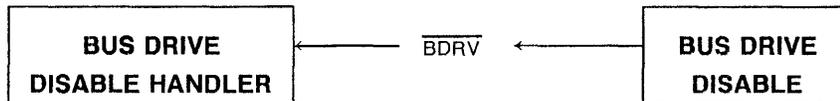


Figure 2-8. Bus Drive Disable Module



Figure 2-9. Bus Timeout Module

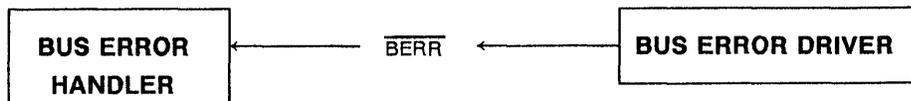


Figure 2-10. Bus Error Handler Module

Bus Subsystems

Defined bus subsystems are shown below:

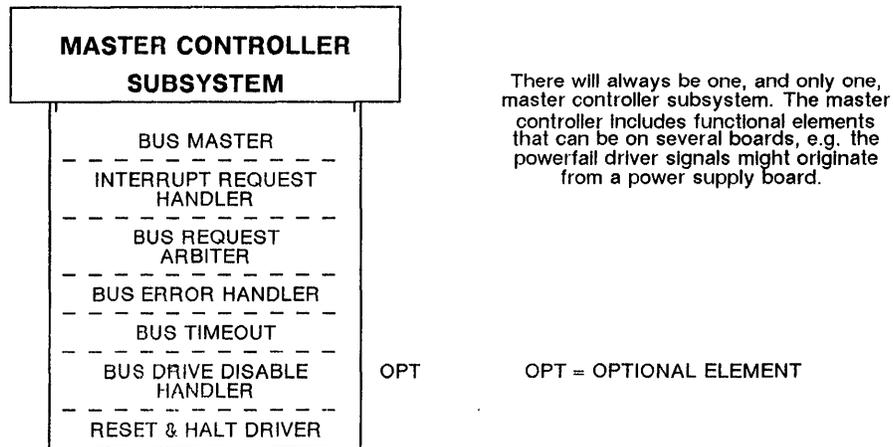


Figure 2-11. Master Controller Subsystem

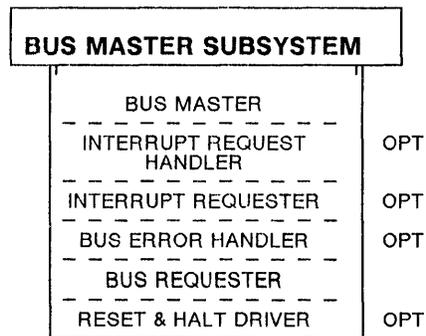


Figure 2-12. Bus Master Subsystem

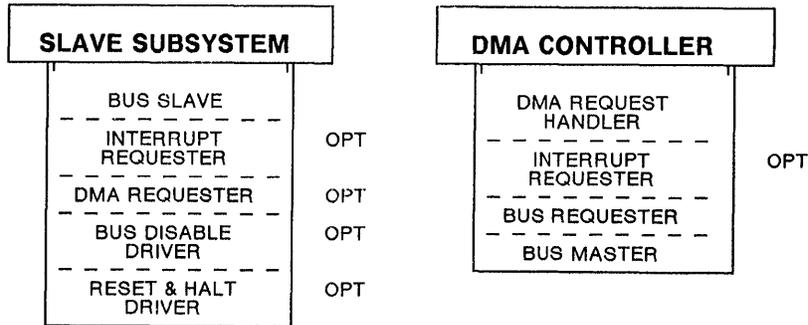


Figure 2-13. Slave and DMA Controller Subsystems

General Bus Timing Background

As was pointed out earlier, the DIO II Bus is a superset of the DIO Bus. DIO II can only exist in the large system size card slots. The DIO can exist either in the large slots or in the smaller DIO slots. Although the slots are of different sizes and capabilities, there is no ordering of address, interrupt capability, etc, due to slot location in the backplane. All timing specifications apply equally to all slots. It is possible, however, that future mainframes may have slot-dependent features (for example: interrupt prioritizing). Thus, designers must be aware of features of each mainframe that their products will operate in.

Note also that timing specifications for the DIO Bus were developed using the 8 Mhz MC68000. Thus the DIO II Bus must live with these timing constraints. Designs incorporating other processors must ensure that the DIO II Bus specifications are met.

The DIO II Bus is asynchronous. That is, there is no clock on the backplane to reference signals to. While address and data generation are related to the CPU clock, the actual clock does not appear on the bus. The presence of address or data is indicated by various control lines which execute interlocked handshaking protocol to convey address and data to their destination. Because the address, data, and control lines are not referenced to a clock on the backplane, signal skew must be controlled to maintain the relative timing between these signals.

For example, the MC68000 is guaranteed to drive the address bus lines 30 nsec prior to asserting Address Strobe. Most receiving devices require at least 15 nsec of address setup time prior to Address Strobe occurrence. In order to guarantee 15 nsec of address setup time, the following rules were developed to control gate delays and bus loading (these are provided in more detail in later chapters).

1. Each board is limited to one LS or ALS load on the address bus, data bus, address strobe, data strobes, and the read/write signal.
2. The PC board trace length on bus signals should be as short as possible and must not exceed 3 inches.
3. The SN74LS245 (or equivalent SN74LS244) is used to drive the above signals (may use ALS).

Rationale for These Rules

The above guidelines originated during the early development of the HP 9826. It became clear that controlling bus capacitance was essential. Hence Rules 1 and 2 were established. Also, to minimize bus skew, it was decided to specify a “standard” bus driver. Hence Rule 3 was established. Next, further analysis was performed on the 74LS245 to develop detailed timing specifications.

Two efforts were made to define bus skew:

1. The first effort was to model the 74LS245 and determine formulas for worst-case minimum and maximum gate delays as a function of bus capacitance.
2. The second effort was to measure delays for a number of 74LS245 parts with different date codes.

The formulas for gate delay, if used with worst-case conditions (fully loaded bus, fast address strobe driver, slow bus driver, etc.), yield unworkable numbers (negative setup times). It was felt that such a worst-case scenario is a low probability and not the appropriate design center method to follow. Therefore, the second effort (measuring delays) was investigated.

Using a sample of 74LS245 parts with different date codes, the worst-case difference in gate delays was measured for a 500 pf load. The difference between the fastest gate and the slowest gate (in different packages) was 9.5 nsec. This was derated by 50% to 15 nsec for margin as well as to cover skew on the bus itself (which is caused by different signal loading). Returning to the MC68000 example, if the address precedes the address strobe by 30 nsec at the MC68000 output, then all receiving devices are guaranteed to have 15 nsec (30 - 15) of address setup time prior to the address strobe (assumes 74LS245's are used to drive the bus).

In addition to determining bus skew, the worst-case high-to-low and low-to-high delay times were determined for a 74LS245 driving a 500 pf load. Delay times were measured relative to the output reaching the nominal device threshold. For example, experimental and published data indicates that 1.7V is sufficient to be seen as a high for the 74LS245; hence, the low-to-high time delay measurement concluded when the output reached 1.7V. The results, which are shown below, have been used in calculating several timing specifications. Drivers for signals such as DTACK (which has a pullup resistor) require approximately 50 nsec to drive the bus from high to low. Likewise, a buffer without a pullup has approximately 40 nsec of delay (for example: the DMA Controller's Fold Buffer). These times include propagation delays.

With 1 kohm pull-up resistor	47.5 nsec high to low
	37.5 nsec low to high
Without pull-up resistor	39.5 nsec high to low
	39.5 nsec low to high

Data was not taken for pull up resistors larger than 1 kohm. However, larger pullups will “group” the rise and fall times closer to the non-pull up rise and fall times.

In summary, the following points can be made:

1. The skews due to the bus drivers and the bus itself are not specified separately. Instead, a “lumped” skew specification of 15 nsecs is provided for the DIO Bus. For the DIO II Bus the skew was calculated at 25 nsecs. Between any two signals driven by 74ALS245 drivers, 25 nsecs of skew can develop between the signal inputs and outputs (where the outputs are measured at the receiving device on the bus). The increased skew is due to potentially light loaded signals on the DIO II backplane compared with potentially heavy loaded signals on the DIO backplane.
2. In many cases, the DIO Bus timing specifications have been derived using the minimum or maximum MC68000 timing specifications (plus or minus the 15 nsec skew, whichever is appropriate).
3. This guideline does not take into account signals driven by devices other than the 74LS245 or the 74ALS245.

DIO II Memory Map

Introduction

This chapter discusses the DIO II address space allocation and the allocation of space for cached, uncached, and DIO use.

Because the new DIO II Bus uses address space essentially the same as the DIO Bus, both buses are discussed in detail. However, this discussion will be limited to the external I/O.

General

The DIO II memory space is a four Gigabyte address space. The I/O and graphic cards register definition are similar to that found in the *HP 9000 Series 200/300 Accessory Development Guide*. A couple of features were added to DIO II operation so as to determine board location and size. Also, the new additional address space has not been defined as specifically as the address space was in the old DIO specification. This may lead to some complications when decoding the address. However, these complications may be alleviated to some extent, by trade offs in the design (For example: It may not be necessary to decode the entire address).

The 4 Gigabyte address space is broken into three areas:

- cached address space,
- uncached address space,
- and the DIO address space.

Cached and uncached are just labels to distinguish between different areas of the memory map. The cached and uncached spaces are defined for specific areas of the available address space regardless of whether the processor board has a cache or not.

Cached Address Space

The cached address space is defined for DIO II Memory boards and other boards that you may want to reside in this area. If the cache is turned on, boards responding through DIO II will have their information put in cache (assuming the processor has a cache). Some boards may be located in this region so that they respond on the system bus. Once the MMU is turned on, areas can be designated for cache or uncache at the MMU's discretion. The uncached and cached areas are just a way of distinguishing different areas of memory and caching may or may not occur. The cached address space **has** pointer registers in the DIO address space, while the uncached address space **does not** have pointer registers.

Uncached Address Space

The uncached address space is an area of the address map that will never be cached (unless the MMU is turned on and wants it cached). Whether a boards cache is on or not does not effect this determination. This space is for I/O cards and frame buffers that are never planned to be cached. Boards in this area have no access to the system bus in our current product implementations. The processor boards on decoding this address start a DIO II cycle immediately, without waiting to determine whether there was a response from the system bus.

DIO Address Space

The DIO address space corresponds to the address space in the old DIO Bus Specification. It has not changed for the current DIO II Specification to ensure compatibility.

DIO II Memory Map and Card Registers

This section shows the three defined areas of the memory address space and explains how cards are found in each of these areas. The *DIO Memory Map and Card Registers* section will discuss the external I/O memory map in more detail.

The three areas of the memory map as shown in Figure 3-1 are the cached address space, uncached address space, and DIO address space.

DIO II Memory Map

The memory map shown in Figure 3-1 is not to scale. The RAM starts from the top of the 4 Gigabyte address space and as more RAM is added grows down toward the uncached region. The upper 8 megabytes of the 4 Gigabyte region is multiple mapped down to the upper 8 megabytes of the DIO region. This allows old DIO bus masters (who have only 24 bits for address designation) access to the top 6 megabytes of RAM (or as much RAM as the system holds, up to 6 megabytes).

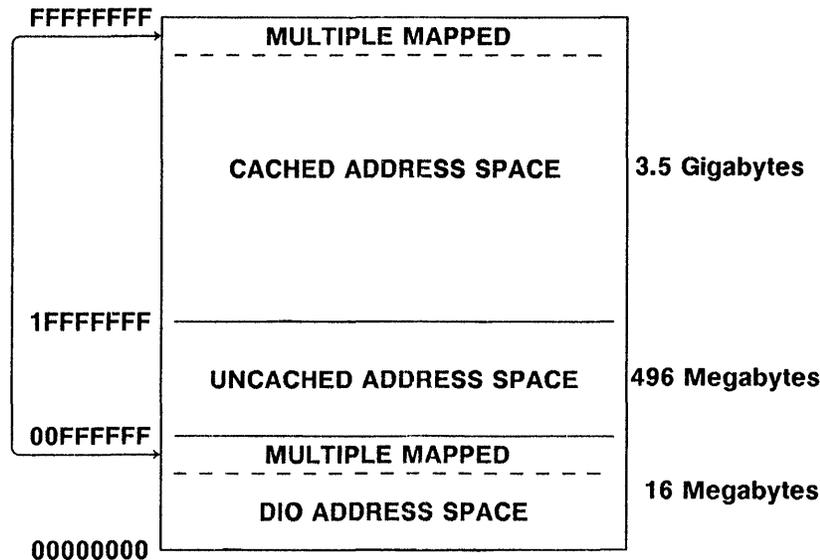


Figure 3-1. DIO II Address Space

Cached Region

The RAM is found by starting at the top of the 4 Gigabyte address space and accessing each long word of memory until a bus error occurs. In this way the size of RAM is found without requiring any external registers. Cards that sit in this region other than RAM have their control registers in DIO External I/O space and will follow DIO register definitions. The DIO register definition defines registers at an offset from the base address of 1, 3, and 5. There is a register defined in DIO II, at an offset of 8, that is 16 bits (one word) wide, and that contains the location of the board in the DIO II cached address space. The register offset of 8(H) may be defined for a different purpose if the board does not reside in the cached space. It is up to the software driver to know whether or not the board will exist in the cached address space.

The address map for cards other than RAM is shown below. The cards in this area should take up one megabyte increments of address space. A label should also be affixed to the card that states the number of megabytes of cached address space the card takes up. The label will help set the extended select code since the installer will know how many select codes to skip. The address format is shown in Figure 3-2.

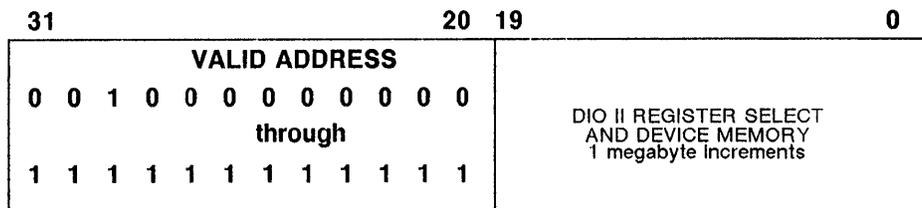


Figure 3-2. Address Map for Cached Address Space

It requires twelve switches to fully decode the address space in the cached address area. This may not be necessary depending on the amount of address space the card requires. This switch selection for the address space in the cached address area requires a 16 bit (one word wide) buffer in DIO address space as a pointer. This buffer is defined at the second available long word. That is, this word resides at an offset from the DIO base address of 8(H).

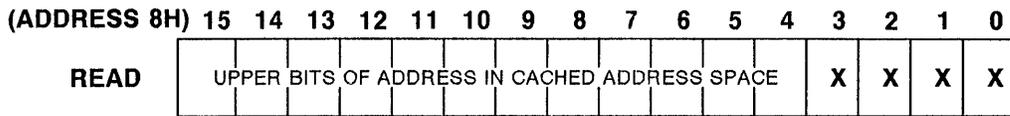


Figure 3-3. DIO Register Pointer to Cached Address Space

The address pointer is a read only register. Writing to it has no effect. It points to the location in the uncached space that is the base address of the area taken up by the card. This register should be set by switches.

Uncached Region

The uncached address space is divided into 124 segments of 4 Megabytes each. The memory boards contain select code switches which determine the physical address of the card in the uncached address space. While only seven switches are required to permit the user to set 124 uncached select codes (from 132 thru 255), to get the proper binary representation of the select code numbers, an eighth switch is needed in the MSD position. This switch has the following characteristics:

- When set to 1, it allows access to the DIO II select codes (132 to 255).
- When set to 0, it allows access to the DIO select codes 0 thru 31 (**IF** the board is designed to work in the DIO external I/O space). It is also recommended that the next two most significant switches be set to zero before select codes 0 thru 31 are accessed.

If the above guidelines are met the user of the DIO II product is able to select both DIO (0-31) and DIO II (132-255) select codes with one switch pack. Furthermore, the proper binary representation for the select code will be visible to the user by looking at the switch pack.

Unlike the DIO select codes(0 thru 31), the DIO II select codes (132 thru 255) allow cards to be mapped over select code boundaries. Since cards can take up multiple select codes in uncached space, a label should be affixed to the card by the cards manufacturer, that tells the number of codes the card takes up. The address format is shown in Figure 3-4.

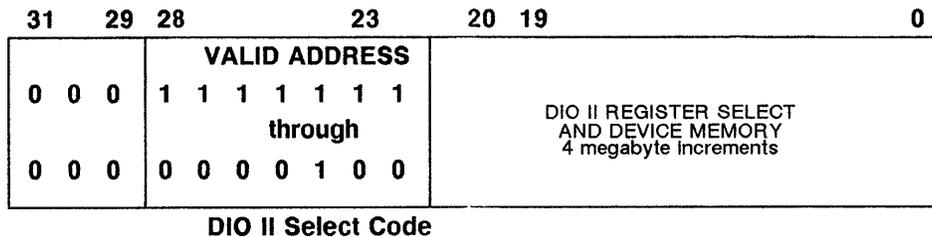


Figure 3-4. Address Map for Uncached Address Space

Cards in the uncached region will be found by scrolling through the address space while looking for ID registers (at an offset of 1(H)) on four megabyte boundaries. When a card is encountered, a register will be read on the card to tell the software how many additional megabytes of memory the card takes up. Note that the number in this register is with respect to one megabyte increments while the select codes are on four megabyte boundaries. The software will then skip over the additional four megabytes of memory before continuing to look for new cards.

Uncached Region Card Registers

The function of certain registers within DIO devices have been defined historically. These definitions have not changed in DIO II Bus specifications. System addresses of these registers are 1, 3, and 5, relative to the card's base address. The only other register necessary to reside in the uncached space is a register at 101(H) which tells the software how many megabytes the board takes up (each uncached select code is four megabytes). The designer is free to implement registers in addition to (but not in lieu of) the ones listed in this chapter. Also, the designer is not required to uniquely map each register within the card's I/O space (That is, registers may be multiple-mapped to simplify address decoding).

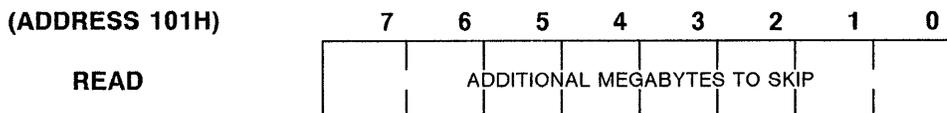


Figure 3-5. Register 101H — Number of extended select codes

DIO Memory Map and Card Registers

The following sections cover information found in the *HP 9000 Series 200/300 Accessory Development Guide*. This information includes a DIO memory map and the defined DIO card registers. The DIO registers are defined for DIO II to maintain compatibility with previous system.

DIO Memory Map

This DIO II Bus Specification is not intended to document in detail the Series 200 (includes Series 300 Models 310 and 320) memory map. Instead, this memory map documentation is limited to the external I/O memory map.

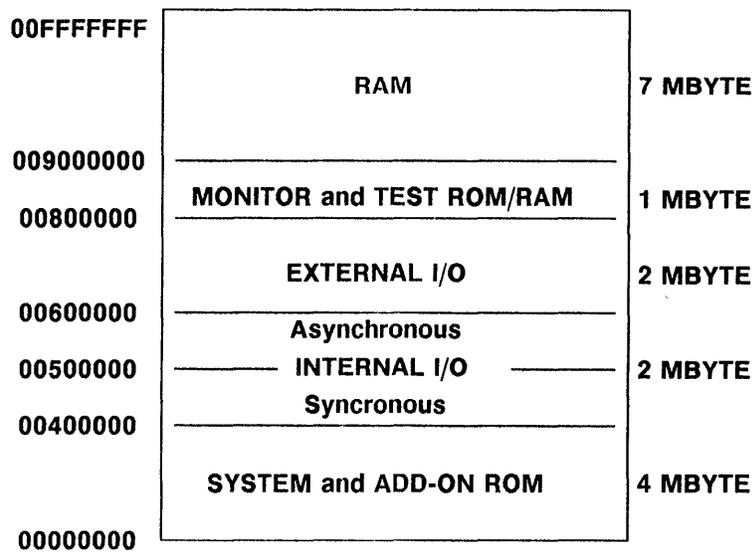


Figure 3-6. DIO Memory Map

The Series 200 memory map is shown in Figure 3-6 and is for reference only (the addresses are in HEX). The MC68000's 24 bit address can address 16 Mbytes of memory. The external I/O occupies 2 Mbytes (600000 - 7FFFFFF). The internal I/O address space (400000 - 5FFFFFF) is used for internal peripherals.

DIO External I/O Memory Map

The DIO external I/O address space is divided into 32 segments of 64 Kbytes each. The I/O cards contain select code switches which determine the physical address of the card in the external I/O address space. 5 switches permit the user to set 32 select codes, from 0 to 31, to determine which 64 Kbyte memory space the card resides in. The address format is shown in Figure 3-7.

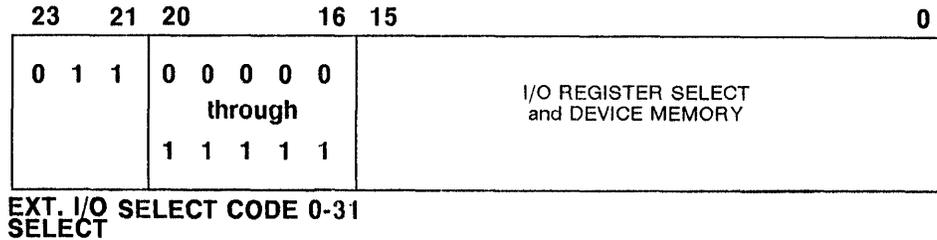


Figure 3-7. Address Map for DIO Address Space

Not all DIO external I/O select codes can be used with existing operating systems. For example, select code 7 is assigned by all operating systems to the internal HP-IB interface which resides in the internal I/O address space. If an I/O card is installed with select code 7, the operating system will ignore it and direct select code 7 activity to the internal HP-IB interface operating in the internal I/O space.

With Pascal and BASIC, all select codes from 0 thru 7 reference internal I/O devices only. It is important to realize that, electrically speaking, I/O cards can be set to select codes 0 thru 7 but that the operating systems map these select codes to addresses in the internal I/O space. Thus, with BASIC and Pascal, I/O cards set from 0 thru 7 are inaccessible. Only I/O cards set from 8 thru 31 can be accessed. At the assembly language level, however, I/O cards with select codes over the entire range of 0 thru 31 can be accessed.

Table 3-1 shows the default select codes for standard HP DIO Bus interfaces. These code assignments are current as of the publication date of this manual. As these default select code assignments change sporadically when new cards are developed, You should check the Operating System manuals and the I/O device manuals (shipped with the I/O cards) to ensure that no two devices have the same select codes.

Table 3-1. Default Select Code Assignments

Select Code	Interface	Address
0-6	Internal Devices	Varies
7	Internal HP-IB	00478000
8	HP 98624 HP-IB	00680000
9	HP 98626 RS-232C Serial I/O, HP 98644 RS-232, Built in RS-232	00690000
10		006A0000
11	HP 98623 BCD	006B0000
12	HP 98622 GPIO	006C0000
13	HP 98642 4-Channel MUX	006D0000
14	HP 98625 Disc Interface	006E0000
15	Custom Card #1	006F0000
16	Custom Card #2	00700000
17		00710000
18	HP 98640 Analog Input	00720000
19		00730000
20	HP 98628 Datacomm	00740000
21	HP 98629 SRM, HP 50961 SRM, HP 98643 LAN	00750000
22	HP 98695 IBM 3270 Coax Interface	00760000
23		00770000
24		00780000
25	HP 98287 Graphics Display Station	00790000
26		007A0000
27	HP 98253 EPROM Programmer Interface	007B0000
28 & 29	HP 98627 Color, HP 98633 Multiprogrammer Interface	007C0000
30	HP 98259 Bubble Memory	007E0000
31	HP 98287 Graphics Display Station HP-HIL Interface	007F0000

DIO I/O Card Registers

The function of certain registers within I/O devices are pre-assigned. Note that because I/O cards are byte-oriented and these registers are connected to the lower byte of the data bus, their system addresses are 1, 3, 5... relative to the card's base address. The designer is free to implement registers in addition to (but not in lieu of) the ones discussed in this chapter. Also, the designer is not required to uniquely map each register within the card's I/O space. That is, registers may be multiple-mapped, which simplifies address decoding, as long as registers do not "exist outside" the card's 64 kbyte I/O space. An exception to this is: when the card is specifically designed to occupy multiple 64K chunks of memory.

The defined I/O registers are:

(ADDRESS 1H)	7	6	5	4	3	2	1	0
READ ID	R/ \bar{L}	SECONDARY ID1	SECONDARY ID0	PRIMARY ID				
WRITE RESET	X	X	X	X	X	X	X	X

Figure 3-8. Register 1H — Card ID and Reset

R/ \bar{L}

REMOTE/ $\overline{\text{LOCAL}}$: A 1 indicates that the mainframe may be controlled from a remote source via this I/O card. For example: under software control a mainframe may receive its keyboard inputs from an RS-232 card and likewise output its display data via the card. This has uses in certain environments where it is desirable to lock out local access and provide remote control of the mainframe.

This feature is not used in current communications cards (except for the HP 98628 card, where the card's firmware monitors this bit). In future data communications cards, this function may be provided (for example: a terminal multiplexer card, Ethernet, etc.). It may be available either by a jumper or a switch. Non-communications oriented cards (such as, an A/D card) should set this bit to 0.

Note

Except in rare cases, software to use this feature is lacking. If the remote feature is to be used, software development is required.

Primary ID

Bits 0-4 contain the Primary ID, which identifies each device. Because 5 bits can only define 32 unique devices, Bits 5 and 6 are defined as Secondary ID bits as discussed below.

Whereas the select code bits are switch-selectable, the ID bits must be hardwired. If possible, the ID and default select code should be the same.

Secondary ID0, 1

The 2 Secondary ID bits are used to provide 4 additional IDs for each Primary ID. Although these bits are located in a higher order position than the Primary ID bits, they should be considered lower order in that they are used to extend the range of each of the Primary ID's defined in Bits 0-4.

Good system design requires that the operating system must be capable of resetting an I/O card to its power-on state. One of two methods must be implemented:

- If the card contains LSI circuitry such as an interface controller chip, a sequence of commands can be defined to reset the interface controller to its power-on state.

- If the card does not have such a sequence, the card must be capable of being reset to its power-on state via a memory write cycle to address 1 as shown in Figure 3-8. The data written is 80 (H) (this may be ignored to simplify the design).

Initially, only bits 0-4 defined the device ID; software typically masked off the upper 3 bits and examined the lower 5 bits. However, because future internal I/O devices as well as I/O cards will have IDs, it became necessary to increase the number of IDs. Accordingly, Bits 5 and 6 have been defined as the Secondary ID bits. Formerly these bits were S (Smart card identifier) and R (Reserved), respectively. Table 3-2 provides an example listing of some of the ID numbers and addresses. This listing was current as of February 1987 but changes sporadically and is not updated at regular intervals.

Unfortunately, not all 128 IDs (7 bits) are available. This is because, as stated above, existing software only looks at the lower 5 bits. For example, the HP 98625 Disc Interface card uses ID 8. If another external I/O card wanted to use this same Primary ID with a different Secondary ID, problems will result. This is because the Secondary ID bits may be ignored, causing the card to be interpreted as the HP 98625. However, this same Primary ID can be used (with a different Secondary ID) by an internal I/O device in a new mainframe since the operating system will look at all 7 bits. Therefore, certain existing Primary IDs are used by new internal I/O devices where the functions are the same or similar.

To be completely safe, additional IDs can be defined using Register 5, the Extension ID Register. If the Primary ID is 0 (Secondary ID = don't care), then the device ID is defined by the Extension ID in Register 5. Register 5 contains the Extension ID only if the Primary ID is 0.

(ADDRESS 3)	7	6	5	4	3	2	1	0
READ STATUS	IE	IR	INTERRUPT LEVEL		X	X	DE1	DE0
WRITE CONTROL	IE	X	X	X	X	X	DE1	DE0

X - These bits are not defined and may be assigned functions by the card designer.

Figure 3-9. Register 3H — Card Status and Control

IE Interrupt request Enabled, set or cleared by a Write Control, is read by a Read Status. IE is cleared by both a bus reset ($\overline{\text{RESET}} = \text{low}$) and a card reset (write to Register 1).

IR Interrupt Request. The card is requesting an interrupt. Used for software polling to determine interrupt origin. If **IR** is true and **IE** is set, one of the DIO Bus interrupt lines ($\overline{\text{IR3}}$, $\overline{\text{IR4}}$, $\overline{\text{IR5}}$ or $\overline{\text{IR6}}$) is asserted, depending on the interrupt level switches. A software-accessible means of clearing **IR** must be provided on the card. **IR** should **not** be cleared by a read of this register.

Interrupt Level These bits permit reading of the interrupt level as shown below:

00	Interrupt Level 3
01	Interrupt Level 4
10	Interrupt Level 5
11	Interrupt Level 6

Current I/O cards have 2 switches to set the interrupt level; these switches map into the two interrupt level bits. Alternately, a card could permit the interrupt level to be programmable (such as, using the corresponding bits in the writeable Control Register). Refer to *Chapter 13, Bus Slave Design Summary* for more details on interrupt levels.

DE0, DE1 DMA Enabled on channel 0, DMA Enabled on channel 1, set or cleared by a Write Control, read by a Read Status. If the card does not implement DMA, these bits can be used for other functions in both the control and status registers. **DE0** and **DE1** are cleared by both a bus reset ($\overline{\text{RESET}} = \text{low}$) and a card reset (write to Register 1).

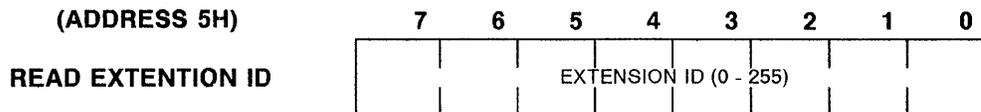


Figure 3-10. Register 5 — Card Extension ID

The Extension ID register is valid only if the Primary ID in Register 1 is 0.

Table 3-2. ID and addresses (As of 2-87 — Subject to Change)

ID	Product	Select Code	Primary Code	Secondary Code	External Code
0	Use ID Extension (Reg. 5)				
1	HP 98624A HP-IB	8	1	0	0
2	HP 98626A RS-232	9	2	0	0
	Series 300 Internal RS-232		2	1	0
	98644A Lowcost RS-232	9	2	2	0
3	98622A GPIO 16-Bit	12	3	0	0
4	98623A BCD Parallel	11	4	0	0
5	98642A 4 Channel MUX	13	5	0	0
6	Free				
7	For Future Use <i>SCSI</i>	14	7		
8	98625A HP-IB Disc <i>B?</i>	14	8	0	0
	Series 300 Internal HP-IB		8	1	0
9	Series 300 Keyboard		9	0	0
10	Floating Point Card	12 (Int)	10	0	0
	286 Emulator Card	13 (Int)	10	1	0
11	Series 300 Internal Timer	11 (Int)	11	0	0
12	98647A Instrument Interface	10	12	0	0
13	Instrument Controllers (Proprietary)	13	13	0	0
14	Free				
15	Customer Card 1	15	15	X	X
16	Customer Card 2	16	16	X	X
17	98646A VME Bus Adapter	24	17	0	0
18	98640A ADC (Meas. and Control)	18	18	0	0
19	For Future Use				

Table 3-2. ID and addresses (Continued)

ID	Product	Select Code	Primary Code	Secondary Code	External Code
20	FDL Master (98691 Based Data Comm)		20	1	0
	98628A DSN-DL (98691 Based)	20	20	1	0
	98629A SRM (98691 Based)	21	20	1	0
	50962A SRM (Direct Coax 98629)	21	20	1	0
	98691A User PDI	20	20	1	0
	98628A ASYNC (98691 Based)	20	20	1	0
	98641A RJE (98691 Based)		20	1	0
21	Ethernet LAN	21	21	0	0
	98643A 802.3 LAN	21	21	1	0
22	98695A 3270 Emul.	22	22	0	0
23	Data Comm Devices				
24	Free				
25	98700A Display	25	25	0	0
	98544B Display	25	25	1	0
	98547A Color Display	25	25	1	0
	98542A Mono Display	25	25	1	0
	98543A Color Display	25	25	1	0
26	Internal <i>Quad</i>				
27	98253A Eprom Prog.	27	27	0	0
28	98627A RBG Video	28	28	0	0
29	Internal <i>Double</i>	29	29	0	0
30	98259A Bubble Memory	30	30	0	0
31	Internal		31	0	0

Data Transfers

Introduction

This portion of the DIO II Accessory Development Guide deals with both DIO and DIO II data transfers. DIO and DIO II are both asynchronous buses, with DIO II being a superset of DIO. Care should be taken when designing for both DIO and DIO II as timing is different for each type of bus.

Masters use the DIO II Bus to talk to slaves, which are devices that manipulate data at the masters request. The master initiates a bus cycle by asserting the address and the appropriate address strobe(s) and data strobe(s). The slave then responds by taking the data off the bus during a Write cycle or by putting data on the bus during a Read cycle. The slave then completes the cycle by asserting one of the DTACK (Data Transfer ACKnowledge) signals. This chapter shows both DIO and DIO II timing during these transactions.

DIO II operation supports four distinct types of bus cycles:

Address Only Cycle
Single Data Transfer Cycle
Block Data Transfer Cycle
Read Modify Write (RMW) Cycle

In DIO operation the Single Data Transfer Cycle is the only one supported over the 16 bit data bus BD15 - BD0. The four types of bus cycles in DIO II as well as the DIO Bus cycles will be discussed in this chapter.

Data Transfer Signal Names

The bus signals that are used in data transfers are defined in Table 4-1.

Table 4-1. Signal Names

Signal Name	Definitions
$\overline{\text{ADACK}}$	Address ACKnowledge is an output from a card that is addressed. This is used by the bus expander to reverse its data bus buffers when a card in the Bus Expander is addressed. All DIO II slaves must generate an $\overline{\text{ADACK}}$ signal.
$\overline{\text{AS}}$	Address Strobe. Not a bus signal. Refers to $\overline{\text{BAS24}}$ or $\overline{\text{BAS32}}$.
BA1	Buffered Address 1 is used to determine if the upper or lower word has the valid data.
BA1-BA31	This is the 31 bit address bus. BA0 is not on the bus. Its value is determined by the data strobes (BUDS and BLDS).
$\overline{\text{BAS}}$	Buffered Address Strobe is not a bus signal but is used to generically refer to either $\overline{\text{BAS24}}$ or $\overline{\text{BAS32}}$.
$\overline{\text{BAS24}}$	Buffered Address Strobe 24 defines when the 23 bit address space (A1-A23) is valid. This signal is the old DIO $\overline{\text{BAS}}$.
$\overline{\text{BAS32}}$	Buffered Address Strobe 32 defines when the 31 address lines are valid. This line is asserted over the complete 32 bit address space. This means that $\overline{\text{BAS32}}$ will be asserted when talking to DIO space with a 32 bit master.
BD0-BD15	Buffered Data 0 - 15 is the DIO data bus. This data port maps to the most significant bits in DIO II.
$\overline{\text{BDS}}$	Buffered Data Strobe is not a bus signal but is used generically to refer to either $\overline{\text{BUDS}}$ and/or $\overline{\text{BLDS}}$.
BFC0-BFC2	Buffered Function Codes 0 - 2 defines the type of transaction occurring on the bus. Refer to the MC68020 manual for correct decoding.
$\overline{\text{BLK}}$	BLock indicates that a block transfer is occurring on the bus.
BR/ $\overline{\text{W}}$	Buffered Read/ $\overline{\text{Write}}$. High for read and low for write.
$\overline{\text{BUDS}}$, $\overline{\text{BLDS}}$	Buffered Upper Data Strobe and Buffered Lower Data Strobe. $\overline{\text{BUDS}}$ indicates that the upper byte of either the upper or lower word is valid. $\overline{\text{BLDS}}$ indicates that the lower byte of either the upper or lower word is valid.
$\overline{\text{DSACK}}$	$\overline{\text{DSACK}}$ is a generic term meaning $\overline{\text{DSACK32}}$ or $\overline{\text{DSACK16}}$ but not $\overline{\text{DTACK16}}$.

Table 2-1. Signal Names (cont'd)

Signal Name	Definitions
$\overline{\text{DSACK16}}$	Data transfer and Size ACKnowledge 16 is asserted by a slave with a 16 bit data port in response to the assertion of $\overline{\text{BAS32}}$. It also informs the bus master that it has completed the bus cycle. Note that $\overline{\text{DSACK16}}$ informs the master as to the size of the data port and contains no information as to number of bytes transferred.
$\overline{\text{DSACK32}}$	Data transfer and Size ACKnowledge 32 is asserted by a slave in response to $\overline{\text{BAS32}}$. It informs the master it has a 32 bit data port and that it has completed the bus cycle. Note that $\overline{\text{DSACK32}}$ informs the master as to the size of the data port and contains no information as to the number of bytes transferred.
$\overline{\text{DTACK}}$	Data Transfer ACKnowledge is not a bus signal but is used to generically refer to either $\overline{\text{DTACK16}}$, $\overline{\text{DSACK16}}$, or $\overline{\text{DSACK32}}$. This signal is used by the accessed slave to inform the bus master that it has completed a bus cycle. During a read cycle, it indicates that the bus slaves data is valid on the bus. During a write cycle it indicates that the bus slave has accepted the data. $\overline{\text{DTACK16}}$ is the old DIO $\overline{\text{DTACK}}$ and is used by those devices that are running a cycle in response to the assertion of $\overline{\text{BAS24}}$. The two new $\overline{\text{DSACK}}$ signals determine the size of the data port and each is asserted in response to a bus cycle beginning with the assertion of $\overline{\text{BAS32}}$.
$\overline{\text{DTACK16}}$	Data Transfer ACKnowledge 16 is asserted by a slave to end a bus cycle started by the assertion of $\overline{\text{BAS24}}$. It must be asserted when responding to $\overline{\text{BAS24}}$ since the master only has a 16 bit port and the slave must emulate the same size port. This signal is the old DIO $\overline{\text{DTACK}}$.
$\overline{\text{ENDT}}$	ENable DTack is defined for DIO only. It is generated by masters and used by bus slaves to control generation of $\overline{\text{DTACK}}$. Slaves must be able to respond with a $\overline{\text{DTACK}}$ if this signal is not generated.
$\overline{\text{LWORD}}$	Long WORD indicates that a long word transfer is being initiated on the bus. It is defined with respect to $\overline{\text{BAS32}}$ and must be driven when $\overline{\text{BAS32}}$ is asserted.
$\overline{\text{RMC}}$	Read Modify write Cycle indicates that a read modify write transfer is being initiated on the bus.
XD0-XD15	eXtended Data 0 - 15, this is the lower 16 bits of the 32 bit DIO II data bus. These signals are not defined for DIO cycles.

Data Transfer Overview

Once a master has control of the data bus it begins a cycle by driving the address bus, function codes, $\overline{\text{BLK}}$ and $\overline{\text{LWORD}}$. The appropriate $\overline{\text{BAS}}$ is then asserted. A master who only has access to the DIO Bus (a 24 bit address port) will not drive $\overline{\text{BAS32}}$, $\overline{\text{LWORD}}$ or $\overline{\text{BLK}}$. On the other hand a master with a 32 bit address port **will always** drive $\overline{\text{BAS32}}$, $\overline{\text{LWORD}}$, and $\overline{\text{BLK}}$. When talking to the lower 16 Megabytes, a 32 bit master will assert both $\overline{\text{BAS32}}$ and $\overline{\text{BAS24}}$. However, if a situation occurs where there is contention between $\overline{\text{BAS32}}$ and $\overline{\text{BAS24}}$, $\overline{\text{BAS32}}$ will be the signal that takes priority.

The slave when responding to $\overline{\text{BAS32}}$ will assert the appropriate $\overline{\text{DSACK}}$ depending on its port size. The slave responding to $\overline{\text{BAS32}}$ will assert $\overline{\text{DSACK16}}$ if it only has a 16 bit port, or it will assert $\overline{\text{DSACK32}}$ if it has a 32 bit port. If the slave is responding to $\overline{\text{BAS24}}$ it will assert $\overline{\text{DTACK16}}$ regardless of port size. The 32 bit slave responding to $\overline{\text{BAS24}}$ will have to emulate a 16 bit port even if it has 32 bit capability.

If the slave does not respond with $\overline{\text{DTACK}}$ within a specified amount of time the system controller (board with the arbiter) asserts $\overline{\text{BERR}}$. $\overline{\text{BERR}}$ is explained in full later in this guide. Slaves that are designed to work in DIO machines should consider disabling the control circuitry from looking at $\overline{\text{BAS32}}$, $\overline{\text{LWORD}}$, and $\overline{\text{BLK}}$ by installing a switch or jumper. These signals are not pulled up in the Series 200 machines or in Series 300 models 310 and 320 machines.

Address and Data Alignment

DIO II has a 4 Gigabyte address space and a 32 bit data port. DIO is a subset of DIO II which accesses the lower 16 Megabytes of the address space with a 16 bit data port. The smallest addressable unit in both DIO II and DIO is the byte location. The largest addressable unit in DIO II is the long word (32 bits), while in DIO it is the word (16 bits). DIO II's data bus consists of BD15 to BD0 in the upper 16 bits and XD15 to XD0 in the least significant 16 bits, while DIO's data bus consists of BD15 to BD0 (Figure 4-1).

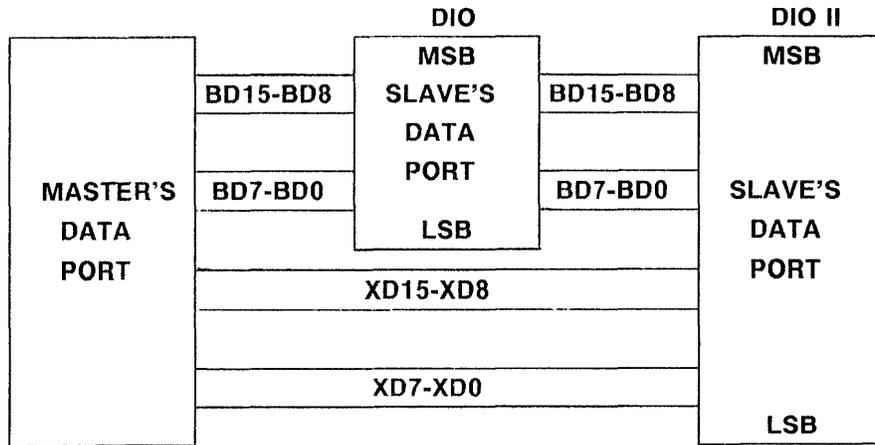


Figure 4-1. Data Transfer Block Diagram

The slave in DIO II transfers determines how the data is aligned and how large the data transfer is by looking at $\overline{\text{BU DS}}$, $\overline{\text{BL DS}}$, BA1 , and $\overline{\text{LWORD}}$. The data bus and byte locations for DIO II are broken up as shown in table 4-1. In DIO the slave determines which bytes are addressed by looking at the data strobes. $\overline{\text{LWORD}}$ and $\overline{\text{BLK}}$ have no meaning in these DIO transfers. Remember, in DIO the largest transfer is the word. Therefore, DIO slaves must be word addressable.

Table 4-2. Byte Addressing

Byte Location	Byte Address
Byte 0	XXXX...XXXX00
Byte 1	XXXX...XXXX01
Byte 2	XXXX...XXXX10
Byte 3	XXXX...XXXX11

Note: Address 0 is shown for clarity but is not present on the bus

Masters can access any byte of a four byte long word in DIO II. The slave determines which byte(s) are accessed during a data cycle by looking at $\overline{\text{LWORD}}$, BA1 , $\overline{\text{BU DS}}$, and $\overline{\text{BL DS}}$. Masters can access any combination of bytes (1, 2, 3, or 4) in a single data transfer during DIO II operation. During DIO cycles only two bytes can be accessed during one cycle. Table 4-3 (on the following page) shows the relationship between the alignment strobes and the accessed data for DIO II transfers.

Although DIO cycles are not explicitly shown in table 4-2, they are similar to the Single Transfers in DIO II (except that $\text{BA31} - \text{BA24}$, $\text{XD15} - \text{XD0}$, $\overline{\text{BLK}}$, and $\overline{\text{LWORD}}$ are undefined and may not be driven). If a slave does not support a particular type of transfer (like block transfers or unaligned transfers), and it is requested to do the unsupported transfer, it **MUST** provide a bus error to tell the controlling device that it cannot complete the request.

Table 4-3. Transfer Signal Level Relationships

TRANSFER TYPE	ACCESSED BYTE	<u>LWORD</u>	BA1	<u>BUDS</u>	<u>BLDS</u>	
Address Only		Note 1		1	1	
Single	Single Byte					
	Byte 0	1	0	0	1	
	Byte 1	1	0	1	0	
	Byte 2	1	1	0	1	
	Byte 3	1	1	1	0	
	Double Byte					
	Byte 0 & Byte 1	1	0	0	0	
Byte 2 & Byte 3	1	1	0	0		
Quad Byte	Byte 0 - 3	0	0	0	0	
	Block	Single Byte	1	Note 2		
		Double Byte	1	Note 3	0	0
Quad Byte		0	0	0	0	
RMW	Single Byte					
	Byte 0	1	0	0	1	
	Byte 1	1	0	1	0	
	Byte 2	1	1	0	1	
	Byte 3	1	1	1	0	
	Double Byte					
	Byte 0 & Byte 1	1	0	0	0	
Byte 2 & Byte 3	1	1	0	0		
Quad Byte	Byte 0 - 3	0	0	0	0	
	Unaligned	Byte 0 - 2	0	0	0	1
Byte 1 - 3		0	0	1	0	
Byte 1 - 2		0	1	0	0	

Note 1 Both data strobes are maintained high during Address Only cycles, but BA1 and LWORD may be either a one or zero.

Note 2 During Single Byte block transfers the data strobes are alternately driven low. Either data strobe might be driven low on the first transfer. BA1 is only valid on the first data transfer and may be either high or low depending on which byte was the first to be transferred.

Note 3 During Word Block transfers both data strobes are asserted on each transfer. BA1 is only valid on the first data transfer and may be either high or low depending on which word was the first to be transferred.

The Single Byte and Double Byte single transfers are supported in DIO. Block, RMW, and unaligned transfers are **not** supported in DIO. Since only 16 bit data ports are supported in DIO the slaves must be word addressable. The data strobes tell the slave which byte(s) of the word carries valid data. The bus master puts valid bytes of data on the data bus and it is up to the slave to determine how the data is routed to the correct location. The data is defined to travel across the data bus in the manner shown in table 4-3 for DIO II operation.

Table 4-4. Transfer Data Lines

TRANSFER TYPE	VALID BYTE(S)	DATA LINES			
		BD15-BD8	BD7-BD0	XD15-XD8	XD7-XD0
Address Only		no bytes transferred			
Single	Single Byte Byte 0 Byte 1 Byte 2 Byte 3	Byte 0	Byte 1 Byte 3		
	Double Byte Byte 0 & Byte 1 Byte 2 & Byte 3	Byte 0 Byte 2	Byte 1 Byte 3		
	Quad Byte Byte 0 - 3	Byte 0	Byte 1	Byte 2	Byte 3
Block	Single	Note 1			
	Double Byte	Note 2			
	Quad Byte	Byte 0	Byte 1	Byte 2	Byte 3
RMW	Single Byte Byte 0 Byte 1 Byte 2 Byte 3	Byte 0	Byte 1 Byte 3		
	Double Byte Byte 0 & Byte 1 Byte 2 & Byte 3	Byte 0 Byte 2	Byte 1 Byte 3		
	Quad Byte Byte 0 - 3	Byte 0	Byte 1	Byte 2	Byte 3
Unaligned	Byte 0 - 2	Byte 0	Byte 1	Byte 2	
	Byte 1 - 3		Byte 1	Byte 2	Byte 3
	Byte 1 - 2		Byte 1	Byte 2	

Note 1 During single byte block transfers the data is transferred 8 bits at a time over BD15 - BD8 or BD7 - BD0. Byte 0 and Byte 2 are transferred over BD15 - BD8. Bytes 1 and Byte 3 are transferred over BD7 - BD0. The bytes are transferred in order after the first one. An example of this would be a single byte transfer beginning with Byte 2, then Byte 3 then Byte 0 then Byte 1 etc. until the last byte is transferred. Any one of the four bytes can begin the transfer:

Note 2 During a double byte block transfer, the data is transferred 16 bits at a time over BD15 - BD0. The bytes are transferred in the same way as a single double byte transfer. An example of this would be to start with the upper 2 bytes, Byte 2 and Byte 3 then Byte 0 and Byte 1 then Byte 2 and Byte 3, etc. Remember that either word could start the transfer.

DIO cycles transfer bytes across the bus only on BD15 - BD0. The Single Byte and Double Byte single transfers define the correct data alignment for DIO cycles.

Data Transfer Timing

The timing for Address Only, Single, Block, and Read Modify Write transfers is given in the following sections. The timing is with respect to the receiving device. In other words, the timing is given with respect to the master or slave (whichever is the receiving device). To help clarify timing issues, notes follow each timing specification figure and table.

Bus Skew

If two drivers begin to drive the DIO II data bus at the same time, this does not mean the receiving device will see both lines valid at the same time. The time between the two lines being seen valid by the receiving device is called skew. Skew must be considered when designing for the DIO II Bus. The skew calculations in this manual assume that similar drivers are used to drive the bus lines.

The worst case skew between signals asserted at the same time was calculated at 18 nsecs (assuming similar drivers). To give some margin, a maximum skew of 25 nsecs is used in DIO II timing (15 nsec in DIO timing). The skew of 15 nsec is appropriate for those signals that have the same loading. The greater skew is a concern when considering the loading on the newly defined DIO II lines with respect to the potentially more heavily loaded DIO signals. The skew **MUST** be considered when meeting the timing requirements of DIO or DIO II. If in doubt when designing, use a skew time of 25 nsecs. Many existing DIO I/O cards only load the lower data strobe. Therefore, the skew between data strobes should be considered as being 25nsecs.

Address Only Cycles

Address only cycles are supported in DIO II operation only and not in DIO. The purpose of the cycle is to allow for future products to pass the address and address strobe through to the DIO II Bus without waiting for the data strobes. This application is not supported for DIO operation. Future products will have to wait for data strobes before asserting address strobes when using a DIO Bus. This will guarantee that a valid data transfer has begun. The address only cycle operates as follows.

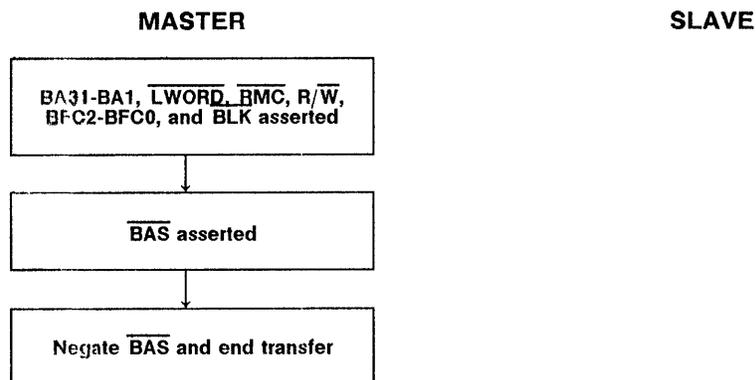


Figure 4-2. Address Only Flowchart

The master does not have to drive \overline{RMC} , or \overline{BLK} . They are pulled high on the bus and the master only needs to drive them if the functionality they define is needed. The master does not wait for a \overline{DTACK} or \overline{BERR} before completing the cycle. Furthermore, the slave cannot respond with \overline{DTACK} or \overline{BERR} during an Address Only cycle. This **correctly** implies that the slave cannot respond to any data transfer cycle unless the data strobes have been asserted. This allows the master, if it has control of the DIO II Bus, to begin a cycle in DIO II space before it has determined that the slave is on the DIO II Bus. If the master does not control the DIO II Bus it must make sure that it wants to run a DIO II cycle before it requests the bus. The slave can respond with \overline{ADACK} .

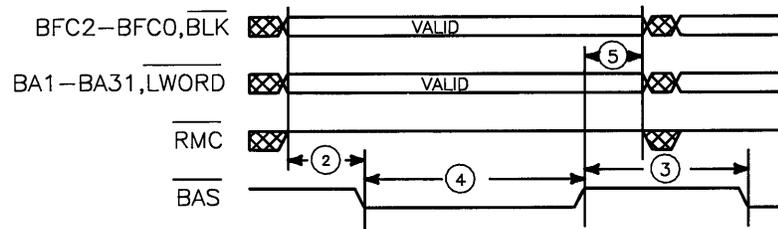


Figure 4-3. Address Only Timing Diagram

Table 4-5. Address Only Timing

NUM	TIMING PARAMETER	MASTER		SLAVE	
		MIN	MAX	MIN	MAX
1a	\overline{Bas} low when Bus Error occurs			5600	
1b	\overline{Bas} low to avoid Bus Error				5100
2	BA1 - BA31, \overline{LWORD} , BFC2 - BFC0, and \overline{BLK} set up time before \overline{BAS}			15	
3	High time for \overline{BAS} between cycles			140	
4	\overline{BAS} low time			70	
5	Hold time on BA1 - BA31, BFC2 - BFC0, \overline{BLK} , and \overline{LWORD} after \overline{BAS} is deasserted			15	

Note 1 a) If \overline{BAS} is asserted for more than the specified amount of time a bus error will occur. The time is dependent on the processor being used. In earlier Series 200 (and Series 300 Models 310 and 320) processors the time was 4 μ secs. This is irrelevant for address only cycles since they are not supported for DIO cycles.

b) This second specification is to provide a margin so that the \overline{BERR} circuitry can be disabled after \overline{BAS} is deasserted.

Note 2 These signals must be set up at the receiving device (slave) with the time specified, or an invalid address could be latched into the slave.

Note 3 The master, when running two consecutive cycles, can not drive \overline{BAS} low until it has been high on the bus for this amount of time. Therefore, all slaves are guaranteed this amount of high time on \overline{BAS} between cycles.

Note 4 Guarantees that \overline{BAS} will be low for this amount of time at the receiving device.

Note 5 This timing parameter guarantees the slave that the address will be valid the entire time that \overline{BAS} is asserted.

SINGLE READ

The single read cycles are defined below for both DIO and DIO II cycles. There are some timing differences between DIO and DIO II. Therefore, care should be taken when designing to either bus. In single transfers only, one data transfer is allowed across the bus. Only one assertion of the data strobe(s) for each assertion of the address strobe is allowed. The steps in Figure 4-4 are necessary for a single byte read transfer.

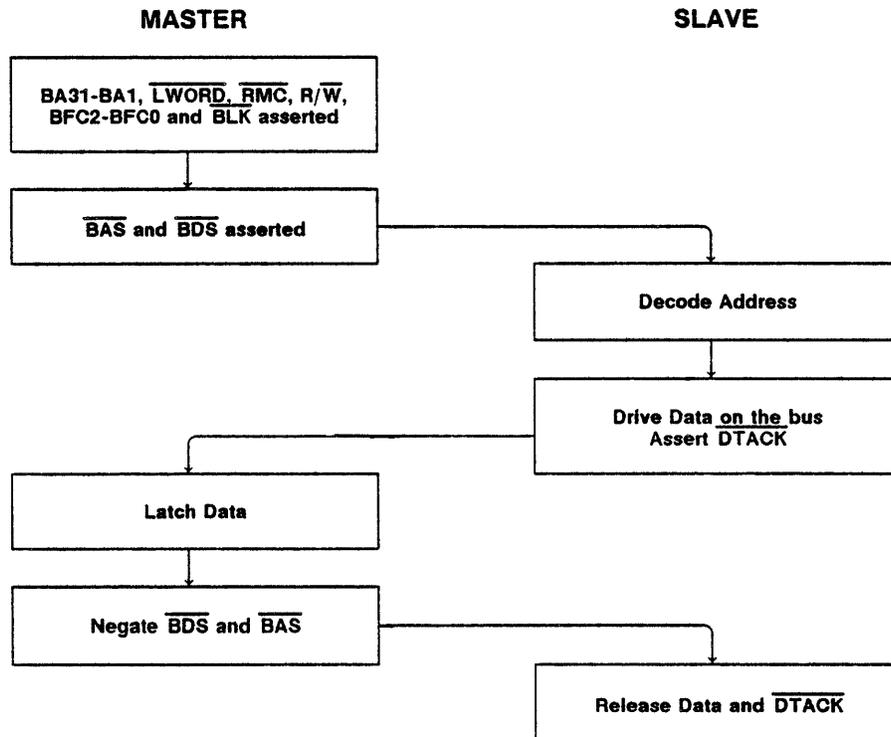


Figure 4-4. Single Read Flowchart

The steps of the flowchart are true for both DIO and DIO II cycles. The timing for both DIO and DIO II is shown on the following pages.

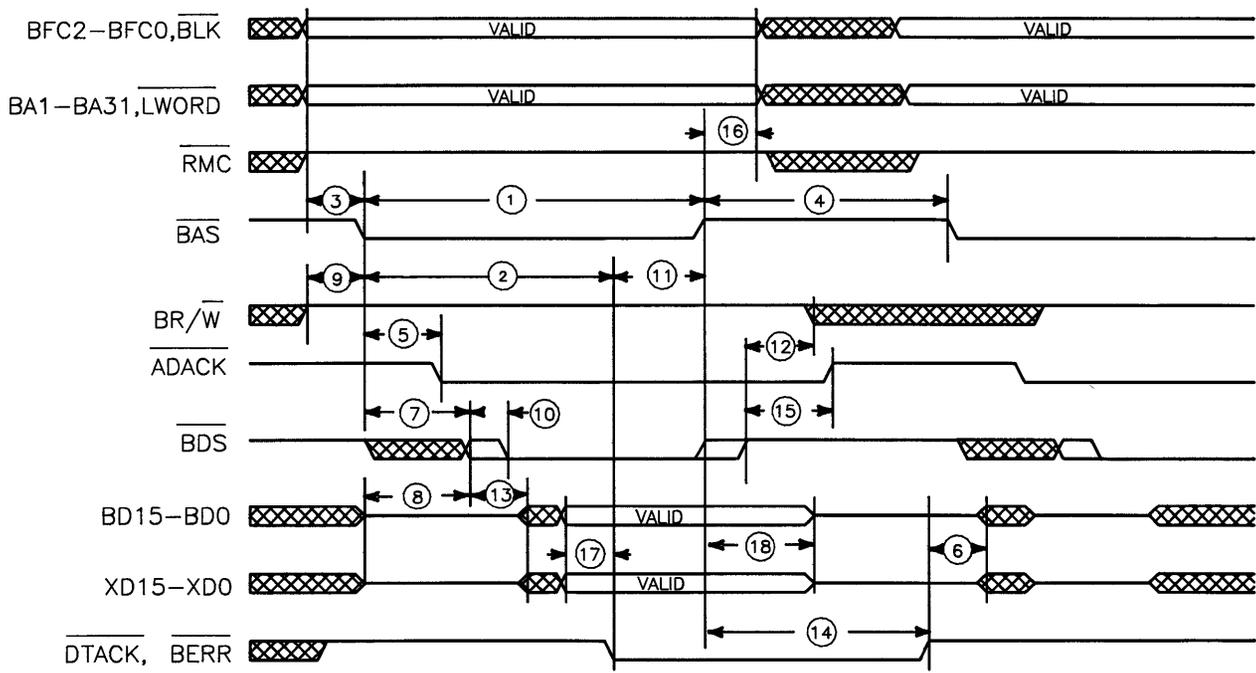


Figure 4-5. Single Read Timing Diagram

Table 4-6. Single Read Timing

NUM	TIMING PARAMETER	MASTER		SLAVE	
		MIN	MAX	MIN	MAX
1a	$\overline{\text{BAS}}$ low when Bus Error Timeout occurs			5600	
1b	$\overline{\text{BAS}}$ low to avoid Bus Error Timeout				5100
2a	$\overline{\text{BAS}}$ low to $\overline{\text{DTACK}}$ low without a Bus Error, DMA		3100		
2b	$\overline{\text{BAS}}$ low to $\overline{\text{DTACK}}$ low without a Bus Error, non DMA		4600		
3	BA1 - BA31, $\overline{\text{LWORD}}$, $\overline{\text{RMC}}$, BFC2 - BFC0, $\overline{\text{BLK}}$ set up time before $\overline{\text{BAS}}$			15	
4	High time for $\overline{\text{BAS}}$ between cycles			140	
5	$\overline{\text{BAS}}$ low on the bus to $\overline{\text{ADACK}}$ low	0	70		
6	Time when master can begin driving data lines after $\overline{\text{DTACK}}$ high	50			
7	$\overline{\text{BAS}}$ low to $\overline{\text{BDS}}$ low			-75	75
8	Masters release of data line before $\overline{\text{BDS}}$ asserted			0	
9	BR/ $\overline{\text{W}}$ setup time before $\overline{\text{BDS}}$ asserted			15	
10	Skew between $\overline{\text{BUDS}}$ and $\overline{\text{BLDS}}$				25
11	$\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ low to $\overline{\text{BAS}}$ deasserted			0	350
12	BR/ $\overline{\text{W}}$ hold time after $\overline{\text{BAS}}$ or $\overline{\text{BDS}}$ deasserted			65	
13	Data bus driven after $\overline{\text{BDS}}$ asserted	0			
14	$\overline{\text{BAS}}$ deasserted to $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ release	0	50		
15	$\overline{\text{BAS}}$ high to $\overline{\text{ADACK}}$ release	0	50		
16	BA1 - BA31, $\overline{\text{LWORD}}$, $\overline{\text{RMC}}$, BFC2 - BFC0, $\overline{\text{BLK}}$ hold after $\overline{\text{BAS}}$ or $\overline{\text{BDS}}$ deasserted			15	
17	Data setup before $\overline{\text{DTACK}}$ asserted	-65,15,55			
18	Data bus release after $\overline{\text{BAS}}$ or $\overline{\text{BDS}}$ deasserted	0	100		

Note 1 a) In Series 200 machines which support only DIO cycles, this time was 4 μsecs , while in Series 300 machines (all Series 300) the minimum was increased to 5.6 μsecs .

b) To allow the $\overline{\text{BERR}}$ timeout circuitry to be disabled this is the maximum allowable amount of time for $\overline{\text{BAS}}$ to be low.

Note 2 The Bus Error timer does not know whether a DMA cycle is being executed or not. The shorter time for the DMA cycle reflects the fact that the memory must be read and the I/O operation completed within a single memory cycle.

- Note 3 This guarantees that $\overline{\text{RMC}}$, A31 - A1, BFC2 - BFC0, $\overline{\text{LWORD}}$ and $\overline{\text{BLK}}$ are valid for this amount of time before $\overline{\text{BAS}}$. For DIO cycles when only $\overline{\text{BAS24}}$ is asserted A31 - A24, $\overline{\text{LWORD}}$, and $\overline{\text{BLK}}$ are undefined.
- Note 4 Guarantees that $\overline{\text{BAS}}$ will be high for the time specified. In DIO cycles the time was originally specified to allow enough $\overline{\text{BAS}}$ precharge time. This is not the case for DIO II, but the high time is necessary so that the signals on the bus have time to go to an idle state.
- Note 5 The $\overline{\text{ADACK}}$ signal must be seen by any potential users by the time specified. If the master is looking for a valid $\overline{\text{ADACK}}$ the designer should also consider the time it takes for the slave to see a valid $\overline{\text{BAS}}$ on the bus.
- Note 6 The master cannot begin driving the data lines for this amount of minimum time until $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ has gone high at the end of the cycle. This reduces the potential for contention on the data bus at the end of the cycle.
- Note 7 Skew must be considered when trying to meet this specification. If a master on the DIO II Bus were to assert the data strobes ($\overline{\text{BDS}}$) at the same time as $\overline{\text{BAS}}$, a slave may see the data strobes 25 nsecs before or after the address strobe. DIO slaves are guaranteed to see $\overline{\text{BDS}}$ within the window specified.
- Note 8 The slave is guaranteed that the master has released the data bus before either data strobe is asserted.
- Note 9 The slaves are assured that the $\text{BR}/\overline{\text{W}}$ signal will be set up for this minimum amount of time before either $\overline{\text{BDS}}$ or $\overline{\text{BAS}}$ is valid (whichever comes first).
- Note 10 Due to uncontrolled loading of some DIO cards that use only the lower data strobe, the maximum skew has been used here. Future cards on both DIO and DIO II should load both data strobes equally so as to reduce skew. Devices driving the data strobes should use the same part for generating the strobes and the same part for driving each strobe. The trace length for both data strobes should be identical. The skew between strobes should be as small as possible at the input of the masters bus drivers.
- Note 11 The slave is guaranteed that $\overline{\text{BAS}}$ will remain low until it drives either $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ low (assuming it does this within the bus timeout period). The 350 maximum time ensures that the $\overline{\text{BAS}}$ will be high before the bus error timer generates a bus error. During DMA cycles this maximum time **cannot** be guaranteed. The DMA controller waits for a $\overline{\text{DMARDY}}$ from the I/O device before deasserting $\overline{\text{BAS}}$.
- Note 12 This guarantees that $\text{BR}/\overline{\text{W}}$ will remain true until BOTH data strobes and $\overline{\text{BAS}}$ have been deasserted in DIO II. The potential skew between $\text{BR}/\overline{\text{W}}$ and the $\overline{\text{BDS}}$ or $\overline{\text{BAS}}$ means the master must drive $\text{BR}/\overline{\text{W}}$ for 40 nsecs after both are deasserted (assuming similar drivers are used).

- Note 13 This assures the slave that the data bus will be released by the time the data strobes have been asserted.
- Note 14 This guarantees the master that \overline{DTACK} or \overline{BERR} will not be released until the master has had a chance to look at the signal and drive \overline{BAS} high.
- Note 15 \overline{ADACK} can be used to turn data buffers around in active expanders. Therefore, it needs to be driven until the end of the cycle. This signal could also be used as an address acknowledge line for buffered writes. All slaves **MUST** drive \overline{ADACK} .
- Note 16 The DIO II slave is guaranteed that BA31 - BA1, \overline{LWORD} , \overline{RMC} BFC2 -BFC0, and \overline{BLK} will remain valid until \overline{BAS} or \overline{BDS} goes high (whichever is last). In DIO operation this specification is from the MC68000 hold time minus the bus skew. Also the HP 98256A requires 40 nsecs of address hold time. This was met in old processors because the address bus is typically driven longer than the specified minimum. New masters that plan to support the HP 98256A should meet the necessary 40 nsec hold time.
- Note 17 For DIO devices, which of the three numbers to use is dependent on whether the data is transferred using a DMA card or not. The -65 nsecs data to \overline{DTACK} is for long word transfers only. The master must keep \overline{BAS} on the bus until it latches the data, since \overline{DTACK} must rise within 50 nsecs after \overline{BAS} is deasserted. The 15 nsecs is for byte and word transfers without DMA. If the slave can be read using the HP 98620 card, or a similar device, the byte transfers must have an additional 20 nsecs (to allow the DMA card to fold the data when the byte is on the wrong eight data bits). Therefore, any card responding to a byte read that supports output DMA cycles to both DIO or DIO II devices (using the DIO DMA card or similar general purpose DMA cards) must allow 55 nsecs of set up before they assert \overline{DTACK} . If the card is responding to \overline{BAS} , and it is not a byte write, the assertion of data and $\overline{DSACK16}$ or $\overline{DSACK32}$ complies with the DIO II specification (15 nsec for word transfers and -55 for long words) regardless of DMA. The DMA device is responsible to assert $\overline{DTACK16}$ when talking to DIO cards.
- Note 18 This guarantees the master that the data buffers will be driven until both data strobes or \overline{BAS} (whichever is first) has been released.

Single Write

The single write cycle is defined in Figure 4-6 for DIO II cycles. The write cycle flow chart shows the logical sequence of events during a bus cycle (assuming no bus error occurred).

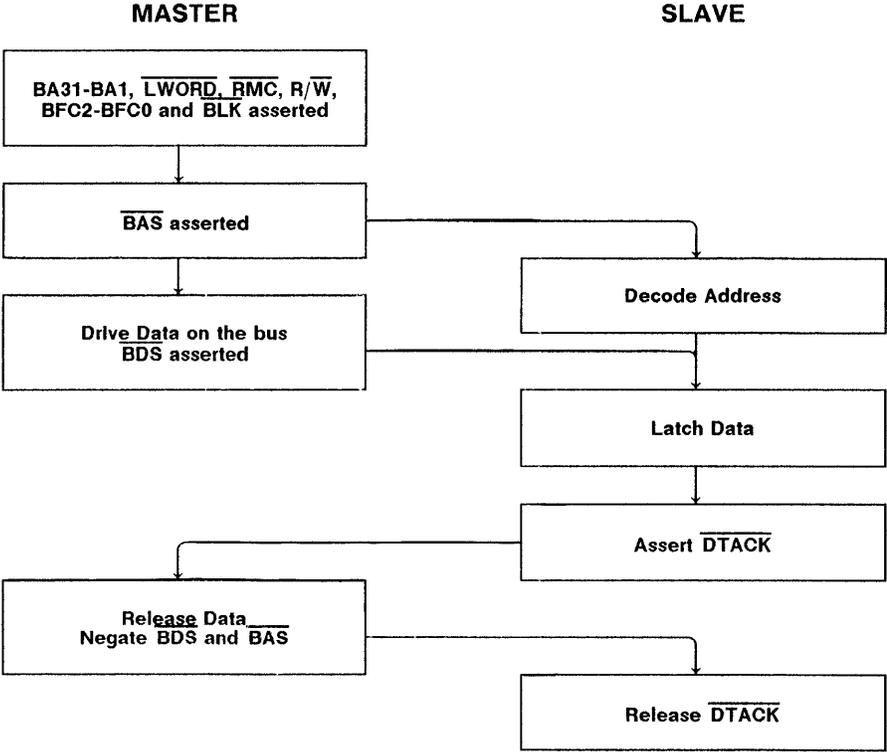


Figure 4-6. Single Write Cycle Flowchart

The steps in Figure 4-6 are true for both DIO and DIO II cycles. The timing for both DIO and DIO II is shown on the following pages.

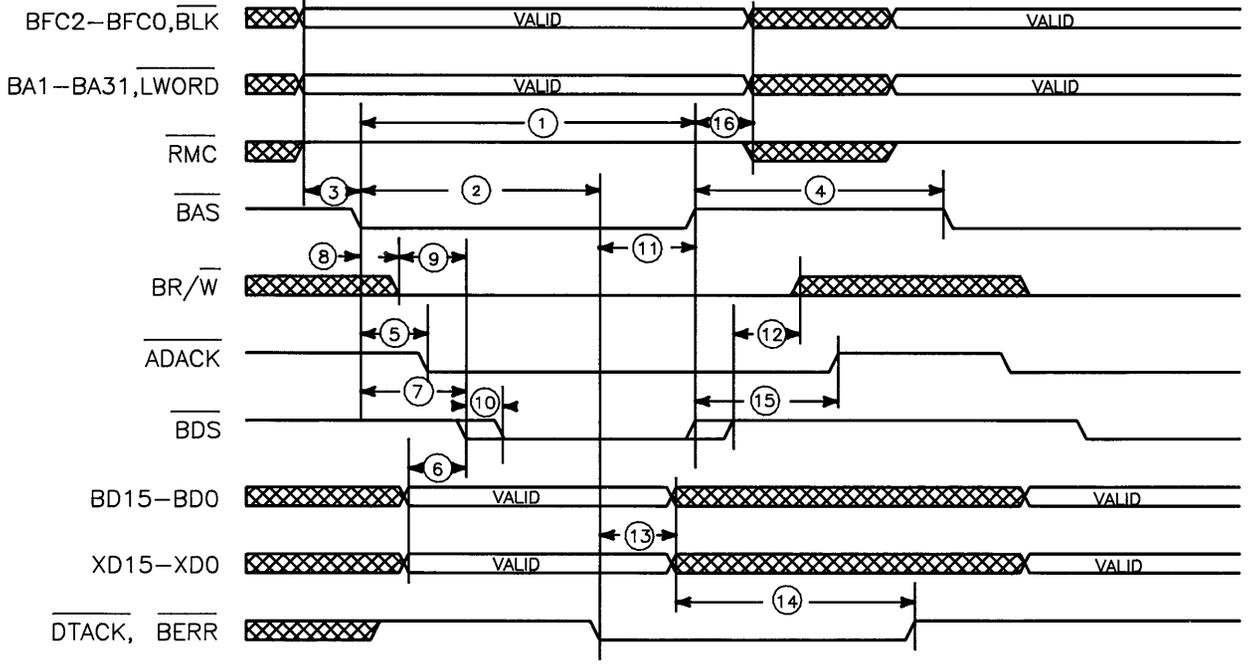


Figure 4-7. Single Write Timing Diagram

Table 4-7. Single Write Timing

NUM	TIMING PARAMETER	MASTER		SLAVE	
		MIN	MAX	MIN	MAX
1a	$\overline{\text{BAS}}$ low when Bus Error Timeout occurs			5600	
1b	$\overline{\text{BAS}}$ low to avoid Bus Error Timeout				5100
2a	$\overline{\text{BAS}}$ low to $\overline{\text{DTACK}}$ low without a Bus Error. DMA		3100		
2b	$\overline{\text{BAS}}$ low to $\overline{\text{DTACK}}$ low without a Bus Error. non DMA		4600		
3	BA1 - BA31, $\overline{\text{LWORD}}$, $\overline{\text{RMC}}$, BFC2 - BFC0, $\overline{\text{BLK}}$ set up time before $\overline{\text{BAS}}$			15	
4	High time for $\overline{\text{BAS}}$ between cycles			140	
5	$\overline{\text{BAS}}$ low to $\overline{\text{ADACK}}$ low	0	70		
6	Data setup before $\overline{\text{BDS}}$ asserted			15	
7	$\overline{\text{BAS}}$ low to $\overline{\text{BDS}}$ low			50	2500
8	$\overline{\text{BAS}}$ low to BR/ $\overline{\text{W}}$ low			-75	45
9	BR/ $\overline{\text{W}}$ low before $\overline{\text{BDS}}$ asserted			65	
10	Skew between $\overline{\text{BUDS}}$ and $\overline{\text{BLDS}}$				25
11	$\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ low to $\overline{\text{BAS}}$ deasserted			85	350
12	BR/ $\overline{\text{W}}$ hold time after $\overline{\text{BDS}}$ deasserted			15	
13	Data release after $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ asserted			85	
14	$\overline{\text{BAS}}$ deasserted to $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ release	0	50		
15	$\overline{\text{BAS}}$ high to $\overline{\text{ADACK}}$ release	0	50		
16	BA1 - BA31, $\overline{\text{LWORD}}$, $\overline{\text{RMC}}$, BFC2 - BFC0, $\overline{\text{BLK}}$ hold after $\overline{\text{BAS}}$ or $\overline{\text{BDS}}$ deasserted			15	

Note 1 a) In Series 200 machines which support only DIO cycles, this time was 4 μ secs. In Series 300 (all Series 300s) the minimum was increased to 5.6 μ secs.

b) To allow the $\overline{\text{BERR}}$ timeout circuitry to be disabled, this amount of time is the maximum allowable $\overline{\text{BAS}}$ low time.

Note 2 The Bus Error timer does not know whether a DMA cycle is being executed. The shorter time for the DMA cycle reflects the fact that the memory must be read and the I/O operation completed within a single memory cycle.

Note 3 This time (measured from the receiving device - slave) guarantees that $\overline{\text{RMC}}$, A31 - A1, BFC2 - BFC0, $\overline{\text{LWORD}}$, and $\overline{\text{BLK}}$ are valid for this amount of time before $\overline{\text{BAS}}$. For DIO cycles (where only $\overline{\text{BAS24}}$ is asserted), A31 - A24, $\overline{\text{LWORD}}$, and $\overline{\text{BLK}}$ are undefined.

Note 4 Guarantees that $\overline{\text{BAS}}$ will be high for the time specified at the slave device. In DIO cycles the time was specified to allow enough $\overline{\text{BAS}}$ precharge time. This is not the case for DIO II, but the high time is necessary so that the signals on the bus have time to go to an idle state.

- Note 5 The $\overline{\text{ADACK}}$ signal must be seen by any potential users by the time specified. If the master is looking for a valid $\overline{\text{ADACK}}$, the designer should also consider the time it takes for the slave to see a valid $\overline{\text{BAS}}$ signal on the bus.
- Note 6 This guarantees the slave that data will be set up by this minimum time before seeing a $\overline{\text{BDS}}$ signal. The master will have to set up data 40 nsecs before asserting the $\overline{\text{BDS}}$ signal in order to meet this timing specification (this assumes similar drivers).
- Note 7 Skew must be considered when trying to meet this specification. A master on DIO II must follow the timing requirements in order to meet the necessary specifications for the slave.
- Note 8 This specifies the maximum limits as to when $\text{BR}/\overline{\text{W}}$ is driven low after $\overline{\text{BAS}}$.
- Note 9 The slaves are assured that the $\text{BR}/\overline{\text{W}}$ signal will be set up for this minimum amount of time before either $\overline{\text{BDS}}$ signal is asserted.
- Note 10 Due to uncontrolled loading of some DIO cards that use only the lower data strobe, the maximum skew has been used here. Future cards on both DIO and DIO II should load both data strobes equally, so as to reduce skew. Devices driving the data strobes should use the same part for generating and for driving each strobe. The trace length for both data strobes should be identical. The skew between strobes should be as small as possible at the input of the masters bus drivers.
- Note 11 The slave is guaranteed that $\overline{\text{BAS}}$ will remain low until it drives either $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ low (assuming it does this within the bus timeout period). The 350 nsec maximum time assures that the $\overline{\text{BAS}}$ will be high before the bus error timer generates an error.
- Note 12 This guarantees that $\text{BR}/\overline{\text{W}}$ will remain true until both data strobes and the $\overline{\text{BAS}}$ have been deasserted in DIO II operation.
- Note 13 This assures the slave that the data will remain valid until $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ is asserted. The data will remain valid for a short period of time after $\overline{\text{DTACK}}$, allowing the slave to finish the cycle before latching in the data.
- Note 14 This guarantees the master that $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ will not be released until the master has had a chance to see the signal and drive $\overline{\text{BAS}}$ high.
- Note 15 $\overline{\text{ADACK}}$ can be used to turn data buffers around in active expanders, therefore, it needs to be driven until the end of the cycle. This signal could also be used as an address acknowledge line for buffered writes. All slaves **MUST** drive $\overline{\text{ADACK}}$.
- Note 16 The DIO II slave is guaranteed that BA31 - BA1, $\overline{\text{LWORD}}$, $\overline{\text{RMC}}$, BFC2 -BFC0, and $\overline{\text{BLK}}$ will remain valid until $\overline{\text{BAS}}$ or $\overline{\text{BDS}}$ goes high (which ever is last). In DIO operation this specification came from the MC68000 hold time minus the bus skew. Also the HP 98256A requires 40 nsecs of address hold time. This was met in old processors because the address bus was typically driven longer than the specified minimum. New masters that plan to support the HP 98256A should meet the necessary 40 nsec hold time.

Block Transfers

Block transfers are similar to single transfers, but are only supported for DIO II cycles. Block transfers are **not** supported for DIO transfers. The block read and write cycles have almost identical timing parameters at the start and finish of the cycle, but different parameters within the transfer. In both read and write operation, the $\overline{\text{BAS32}}$ signal is asserted over the entire transfer. It may be deasserted by the master immediately upon seeing the last $\overline{\text{DSACK}}$ asserted by the slave. Each block of data is transferred by deasserting and reasserting the data strobes. The requirements for block transfers is that they can not cross 256 byte boundaries. This means the slave needs only one eight bit counter on the lower address byte. $\overline{\text{LWORD}}$ and the address are only valid until the first $\overline{\text{DSACK}}$ is seen by the master. $\text{BFC2} - \text{BFC0}$, $\overline{\text{RMC}}$, and $\overline{\text{BLK}}$ are valid until the last $\overline{\text{DSACK}}$ is seen by the master. If the slave is requested to do block transfers and does not support them, the slave must bus error (or not respond).

Note

Slaves that do not support block transfers should use $\overline{\text{BLK}}$ to modify the board select signal. Also, they should not respond to the block transfer since the address on the bus is only valid for the first transfer.

Block Read

Block reads are very similar to single reads. As mentioned above, $\overline{\text{BAS}}_{32}$ is held down for the entire block transfer (along with $\overline{\text{BLK}}$ and valid function codes). $\text{BA}_{31} - \text{BA}_1$ and $\overline{\text{LWORD}}$ are only valid until the first $\overline{\text{DTACK}}$. $\overline{\text{RMC}}$ should not be asserted during a block transfer. The $\text{BR}/\overline{\text{W}}$ line is also asserted for the entire cycle. The slave is allowed to drive the data bus the entire time during block read transfers. It does not have to release the data bus until the last transfer is completed. The steps for a block read are shown in Figure 4-8.

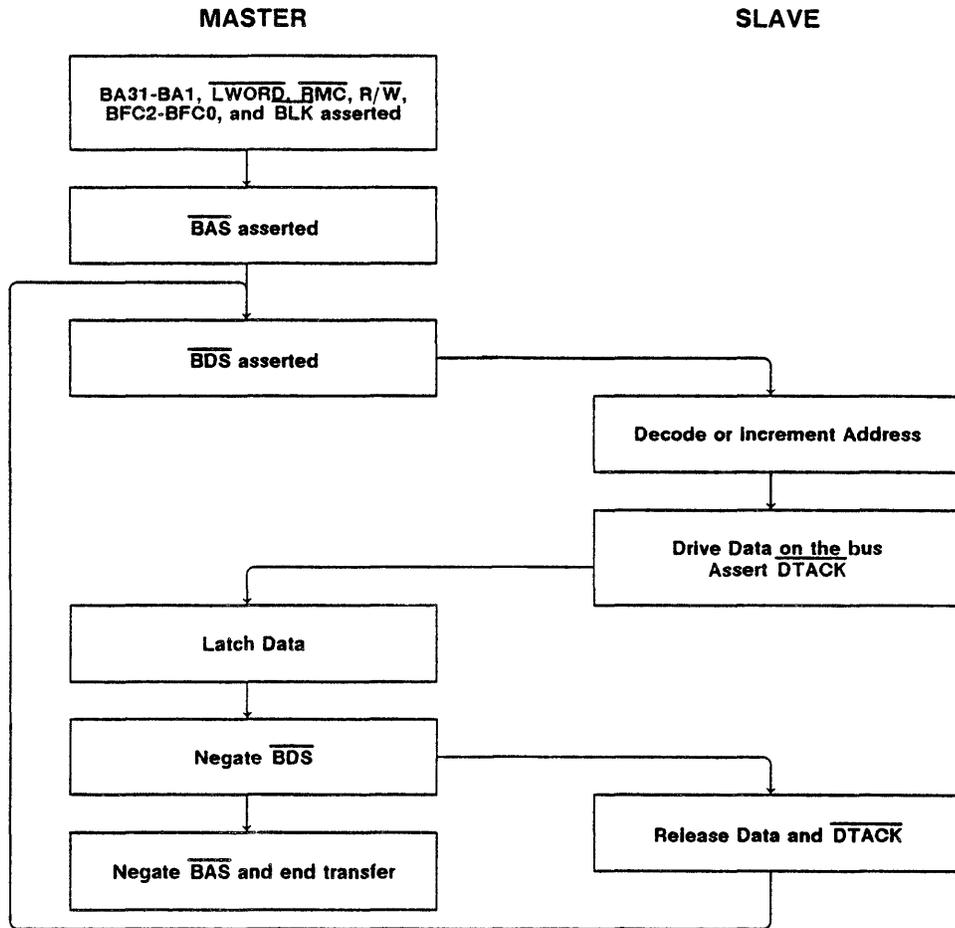


Figure 4-8. Block Read Flowchart

The timing for a block read with two transfers is shown below. Note that the internal timing, with respect to the data strobes and $\overline{\text{DSACK}}$, is different at the beginning of the first and end of the second cycle. The block transfers must be acknowledged using either $\overline{\text{DSACK16}}$ or $\overline{\text{DSACK32}}$, and not $\overline{\text{DTACK16}}$.

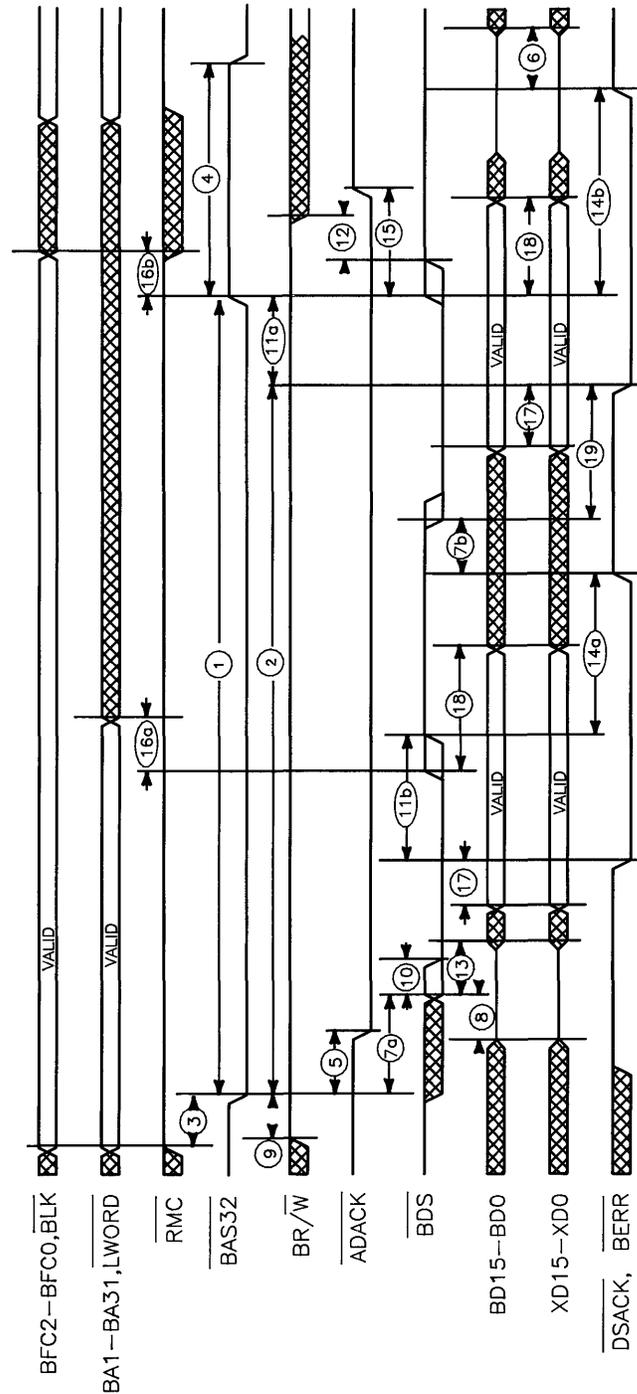


Figure 4-9. Block Read Timing

Block Write

Block writes are very similar to single writes and behave in the same manner as block reads. As mentioned above $\overline{\text{BAS}}_{32}$ is held down for the entire block transfer (along with $\overline{\text{BLK}}$ and valid function codes). $\text{BA}_{31} - \text{BA}_1$ and $\overline{\text{LWORD}}$ are only valid until the first $\overline{\text{DSACK}}$. The $\text{BR}/\overline{\text{W}}$ line is also asserted for the entire cycle. The master is allowed to drive the data bus the entire time of block write transfers. The steps for a block write are shown in Figure 4-10.

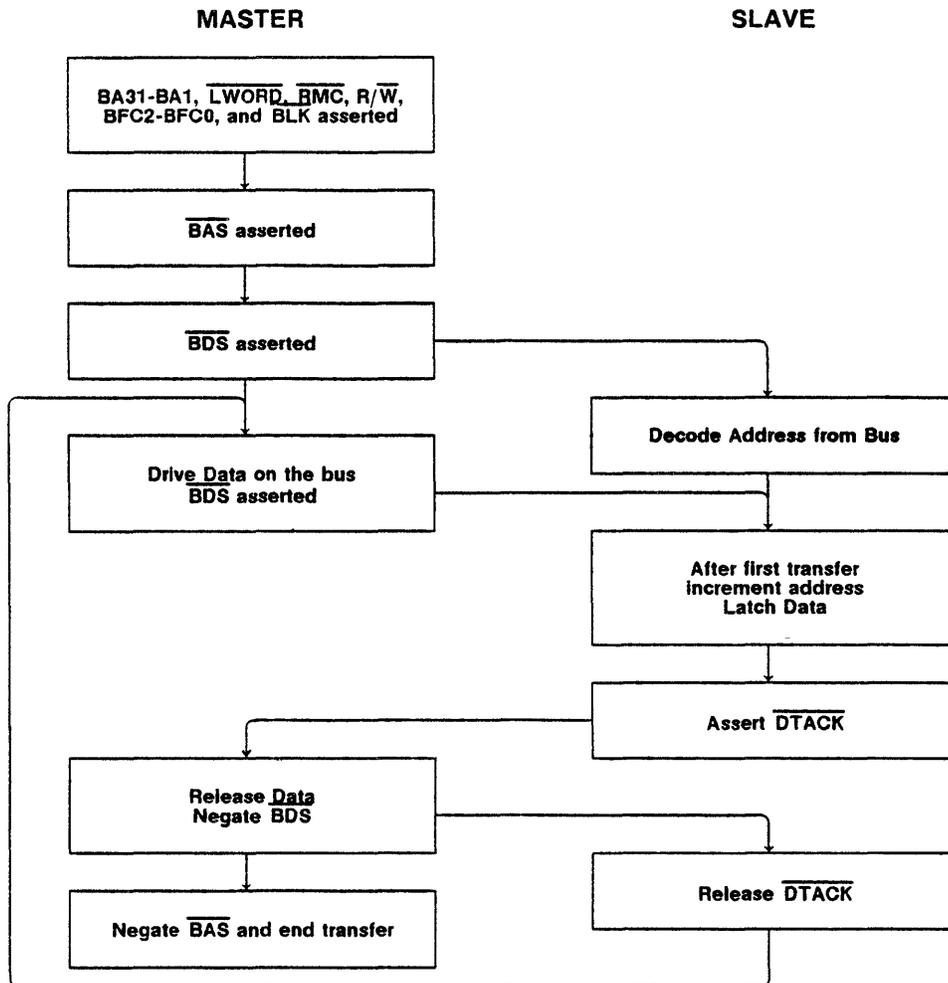


Figure 4-10. Block Write Flowchart

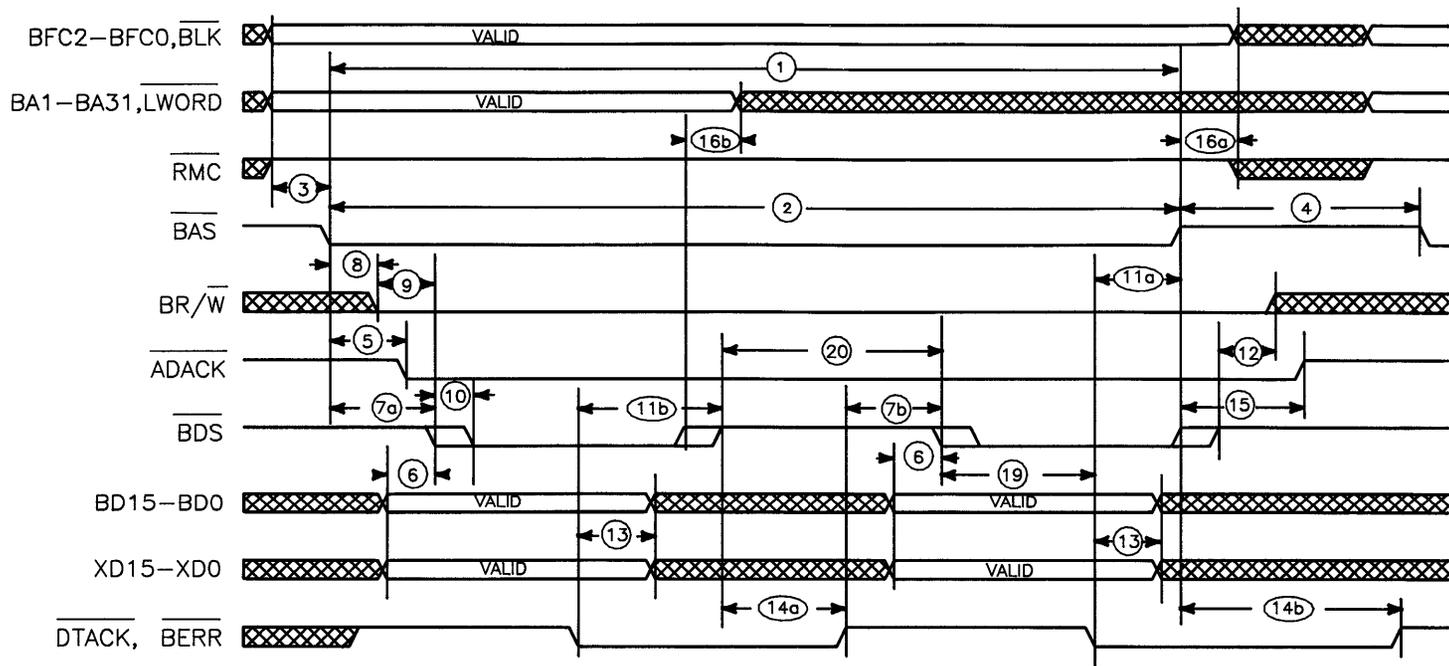


Figure 4-11. Block Write Timing Diagram

Table 4-8 shows the timing for both the block read and write cycle. Only those timing parameters that are different from single transfers are shown. Since not all devices respond to block transfers, those that don't should cause a bus error when asked to do a block transfer. Slaves that do not respond to block transfers should have their board select circuits modified with the $\overline{\text{BLK}}$ signal. They should not respond as the address on the bus is only valid on the first transfer. When the slave does not respond, the master requesting the block transfer will time out and generate a bus error.

Table 4-8. Block Read/Write Timing

NUM	TIMING PARAMETER	MASTER		SLAVE	
		MIN	MAX	MIN	MAX
	The timing parameters 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 13, 15, 17, and 18 are identical to the Single Read or Write cycle respectively and are not repeated here.				
7a	$\overline{\text{BAS32}}$ low to $\overline{\text{BDS}}$ low (first cycle)			50	2500
7b	$\overline{\text{DSACK}}$ high previous cycle to $\overline{\text{BDS}}$ low (all but first transfer)			0	
11a	$\overline{\text{DSACK}}$ or $\overline{\text{BERR}}$ low to $\overline{\text{BAS32}}$ deasserted (last transfer)			0	350
11b	$\overline{\text{DSACK}}$ or $\overline{\text{BERR}}$ low to $\overline{\text{BDS}}$ deasserted (all but last transfer)			0	
14a	$\overline{\text{BDS}}$ deasserted to $\overline{\text{DSACK}}$ or $\overline{\text{BERR}}$ release (all but last transfer)	0			
14b	$\overline{\text{BAS32}}$ deasserted to $\overline{\text{DSACK}}$ or $\overline{\text{BERR}}$ release	0	50		
16a	BFC0 - BFC2, $\overline{\text{RMC}}$, and $\overline{\text{BLK}}$ hold after the deassertion of $\overline{\text{BAS32}}$ or $\overline{\text{BDS}}$			15	
16b	BA1 - BA31 and $\overline{\text{LWORD}}$ hold after first $\overline{\text{BDS}}$ deasserted			15	
19	$\overline{\text{BDS}}$ asserted to $\overline{\text{DSACK}}$ or $\overline{\text{BERR}}$ asserted	30			
20	$\overline{\text{BDS}}$ high time between cycles			30	

- Note 7 a) Skew must be considered when trying to meet this specification. If a master on the DIO II Bus were to assert the data strobes ($\overline{\text{BDS}}$) at the same time as $\overline{\text{BAS}}$, a slave may see the data strobes 25 nsecs before or after the address strobe (assuming similar drivers). DIO slaves are guaranteed that $\overline{\text{BDS}}$ will be within the specified window.
- b) $\overline{\text{BDS}}$, once it has been deasserted, must remain high until a high $\overline{\text{DSACK}}$ has been seen on the bus.
- Note 11 a) The slave is guaranteed that $\overline{\text{BAS32}}$ will remain low until it drives either $\overline{\text{DSACK}}$ or $\overline{\text{BERR}}$ low (assuming it does this within the bus timeout period). The 350 nsec maximum time assures that $\overline{\text{BAS32}}$ will be high before the bus error timer generates an error.
- b) $\overline{\text{DSACK}}$ or $\overline{\text{BERR}}$ must be high from the previous cycle before the master can assert $\overline{\text{BDS}}$ for the next cycle.
- Note 14 a) This guarantees the master that $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ will not be released on all but the last transfer, until $\overline{\text{BDS}}$ has been deasserted.
- b) This guarantees the master that $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ will not be released on the last transfer, until the master has had a chance to look at the signal and drive $\overline{\text{BAS}}$ high.
- Note 16 a) The DIO II slave is guaranteed that $\overline{\text{RMC}}$, BFC2 -BFC0, and $\overline{\text{BLK}}$ will remain valid until $\overline{\text{BAS32}}$ or $\overline{\text{BDS}}$ goes high (whichever is last).
- b) The DIO II slave is guaranteed that BA1 - BA31, and $\overline{\text{LWORD}}$ will remain valid until the first deassertion of the data strobes. After the first transfer the slave is responsible for address calculations.
- Note 19 This guarantees that the entire bus has seen $\overline{\text{DSACK}}$ from the previous cycle, before the signal is reasserted for the next transfer.
- Note 20 This guarantees the slave that the minimum high time is seen on both data strobes.

Read Modify Write Cycles

The Read Modify Write cycle (defined only for DIO II) insures that the control of the data bus cannot be passed to another master until the RMW cycle is completed. This ensures that the status of a memory location can be checked and written back without interference. When RAM is dual ported, the complexity of RMW cycles increases. The RAM must be told not to allow the second port to access the location where the read modify write cycle is occurring. A way to do read modify write cycles that solve the problem of dual ported RAMs is defined for DIO II.

The only way in DIO to do RMW transfers is to do a read cycle, then a write cycle, without giving up the bus (deasserting $\overline{\text{BGACK}}$ - Bus Grant ACKnowledge). This does not solve the dual ported RAM problem. Therefore, a signal was defined on the bus called $\overline{\text{RMC}}$ which, when asserted, tells slaves that a RMW cycle is occurring. This line is set up the same as address on the read portion of the cycle, and remains asserted until the end of the write portion of the cycle. The other bus signals operate identical to the single read and single write cycles. Masters wanting to do RMW cycles in DIO will have to run two individual cycles without deasserting $\overline{\text{BGACK}}$. **IF** the master has a possibility of talking to a dual ported card, and cannot get control of both buses, it must tell the dual ported card not to arbitrate the DIO II port away.

The method of doing RMW cycles that solve the dual port problem is shown in Figure 4-12. The preferable method is to assert $\overline{\text{RMC}}$ and run two back to back cycles. The flow diagram for DIO II RMW cycle is shown in Figure 4-12.

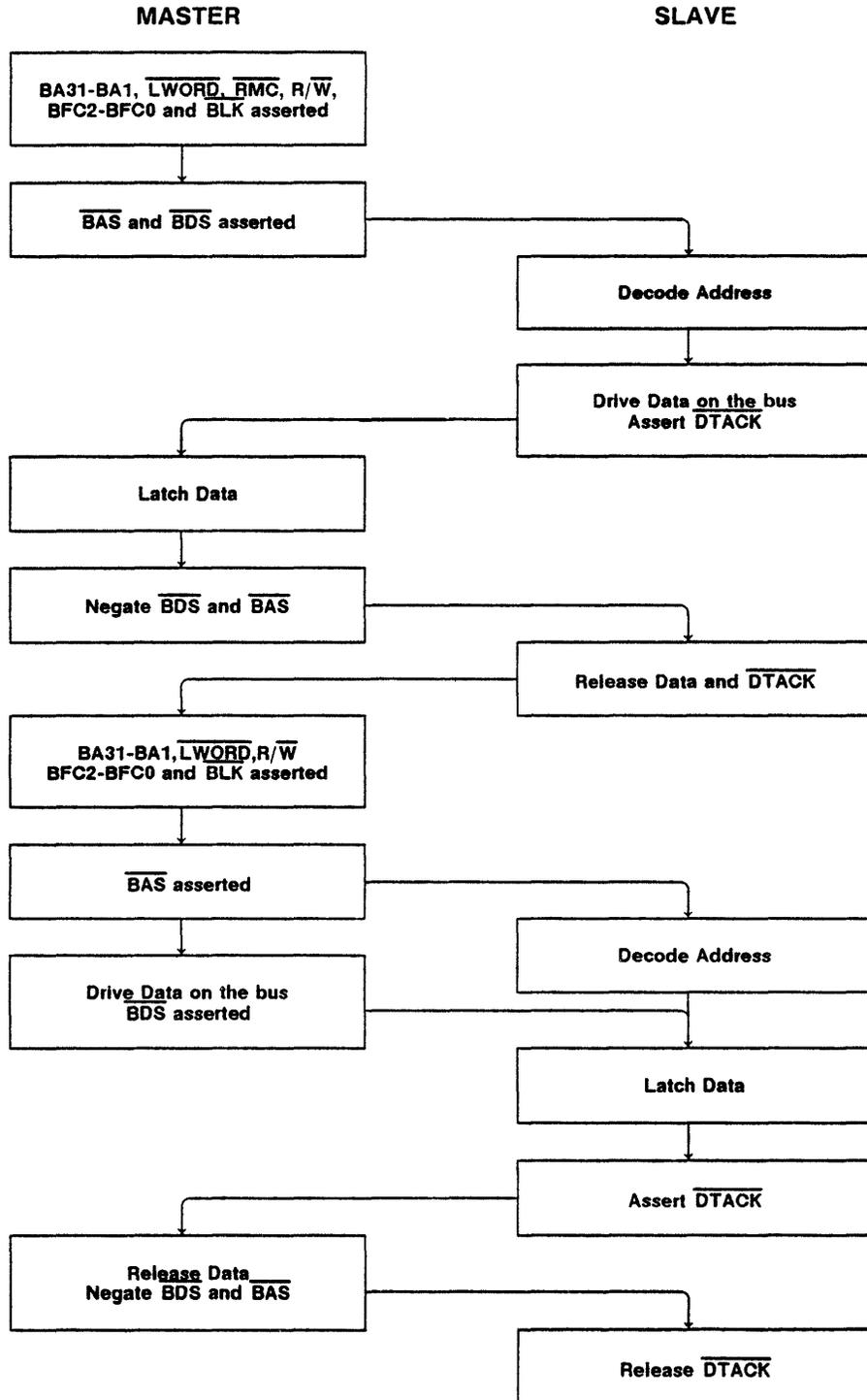


Figure 4-12. Read Modify Write

Timing for the RMW cycle is given in Figure 4-13.

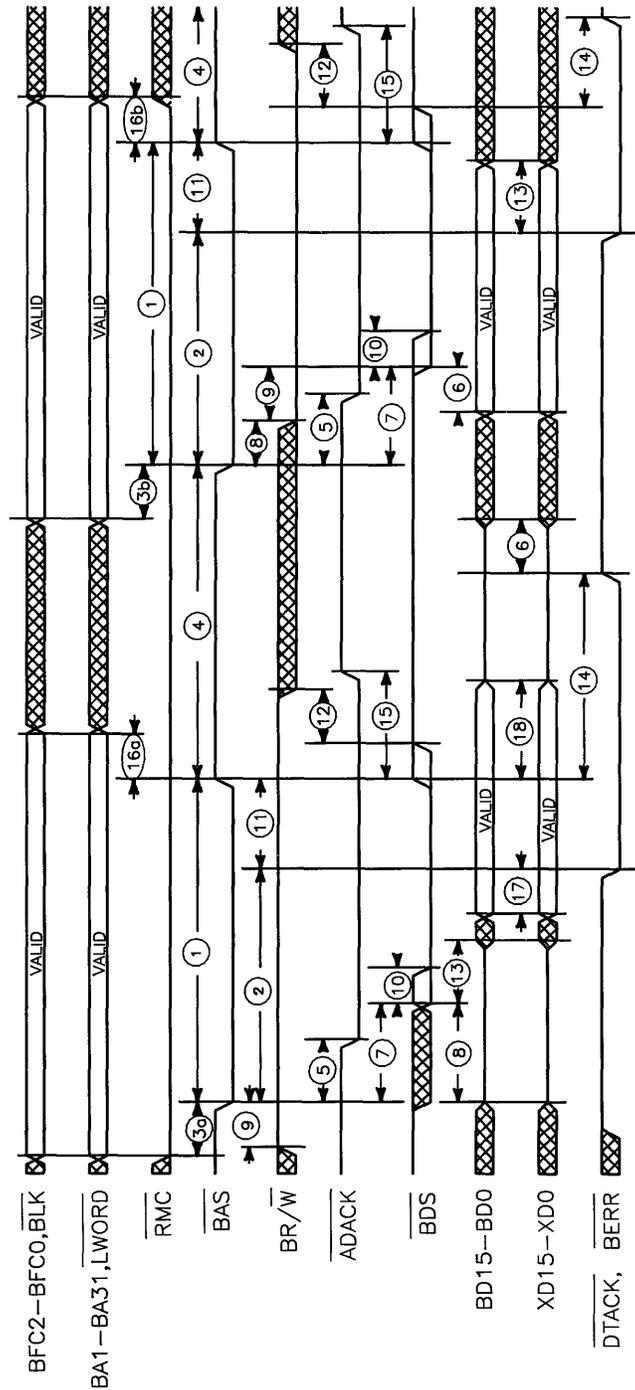


Figure 4-13. Read Modify Write Timing Diagram

Table 4-9. Read Modify Write Timing

NUM	TIMING PARAMETER	MASTER		SLAVE	
		MIN	MAX	MIN	MAX
	Timing parameters 1, 2, 4 - 15, 17 and 18 are identical to the Single Read or Write cycle and are not repeated here.				
3a	BA1 - BA31, $\overline{\text{LWORD}}$, $\overline{\text{RMC}}$, BFC0 - BFC2, and $\overline{\text{BLK}}$ set up time before $\overline{\text{BAS32}}$			15	
3b	BA1 - BA31, $\overline{\text{LWORD}}$, BFC2 - BFC0, and $\overline{\text{BLK}}$ set up time before $\overline{\text{BAS32}}$			15	
16a	BA1 - BA31, $\overline{\text{LWORD}}$, BFC2 - BFC0, $\overline{\text{BLK}}$ hold after $\overline{\text{BAS32}}$ or $\overline{\text{BDS}}$ deasserted			15	
16b	BA1 - BA31, $\overline{\text{LWORD}}$, $\overline{\text{RMC}}$, BFC2 - BFC0, $\overline{\text{BLK}}$ hold after $\overline{\text{BAS32}}$ or $\overline{\text{BDS}}$ deasserted			15	

Note 3 a) This guarantees the slave that $\overline{\text{RMC}}$, BA31 - BA1, BFC2 - BFC0, $\overline{\text{LWORD}}$, and $\overline{\text{BLK}}$ are valid for this amount of time before $\overline{\text{BAS32}}$. $\overline{\text{BLK}}$ cannot be asserted for RMW cycles, it must be high.

b) This guarantees the slave that BA31 - BA1, BFC2 - BFC0, $\overline{\text{LWORD}}$, and $\overline{\text{BLK}}$ are valid for this amount of time before $\overline{\text{BAS32}}$. $\overline{\text{RMC}}$ remained low between the read and write portions of the cycle.

Note 16 a) This guarantees the slave that BA31 - BA1, BFC2 - BFC0, $\overline{\text{LWORD}}$, and $\overline{\text{BLK}}$ are valid for this amount of time after $\overline{\text{BAS32}}$ or $\overline{\text{BDS}}$ goes high (whichever is last). $\overline{\text{RMC}}$ remains asserted and does not go high until the end of the write cycle.

b) This guarantees the slave that $\overline{\text{RMC}}$, BA31 - BA1, BFC2 - BFC0, $\overline{\text{LWORD}}$, and $\overline{\text{BLK}}$ are valid for this amount of time after $\overline{\text{BAS32}}$ or $\overline{\text{BDS}}$ goes high (whichever is last).

ENDT Overview

This section is added for historical reasons and may or may not be required.

ENable DTack ($\overline{\text{ENDT}}$) is defined for DIO cycles only and is not defined for DIO II cycles. It was used in Series 200 machines (and Series 300 Models 310 and 320) to improve the response time of certain DIO Bus devices such as DIO RAM cards. These devices use $\overline{\text{ENDT}}$ (which is basically $\overline{\text{BAS}}$ delayed by $1\frac{1}{2}$ clock cycles at a rate of 8Mhz) to generate an “early” $\overline{\text{DTACK}}$. For example, during a CPU read, $\overline{\text{ENDT}}$ is used to generate $\overline{\text{DTACK}}$ before the data is even on the bus. However, because the MC68000 accepts the data up to 90 nsec after $\overline{\text{DTACK}}$, the read cycle will be completed successfully (as long as the device provides the data within this time). It is necessary to observe the following when using $\overline{\text{ENDT}}$.

- Five cycle read/write accesses can be provided using $\overline{\text{ENDT}}$ with Series 200 machines (and Series 300 Models 310 and 320). However, during those memory cycles where the card is unable to accept or provide data within the required time (for example, during a RAM refresh cycle), then the card must inhibit use of $\overline{\text{ENDT}}$ to generate the early $\overline{\text{DTACK}}$.
- Cards that use $\overline{\text{ENDT}}$ must also be able to work without it. For example, the HP 98620A DMA Controller does not generate $\overline{\text{ENDT}}$. Also, new bus masters are not required to generate $\overline{\text{ENDT}}$.
- If a new DMA Controller is designed that generates $\overline{\text{ENDT}}$ during a DMA cycle, $\overline{\text{ENDT}}$ should be used by memory (not the I/O card) during the DMA cycle.
- On faster masters (Series 300 processors) $\overline{\text{DTACK}}$ is delayed so that the processor does not see it until data is valid.

In the timing discussions that follow, the timing margins appear very tight (if not unworkable). This is attributable to using worst-case timing specifications for all parameters. In actuality, sufficient margin exists to ensure that 5 cycle accesses are achievable. For example, the worst-case $\overline{\text{ENDT}}\text{-}\overline{\text{DTACK}}$ timing on the 09826-66522 board violates the timing specification. However, the production test for 5 cycle timing has never detected a failure of this board when attempting to achieve 5 cycle accesses (except, of course, during memory refresh cycles). The maximum time from $\overline{\text{ENDT}}$ -in to $\overline{\text{DTACK}}$ -out is calculated on the next page. This applies to both read and write cycles.

TIME	DESCRIPTION
125	Clock cycle period. $\overline{\text{ENDT}}$ is generated on the falling edge of the clock and $\overline{\text{DTACK}}$ must be valid prior to the next falling edge in order to achieve 5 cycle accesses.
-9	Maximum delay in the 74S74 flip-flop that generates $\overline{\text{ENDT}}$.
-30	Maximum delay in the 74LS245 buffer that drives $\overline{\text{ENDT}}$. This delay is less than the standard 47.5 nsec delay since the loading on $\overline{\text{ENDT}}$ is typically less than 500 pf (call this engineering judgement).
-12	Delay of the Processor's $\overline{\text{DTACK}}$ receiver (LS125). The typical specification (not the maximum) is used here since the loading on the signal is much less than the 45 pf load that the part is characterized as having (more engineering judgement).
-20	Maximum $\overline{\text{DTACK}}$ setup time prior to the trailing edge of the clock.
54	This represents the bus slave's requirement for generating $\overline{\text{DTACK}}$ from the $\overline{\text{ENDT}}$ input. Because the $\overline{\text{DTACK}}$ bus driver requires a maximum of 47.5 nsec, approximately 6.5 nsec remain for internal logic delay. However, because all of these timing specifications are worst-case and thus statistically unlikely to occur together, more delay than 6.5 nsec is allowed. Specifying the permissible time delay is difficult. The guideline is to keep the internal delay as small as possible. The penalty for larger time delays is 6 cycle accesses in some systems, not the desired 5 cycle accesses.

Read Cycle Using ENDT

The following read cycle timing using $\overline{\text{ENDT}}$ is based on Series 200 bus masters. As discussed above, devices that use $\overline{\text{ENDT}}$ to generate $\overline{\text{DTACK}}$ must provide data within a certain time after $\overline{\text{DTACK}}$. Because bus slaves begin memory accesses using $\overline{\text{BAS}}$, the time from $\overline{\text{BAS}}$ going low to data valid on the bus is calculated. The method of calculating this timing is explained so that users can apply $\overline{\text{ENDT}}$ to their application.

TIME	DESCRIPTION
437.5	Represents 3.5 clock cycles. $\overline{\text{BAS}}$ starts with the rising edge of the CPU clock; data must be valid just before the clock's falling edge, 3.5 cycles later.
-60	Maximum delay from rising edge of clock to $\overline{\text{AS}}$ (Address Strobe).
-47.5	Maximum delay of $\overline{\text{BAS}}$ bus driver.
-12	Maximum data delay on CPU board due to bus receivers.
-15	CPU data setup time before trailing edge of clock
303	Thus, the bus slave must have data valid within 303 nsec of $\overline{\text{BAS}}$ going low, if $\overline{\text{ENDT}}$ is used to generate $\overline{\text{DTACK}}$ as discussed above. Because the bus slave's data bus drivers can require up to 39.5 nsec, internal delays in accessing data must be less than $303 - 39.5 = 263.5$ nsec.

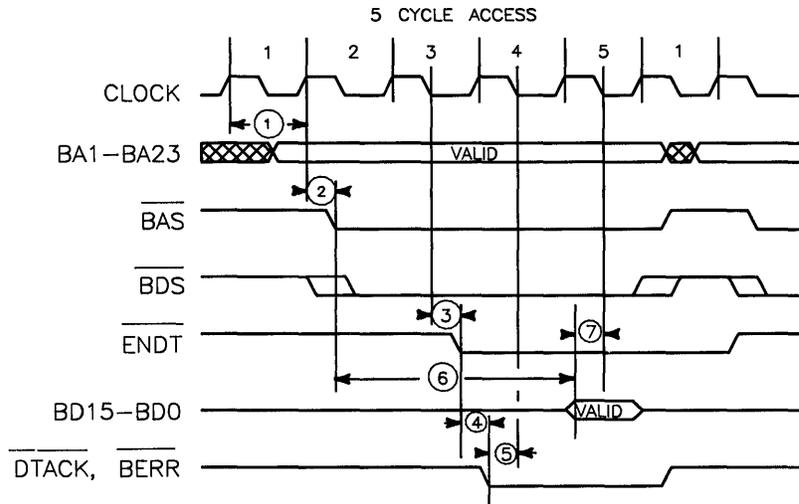


Figure 4-14. $\overline{\text{ENDT}}$ Read Timing Diagram

Table 4-10. $\overline{\text{ENDT}}$ READ Timing

NUM	TIMING PARAMETER	DIO		NOTE
		MIN	MAX	
1	Clock period @ 8 Mhz	125 nominal		1
2	$\overline{\text{BAS}}$ delay from clock high		107.5	
3	$\overline{\text{ENDT}}$ delay from clock low		39	
4	$\overline{\text{ENDT}}$ low to $\overline{\text{DTACK}}$ low		54	
5	$\overline{\text{DTACK}}$ low setup before clock low	32		2
6	$\overline{\text{BAS}}$ low to read data valid		303	
7	Data setup before clock	27		

Note 1 The clock is on the bus master and does not appear on the DIO Bus. It is shown for **Reference Only**.

Note 2 As it should, the times specified for numbers 3 + 4 + 5 = 125 nsec.

Note 3 These times assume $\overline{\text{ENDT}}$ loading is less than the 500 pf assumed for the data bus, address bus, etc.

Write Cycle Using ENDT

During write cycles, the time from $\overline{\text{BAS}}$ going low to data becoming invalid (indicating completion of the cycle) is of interest. Also, the time from the data strobes BLDS and BUDS going true (indicating valid data on the bus) to the data becoming invalid is also of interest. These times are calculated below:

MINIMUM $\overline{\text{BAS}}$ LOW TO DATA INVALID TIME

TIME (nsec) DESCRIPTION

437.5	3.5 clock cycles
-60	Maximum delay from rising edge of clock to $\overline{\text{AS}}$ (Address Strobe).
-47.5	Maximum delay of $\overline{\text{BAS}}$ bus driver.
-15	Maximum skew between $\overline{\text{BAS}}$ and data
0	Minimum delay of data bus drivers. Any delay extends the time from $\overline{\text{BAS}}$ low to data invalid. However, to continue with the worst-case analysis, 0 must be used.
+30	Minimum data hold after $\overline{\text{BAS}}$ goes high. Because the data bus buffers continue to drive the bus after $\overline{\text{BAS}}$ and $\overline{\text{BDS}}$ go high, this hold time occurs on the bus.

345

MINIMUM BDS LOW TO DATA INVALID TIME

220	The data strobes are delayed from $\overline{\text{BAS}}$ by one clock cycle (125 nsec). Therefore, the time from the data strobes going low to data invalid is $345 - 125 = 220$ nsec.
-----	---

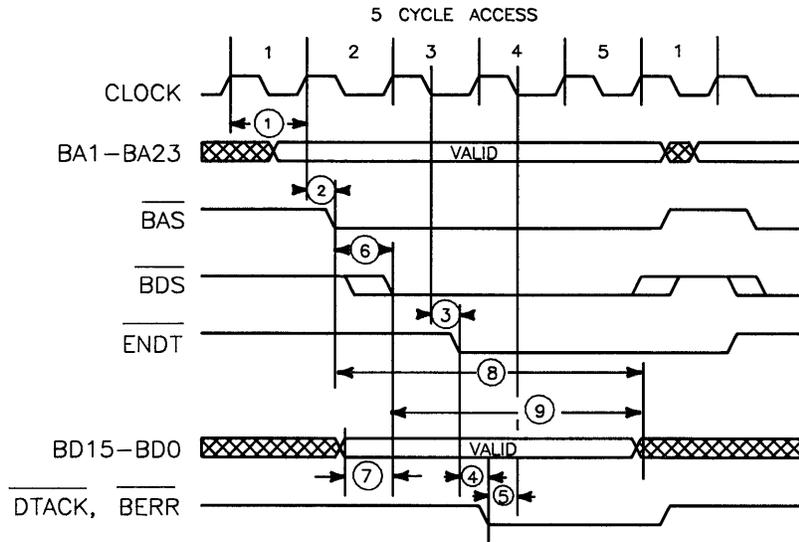


Figure 4-15. $\overline{\text{ENDT}}$ Write Timing Diagram

Table 4-11. $\overline{\text{ENDT}}$ Write Timing Table

NUM	TIMING PARAMETER	DIO		NOTE
		MIN	MAX	
1	Clock period @ 8 Mhz	125 nominal		1
2	$\overline{\text{BAS}}$ delay from clock high		107.5	
3	$\overline{\text{ENDT}}$ delay from clock low		39	
4	$\overline{\text{ENDT}}$ low to $\overline{\text{DTACK}}$ low		54	
5	$\overline{\text{DTACK}}$ low setup before clock low	32		
6	$\overline{\text{BAS}}$ low to $\overline{\text{BDS}}$ low (nominal)		125	
7	Data setup before $\overline{\text{BDS}}$ low	15		
8	$\overline{\text{BAS}}$ low to data invalid	345		
9	$\overline{\text{BDS}}$ low to data invalid	220		

Note 1 The clock is on the bus master and does not appear on the DIO Bus. It is shown for **Reference Only**.

Note 2 These times assume $\overline{\text{ENDT}}$ loading is less than the 500 pf assumed for the data bus, address bus, etc.

Bus Error

Introduction

This chapter discusses the Bus Error (BERR) signal.

When the CPU's bus error input signal ($\overline{\text{BERR}}$) is asserted, an exception sequence is generated. This signal (BERR) is put on the bus by an open collector gate, permitting it to be generated by any device including the processor board. Applications of the bus error signal are device dependent. Therefore, several applications that DIO Bus designers should be aware of are discussed in this chapter.

Bus Error Signals

Table 5-1. Bus Error Signals Table

SIGNAL	DEFINITION
$\overline{\text{BERR}}$	Bus ERRor, when asserted, causes the processor to terminate the bus cycle. When BERR is negated, the processor begins its exception processing.

Bus Error Timing

BERR timing is different for MC68000, MC68010, and MC68020 processors. The MC68000 BERR input can be asynchronous if the BERR signal arrives before DTACK. As was mentioned previously, the MC68010 permits BERR to arrive after DTACK, but the BERR signal must meet a setup time before the CPU clock. This setup time is critical to proper operation, and the MC68010 may exhibit erratic behavior if it is violated. Because BERR timing varies for different mainframes and different processor boards, timing diagrams are not included in this document. The reader is referred to the section *Guidelines for Utilizing BERR* in this chapter.

Bus Timeout

Series 200 and Series 300 processor boards generate the bus error signal, BERR, when an accessed card fails to respond within a certain time. As discussed in previous chapters, a device responds with DTACK, so logically the time from BAS going true to the arrival of DTACK would be monitored to determine an error condition. However, DTACK causes BAS to go false (with some delay) so the BERR circuit simply monitors the length of BAS. If DTACK occurs too late or fails to occur, BAS will remain true and a counter will time out.

A 4-bit BERR counter is located on Series 200 processor boards and is cleared when BAS is false. When BAS is true, it begins counting (4 Mhz clock) and generates a BERR when it overflows (16 counts x 250 nsec = 4.0 usec). The Series 300 processors provide a similar counter, but the timeout is extended to 5.6 usecs. Since many designers are planning to support series 200 devices the timing is discussed below. Series 300 will not be discussed in detail, but it can be derived in a similar manner as the Series 200.

To provide margin in the Series 200, the maximum width of BAS is specified at 3.5 usec. To ensure that DTACK has time to reset BAS prior to 3.5 usec, DTACK must arrive 2 CPU clocks (250 nsec, synchronizing time) earlier. Because of this time, other delays, and to provide ample margin, the maximum time from BAS going low to DTACK going low has been specified at 3.0 usec.

From the above discussion, it appears that devices have 3.0 usec from BAS to DTACK. However, for devices that implement DMA, it is not that simple. For example, during a DMA output cycle, RAM must be read **and** the I/O card written to within one BAS cycle. Thus, the combined RAM read time and I/O card write time must be less than 3.0 usec.

For a DMA input (I/O read, memory write), DMARDY must be true (indicating valid I/O card data) within 2.5 usec of DMACK. As usual, DTACK (indicating the memory card has accepted the data) must be true within 3.0 usec of BAS going true to prevent a BERR timeout.

For a DMA output (memory read, I/O write), memory is ready within 1.5 usec as evidenced by DTACK. The I/O card has another 1.5 usec to accept the data, as evidenced by DMARDY.

Because the master controller implements the bus timeout function, other bus masters do not have to generate it. However, bus masters **do have to respond to the bus timeout** if the master controller generates it while another bus master has control of the bus. Acceptable responses are: (1) handle the bus timeout problem or (2) give up the bus to the master controller and let it resolve the problem. Refer to the Series 200 DIO manual, *HP 9000 Series 200/300 Computer Accessory Development Guide*, for more details.

Auto-Locate of Processor RAM

The BERR signal generated by the BAS timeout counter is used to auto-locate RAM on HP 9826/36 computers and on the Model 310 processor boards. The processor RAM auto-locates itself at the bottom (lower addresses) of RAM memory, eliminating the need for the user to set switches on an internal board. This works as follows: at power-on, the processor writes to and reads from each block of memory to determine how much memory is available. When there is no response from a memory card (indicating that the access is below all plug-in memory), the BERR counter times out and generates BERR. A latch then latches the bus-error address as the processor board's RAM address. Auto-location is a Series 200 implementation detail, not a bus master requirement.

BERR for Page Faulting

Processor boards that have MMU's generate BERR upon detection of a page fault. DTACK is not generated so that a normal BERR trap occurs.

Guidelines for Utilizing BERR

This section gives guidelines for using BERR in new designs. It should be noted that presently (January 1987) BERR is generated only by processor boards. The only card which uses BERR (in addition to the processor boards themselves) is the HP 98620 DMA Controller.

Because of the critical nature of BERR timing and the differences in operation with different mainframes and processor boards, the general guideline is: $\overline{\text{BERR}}$ is generated on the processor card, and can be recognized by bus slaves. It should not be originated by I/O cards and other accessories.

Prior to using the $\overline{\text{BERR}}$ signal in any design, the following information should be studied:

1. BERR specifications for the different MC680xx Processors.
2. Hardware and software documentation on BERR operation and limitations in current mainframes. This information is usually available in the Theory of Operation for specific boards.

Even though a processor board or another board may generate BERR, a bus slave may not “see” it. For example, BERR is an **input only** signal for the HP 9888 Bus Expander. A processor generated BERR is not seen by cards in the expander. Therefore, cards may utilize BERR if it is present, but they must not depend on it being available unless the stipulation is made that the card **must** reside in the mainframe. Such a stipulation is acceptable for a one-per-system type of card (for example: the HP 98620 DMA Controller) but is not acceptable for generic I/O cards.

If a bus slave does respond to BERR, it should terminate all DIO Bus transactions (that is, “get off of the bus”).

Interrupt Operation

Introduction

This chapter discusses the DIO II interrupt signals, their timing, and the cycles they use.

The DIO II Bus supports seven interrupt levels and 2 methods of responding to interrupts: (1) external vectored and (2) autovectored. External vectoring requires the interrupting device to assert the VECTOR32 signal then put an 8-bit vector on the bus and assert DTACK. With autovectored, the interrupting device does not provide a vector and the processor generates its own default vector.

Note

Previous documentation describing external vectoring on the Series 200/300 computers does not apply to the DIO II bus. Existing processor boards do not generate several of the control signals described in the *Designers Guide To The 9826A Cardcage*. Also, not all processors support external vectored interrupts as described in the DIO Bus manual, *HP 9000 Series 200/300 Computers Accessory Development Guide*.

I/O card designers should not design cards that implement externally vectored interrupts. Only autovectored interrupts should be used. Externally vectored interrupts on DIO II are defined in this document so that transparent adapters to other I/O buses are possible. While external vectored interrupts should not be used on I/O cards or other products, they may be used in special, one-of-a-kind projects as long as the designer accepts the risks involved.

Interrupt Signals

The interrupt signals on the DIO II Bus are shown below. Interrupts 3 thru 6 are for external I/O cards. Interrupts 1, 2 and 7 are for internal I/O. The assignment of these interrupt levels are shown below as an example.

Table 6-1. Interrupt Signal Table

SIGNAL	DEFINITION	
$\overline{IR1}$	Interrupt 1 - Keyboard/real time clock	LOWEST LEVEL
$\overline{IR2}$	Interrupt 2 - 9826/36 internal floppy controller	-
$\overline{IR3}$	Interrupt 3 - External I/O	-
$\overline{IR4}$	Interrupt 4 - External I/O	-
$\overline{IR5}$	Interrupt 5 - External I/O	-
$\overline{IR6}$	Interrupt 6 - External I/O	-
$\overline{IR7}$	Interrupt 7 - Reset key, powerfail	HIGHEST LEVEL
\overline{IACK}	Interrupt ACKnowledge - output from a DIO Bus Master, not supported on Series 300 Models 330 and 350 CPU's.	
$\overline{IACK32}$	Interrupt ACKnowledge 32 - output from the Bus Master and is supported by DIO II processors.	
\overline{VECTOR}	Output of interrupting device if it has an interrupt vector to put on BD0-BD7. This signal is not supported on Series 300 CPU's	
$\overline{VECTOR32}$	Output of interrupting devices if it has an interrupt vector to put on BD0-BD7 or XD0-XD7 depending on which DSACK is asserted. This signal is supported by DIO II processors.	

Note

There is no $\overline{IR0}$ signal. $\overline{IR0}$ is the quiescent (non-interrupting) state.

For MC68000 based bus masters, logic is needed to encode the interrupt signals into the 3 processor inputs, IPL0, IPL1, and IPL2. For example: Series 200 bus masters use a 74LS148 8-to-3 priority encoder, to encode INT1 - INT7 and to generate IPL0, IPL1 and IPL2.

The following interrupt levels are the only I/O interrupt levels which have been “hardwired”. All plug-in DIO I/O cards and DIO II I/O cards have a 2-bit switch to select interrupt levels 3 thru 6.

Internal Use	Level 1
Internal Use	Level 2
HP 98620A DMA Controller	Level 3
HP 98620B DMA Controller	Programmable from 3 thru 6
1TQ4-0401 DMA Chip	Programmable from 3 thru 6
Internal HP-IB (HP 9816/26/36)	Level 3
Internal RS-232	Level 4
6840 timer	Level 6
Internal Use	Level 7

Even though IR1, IR2 and IR7 are not currently used for external I/O, the signals have been put on the backplane for expandability and compatibility with future products. Any cards which use these interrupt levels should be designed to respond to the internal I/O memory space since the operating system protocol for these levels is designed around a known set of internal peripherals and is different than the protocol for the external I/O memory space.

Level 1 and 7 interrupts are driven by open collector gates in the HP 9826/36. However, level 2 is used by the 09826-66561 and 09826-66562 floppy controller boards, and is driven by a standard LS-TTL gate. Thus it cannot be used by other devices that you may wish to interface with the HP 9826/36.

External Vectored Interrupt Cycle

If the priority of the pending interrupt is greater than the current processor priority, the following interrupt sequence begins. The bus master attempts to read an 8-bit interrupt vector from the interrupting I/O card. This operation is the same as a normal read, except $\overline{\text{IACK32}}$ is used in lieu of either of the $\overline{\text{BAS}}$ signals. BAS remains false. Interrupt timing is shown on the following page (Figure 6-1).

Prior to the beginning of the cycle, the lowest three address bits (BA1-BA3) are set equal to the interrupt level being acknowledged. Likewise, the MC68000 Function Code bits FC0 , FC1 and FC2 are set to 1. The bus master decodes the function code bit and gates with Address Strobe, and generates $\overline{\text{IACK32}}$.

When FC0 , FC1 and FC2 are 1, the bus drivers for $\overline{\text{BLDS}}$, $\overline{\text{ENDT}}$, $\overline{\text{BUDS}}$, $\overline{\text{BR/W}}$, $\overline{\text{BAS}}$, BFC0 , BFC1 and BFC2 on existing CPU boards should be considered undefined.

In response to $\overline{\text{IACK32}}$, each interrupting card detects: (1) if it is interrupting and (2) if its interrupt level is being acknowledged. The card does this by looking at BA1-BA3 . If both of these conditions are occurring, the card asserts the $\overline{\text{VECTOR32}}$ signal within 100 nsec after $\overline{\text{IACK32}}$. This indicates to the processor that it has an 8-bit interrupt vector to put on the lower byte of the data bus. If the interrupting card does not respond within 100 nsec, the bus master circuitry will initiate an autovector interrupt response. Autovector is discussed in the next section.

If the $\overline{\text{VECTOR32}}$ signal is asserted within 100 nsec of $\overline{\text{IACK32}}$, reading of the 8-bit vector occurs. Because the data strobes are disabled, the interrupting card provides its vector in response to $\overline{\text{IACK32}}$, as described below:

1. After detecting $\overline{\text{IACK32}}$, the interrupting card drives its vector on BD0-BD7 (or XD0-XD7 depending on the size of the port). Data is set up a minimum of 15 nsec prior to assertion of $\overline{\text{DTACK}}$ (as measured at the receiver).
2. The CPU detects that $\overline{\text{DTACK}}$ has occurred, accepts the vector, and ends the interrupt acknowledge by setting $\overline{\text{IACK32}}$ false.
3. The interrupting card then releases $\overline{\text{VECTOR32}}$ signal within 50 nsec after $\overline{\text{IACK32}}$ goes false and releases the drive on the data bus within 100 nsec after $\overline{\text{IACK32}}$ is false.

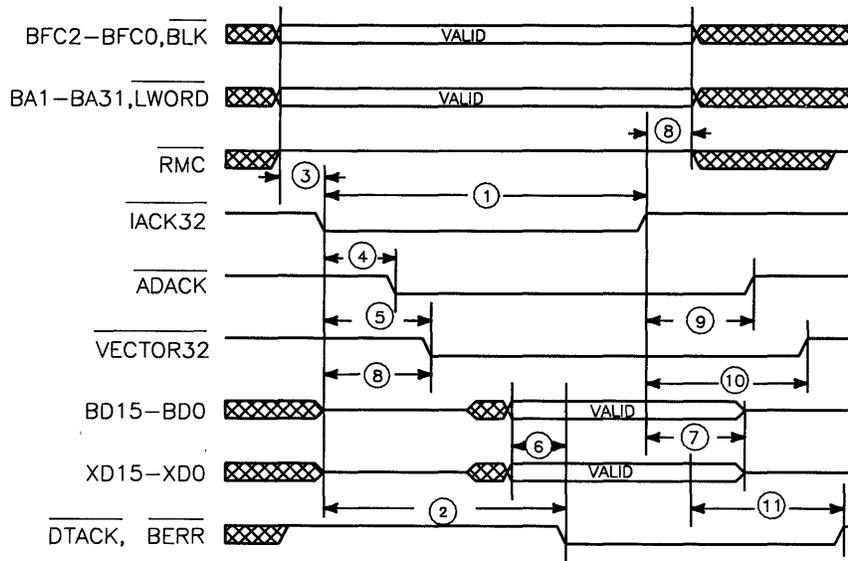


Figure 6-1. Interrupt Timing Diagram

Table 6-2. Interrupt Timing Table

NUM	TIMING PARAMETER	MASTER		SLAVE	
		MIN	MAX	MIN	MAX
1	$\overline{\text{IACK32}}$ low				5100
2	$\overline{\text{IACK32}}$ low to $\overline{\text{DTACK}}$ low w/o bus error		4600		
3	Address setup before $\overline{\text{IACK32}}$ low			15	
4	$\overline{\text{IACK32}}$ low to $\overline{\text{ADACK}}$ low	0	70		
5	$\overline{\text{IACK32}}$ low to $\overline{\text{VECTOR32}}$ low	0	100		
6	Data setup before $\overline{\text{DTACK}}$ low	15			
7	Data hold after $\overline{\text{IACK32}}$ high	0	100		
8	Address hold after $\overline{\text{IACK32}}$ high			15	
9	$\overline{\text{IACK32}}$ high to $\overline{\text{ADACK}}$ high	0	50		
10	$\overline{\text{IACK32}}$ high to $\overline{\text{VECTOR32}}$ high	0	50		
11	$\overline{\text{IACK32}}$ high to $\overline{\text{DTACK}}$ high	0	50		

External vectoring has not been implemented on any I/O card to date (1-87). Thus its operation has not been tested. If external vectoring is implemented, extensive qualification will be required to verify that it works with the different mainframes and processor boards. Also, designers should be aware that only **one** vectoring card per interrupt level is allowed unless a hardware arbitration scheme is included. Multiple autovectoring I/O cards may reside on the same level as a single vectoring card since the vectoring card will override autovectoring cards.

Refer to Chapter 12, *Operation In The 9888A Bus Expander* for problems relative to vectored interrupt operation in the bus expander.

Autovectorred Interrupt Cycle

If an interrupting card does not generate $\overline{\text{VECTOR32}}$, the bus master automatically generates its own vector as follows:

Note

Steps 1-3 of the *External Vectored Interrupt Cycle* are attempted, as the bus master does not “know” yet that it is performing an autovector cycle.

1. With autovectored, the interrupting card does not assert $\overline{\text{VECTOR32}}$ in response to $\overline{\text{IACK32}}$, nor does it provide the vector itself. Logic on the processor board detects that $\overline{\text{VECTOR32}}$ remains high and, within 100 nsec of $\overline{\text{IACK32}}$, asserts AVEC (AutoVECTor), an input to the MC68020.
2. The processor recognizes that, if AVEC is asserted during the interrupt acknowledge cycle, it is an autovector cycle. The processor uses an internally generated vector that is a function of the interrupt level being serviced. The 7 interrupt vectors, corresponding to interrupt levels 1 thru 7, are 25 through 31 (Decimal). These are used to access 4-byte addresses from 100 - 124 which vector operation to RAM. Refer to the MC68000 Data Sheet for more information.
3. If more than one I/O card is on the same interrupt level, then it is not possible to tell which card is interrupting. Thus, a machine level service poll routine must be used to determine which card to service. The two most significant bits of the status register contain interrupt information. The most significant bit indicates if the card is enabled to interrupt. The next most significant bit indicates if the card is in an interrupt state.

Bus Arbitration

The DIO II Bus design permits other bus masters to acquire the DIO Bus and perform memory read/write operations. To support multiple bus masters (up to 7 in addition to the master controller), a bus arbitration scheme is used.

Note

Designers considering designing bus master I/O cards should ensure that they have a thorough understanding of the system implications of such plans. For example, the HP 9000 Model 350 computer has a separate system bus and I/O bus. Therefore, the system bus is not controllable from the I/O bus.

Bus Arbitration Signals

The signals used to arbitrate control of the bus are shown in Table 7-1 on the following page:

Table 7-1. Bus Arbitration Signals

SIGNALS	DEFINITION
$\overline{\text{BR}}$	Bus Request. Asserted by the device(s) requesting control of the bus. It is wire-ORed among all bus masters.
$\overline{\text{XBG3}}$	Extended Bus Grant 3. This is output from a master controller only. It goes to the bus grant input of bus master A.
$\overline{\text{XBG2}}$	Extended Bus Grant 2. It may be a master controller output (if $\overline{\text{XBG3}}$ is not used), or it is a bus grant output from bus master A. It is connected to the bus grant input of bus master B.
$\overline{\text{XBG1}}$	Extended Bus Grant 1. It may be a master controller output (if $\overline{\text{XBG3}}$ and $\overline{\text{XBG2}}$ are not used), or it is a bus grant output from bus master B. It is connected to the bus grant input of bus master C.
$\overline{\text{BG}}$	Bus Grant. It may be a master controller output (if $\overline{\text{XBG3}}$, $\overline{\text{XBG2}}$ and $\overline{\text{XBG1}}$ are not used), or it is a bus grant output from bus master C. It is connected to the bus grant input of bus master D. This is the default shipping condition for all master controllers currently being produced as of January 1987.
$\overline{\text{BG1}}$	Bus Grant 1. Bus grant output from bus master D. It is connected to the bus grant input of bus master E.
$\overline{\text{BG2}}$	Bus Grant 2. Bus grant output from bus master E. It is connected to the bus grant input of bus master F.
$\overline{\text{BG3}}$	Bus Grant 3. Bus grant output from bus master F. It is connected to the bus grant input of bus master G. Note that bus master G's bus grant output is not connected to another device. It should also be noted that $\overline{\text{BG3}}$ is not defined for DIO II. It is not a valid pin on the connector. The only way it can be used is if bus master F and G are in the DIO I/O or option slots and not in the DIO II system slots. Refer to the pinouts of the DIO II and DIO connectors in Chapter 11.
$\overline{\text{BGACK}}$	Bus Grant ACKnowledge. A tri-state signal generated by the bus master accepting the bus, to verify that the bus is now controlled by the new bus master.

Bus Arbitration Overview

The bus arbitration scheme permits other bus masters to request and receive control of the DIO Bus. The bus grant signals are daisy chained from the master controller to each bus master. While it may appear that the daisy chain supports a priority scheme, in actuality it does not. It is basically a first-come first-serve arrangement. The current bus master (but not the master controller) can keep the bus indefinitely regardless of other devices which want the bus. Current master controllers, however, give up the bus immediately whenever any other bus master requests it.

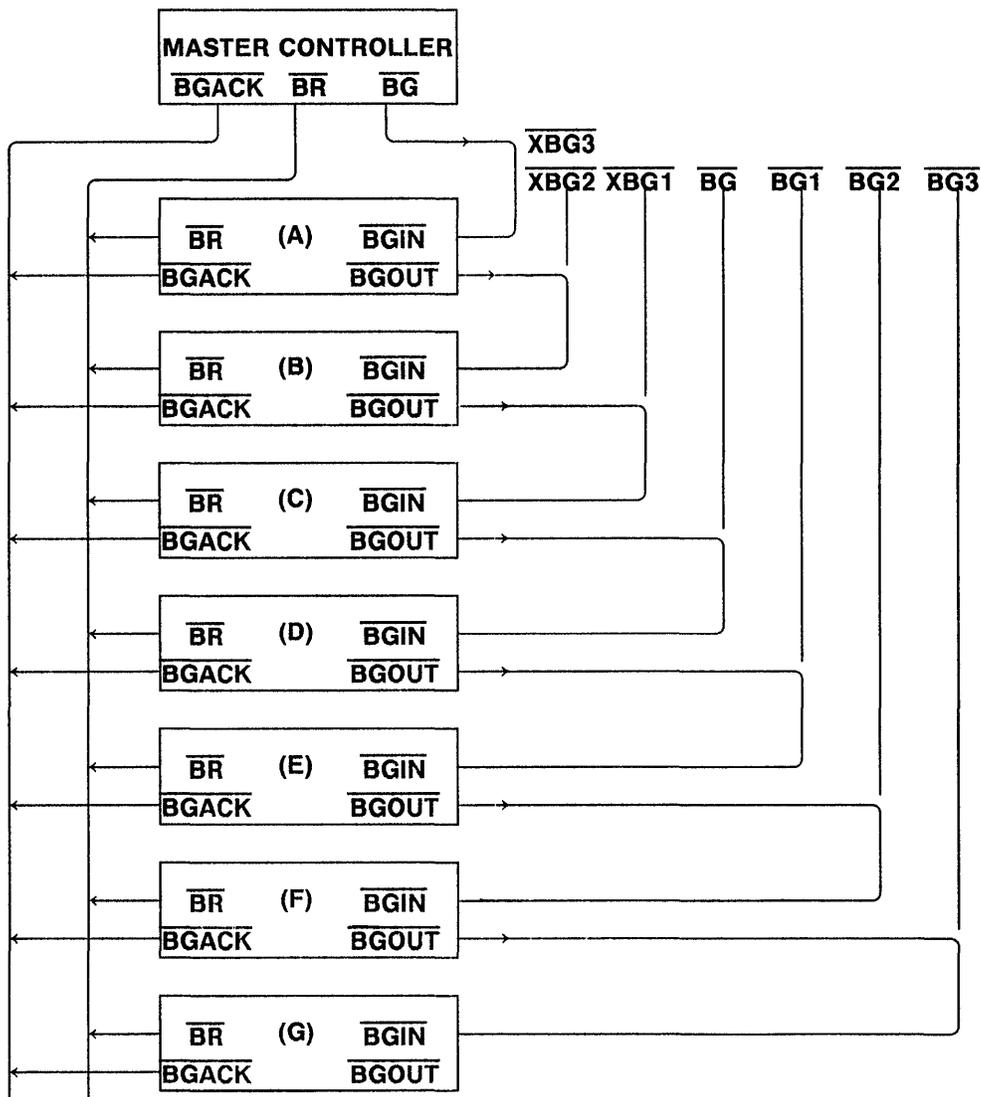
Arbitration is handled via the bus grant signals (XBG3, XBG2, XBG1, BG, BG1, BG2 and BG3) which are daisy-chained from the master controller to bus masters A, B, C, D, E, F, G (see Figure 7-1 on the following page). Connections to the desired bus grant signals are made by jumpers on each bus master. Alternatively, switches could be used. Thus, the daisy chain is NOT hardwired on the backplane itself but requires the user to configure it.

The master controller's \overline{BG} signal controls arbitration. When this signal is high (false), the \overline{BGIN} signal of all bus masters is high. Each bus master uses this high signal to enable a latch which samples the on-board "My Request" signal. When \overline{BG} goes low (in response to BR from a bus master seeking control of the bus), the first bus master in the daisy chain that has latched a valid "My Request" signal will maintain a high \overline{BGOUT} output to inhibit "downstream" bus masters from assuming control. When BGACK (it may have been true if another bus master had control of the bus) and the bus control lines are false, then the bus master which is asserting a high \overline{BGOUT} will assume control of the bus as indicated by BGACK true.

The bus master that assumes control asserts BGACK to inform the master controller that it has the bus. In response to BGACK, the master controller sets BG false, which re-initiates sampling by each bus master of the on-board "My Request" signal in preparation for the next arbitration. If BR is true, the master controller re-asserts BG to re-arbitrate while the current bus master has control. The master controller will assume control of the bus when BGACK goes false if BR is also false.

Note

A bus controller in a DIO II machine that supports DIO masters should only grant the bus between cycles. DIO masters cannot see $\overline{BAS32}$ to know when the bus is idle.



NOTE: The $\overline{\text{BR}}$ output is open-collector. The $\overline{\text{BGACK}}$ output is tri-state.

Figure 7-1. Configuration of Bus Arbitration Signals

Bus Arbitration Sequence

The bus arbitration sequence is discussed below:

1. A bus master desiring the bus asserts BR to the master controller. The master controller eventually responds with BG (response time is 0 nsec to infinity).
2. The assertion of BG by the master controller begins the arbitration among the bus masters. If BG reaches a device that is not requesting the bus, it is simply passed along (from BGIN to BGOUT). When BG reaches a device that is requesting the bus, the signal is blocked from going further.
3. The device that is requesting the bus then waits for bus availability as indicated by $\overline{\text{BAS32}}$, $\overline{\text{BAS24}}$, $\overline{\text{DTACK16}}$, $\overline{\text{DSACK32}}$, $\overline{\text{DSACK16}}$, and $\overline{\text{BGACK}}$ going false while BG remains true (again, the response time can be infinity). When these conditions are met, the new bus master must then wait a minimum of 100 nsec to allow the old bus master to tri-state its signals before asserting BGACK.

Note

Some processor boards enable their drivers in less than 100 nsec after regaining control of the bus. Therefore, there will be bus contention while the former bus master tri-states its drivers.

4. After asserting BGACK, the device then releases BR within 100 nsec. BR must be removed within 100 nsec after assertion of BGACK so that the master controller does not decide that another bus master is also requesting the bus.
5. When BGACK is asserted, the master controller negates BG which ripples through the arbitration daisy-chain and prepares the arbitration logic for the next arbitration decision.
6. When the current bus master is finished, it drives the control signals BAS32, BAS24, DTACK16, DSACK32, DSACK16, and BGACK false prior to tri-stating the bus drivers. The bus master must tri-state its drivers within 100 nsec of negating BGACK so the next bus master can take over the bus. However, certain CPU boards, when regaining control of the bus, drive the address bus immediately upon negation of BGACK. Therefore, the device relinquishing the bus should tri-state its address bus simultaneously with negating BGACK.

Bus handoff timing and an example of a bus handoff involving multiple bus masters, is shown on the following pages.

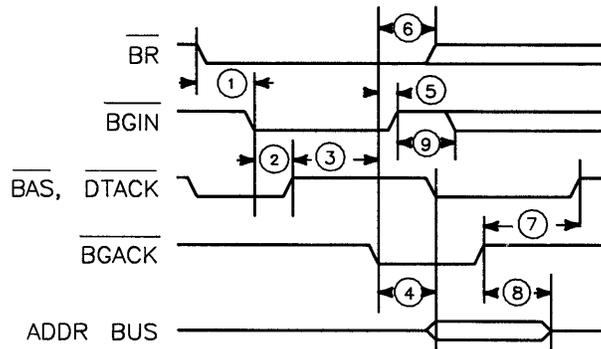


Figure 7-2. Bus Arbitration Timing Diagram

Table 7-2. Bus Arbitration Timing Table

NUM	TIMING PARAMETER	MASTER CONTROLLER		REQUESTER	
		MIN	MAX	MIN	MAX
1	\overline{BR} low to \overline{BG} low			0	infinity
2	\overline{BG} low to \overline{BGACK} , \overline{BAS} and \overline{DTACK} high (bus available)			0	infinity
3	Bus available to \overline{BGACK} low	100			
4	\overline{BGACK} low to bus signals driven	0			
5	\overline{BGACK} low to \overline{BG} high			190	465
6	\overline{BGACK} low to \overline{BR} release	0	100		
7	\overline{BGACK} high to bus signals tri-state except for address bus	0	100		
8	\overline{BGACK} high to address bus tri-state		0		
9	\overline{BG} high (arbitration reset time)			187	

- Note 1 The time from which the requesting devices asserts a request to the time it receives a grant may be infinitely long. There is no way to guarantee a maximum time with the daisy chain architecture.
- Note 2 The device receiving the bus grant must wait for both BAS's, all three DTACK's, and BGACK to be deasserted before the bus is available. Since old DIO masters cannot look at $\overline{\text{BAS32}}$ or either $\overline{\text{DSACK}}$, new master controllers may not want to do overlapped arbitration. If arbitration is used that does not overlap, both BAS's and all three DTACK's will always be high when BG is asserted.
- Note 3 The requesting device must wait this minimum amount of time before driving the bus after seeing it idle. This is to prevent contention on the bus and to guarantee that the master controller sees the change in ownership of the bus.
- Note 4 This is the minimum time from acknowledging the bus grant to driving the bus with valid address, data, and control signals.
- Note 5 $\overline{\text{BG}}$ from the master controller **MUST** be driven false within the time specified. There must be enough time for the bus grant to ripple through the arbitration daisy chain.
- Note 6 $\overline{\text{BR}}$ from the requesting device **MUST** release $\overline{\text{BR}}$ within this maximum time so that the master controller does not see an erroneous request for use of the DIO II Bus.
- Note 7 This minimum amount of time is allowed to release the control and data buses after $\overline{\text{BGACK}}$ is deasserted. Older Series 200 processor boards, when taking control of the bus, do not wait the minimum amount of time before asserting their own buffers. Therefore, contention may result while trying to tri-state the previous bus master's buffers.
- Note 8 The address bus must be tri-stated at or before $\overline{\text{BGACK}}$ is deasserted.
- Note 9 $\overline{\text{BG}}$ must be high this minimum amount of time to guarantee that the bus grant daisy chain has time to clear before the next grant occurs.

Example

This is an example of the bus handoff when multiple masters are involved. Following this diagram (on the next page) is Table 7-3 which describes the events taking place in the diagram.

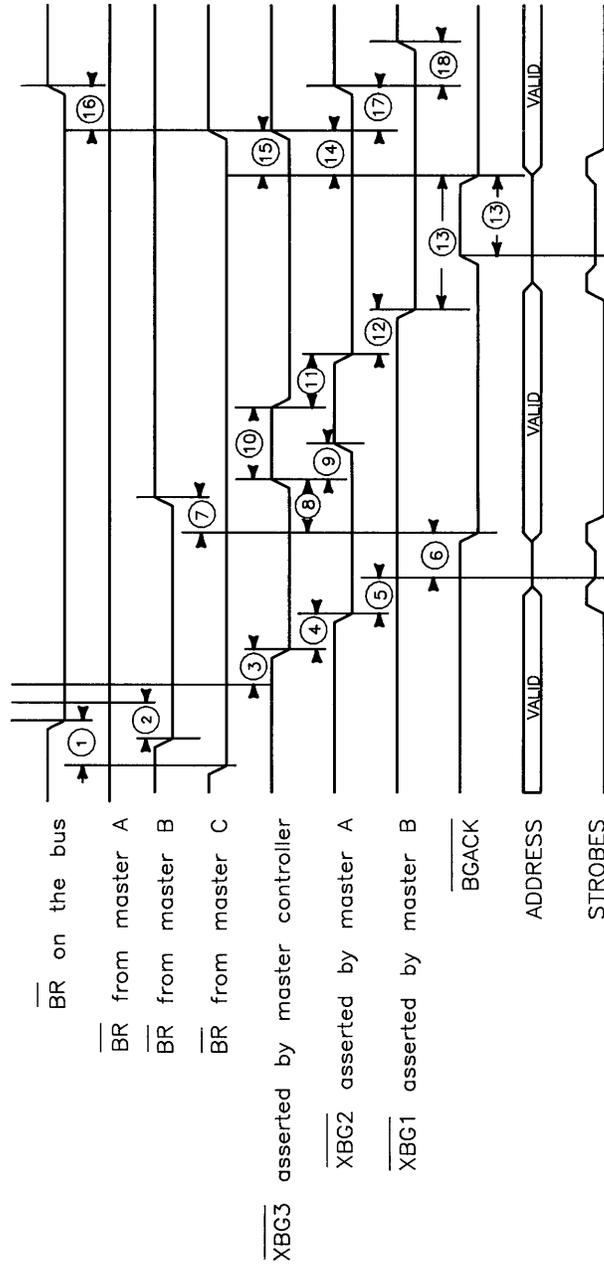


Figure 7-3. Example of Bus Handoff

Table 7-3. Bus Handoff Sequence of Events

SEQUENCE NUMBER	ACTION
1	\overline{BR} asserted by master C.
2	\overline{BR} asserted by master B before \overline{BG} low on bus.
3	\overline{BR} seen low by master controller and $\overline{XBG3}$ is asserted.
4	$\overline{XBG3}$ seen low by master A which passes grant down daisy chain by asserting $\overline{XBG2}$ because master A is not requesting the bus.
5	$\overline{XBG1}$ IS NOT asserted because master B is requesting the bus.
6	Master B waits for inactivity on the bus and asserts \overline{BGACK} .
7	Master B deasserts \overline{BR} after assertion of \overline{BGACK} . This does not change \overline{BR} on bus because master C is asserting \overline{BR} also.
8	Master controller deasserts $\overline{XBG3}$ in response to \overline{BGACK} .
9	Master A deasserts $\overline{XBG2}$ in response to $\overline{XBG3}$ being deasserted.
10	The master controller waits the minimum arbitration reset time and then reasserts $\overline{XBG3}$.
11	Master A not requesting bus so $\overline{XBG2}$ is asserted.
12	Master B not requesting bus so $\overline{XBG1}$ is asserted. Note that master B currently owns the bus.
13	Master C waits for strobes to deassert, \overline{BGACK} to go high, and $\overline{XBG1}$ to assert before taking control of the bus and asserting \overline{BGACK} .
14	In response to assertion of \overline{BGACK} , $\overline{XBG3}$ is deasserted.
15	In response to it's assertion of \overline{BGACK} master C releases \overline{BR} .
16	Deassertion of \overline{BR} by master C cause \overline{BR} on bus to rise since no other bus master is requesting.
17	In response to $\overline{XBG3}$ deasserting $\overline{XBG2}$ is deasserted.
18	In response to $\overline{XBG2}$ deasserting $\overline{XBG1}$ is deasserted.

DMA Operation

8

Introduction

This chapter gives an overview of DMA operation and discusses DMA input and output operation.

The DIO II Bus supports 2 DMA channels.

DMA transfers are possible in byte, word, or long word mode.

A DMA Controller performs three tasks:

- Monitors DMA requests from I/O cards
- Requests control of the bus
- Orchestrates DMA data transfers

The Series 200, and Series 300 Models 310 and 320 computers, use the HP 98620A or HP 98620B DMA Controller cards for transfers.

The Series 300 Models 330 and 350 computers, use the 1TQ4-0401 DMA chip.

DMA Signals

The signals unique to DMA operation are listed below. In addition to these signals, the normal master-slave data transfer signals are used.

Table 8-1. DMA Signals

SIGNAL	DEFINITION
$\overline{\text{DMAR0}}, \overline{\text{DMAR1}}$	DMA Request. Asserted by an I/O card to request a DMA transfer on DMA Channel 0 or DMA Channel 1.
$\overline{\text{DMACK0}}, \overline{\text{DMACK1}}$	DMA ACKnowledge. Response from the DMA Controller, acknowledges DMA request on Channel 0 or Channel 1.
$\overline{\text{DMARDY}}$	DMA ReaDY. Indicates that the I/O card has provided the data (DMA input) or accepted the data (DMA output).
$\overline{\text{DONE}}$	DONE. An output from the DMA Controller to flag the last DMA transfer. DONE can be used at the option of the I/O card designer to determine when DMA is done (For example: to assert EOI on the last byte in an HP-IB transfer).
$\overline{\text{FOLD}}$	FOLD. An output from the DMA Controller to indicate that the data byte needs to be folded from the upper byte of the data bus (from memory) to the lower byte (to an I/O card) or from the lower byte (from an I/O card) to the upper byte (to memory).

In the discussions that follow, DMAR0 and DMACK0 are used. However, all operations apply equally to DMAR1 and DMACK1.

DMA Overview

To enable a DMA transfer, the operating system performs two operations:

1. Programs the DMA Controller with the type of transfer (long word, word, or byte; input/output; priority; etc.).
2. Enables DMA channel 0 or DMA channel 1 on the I/O card by writing to its Control Register. When the I/O card is ready to transfer data, it will request a DMA operation on the assigned channel using DMAR0 or DMAR1. In response to this request, the DMA Controller requests and eventually receives control of the bus from the bus master. It then executes the DMA transfer.

A DMA transfer takes a single bus cycle, during which data is both read and stored. The reference to “a single bus cycle” means that all operations happen during a single cycle to (or from) RAM. For a DMA output cycle, the data is fetched from memory and written to the I/O card. For a DMA input cycle, the data is read from the I/O card and stored in memory. The I/O card itself is programmed to request the DMA transfer. Upon seeing this DMA request, the DMA Controller requests (and receives) control of the bus and provides the necessary address and control signals to perform the transfer.

During a DMA operation, the memory device does a normal data transfer using BAS, BR/W, BLDS/BUDS, DTACK, etc. Therefore, the I/O card must use different signals to handshake during the transfer of data. As discussed above, the I/O card asserts DMAR0 to request a DMA transfer. Once the DMA Controller has control of the bus, it asserts BAS (which initiates the RAM access) and DMACK0 (which initiates the I/O card operation). When the I/O card has provided or accepted the data, it responds with DMARDY which is analagous to DTACK during a normal transfer.

Note

I/O card designers **must not** rely on either $\overline{\text{BAS}}$ being asserted during DMA cycles. The I/O card should use $\overline{\text{DMACK}}$ or another DMA signal for qualification and not those signals used for RAM.

Byte, word, and long word DMA transfers are supported by the 1TQ4-0401 chip. In long word mode, data is transferred a long word at a time between memory and the I/O device. In word mode, data is transferred a word at a time between memory and the I/O device. However, most I/O cards are byte oriented and do not connect to the upper byte of the data bus. Byte mode supports these I/O cards. In byte mode, I/O data is transferred on the low byte of the data bus to or from both the upper and lower bytes of memory. The DMA Controller supports this via a “Fold Buffer” which is used to transfer data between the upper byte of memory and the lower byte of the data bus for I/O cards. During a DMA input operation, the Fold Buffer is alternately used to transfer I/O data to the upper byte of memory (BD8-BD15).

Likewise, during a DMA output operation, the Fold Buffer is used to transfer memory data on BD8-BD15 to the lower data byte for the I/O card. The DMA Controller provides the FOLD signal indicating when folding is to occur.

Note that, depending on programming of the DMA Controller, the speed of the I/O card, and the speed of the peripheral, the DMA Controller may give up control of the bus between DMA cycles.

Other important information to keep in mind is as follows:

1. While an I/O card is enabled for a DMA transfer, it must still respond to normal bus cycles between DMA cycles (so that its control and status registers can be read).
2. The HP 98620A DMA Controller and the 1TQ4-0401 chip do not generate ENDT.
3. In Series 200 and Series 300 machines, it is not possible to perform DMA operations from alpha or graphics memory.

HP 98620 DMA Controller

An overview of the performance of the HP 98620A/B DMA Controllers, and the 1TQ4-0401 DMA chip follows this paragraph. Remember that the HP 98620A/B DMA Controller is used with the Series 200 and the Series 300 Models 310 and 320 computers. The 1TQ4-0401 DMA Chip is used in the Series 300 Models 330 and 350 Computers. Designers implementing DMA on an I/O card should obtain additional documentation on these devices.

The key features of the HP 98620 DMA Controller are:

- Provides 2 independent DMA channels with programmable priorities.
- Capable of 1.2 Mega-transfers per sec. This provides 1.2 Mbytes/sec in byte mode, 2.4 Mbytes/sec in word mode.
- Supports I/O-to-memory and memory-to-I/O transfers, but not memory-to-memory transfers.
- The HP 98620A and HP 98620B provide a 16 Mbyte address range on both channels and a maximum transfer length of 64 Kbyte transfers. That is, it provides 64 Kbyte transfers in byte mode, or 128 Kbyte transfers in word mode.

The 1TQ4-0401 DMA chip provides 4 Gigabytes of address range on both channels and a maximum transfer length of 4 Giga transfers.

- Memory mapped to asynchronous internal I/O address space 500 000 (H) on Channel 0, and 500 008 (H) on Channel 1.
- “Hardwired” to be bus master between \overline{BG} and $\overline{BG1}$ (see Chapter 7, *Bus Arbitration*).
- Available as an HP 98620A, HP 98620B or 1TQ4-0401 DMA chip.

DMA Output Cycle

Figure 8-1 and Table 8-2 shows the DMA output cycle timing. To do a DMA output, a memory read is followed by an I/O write. Again, DMA Channel 0 is assumed. All operations apply equally to DMA Channel 1.

1. The I/O card asserts $DMAR_0$, indicating that it is ready to begin a DMA output operation.
2. The DMA Controller detects this request and, if it is not the current bus master, it requests and is eventually granted, the system bus.
3. The DMA Controller then initiates what looks like a normal memory read cycle:
 - a. Memory address is put on the bus and BR/\overline{W} line is set to read (high).
 - b. BAS and BDS are asserted for the memory device and $DMA\ ACKnowledge$ ($DMACK_0$) is asserted to indicate to the I/O card that a DMA cycle has started. The I/O card responds to $DMACK_0$ as it does to BAS during a normal transfer. If the DMA transfer is a word transfer, $BLDS$ and $BUDS$ are strobed simultaneously. If a byte transfer, $BLDS$ or $BUDS$ is strobed (depending on the byte being read). If the upper byte is being read, the Fold Buffer is used to transfer data from the upper byte to the lower byte of the data bus for the I/O card.
4. When the I/O card detects $DMACK_0$, it can optionally release $DMAR_0$. This is discussed in more detail below.
5. The memory device fetches the data, places it on the bus with 55 nsec of set up time and asserts $DTACK$. The I/O card detects $DTACK$ and, reacting to it like it normally reacts to BDS , begins its own sequence to accept the data. If in burst mode and $DMAR_0$ was set false after $DMACK_0$, the I/O card must re-assert $DMAR_0$ to request the next cycle. In either case, the I/O card asserts $DMARDY$ when it has accepted the data.
6. The DMA Controller detects that $DMARDY$ has occurred and, responding like bus masters normally respond to $DTACK$, ends the cycle by removing BAS , $BLDS/BUDS$ and $DMACK_0$. Once the I/O card detects that $DMACK_0$ is gone, it removes $DMARDY$.
7. On the last transfer, the DMA Controller generates the $DONE$ signal to tell the I/O card that "this is the last byte". An I/O card can, at its option, use this bit to inhibit further DMA requests. $DONE$ is valid only when $DMACK_0$ is asserted. Therefore it should be qualified with $\overline{DMACK_0}$. Once the transfer count is satisfied, the DMA controller ignores further DMA requests and relinquishes the bus.

The $DONE$ signal works as follows: $DONE$ is asserted by the DMA Controller on the last DMA transfer with the same timing as $DMACK_0$. For an input operation, $DONE$ can be used by the I/O card to inhibit further acceptance or handshaking required by transfer of data from the peripheral. Without the $DONE$ signal, the I/O card, not realizing that the transfer is complete, could accept the next byte from the peripheral. This could result in data being lost (unless the transfer count is set to the actual size minus 1).

For an output operation, the $DONE$ signal is not typically needed since the DMA Controller simply ignores $DMAR_0$ from the I/O card when the transfer count is satisfied. If desired, the $DONE$ signal can be used by the I/O card to inhibit an extra DMA request. Also, the I/O card can use $DONE$ to flag the last byte with an indicator (for example: EOI on the HP 98625 card).

Note that DONE floats when the DMA Controller does not have control of the bus. Designers should use appropriate techniques (such as modifying DONE with another signal) to ensure that its undefined state does not cause problems.

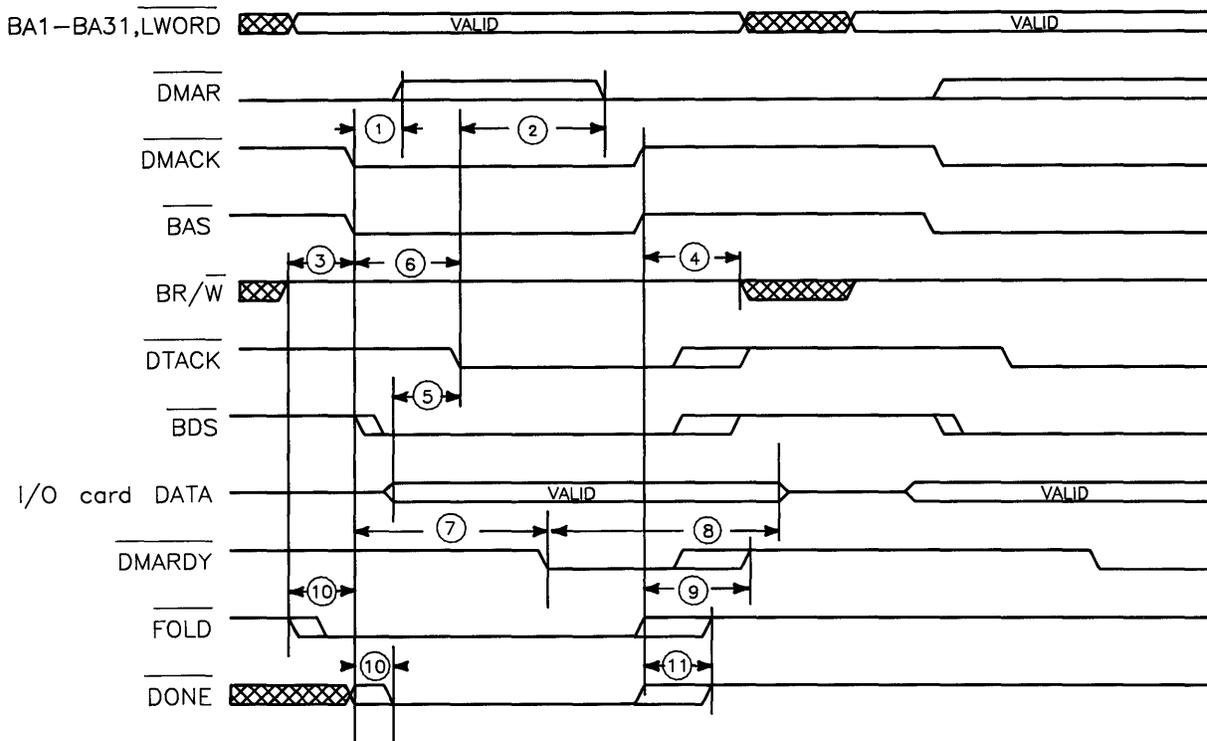


Figure 8-1. DMA Output Timing Diagram

Table 8-2. DMA Output Timing Table

NUM	TIMING PARAMETER	DMA CONTROLLER		DMA REQUESTOR	
1	$\overline{\text{DMAR}}$ release after $\overline{\text{DMACK}}$	0			
2a	$\overline{\text{DMAR}}$ low after $\overline{\text{DTACK}}$ low, with Priority = 0		65		
2b	$\overline{\text{DMAR}}$ low after $\overline{\text{DTACK}}$, with Priority = 1		1600		
3	$\overline{\text{BR/W}}$ setup before $\overline{\text{DMACK}}$ low			15	
4	$\overline{\text{BR/W}}$ hold after $\overline{\text{DMACK}}$ high				15
5	Data setup before $\overline{\text{DTACK}}$ low	-65,15,55			
6	$\overline{\text{BAS}}$ low to $\overline{\text{DTACK}}$ low	0	2300		
7	$\overline{\text{DMACK}}$ low to $\overline{\text{DMARDY}}$ low w/o bus error		4600		
8	Data hold after $\overline{\text{DMARDY}}$ low			85	
9	$\overline{\text{DMARDY}}$ release after $\overline{\text{DMACK}}$ high	0	50		
10	$\overline{\text{DMACK}}$ low to $\overline{\text{FOLD}}$, $\overline{\text{DONE}}$ low			-15	53
11	$\overline{\text{DMACK}}$ high to $\overline{\text{FOLD}}$, $\overline{\text{DONE}}$ high			-15	53

- Note 1 The I/O card may keep $\overline{\text{DMAR}}$ low after its request is acknowledged if it intends to do multiple DMA cycles.
- Note 2 To request another DMA cycle, the I/O card must assert $\overline{\text{DMAR}}$ within the specified time after the memory device's $\overline{\text{DTACK}}$. After satisfying this requirement (Specification 2 in Table 8-2), the HP 98625A disc interface card can set $\overline{\text{DMAR}}$ high. Thus, during the next cycle, $\overline{\text{DMAR}}$ may still be high when $\overline{\text{DMACK}}$ goes low.
- Note 3 $\overline{\text{BR}/\overline{\text{W}}}$ must be set up this minimum amount of time at the DMA requestor before $\overline{\text{DMACK}}$ is asserted by the DMA Controller.
- Note 4 The DMA Controller must hold $\overline{\text{BR}/\overline{\text{W}}}$ true at the DMA requestor for this minimum amount of time after $\overline{\text{DMACK}}$ is deasserted.
- Note 5 On long word transfers the memory must provide -65 nsecs of data setup with respect to $\overline{\text{DTACK}}$ at the I/O device. On word transfers the memory must provide 15 nsecs of data set up with respect to $\overline{\text{DTACK}}$ at the I/O device. On byte transfers, memory must provide a minimum of 55 nsec data set up time prior to $\overline{\text{DTACK}}$ in order to provide 15 nsec of data set up time for the I/O device (due to 40 nsec delay through the Fold Buffer).
- Note 6 $\overline{\text{DTACK}}$ must be low within this maximum amount of time after $\overline{\text{BAS}}$, to guarantee that a bus error will not occur. The maximum number was determined by taking the allowable $\overline{\text{BAS}}$ low time and allowing the slave device only half of the bus cycle time.
- Note 7 $\overline{\text{DMARDY}}$ must be low for this maximum amount of time after $\overline{\text{DMACK}}$, to guarantee that the cycle will end before a bus error will occur.
- Note 8 Data must be guaranteed valid at the I/O device for this minimum amount of time after $\overline{\text{DMARDY}}$ is asserted.
- Note 9 $\overline{\text{DMARDY}}$ must be released for this maximum amount of time after $\overline{\text{DMACK}}$ is deasserted to guarantee that $\overline{\text{DMARDY}}$ is high by the beginning of the next cycle.
- Note 10 $\overline{\text{FOLD}}$ and $\overline{\text{DONE}}$ must be asserted within the specified window to guarantee that the fold buffers and I/O card see the true signals on the correct cycle. $\overline{\text{FOLD}}$ and $\overline{\text{DONE}}$ are only valid when $\overline{\text{DMACK}}$ is asserted.
- Note 11 $\overline{\text{FOLD}}$ and $\overline{\text{DONE}}$ must be deasserted within the specified window to guarantee that the fold buffers and I/O card do not see the true signals on the incorrect cycle.

DMA Input Cycle

Figure 8-2 and Table 8-3 shows the DMA Input cycle timing. To do a DMA input operation, an I/O read is followed by a memory write. Again, DMA Channel 0 is assumed. All operations apply equally to DMA Channel 1.

1. The I/O card asserts DMAR0, indicating that it is ready to begin a DMA input operation.
2. The DMA Controller detects this request and if not the current bus master, it requests and is eventually granted the system bus.
3. The DMA Controller then initiates what looks like a normal memory write cycle:
 - a. Memory address and $\overline{\text{LWORD}}$ are put on the bus and $\text{BR}/\overline{\text{W}}$ line is set to write (low). Notice that $\text{BR}/\overline{\text{W}}$ is set low prior to BAS contrary to a normal write cycle in which $\text{BR}/\overline{\text{W}}$ may go low after BAS. Since the DMA Controller knows that a memory write operation is to occur, it can assert $\text{BR}/\overline{\text{W}}$ immediately.
 - b. BAS is asserted for the memory device and DMACK0 is asserted to indicate to the I/O card that a DMA cycle has started.
4. The I/O card responds to DMACK0 the same as it does to BAS during a non-DMA transfer in that it enables the data transfer. When the I/O card detects DMACK0, it can optionally release DMAR0. This is discussed in more detail in the notes after Table 8-3 (the DMA Input Timing Table). In response to DMACK0, the I/O card fetches the data, places it on the bus and asserts DMARDY with a minimum data set up time of 15 nsec. Because this set up time is measured at the **receiver** end of the bus, the set up time (for data and DMARDY) at the inputs to the drivers on the I/O card must be greater than 15 nsecs (time depends on the driver used). DMARDY indicates to the DMA Controller that the bus data is valid.
5. IF the DMA transfer is a byte transfer and the data is to be written to the upper byte of memory, the DMA Controller uses its Fold Buffer to move the byte from the lower data byte to the upper data byte. In either case, the DMA Controller detects that DMARDY has occurred and asserts the BLDS and/or BUDS to indicate to memory that data is valid on the bus.
6. The memory then stores the data and asserts DTACK to indicate that data has been accepted. The DMA Controller detects that DTACK has occurred and ends the cycle by removing BAS, BLDS/BUDS and DMACK0. In response to the removal of BAS, the memory card removes DTACK. Also, in response to the removal of DMACK0, the I/O card removes DMARDY.
7. On the last transfer, the DMA Controller generates the DONE signal to tell the I/O card that "this is the last byte". An I/O card can, at its option, use this bit to inhibit further DMA requests. Once the transfer count is satisfied, the DMA Controller ignores further DMA requests and relinquishes the bus.
8. A bus error also causes the DMA Controller to terminate the DMA transfer and relinquish the bus. A bus error occurs if a DMARDY does not occur within approximately 5 μsec of DMACK0 going true.

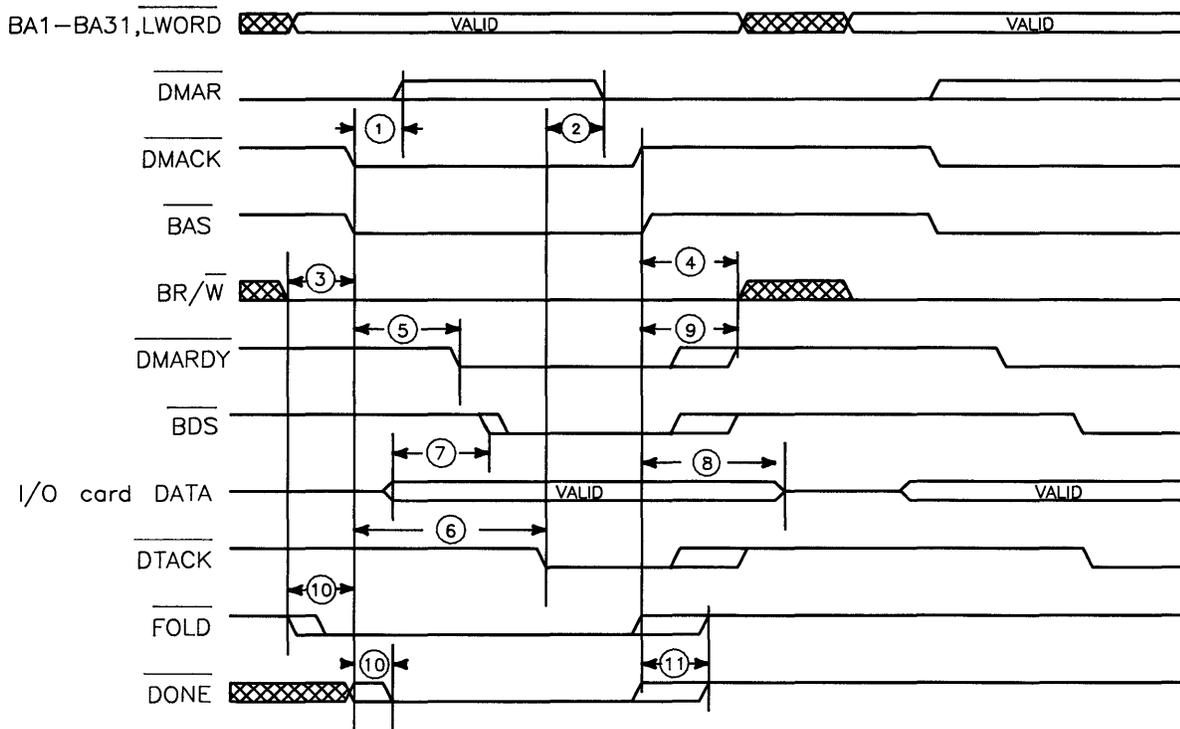


Figure 8-2. DMA Input Timing Diagram

Table 8-3. DMA Input Timing Table

NUM	TIMING PARAMETER	DMA CONTROLLER		DMA REQUESTOR	
1	$\overline{\text{DMAR}}$ release after $\overline{\text{DMACK}}$	0			
2a	$\overline{\text{DMAR}}$ low after $\overline{\text{DTACK}}$ low for burst mode, with Priority = 0		65		
2b	$\overline{\text{DMAR}}$ low after $\overline{\text{DTACK}}$, with Priority = 1		1600		
3	$\overline{\text{BR}}/\overline{\text{W}}$ low before $\overline{\text{DMACK}}$ low			15	
4	$\overline{\text{BR}}/\overline{\text{W}}$ high after $\overline{\text{DMACK}}$ high			15	
5	$\overline{\text{DMACK}}$ low to $\overline{\text{DMARDY}}$ low w/o bus error		2500		
6	$\overline{\text{BAS}}$ low to $\overline{\text{DTACK}}$ low w/o bus error		3000		
7	Data setup before $\overline{\text{DMARDY}}$ low	15			
8	Data hold after $\overline{\text{DMACK}}$ high	0	100		
9	$\overline{\text{DMARDY}}$ release after $\overline{\text{DMACK}}$ high	0	50		
10	$\overline{\text{DMACK}}$ low to $\overline{\text{FOLD}}$, $\overline{\text{DONE}}$ low			-15	53
11	$\overline{\text{DMACK}}$ high to $\overline{\text{FOLD}}$, $\overline{\text{DONE}}$ high			-15	53

- Note 1 The I/O card may keep $\overline{\text{DMAR}}$ low after its request is acknowledged, if it intends to do multiple DMA cycles.
- Note 2 To request another DMA cycle, the I/O card must assert $\overline{\text{DMAR}}$ within the specified time after the memory device's $\overline{\text{DTACK}}$. After satisfying this requirement (Specification 2 in Table 8-3), the HP 98625A disc interface card can set $\overline{\text{DMAR}}$ high. Thus, during the next cycle, $\overline{\text{DMAR}}$ may still be high when $\overline{\text{DMACK}}$ goes low.
- Note 3 $\text{BR}/\overline{\text{W}}$ must be setup this minimum amount of time at the DMA requestor before $\overline{\text{DMACK}}$ is asserted by the DMA Controller.
- Note 4 The DMA Controller must hold $\text{BR}/\overline{\text{W}}$ true at the DMA requestor for this minimum amount of time after $\overline{\text{DMACK}}$ is deasserted.
- Note 5 $\overline{\text{DMARDY}}$ must be low by this maximum amount of time after $\overline{\text{DMACK}}$ to guarantee that the cycle will end before a bus error will occur.
- Note 6 $\overline{\text{DTACK}}$ must be low within this maximum amount of time after $\overline{\text{BAS}}$ is asserted to prevent a bus error.
- Note 7 Data must be set up this minimum amount of time at the RAM before $\overline{\text{DMARDY}}$ is asserted.
- Note 8 Data must be held within the specified window to guarantee that the RAM has time to take the data and to guarantee that no contention will occur on the next bus cycle.
- Note 9 $\overline{\text{DMARDY}}$ must be released this maximum amount of time after $\overline{\text{DMACK}}$ is deasserted to guarantee that $\overline{\text{DMARDY}}$ is high by the beginning of the next cycle.
- Note 10 $\overline{\text{FOLD}}$ and $\overline{\text{DONE}}$ must be asserted within the specified window to guarantee that the fold buffers and I/O card see the signals true on the correct cycle. $\overline{\text{FOLD}}$ and $\overline{\text{DONE}}$ are only valid when $\overline{\text{DMACK}}$ is asserted.
- Note 11 $\overline{\text{FOLD}}$ and $\overline{\text{DONE}}$ must be deasserted within the specified window to guarantee that the fold buffers and I/O card do not see the signals true on the incorrect cycle.

DMA Speed Considerations

To optimize the speed of DMA transfers, the transfer (data read, data write) must be completed during a single bus cycle. Also, the overhead time of the DMA Controller must be minimal and the device connected to the I/O card must be able to provide or accept the data immediately. The time for the existing DMA Controller (HP 98620) to synchronize the handshake signals is similar to the response of the MC68000 and adds minimal overhead.

The following discussion is concerned with the HP 98620 Card and does not apply to the 1TQ4-0401 chip operation.

To meet the desired performance, the DMA Controller must also minimize overhead time between bus cycles. The DMA Controller is designed to hold the bus continuously, providing that the I/O card can generate another DMA request (DMAR0) within a certain length of time after DTACK. This time is either 65 nsec (DMA priority bit = 0) or 1.6 μ sec (DMA priority bit = 1). See note 2 in Figure 8-2 and Table 8-3. Burst mode requires a 65 nsec response. If the I/O card does not assert DMAR within 65 nsec, the DMA Controller relinquishes the bus at the end of the cycle. The next highest priority bus master then assumes control of the bus (typically the master controller). The response to the 65 nsecs is not guaranteed, since it is based on the original state machine of the HP 98620.

Because designing an I/O card to respond to DTACK with DMAR0 within 65 nsec adds complexity to the I/O card, the DMA Controller can be programmed for the 1.6 μ sec DTACK-DMAR0 response time (in the hope that it will generate another DMA request). As mentioned above, selection of this time is done with the priority bit. This works as follows:

1. In burst mode, the priority bit is 0 and the I/O card must assert DMAR0 within 65 nsec of DTACK. This ensures that the DMA Controller keeps control of the bus and provides 1.2 Mbytes/sec.
2. If the priority signal for the channel is 1, then the bus is not relinquished until 1.6 μ sec after the last transfer is complete.

Terminating DMA Transfers

DMA transfers can be terminated in several ways:

1. The DMA Controller can be programmed to interrupt the master controller after the transfer is complete and the bus relinquished.
2. The bus master can monitor the ARM bit in the DMA Controller between DMA cycles (assuming that the bus is released between DMA cycles). When the ARM bit is 0 after completion of the DMA transfer, the bus master can react accordingly.
3. The I/O card can use the DONE signal from the DMA Controller to interrupt the bus master. The HP 98625 disc interface card uses this technique.

Software Interface

The 1TQ4-0401 DMA chip appears to the CPU as a set of memory mapped registers that consist of status and control operations. From the system perspective, the DMA chip is a 16-bit wide peripheral. The chip contains the complete register set of the HP 98620B DMA Controller plus an extended register set. Most registers must be accessed on even addresses. The exceptions are HP 98620B status register and the 1TQ4-0401 status and control registers. These may be byte addressed.

For DMA transfers to occur, the following two conditions must be met:

- A DMA channel must be set-up to transfer data.
- An I/O card must be set-up to do the DMA transfer on the channel.

Operating system or user-written code must provide the following coordination:

- Match the DMA channel useage and priority to I/O cards.
- Ensure that only one card per DMA channel is enabled to perform DMA transfers at any given time.

Three sets of registers will be discussed in this section.

1. HP 98620B Compatible Registers.
2. General Status/Control Registers.
3. Channel Specific Registers.

HP 98620B Compatible Registers

The 1TQ4-0401 DMA chip provides the complete HP 98620B interface for backwards compatibility, but has a full 32-bit physical address range in order to support both the DIO and DIO-II buses. The definition of these registers is identical to that of the HP 98620B register definition and is provided here for reference.

Channel 0 control register base address is 00500000 (H).

Channel 1 control register base address is 00500008 (H).

RESET ARM and INTERRUPT Register

Characteristics:

- It has an offset address of 0.
- It is a READ only register.
- Byte operations are NOT allowed on the RESET ARM and INTERRUPT register.
- It has no data associated with it.

The function of this register is to:

- Clear a pending interrupt (when ARM is already cleared).
- Abort a DMA transfer. (ARM is set but no normal termination has been reached).

If a transfer is aborted, the ADDRESS and TRANSFER COUNT registers are not disturbed so that the transfer may be resumed by simply re-arming the channel.

ADDRESS Register

Characteristics:

- Their offset addresses are 0 and 2 (High Address and Low Address respectively). The High Address is bits 31 thru 16 while the Low Address is bits 15 thru 0.
- These are WRITE registers.
- Byte operations are **NOT** allowed with these registers.

The ADDRESS register is a 32-bit write-only register that increments and holds the DMA memory address pointer. Although the DMA address cannot be obtained by reading the address registers, it can be derived from the current transfer count and the buffer parameters. The address register can be loaded with one “move long word” instruction (MOVE.L), since the high and low words of the address are in sequence.

TRANSFER COUNT Register

Characteristics:

- It has an offset address of 4.
- It is a READ/WRITE register.
- Byte operations are **NOT** allowed on the TRANSFER COUNT register.

The TRANSFER COUNT register is a 16-bit read/write register that holds and decrements the current number of transfers remaining in the transfer. The register is decremented once for each transfer. Since the register is 16-bits wide, this permits continuous transfers of up to 64K bytes in byte-mode and 128K bytes in word-mode without resetting the HP 98620B compatible registers.

The transfer count register holds the $COUNT-1$. Upon normal termination, the register should hold -1 (FFFF (H)).

The transfer count register may be read at any time to allow monitoring the progress of the transfer. This register should only be accessed by WORD instructions.

CONTROL and ARM Register

Characteristics:

- It has an offset address of 6.
- It is a WRITE register.
- Byte operations are **NOT** allowed on this register.

The CONTROL and ARM register is a 7-bit wide write-only control register that enables DMA. Any write to a Channel Control register arms the corresponding DMA channel.

The normal programming sequence to set up and arm either of the two DMA channels is:

1. Write ADDRESS.
2. Write COUNT.
3. Write CONTROL, and thereby ARM the channel.

DMA on the programmed and armed channel will commence when a DMA-capable I/O card requests DMA transfer on that channel. The four control bits that control the actual transfer are shown in Table 8-4.

Table 8-4. CONTROL and ARM Register Bit Assignment

Control Bit	Definition
INT (bit 0)	If this bit is set, a normal or bus error termination will set the interrupt line and the interrupt status bit. The DMA controller interrupts on a level determined by bits 4 through 6 of this register, and must be polled to check interrupt status.
W/ \bar{B} (bit 1)	If set, word transfers are executed and the address register is incremented by two for each transfer. If cleared, byte transfers are done with memory packing. The type of transfer must be coordinated to match the DMA transfer type supported by the I/O card.
O/ \bar{I} (bit 2)	This defines the direction of the DMA transfer relative to the I/O card. If set, data is read from memory and OUTPUT to the card. If cleared, data is read from the card and INPUT to memory.
PRI (bit 3)	This controls the interaction between DMA channel 0 and DMA channel 1. When the priority bit is SET, high bandwidth transfers are expected. The chip will immediately set up for the next transfer upon completion of the current transfer. It will wait 1.5 usec for an I/O card to handshake. The chip will terminate the DMA cycle and give up the bus if the I/O card doesn't handshake within this period. When the priority bit is CLEAR, low bandwidth transfers are expected. The chip will terminate the DMA cycle and give up the bus upon completion of the current transfer. The interaction between the two channels, as a function of priority bit setting, is shown in Table 8-5.
INTERRUPT LEVEL (bits 4-6)	The interrupt level of the HP 98620B compatible chip is programmable between levels 3 and 7. The default interrupt level is 3. The value stored in bits 4-6 are added to the default interrupt level to yield the programmed interrupt level. Bit 6 is the most significant bit. The values required to achieve each of the programmable interrupt levels are shown in Table 8-6.

Table 8-5. PRIORITY Bit Setting Versus Channel Operation

Channel 0	Channel 1	Result
0	0	Priority will be allocated on a circular priority basis, so that neither channel will be locked out. Low bandwidth (less than 200K transfers/second) are expected on both channels.
1	0	High bandwidth requests are expected on channel 0 and low bandwidth requests are expected on channel 1. Channel 1 cannot hinder channel 0, hence channel 0 has priority over channel 1. Channel 1 gets any free bandwidth available.
1	1	High bandwidth requests are expected on both channels. Priority will be allocated on a circular basis so that neither channel will be locked out.

Table 8-6. Interrupt Level Bits

Interrupt Level	Bit 6	Bit 5	Bit 4
3 (default)	0	0	0
4	0	0	1
5	0	1	0
6	0	1	1
7	1	0	0

STATUS Register

Characteristics:

- It has an offset address of 6.
- It is a READ register.
- Byte operations are allowed on the Status register.

The STATUS register is a two-bit read-only register that contains the current status of the channel. The status is not destroyed by reading, and it may be read at any time.

The two bits that indicate channel status are:

ARM (bit 0) If set, indicates that the channel is enabled to do DMA transfers. Any of four conditions will clear the ARM bit:

- Normal DMA termination.
- Bus error termination.
- General system hardware reset.
- Software-controlled reset (RESET ARM and INT).

A bus error termination can be distinguished from a normal termination by reading the contents of the TRANSFER COUNT register. The contents of the TRANSFER COUNT register should always be -1 upon a normal termination.

INT (bit 1) If set, indicates that the DMA interface is interrupting on a level determined by bits 4-6 of the CONTROL register. INT can only be set if all of the following conditions are met:

- Bit 0 (interrupt control) is already set in the CONTROL register.
- A normal or bus error termination occurs.

The only way to reset the INT bit is to read the RESET ARM and INTERRUPT register. Note that since the DMA channels are independent, clearing the interrupt bit for one channel has no effect upon that status of the other channel.

General Control Register

General control register base address is 00500010 (H)

The 1TQ4-0401 DMA chip provides the complete interface to support both the DIO and DIO-II bus. The chip architecture is similar to the HP 98620B DMA Controller, but provides the following additional features:

- Read/write address, transfer count, control registers.
- DIO-II 32-bit transfers (word.l).
- Programmable priority wait time.
- Programmable bus utilization percentage.
- 32-bit wide transfer count register (up to 4 Giga transfers without rearming).

The two general registers are:

- ID Register.
- General Control Register.

ID Register

Characteristics:

- It has offset addresses of 0 and 2.
- It is a READ register.

The register is actually one long word which uniquely identifies the chip. The four ID register bytes of the long word (offset 0) are ASCII encoded and have the following meaning (Table 8-7).

Table 8-7. ID Register Byte Contents

Field Definition	Location (Offset from 500010 (H))
General architecture type	0, 1
Major revision letter	2
Minor revision number	3

As of this writing, the general architecture is HP 98620 (20 for short). The major revision letter is C, and the minor revision number is 0. Hence, reading the long word at Address 00500010 (H) yields the following ascii ID:

20C0

GENERAL CONTROL Register

Characteristics:

- It has an offset address of 4.
- It is a READ/WRITE register.
- Byte operations are allowed.

This register contains four valid fields. These are:

BR (bit 1 and 0) Bandwidth Restriction. These bits set the upper limit for the amount of DIO-II bus bandwidth that the DMA chip is allowed to have. The restriction is defined according to the following table:

Table 8-8. Bandwidth

Bit 1	Bit 0	Bandwidth Limit (%)
0	0	100
0	1	50
1	0	25
1	1	12.5

BT (bit 3 and 2) Burst Wait Time. These bits set the upper limit for the time that the DMA chip will wait for an I/O card to request a DMA transfer when the priority bit is set for the channel. The restriction is defined according to the following table:

Table 8-9. Burst Time

Bit 3	Bit 2	Burst time (clocks)
0	0	4
0	1	8
1	0	16
1	1	32

RESET0 (bit 4) When set (high), DMA channel 0 is reset. The chip automatically clears this bit upon completion of the reset. Resetting the channel does not affect the address or transfer count registers. However, resetting the channel does set the software halt status bit, and clears the interrupt and arm status bits.

RESET1 (bit 5) Setting this bit has the same effect as setting RESET0 except the reset is performed on channel 1 instead of channel 0.

CHANNEL SPECIFIC Registers

Channel 0 control register base address is 00500100 (H)

Channel 1 control register base address is 00500200 (H)

ADDRESS Register

Characteristics:

- It has an offset address of 0
- It is a READ/WRITE register
- Byte operations are **NOT** allowed

The ADDRESS REGISTER is a 32-bit read/write register that increments and holds the DMA memory address pointer. The address register can be loaded with one “move long word” (MOVE.L) instruction, since the high and low words of the address are in sequence. Byte operations are not allowed.

TRANSFER COUNT Register

Characteristics:

- It has offset addresses of 4.
- It is a READ/WRITE register.
- Byte operations are **NOT** allowed.

The TRANSFER COUNT register is a 32-bit read/write register that holds and decrements the current number of transfers remaining. The register is decremented once for each completed transfer. Since the register is 32-bits wide, this permits continuous operation of up to 4 Giga transfers.

The transfer count register holds COUNT-1. Upon normal termination, the register should hold -1 (FFFFFFFF (H)).

The transfer count register may be read at any time to allow monitoring the progress of the transfer. This register should only be accessed by word or long word (word.l) instructions. Byte operations are not allowed.

CONTROL Register

Characteristics:

- It has an offset address of 8
- It is a READ/WRITE register
- Byte operations are **NOT** allowed

Valid fields for this register are:

- INT (bit 0) If this bit is set, a normal or bus error termination will set the interrupt line and the interrupt status bit. The DMA controller interrupts on a level determined by bits 4 through 6 of this register; and must be polled to check interrupt status.

- W/ \overline{B} (bit 1) If set, word transfers are executed and the address register is incremented by two for each transfer. If cleared, byte transfers are done with memory packing. The type of transfer must be coordinated to match the DMA transfer type supported by the I/O card.

- O/ \overline{I} (bit 2) This defines the direction of the DMA transfer relative to the I/O card. If set, data is read from memory and OUTPUT to the card. If cleared, data read from the card and INPUT to memory.

- PRI (bit 3) This controls the interaction between DMA channel 0 and DMA channel 1.
 When the priority bit is SET, high bandwidth transfers are expected. The chip will immediately set up for the next transfer upon completion of the current transfer. It will wait 1.5 usec for an I/O card to handshake. The chip will terminate the DMA cycle and give up the bus if the I/O card does not handshake within this period.
 When the priority bit is CLEAR, low bandwidth transfers are expected. The chip will terminate the DMA cycle and give up the bus upon completion of the current transfer.
 The interaction between the two channels as a function of priority bit setting is as follows:

Table 8-10. Channel Priority Bit Operation

Channel 0	Channel 1	Result
0	0	Priority will be allocated on a circular priority basis, so that neither channel will be locked out. Low bandwidth (Less than 200K transfers/second) are expected on both channels.
1	0	High bandwidth requests are expected on channel 0 and low bandwidth requests are expected on channel 1. Channel 1 cannot hinder channel 0, therefore, channel 0 has priority over channel 1. Channel 1 gets any free bandwidth available.
1	1	High bandwidth requests are expected on both channels. Priority will be allocated on a circular basis so that neither channel will be locked out.

INTERRUPT LEVEL (bits 6 thru 4) The interrupt level of the HP 98620B DMA Controller is programmable between levels 3 and 7. The default interrupt level is 3. The value stored in bits 4 thru 6 are added to the default interrupt level to yield the programmed interrupt level. Bit 6 is the most significant bit. The values required to achieve each of the programmable interrupt levels are:

Table 8-11. Interrupt Level

Interrupt Level	Bit 6	Bit 5	Bit 4
3 (default)	0	0	0
4	0	0	1
5	0	1	0
6	0	1	1
7	1	0	0

LWORD (bit 8) LWORD is a single bit field which defines whether a 32-bit transfer will be performed. The DMA chip will not allow 32-bit (word.1) DMA transfers to memory locations which aren't aligned to long word boundaries.

System software should take this into consideration when setting up the buffers.

START (bit 15) START. Writing a 1 to this bit does not actually set the START bit. Instead, the status register arm bit is set, the channel is armed, and the error codes in the channel status register are cleared.

The power-up state of both channel control registers is as follows:

Table 8-12. Power-Up of Channel Control Registers

Control Name	Bit Line	Default	Effect
INT	0	0	Do not interrupt on done or error.
W/ \bar{B}	1	0	Byte transfers.
O/ \bar{I}	2	0	DMA direction is IN.
PRI	3	0	Channel priority is 0 (lowest).
INT LVL	6 thru 4	0	Channel interrupt level is 3.
LWORD	7		
	8	0	Transfer size is partial (byte or word).
	11		
	12		
	13		
	14		
START	15	0	Do not arm DMA channel.

CHANNEL STATUS Register

Characteristics:

- It has an offset address of “A”(H).
- It is a READ register.
- Byte operations are allowed.

The following fields are valid for the Channel Status register:

ARM (bit 0)	If set, indicates that the channel is enabled to do DMA transfers. Any of four conditions will clear the ARM bit: <ul style="list-style-type: none">• Normal DMA termination.• Bus error termination.• General system hardware reset.• Software-controlled reset.
INT (bit 1)	If set, indicates that the DMA interface is interrupting on a level determined by bits 4 thru 6 of the channel control register. INT can only be set if the following conditions are true: <ul style="list-style-type: none">• Bit 0 (interrupt control) is already set in the CONTROL register.• A normal or error termination occurs. <p>The only way to reset the INT bit is to reset the channel. Note that since the DMA channels are independent, clearing the interrupt bit for one channel has no effect upon that status of the other channel.</p>
ACC (bit 2)	If set, indicates the channel specific control register was written to while the channel was armed. Hence, the DMA control conditions changed before normal termination of the current DMA process.
HALT (bit 3)	If set, indicates the current DMA process was halted programmatically by a writing to the channel reset bits in the General Control register.
BERR (bit 4)	If set, indicates the current DMA process was halted as a result of some outside party asserting the NBERR input to the chip for more than 2 full clock periods. (Greater than 200 nanoseconds, assuming a 10 MHz clock).
ALIGN (bit 5)	If set, indicates an attempt was made to perform DMA to non-aligned boundaries. That is, word-oriented DMA to an odd-byte boundary, or long word (word.l) oriented DMA to a non long word aligned boundary.
ARO (bit 6)	If set, indicated the DMA address counter rolled over from FFFFFFFF (H) to 00000000 at some time during the just-ended DMA process.

Software Performance

Use of Interrupts Versus Fast Handshake

A DMA transfer can be monitored by the processor of either the fast handshake method (continuously polling the DMA chip status register), or by setting the DMA chip (or I/O card) to interrupt when the DMA process is complete.

There is a subtle difference in behavior of the 1TQ4-0401 DMA chip versus the HP 98620B card. This difference in behavior may affect the philosophy of how certain drivers are implemented.

The difference is related to the decision of whether to set the DMA channel priority bit. By definition, if the bit is 0 then less than 200K transfers per second are expected. If the bit is 1, then more than 200K transfers per second are expected (up to maximum bus bandwidth). The DMA control state machine retains control of the bus for a predefined period while waiting for a valid DMA request.

As a result of the above definition, the DMA chip always interleaves with the processor when the following conditions are satisfied (by interleave we mean — releases ownership of the I/O bus, then immediately asserts the bus request):

- One DMA channel armed.
- Priority bit (control register bit 3) is not set.

In contrast, the HP 98620B DMA card retains ownership of the I/O bus if a valid DMA request is seen by the state machine controlling the card - regardless of whether the channel priority bit is set.

The differences between the 1TQ4-0401 DMA chip and the HP 98620B card may surface in the HP 98625A/HP 98620B disc driver. The HP 98625A was designed to be highly tuned to the timing of the HP 98620x DMA state machine. Therefore, the HP 98625A could request additional DMA transfers such that the request would be seen, after synchronization, by the DMA control state machine. The HP 98620B DMA card would retain ownership of the bus and process the next DMA request - even if the priority bit is not set. Software drivers were written to take advantage of this relationship and do not set the DMA channel priority bit, even though the HP 98625A is a high-bandwidth device. Therefore, the performance of high-speed disc transfers will probably degrade when using the DMA chip.

Programmable BUS Bandwidth

The 1TQ4-0401 DMA architecture includes a general control register through which certain global parameters for the chip may be set. Bits 3 and 0 of this register (address 00500014 H) contains two 2 bit fields: BT (burst wait) and BR (bandwidth restriction) respectively.

On the active channel, when the priority bit is set, Burst wait time is the amount of time expressed in number of DMA input clock periods (100 nsec each), that the DMA channel will wait, following a completed DMA cycle, for DMA Request to become valid. The HP 98620B burst wait time is always 16 clocks. The 1TQ4-0401 chip burst wait time is programmable to 4, 8, 16 or 32 clocks.

Bandwidth restriction is the maximum percentage of a specific period of time which the DMA chip will be allowed to use the I/O bus. The HP 98620B bandwidth limit is 100%. The 1TQ4-0401 chip bandwidth limit is programmable to 100%, 50%, 25%, or 12.5%.

The total time which the DMA may control the bus is the DMA period. The total time guaranteed to other bus masters is the MPU period. The sum of the DMA period and the MPU period is the total time allowed for one complete bus monitoring period (TTL period). The length of these periods is determined by the settings of the BT and BR fields and is shown in the following table:

Table 8-13. Bandwidth Timing

Burst Time	Bandwidth Percent	DMA Period (clocks)	MPU Period (clocks)	TTL Period (clocks)
4	100	all	0	(na)
4	50	8	8	16
4	25	8	24	32
4	12.5	8	56	64
8	100	all	0	(na)
8	50	16	16	32
8	25	16	48	64
8	12.5	16	112	128
16	100	all	0	(na)
16	50	32	32	64
16	25	32	96	128
16	12.5	32	224	256
32	100	all	0	(na)
32	50	64	64	128
32	25	64	192	256
32	12.5	64	448	512

This chapter identifies and defines the signal lines which serve utility-type functions on the DIO II Bus. These utility lines supply initialization and diagnostic capability for the bus and consist of the following signals:

- Bus Drive Disable ($\overline{\text{BDRV}}$)
- $\overline{\text{RESET}}$
- $\overline{\text{HALT}}$
- Function Codes (BFC0, BFC1, BFC2)

Bus Drive Disable

The 09826-66516, 09826-66517, 09816-66511 and all Series 300 processor boards respond to the $\overline{\text{BDRV}}$ signal (this signal is called $\overline{\text{BMON}}$ in the HP 9816). The HP 98206A Test Stimulus Board generates BDRV to disable the CPU board boot ROM and keep it from responding, while replacing the boot ROM code with test stimulus code. This permits testing of the computer without depending on a working boot ROM.

Because $\overline{\text{BDRV}}$ is primarily a diagnostic/development tool and is not used by I/O cards, detailed information is not provided here.

Reset Operation

In the HP 9826, and the HP 9836, RESET and HALT are asserted by the power supply at power-on and remain true until 120 msec after +5V reaches 4.5 volts. In all other Series 200 and Series 300 computers, the RESET and HALT are asserted by the processor boards under similar conditions. When RESET goes false, the 12V supply will have been within its 11.5V limit for at least 55 msec. At power-down, RESET is re-asserted within 15 msec after 5V drops to approximately 4 volts. Note that RESET comes too late to properly reset devices at power-down (RESET would ideally be asserted before 5V reaches its TTL limit of 4.75). Refer to the *HP 9000 Series 200/300 Computers Accessory Development Guide* for more information about the power-on RESET.

In Series 200 and 300 computers, activating RESET on the DIO Bus does **not** reset the processor board unless HALT is simultaneously asserted. Thus, RESET by itself only resets other bus devices. When RESET and HALT are both asserted, the processor board is reset to its power-on state. In response to RESET and HALT, certain processor boards refloat their RAM in preparation for auto locate while others keep the address fixed.

In addition to responding to an external RESET, 68000-based processor boards can use the RESET instruction to strobe $\overline{\text{RESET}}$ low on the DIO Bus.

Thus, in specifying $\overline{\text{RESET}}$ timing, there are 3 areas that need to be covered:

1. The duration of $\overline{\text{RESET}}$ at power-on to properly reset the processor. In new designs, $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ should be low at power-on and remain low until all supplies have stabilized within their operating limits for at least 150 ms. At power-down, $\overline{\text{RESET}}$ should go low before any supply exceeds its operating limits.
2. The duration of $\overline{\text{RESET}}$ (and $\overline{\text{HALT}}$) after power-on to reset the processor. $\overline{\text{RESET}}$ (and $\overline{\text{HALT}}$) should have a 50 μsec minimum time duration.
3. The width of the $\overline{\text{RESET}}$ output pulse generated by the processor in executing the RESET instruction. The minimum $\overline{\text{RESET}}$ pulse width is 8 μsec .

Halt Operation

HALT is asserted on the DIO II Bus at power-up in Series 200 and 300 computers with the same timing as RESET. In addition, HALT can be asserted by a DIO II Bus device to halt the CPU. HALT cannot be asserted on the DIO II Bus by existing processor boards because there is no output driver from the CPU to the HALT pin. However, there is a receiving buffer which permits a DIO II Bus device to halt the processor.

Processor boards with MC68010 and MC68020 CPU's support re-run if $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ are asserted per the CPU re-run timing requirements. However, re-run has never been tested in Series 200 or 300 computers so it is not known whether it functions correctly.

Function Code Signals

The Function Codes (FC0, FC1 and FC2) generated by the 68000 are buffered (BFC0, BFC1 and BFC2) and brought out on the bus for general purpose use and for future expandability. As would be expected, the function code buffer is disabled when bus control is passed to another bus master. However, this buffer is also disabled (on certain processor boards) during an interrupt acknowledge cycle. This inhibits certain control signals which happen to share the same buffer. Therefore, when $\text{FC0} = \text{FC1} = \text{FC2} = 1$ (interrupt acknowledge), the buffered function codes on the bus will float (no pullups) and are undefined. To use function codes, these guidelines should be followed:

1. When BAS occurs, the buffered function codes on the DIO Bus are valid. $\overline{\text{BAS}}$ is one of the control signals disabled during an interrupt acknowledge cycle. However, since $\overline{\text{BAS}}$ is pulled up by a pullup resistor, it does not float (in contrast to BFC0 thru BFC2, which do float).
2. During an interrupt acknowledge cycle, BFC0, BFC1 and BFC2 on the bus are undefined and cannot be used.

Electrical Specifications

10

This chapter defines the non-timing electrical specifications for the DIO and DIO II Bus. Included in this chapter are the power supply tolerances, the I/O card power dissipation specifications and signal loading recommendations. Pinouts of the DIO and DIO II Bus are discussed in Chapter 11, *Mechanical Specifications*.

Note that all slots in the I/O backplane are identical in the Series 200 and Series 300 Models 310 and 320, while in the Series 300 Models 330 and 350, there are large (System) slots and smaller (DIO) slots. There is no ordering or prioritizing of signal lines by location in the backplane. That is, address lines, data lines and control signal lines do not have a specific order, nor are interrupt lines prioritized by location. However, future products may have slot-dependent features.

Power Distribution and Grounding

Power on the DIO II Bus is distributed on the backplane as regulated DC supplies. The supplies are:

- +5 Vdc Main logic supply
- +12 Vdc Provided for I/O circuitry requiring multiple voltages. Can also be used for analog applications.

Note

There is a potential for noise on the +12 Vdc line as this supply is used for the floppy drive motor in HP 9826A's. Designers should be aware of potential noise on the line.

- 12 Vdc Provided for I/O circuitry requiring multiple voltages. It can also be used for analog applications.

The DIO II Bus provides several logic ground lines (GND — see Chapter 11, *Mechanical Specifications* for pinout information). On the DIO Bus, two lines are used for dirty ground (DGND). They are to be used for future cards which generate excessive ground noise (such as, cards with relays). In Series 200 products DGND lines run separately from the I/O backplane thru the motherboard to the power supply. In Series 300 products DGND and GND are connected together at the connector. Therefore, care should be taken when applying noise signals to DGND lines.

Power Supply Tolerances

The supply performance specifications shown below allow for the effects of line regulation, load regulation, cross regulation, initial accuracy, temperature stability and ripple. The tolerances represent the worst-case tolerances for existing and planned DIO Bus devices.

Table 10-1. Power Supply Tolerance

SUPPLY	TOLERANCE	RANGE
+5	+5 and -4.3 %	4.78 to 5.25
+12	+6 and -4 %	+11.5 to +12.7
-12	+10 and -4 %	-11.5 to -13.2

Current Requirements

The requirements for both DIO and DIO II cards were determined using current product designs. In the future there may be more power available depending on product design changes. To guarantee compatibility with all existing products, the requirements listed in Tables 10-2 or 10-3 should be met.

DIO II Current Requirements

The current specification for each slot in the DIO II system is given in Table 10-2.

Table 10-2. DIO II Power and Current Requirements (per slot)

	MAXIMUM POWER & CURRENT			
	MAX. PWR/BD (WATTS)	MAXIMUM AVERAGE DC AMPS		
		+5	+12	-12
Standard	39.1	5.0	0.9	0.275

- The “average” DC current represents the average current over some small period of time. Thus, peak currents (such as might occur during a memory refresh) are not a concern.
- Maximum power and current data is listed for reference only. The designer should ensure that these power and current specifications are not exceeded.
- If designers decide that current and power specifications must be exceeded a configuration guide will have to be made to show valid and invalid configurations.
- Expression of card power requirements in terms of a standard deviation about a mean is not necessary as the hardware has demonstrated that the spread is very small.

DIO I/O Card Current Requirements

The current specification for each slot in the I/O cardcage is given below. Note that I/O cards that are “double high” can consume twice this amount of power since they occupy two slots in the cardcage.

Table 10-3. DIO Power and Current Requirements (per slot)

	TYPICAL POWER & CURRENT				MAXIMUM POWER & CURRENT			
	TYP. PWR/BD (WATTS)	TYPICAL AVERAGE DC AMPS			MAX. PWR/BD (WATTS)	MAXIMUM AVERAGE DC AMPS		
		+5	+12	-12		+5	+12	-12
Standard	4.4	.8	.096	.064	5.3	.96	.120	.080
High Pwr	7.6	1.4	.166	.110	9.2	1.68	.207	.138

- The “average” DC current represents the average current over some small period of time. Thus, peak currents (such as might occur during a memory refresh) are not a concern.
- Maximum power and current data is listed for reference only. The designer should ensure that typical power and current specifications are not exceeded.
- Any combination of currents may be used as long as current and power specifications are not exceeded. For example, to stay within 4.4 watts for a low-power card, it is not possible to use the typical currents on all supplies, which would dissipate 5.92 watts. If the typical current on +5V is .8 amps, then .4 watts ($4.4 - 5 \times 0.8$) can be drawn from +12V and -12V.
- Expression of I/O card power requirements in terms of a standard deviation about a mean is not necessary because: (1) sampling of hardware in production demonstrated that the spread is very small and (2) the power supply has sufficient capacity to handle heavier loads.

On-Card Fuse Specification

A UL/CSA/IEC requirement is that any device operating from a power supply capable of supplying more than 8 amps must be fused. Because the +5V supply in Series 200 and Series 300 mainframes is capable of supplying more than 8 amps, a fuse is required in series with the +5V supply on each plug-in board. The suggested fuse is a 4 amp fuse, HP P/N 2110-0592, which is soldered into the circuit board. A plug-in fuse is also available. The part number for the right-angle holder for this fuse is 2110-0691. The part number for a 5 amp fuse to fit this holder is 2110-0520. Fuses with other current ratings are also available.

In addition to fusing +5V, the Data Comm cards (HP 98628, HP 98629, and HP 98691) also have a fuse on the +12V and -12V. This was done because these supplies appear on the I/O connector and are used for powering external devices such as the HP 98629 (SRM Interface) card’s external pod.

However, FSD’s Product Regulation Department states that because of the possibility that I/O cards can operate in mainframes with +12V and/or – 12V supplies capable of supplying more than 8 amps, these power supply lines on the card should also be fused. Because the fuses are inexpensive to buy and insert on the board, this is not seen as a serious disadvantage (fuses can be autoloading).

Signal Loading

Both DIO and DIO II signal loading is shown in Tables 10-4 and 10-5. Designers should design their boards for the appropriate backplane.

DIO II Signal Loading

Table 10-4 shows the receiver loading and recommended driver for each of the signals on the **DIO II** backplane.

Table 10-4. DIO Signal Loading Table

SIGNAL NAME	RECEIVE LOADING	SIGNAL DRIVER
$\overline{\text{BAS32}}$, $\overline{\text{BAS24}}$, $\overline{\text{BUDS}}$, $\overline{\text{BLDS}}$, $\overline{\text{LWORD}}$, $\overline{\text{BLK}}$, and $\overline{\text{BR/W}}$	1 LS or ALS load max	SN74LS245 buffer
$\overline{\text{DTACK16}}$, $\overline{\text{IACK32}}$, $\overline{\text{BG}}$, $\overline{\text{DONE}}$, $\overline{\text{RMC}}$, $\overline{\text{DMACK0}}$, $\overline{\text{DMACK1}}$, $\overline{\text{BERR}}$, $\overline{\text{INT3}}$ - $\overline{\text{INT6}}$, $\overline{\text{VECTOR32}}$, $\overline{\text{BR}}$, $\overline{\text{DMARDY}}$, $\overline{\text{DMAR0}}$, $\overline{\text{DMAR1}}$	1 LS or ALS load max	Any 3S/OC LS gate
$\overline{\text{DSACK32}}$, $\overline{\text{DSACK16}}$, $\overline{\text{ADACK}}$	Any ALS or LS load with hysteresis	Any 3s/OC LS or ALS gate capable of 25 mA.
$\overline{\text{BGACK}}$, $\overline{\text{ENDT}}$	1 LS or ALS load max	Any 3 state LS gate
$\overline{\text{XBG3}}$ - $\overline{\text{BG3}}$	2 LS or ALS loads max.	Any LS gate
$\overline{\text{BFC0}}$ - $\overline{\text{BFC2}}$	2 LS or ALS loads max.	SN74LS245
$\overline{\text{RESET}}$	5 LS or ALS loads max.	ALS1035 OC Buffer
$\overline{\text{HALT}}$.8 ma TOTAL for cardcage	Any 3S/OC LS or ALS gate
BA1 - BA31	2 ALS or LS loads max.	SN74LS245 or SN74ALS245 Buffer
BD0 - BD15, XD0-XD15	2 ALS or LS loads max.	SN74LS245 or SN74ALS245 Buffer

DIO Signal Loading

Table 10-5 shows the receiver loading and recommended driver for each of the signals on the DIO back-plane.

Table 10-5. DIO Signal Loading Table

SIGNAL NAME	RECEIVE LOADING	SIGNAL DRIVER
$\overline{\text{BAS}}$, $\text{BR}/\overline{\text{W}}$, $\overline{\text{BUDS}}$, and $\overline{\text{BLDS}}$	1 LS load max	SN74LS245 buffer
$\overline{\text{DTACK16}}$, $\overline{\text{IACK}}$, $\overline{\text{BG}}$, $\overline{\text{DONE}}$, $\overline{\text{BERR}}$, $\overline{\text{DMACK0}}$, $\overline{\text{DMACK1}}$, $\overline{\text{INT3}}$ - $\overline{\text{INT6}}$, $\overline{\text{VECTOR}}$, $\overline{\text{BR}}$, $\overline{\text{DMARDY}}$, $\overline{\text{IMA}}$, $\overline{\text{DMAR0}}$, $\overline{\text{DMAR1}}$	1 LS load max	Any 3S/OC LS gate
$\overline{\text{BGACK}}$, $\overline{\text{ENDT}}$	1 LS load max	Any 3 state LS gate
$\overline{\text{BG1}}$ - $\overline{\text{BG3}}$	2 LS loads max.	Any LS gate
$\overline{\text{BFC0}}$ - $\overline{\text{BFC2}}$	2 LS loads max.	SN74LS245
$\overline{\text{RESET}}$	5 LS loads max.	ALS1035 OC Buffer
$\overline{\text{HALT}}$.8 ma TOTAL for cardcage	Any 3S/OC LS gate
BA0 - BA23	1 LS load max.	SN74LS245 Buffer
BD0 - BD15	SN74LS245 Buffer	SN74LS245 Buffer

- 3S/OC = 3-State driver or Open Collector
- “Any 3S/OC LS gate” was indicated as the signal driver for Bus Grant Acknowledge ($\overline{\text{BGACK}}$) in previous versions of the DIO document. However, problems were encountered with the slow rise times of an open-collector driver on this line. Thus, the specification has been changed to require a 3 state driver.
- ALS gates are acceptable as receiving devices. Also, Schmitt trigger input devices should be used where possible as receivers.
- Until ALS drivers are characterized on the bus, only 74LS244/245 drivers should be used.

Mechanical Specifications

Introduction

The information contained in this chapter should be sufficient so that DIO II PC boards (sometimes referred to as “system boards”), piggy-backed boards (boards that plug on to system boards to extend the functionality), DIO I/O cards, and DIO non I/O cards (for use in Series 200 and Series 300 Models 310/320), can be built in a compatible configuration with current product designs. The mechanical dimensions of cards designed for use with the DIO or DIO II Bus must conform to the specifications in this chapter.

Since production of the Series 300 Models 330 and 350, there are two different board sizes available for the HP Series 200/300 computers. They require two different card cages. The first is the original DIO card cage (for use with Series 200 and Series 300 Models 310/320) which holds DIO I/O cards and DIO non I/O cards such as DIO memory and DIO DMA. The larger DIO II (system) cards cannot be used in this card cage.

The second card cage was developed for the DIO II and is the card cage which hold the larger DIO II (System) boards. You can use DIO cards in the this card cage if you use an expander, or an accessory to adapt the slot size and connector to the DIO Card.

The Series 300 Models 330/350 computers were the first to introduce the new size format for system boards. The processor and video boards needed the larger format and a two piece connector system. The original DIO cards use one 96 pin edge card connector and were based on 1/6th of an 18 x 24 inch panel. The DIO II board is 1/4th of an 18 x 24 inch panel and requires the use of the large board because of the addition of a 48 pin connector. This means that the larger DIO II (System) boards can be DIO or DIO II systems boards as the 48-pin connector is optional.

The three types of cards (DIO II System boards, DIO I/O cards, and DIO non I/O cards) and two types of card cages are discussed in the next two sections. Following this information are discussions covering the RFI, layout rules, and pin assignments.

DIO II Boards and DIO Card Specifications

The smaller DIO cards (I/O or non I/O) are very similar to each other and fit in the Series 200 (and Series 300 Models 310/320) DIO card cage. The DIO I/O card has an attached metal backplate (cover) and is only allowed in every other slot. The DIO non I/O cards have no cover plate and are notched to allow clearance for the I/O card connectors attached to the cover plate on the I/O card in the slot directly below (see Figures 11-4 and 11-6).

The larger, DIO II (System) boards, which perform either DIO or DIO II functionality, may have connectors and attached cover plates. If the system board has no I/O ports the card is not required to have an attached cover plate. A cover plate with air circulating holes, 98561-04107, will be installed behind this type of board for safety purposes and to provide protection to the board. To prevent shorting of leads and for mechanical integrity a board support, 5041-2415, should be installed on the cover plate.

A board connecting to the system bus will not follow these guidelines as the system bus boards include a cover plates with a connector (see Figure 11-1).

DIO II (System) Boards

The DIO II board specification has been reproduced in Figure 11-1 and 11-2.

Figures 11-1 and 11-2 shows all the requirements for making DIO II (System) boards whether their functionality is DIO or DIO II. Figure 11-1 also shows the location of the system bus connector. The DIO II and the system bus connectors are optional. As changes occur these figures may not reflect the latest changes. Therefore, it is recommended that a design engineer (before designing a new system board) contact Hewlett-Packard's support personnel to confirm that specifications in these figures are still current.

(Dimensions in Figure 11-1 are in inches)

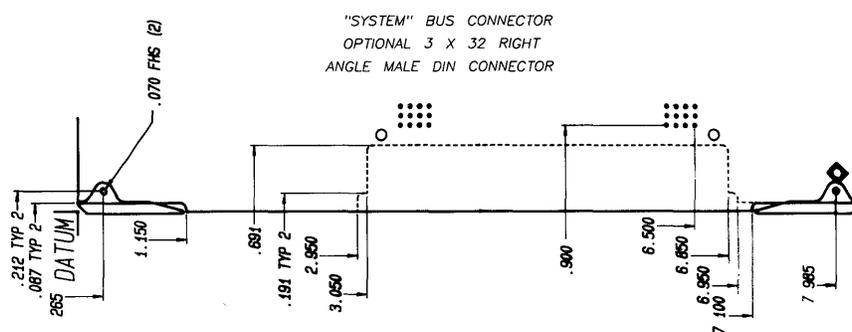
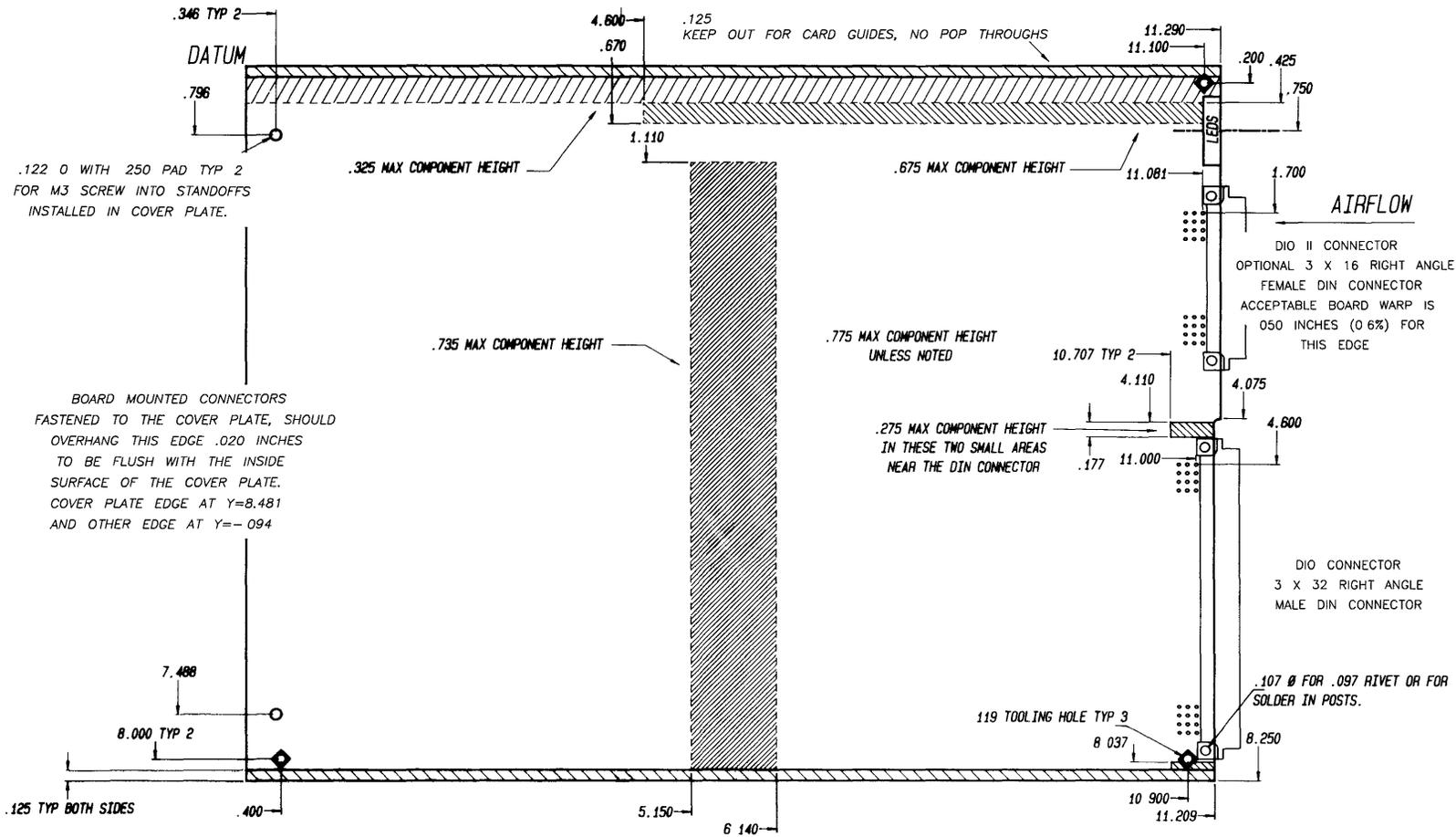


Figure 11-1. DIO II Ejector Lever Drawing

Figure 11-2. DIO II (System) Board Drawing



(Dimensions in Figure 11-2 are in inches)

There are several points that should be considered when designing a DIO II (System) board.

- DIO II (System) boards fit in any slot of the Series 300 Models 330/350 motherboard (also known as the DIO II backplane) and are secured by a cover plate. As mentioned previously the cover plate may be loose or physically attached to the board. The cover plate has captive, retractable thumbscrews.
- The bottom surface of the DIO II (System) board is nominally 4.4 mm above the flange of the cover plate. (A 6 mm standoff is often pressed into the bottom surface of the 1.6 mm thick flange.) Care must be taken to insure that leads do not short out during vibration. A board support, 5041-2415, may be used on the 98561-04107 cover plate or the board can be fastened to the cover plate with M3 screws or fastened via I/O connectors. These are the techniques used with current boards.
- DIO II (System) boards also have a “keep out” area along each side to keep components and pop-thrus away from the cardguides. Solder can bead up through pop-thrus, causing jamming in the cardguide. If the pop-thrus are covered with solder resist there will not be a problem. Refer to Figure 11-2 for the dimensions. Also be aware that there are strict height guidelines near the edges of the board due to the sheet metal chassis and its fasteners.
- The spacing of system boards is 30 mm center to center. The maximum component height is 19.7 mm which will allow components to slide underneath the DIO backplane if it is present in the slot above. As mentioned previously other component heights are more stringent, refer to Figure 11-2.

Piggy-backed boards may be mounted on top of system boards to extend their functionality. Figure 11-3 gives the maximum permissible dimensions for such a board.

Piggy-backed boards cannot reside in the system slot directly below the 2 or 4 slot DIO card cage unless the connector for the piggy-back is physically located in back of the DIO backplane. Piggy-back boards currently mate component side to component side using the DIN family of connectors with a surface to surface spacing of 16.67 mm. The pins on the circuit side of the piggy-backed board will interfere with the DIO backplane, not allowing the board to be fully inserted and perhaps damaging the board. (The Human Interface Board, 98562-66530, has its connector in back of the backplane.)

(Dimensions in Figure 11-3 are in inches)

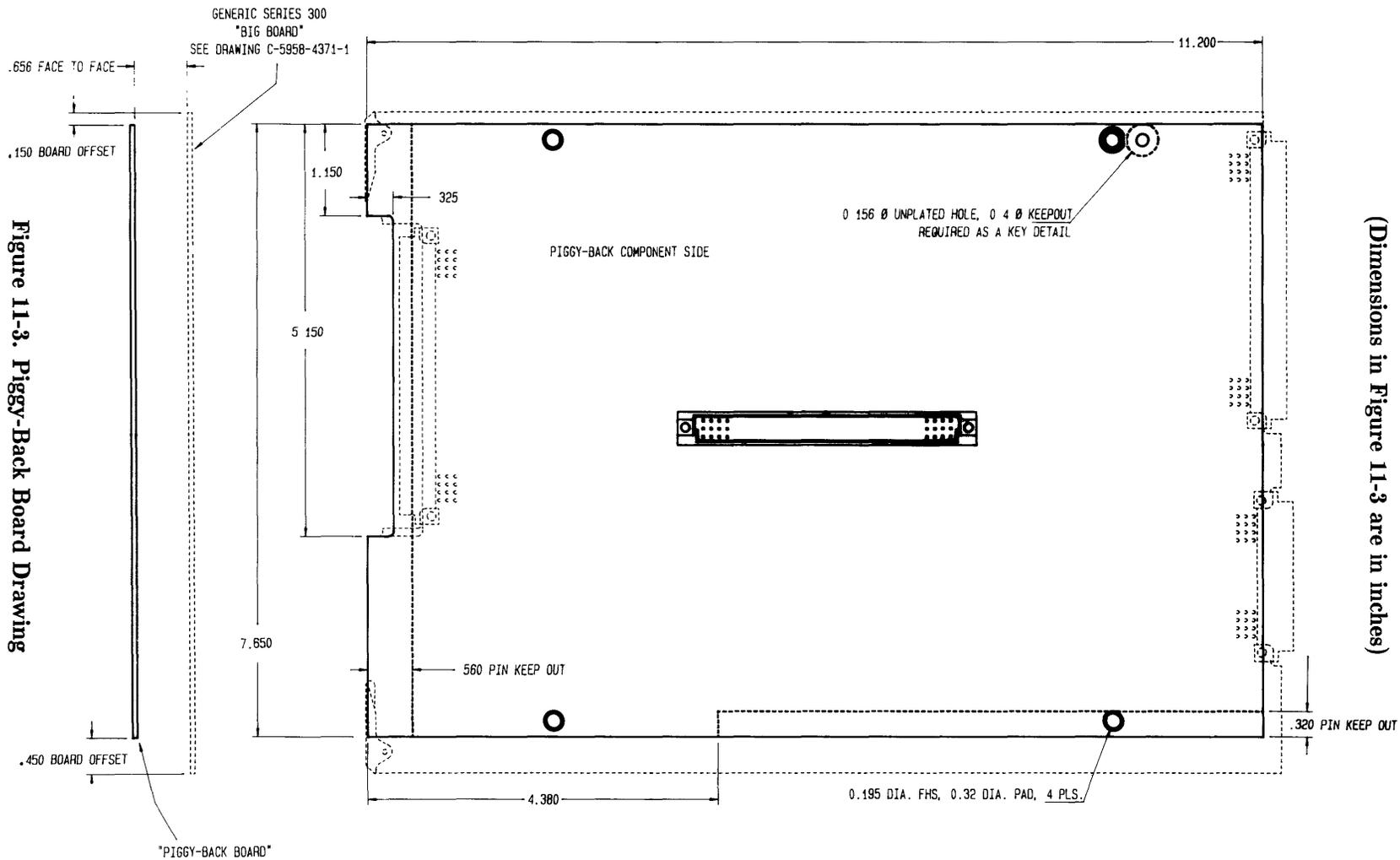


Figure 11-3. Piggy-Back Board Drawing

DIO Cards

The drawings for I/O cards, non-I/O cards and the metal cover plate are reproduced in Figures 11-4, 11-5, and 11-6 for your reference.

(Dimensions in Figure 11-4 are in inches)

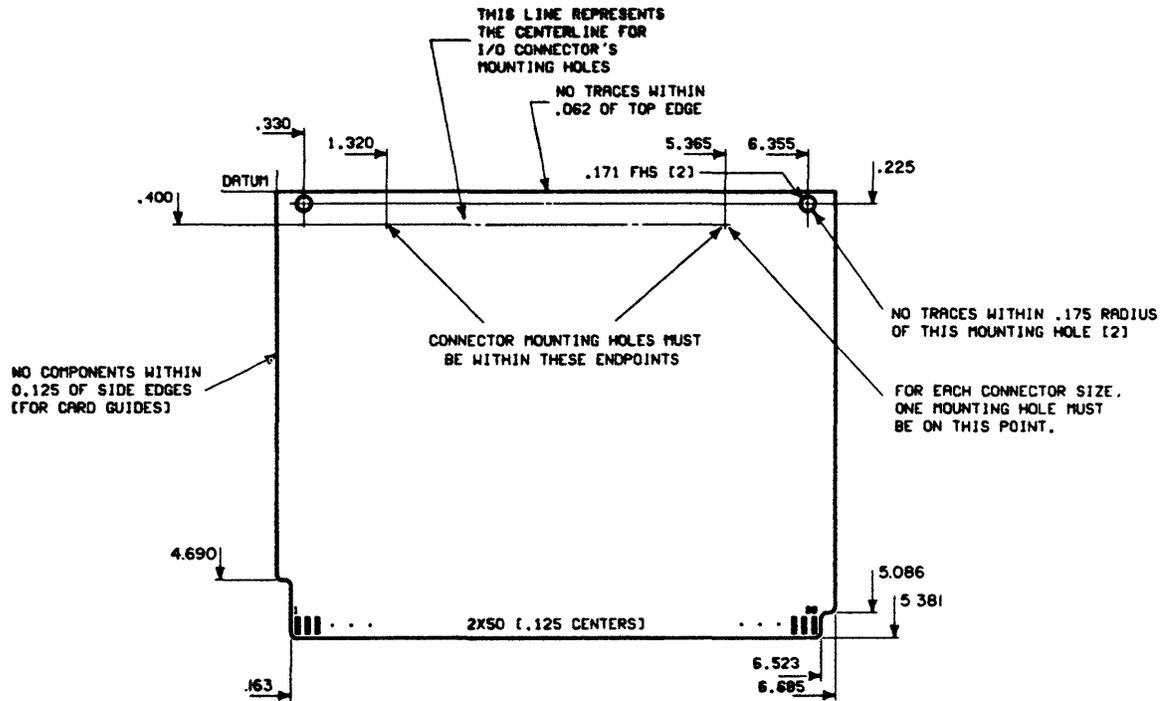


Figure 11-4. DIO Board Mechanical Diagram

(Dimensions in Figure 11-5 are in millimetres)

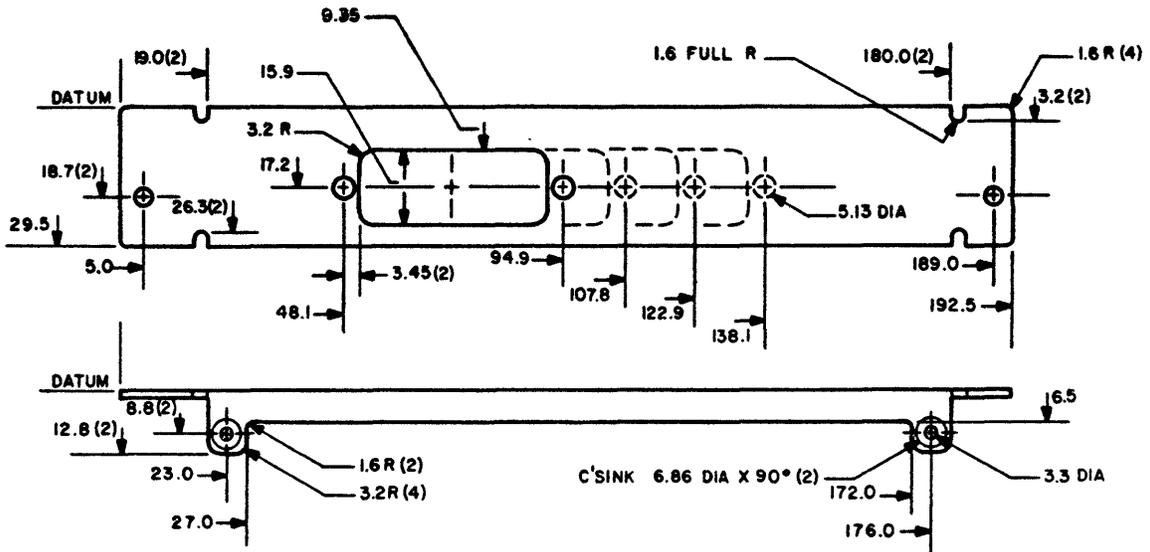


Figure 11-5. I/O Card Cover Plate

Figure 11-5 shows cutouts for 24, 36, 50 and 64 pin connectors. Note that connector openings are always left justified to a fixed reference point. Cutouts are designed to accept these Amphenol connectors:

Size	Amphenol Part Number
24 Pin	57-92245-12
36 Pin	57-92365-12
50 Pin	57-92505-12
64 Pin	57-92645-12

The following are examples of cards used in the DIO Cardcage:

I/O card: System Interface Board	98561-26531
Non I/O card: RAM Board	98257-26524
DIO coverplate:	98561-04103

The main points for DIO cards are:

- DIO I/O cards fit in every other slot of the DIO I/O backplane and are secured by thumbscrews thru a metal backplate. The slots in between hold non-I/O cards such as RAM and ROM cards.
- Non-I/O cards have a recess at the rear to allow clearance for the connector of the next-lower I/O card.
- I/O cards have a “keep-out” area in the rear where traces and parts are not allowed, to prevent them from shorting to the metal backplate.
- I/O cards also have a “keep-out” area along each side to keep traces (specifically pop-thrus) away from the cardguides. Solder can bead up through pop-thrus, causing jamming with the I/O card guide.
- For both types of cards, space must be left on either side of the board to prevent components from interfering with the cardcage guides.
- I/O connectors are left-justified and extended to the right as needed for the size of the connector.
- Note that keying (to prevent inserting a card upside down) is provided by the 2 different length notches on each side of the edge connector.

Cardcage Specifications

As mentioned above there are two cardcages; one exclusively for DIO cards, and one for DIO and/or DIO II (System) boards. The DIO card cage holds cards designed for the Series 200 (and Series 300 Models 310/320) machines. This allows them to be mechanically compatible with future products. The two cages are discussed in detail below.

DIO II (System) Cardcage Specifications

There is no single drawing showing the dimensions of the Series 300 Model 330/350 cardcage. However, this manual contains board specifications in Figure 11-1, 11-2, and 11-3 suitable for developing interface or systems boards for use in these computers. These boards are specific to the Series 300 Models 330/350 and are created for the 5001-3696 chassis. The following points are worth noting:

- While a component “keepout” area is specified on the cards, PC traces can (and do) run inside of this area. Therefore, to prevent shorts, metal cardguides **will not** be used in any future designs housing system boards.
- The center-to-center board spacing is 30 mm (1.181 inches).

DIO Cardcage Specifications

In Figure 11-6, a sketch of the Series 200 Model 26/36 backplane is shown.

(Dimensions in Figure 11-6 are in millimetres)

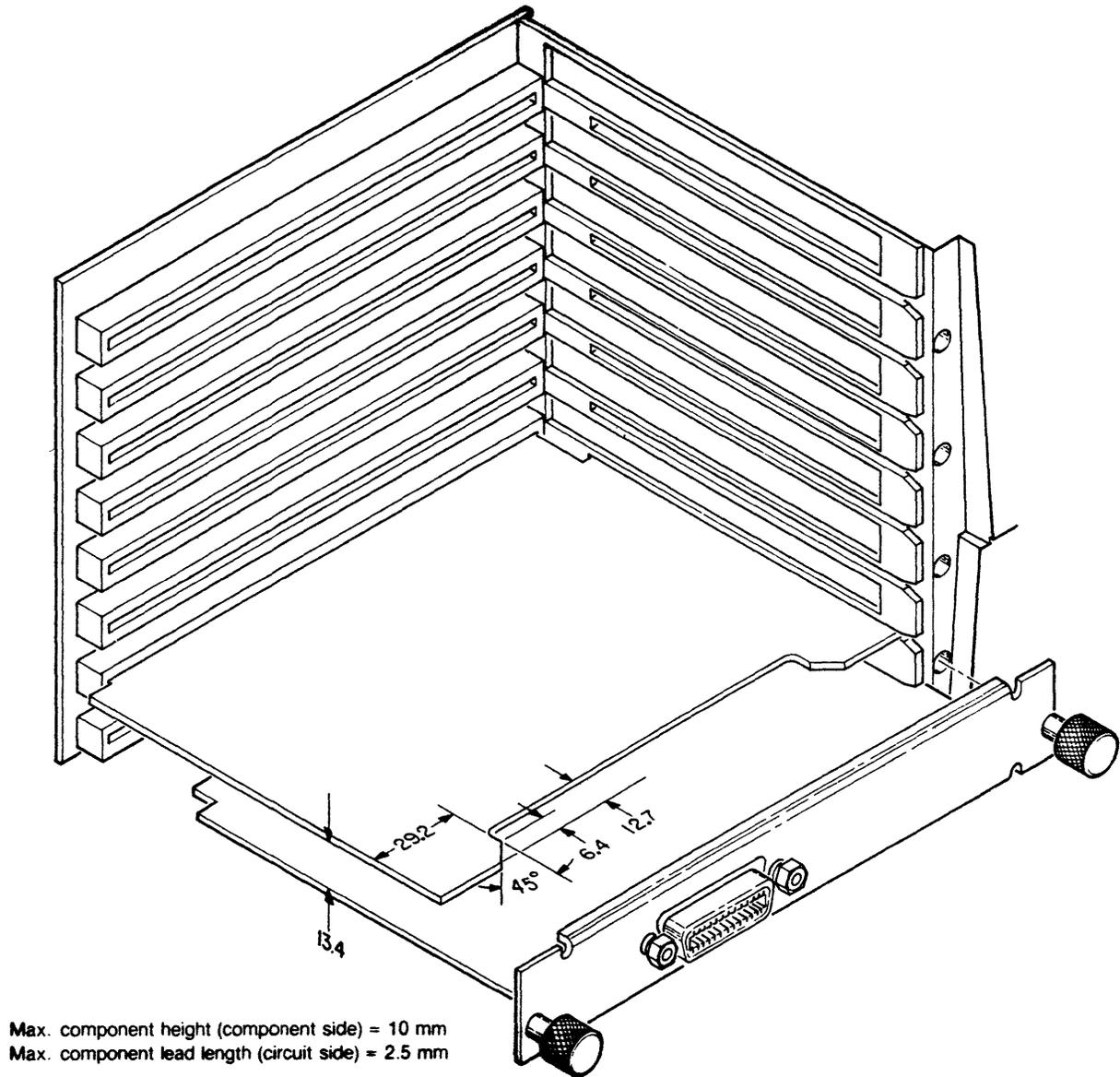


Figure 11-6. DIO Cardcage (Shows Card Clearances)

The following points are worth noting about the DIO cardcage:

- While a component “keepout” area is specified for cards, PC traces can (and do) run inside this area. Therefore, to prevent shorts, metal cardguides **will not** be used in any future designs housing DIO cards.
- Even though I/O cards only use every other connector on the backplane, new designs should continue to load all connectors. In addition to providing slots for RAM cards, ROM cards, etc., this scheme also supports “double-high” I/O cards that require both boards to connect to the bus.
- The center-to-center board spacing is 15 mm (0.59 inches). (I/O boards can be spaced 30 mm center-to-center).

Minimizing RFI

To minimize EMI problems, the following rules should be followed:

- PC boards should be a minimum of 4 layers, with planes 2 and 3 reserved for power and ground, respectively. Boards greater than 4 layers should maintain power and ground on the middle layers so that a good high frequency bypass capacitor is formed between the power and ground planes. This minimizes the current loop area.
- Even though it has been suggested by other sources, that power and ground layers should be placed nearer the outer layers to act as shielding for inner layer traces, it has generally been found that placing power and ground on the middle layers is more effective in minimizing RFI.
- I/O cards must contain a sheet metal cover plate with thumbscrews to securely mount the board in the computer and provide good electrical connection to safety ground.
- Grounding of I/O cable housings or shields is done through the I/O cover plate to the mainframe rear panel. Appropriate techniques must be used to ensure solid contact between the connector housing and the cover plate. Also, the thumbscrews must provide good contact between the cover plate and the rear panel. If the shield ground appears on a pin of the connector (as with HP-IB), a generously-sized PC trace must go between the connector pin to one of the screw pads which secure the cover plate to the PC board.

PC Layout Rules

The following PC layout rules should be met:

- Each card must limit loading on the bus lines. Refer to Chapter 10, *Electrical Specifications* for clarification.
- The PC board trace length for the bus lines should be as short as possible and no more than 3 inches. Where possible, these signals should be isolated from the ground and power planes to minimize capacitance. Therefore, for 6-8 layer boards containing power and ground on the middle layers, the bus signals should be in the outer layers.
- If the I/O cable connector contains a drain or shield wire, it should be connected by a generously-sized trace to the screw which secures the I/O backplate to the PC board. The connection to safety ground is then completed by the backplate, thumbscrew and rear panel assembly. The drain or shield wire should not be connected to logic ground.
- PC boards should be a minimum of four layers, with planes two and three being power and ground, respectively. For greater than four layers, the middle layers should be power and ground for a good RF bypass.
- IC's are mounted parallel to the connector with pin one of the IC oriented to the lower left when viewing the board from the component side with connector pin one also oriented to the lower left. NOTE: This is a recommendation since some devices (that is, the HP 98629 card) already violate this rule.
- For DIO cards the maximum component height is 10 mm (0.39 inches) to prevent components from shorting to the leads of the next higher board. As discussed previously, the board center-to-center spacing for DIO cards is 15 mm (0.59 inches). The maximum component height varies for system boards and Figure 11-2 should be referenced for exact locations for component height restrictions.
- The DIO PC edge connector is a standard S100 connector with 100 pins on 0.125" centers. Furthermore, the edge connector finger width is 0.060". This dimension was carefully chosen and ensures that adjacent traces won't short in the connector. The DIO II (System) board connectors are of the DIN family of connectors. On the DIO II (system) board a 96 pin right angle male is the DIO connector and the system bus connector, a 48 pin right angle female is used for DIO II specific signals. The DIO PC edge connector (system bus) and DIO II connectors are optional.

DIO II Pin Assignment

The pin assignments for the two boards (three connectors) are shown below. The connectors are referenced with respect to which board they are mounted on.

DIO II (System) Boards

There are two connectors on the DIO II boards that connect to the motherboard. Connector J1 is electrically connected to the DIO backplane (except for seven signals) and consists of signals listed in Table 11-1. The J2 connector is only available to DIO II boards and is not available for use on the DIO backplane. The J1 and J2 connectors are shown in Tables 11-1 and 11-2.

Table 11-1 System Board Connector J1 Pinout

PIN NO.	ROW		
	A	B	C
1	+12	GND	-12V
2	+12	+5V	GND
3	+5V	BA23	BD15
4	$\overline{\text{HALT}}$	BA22	BD14
5	$\overline{\text{BUDS}}$	BA21	BD13
6	GND	BA20	BD12
7	$\overline{\text{BLDS}}$	BA19	BD11
8	$\overline{\text{BAS24}}$	BA18	BD10
9	GND	GND	BD9
10	BR/ $\overline{\text{W}}$	BA17	BD8
11	$\overline{\text{ENDT}}$	BA16	+5V
12	GND	BA15	BD7
13	$\overline{\text{DTACK16}}$	BA14	BD6
14	$\overline{\text{BG1}}$	BA13	BD5
15	GND	BA12	BD4
16	$\overline{\text{DONE}}$	BA11	BD3
17	$\overline{\text{DMARDY}}$	BA10	BD2
18	GND	+5V	BD1
19	$\overline{\text{DMAR1}}$	BA9	BD0
20	$\overline{\text{DMAR0}}$	BA8	$\overline{\text{BR}}$
21	GND	BA7	+5V
22	$\overline{\text{DMACK1}}$	BA6	$\overline{\text{BG}}$
23	$\overline{\text{DMACK0}}$	BA5	$\overline{\text{BGACK}}$
24	GND	BA4	$\overline{\text{FOLD}}$
25	$\overline{\text{BERR}}$	BA3	$\overline{\text{BDRV}}$
26	$\overline{\text{IR7}}$	BA2	$\overline{\text{IR6}}$
27	GND	BA1	$\overline{\text{IR5}}$
28	PSPARE	BFC0	$\overline{\text{IR4}}$
29	SPARE 1	BFC1	$\overline{\text{IR3}}$
30	GND	BFC2	$\overline{\text{IR2}}$
31	$\overline{\text{BG2}}$	+5V	$\overline{\text{IR1}}$
32	$\overline{\text{RESET}}$	+5V	$\overline{\text{PURESET}}$

Only the processor can use PSPARE.

The J2 connector is **NOT** electrically connected to the DIO backplane and consists of signals defined only in the DIO II bus specifications.

Table 11-2 System Board Connector J2 Pinout

PIN NO.	ROW		
	A	B	C
1	SPARE 6	XD15	SPARE 5
2	GND	XD13	GND
3	BA31	XD11	SPARE 4
4	BA30	XD9	XD14
5	BA29	XD7	XD12
6	BA28	XD5	XD10
7	GND	XD3	XD8
8	BA27	XD1	XD6
9	BA26	GND	XD4
10	BA25	$\overline{\text{BLK}}$	XD2
11	BA24	$\overline{\text{BAS32}}$	XD0
12	$\overline{\text{IACK32}}$	$\overline{\text{XBG1}}$	GND
13	$\overline{\text{VECTOR32}}$	$\overline{\text{ADACK}}$	$\overline{\text{DSACK32}}$
14	GND	$\overline{\text{XBG2}}$	$\overline{\text{DSACK16}}$
15	$\overline{\text{LWORD}}$	GND	GND
16	$\overline{\text{RMC}}$	$\overline{\text{XBG3}}$	SPARE 3

DIO Cards

Pin assignments for the DIO Bus backplane (Series 200 and Series 300 Models 310/320) are shown in Table 11-3. Odd pins are on the component side, even pins on the circuit side. Relative to viewing the board from the component side with the connector pointing down, the pins are numbered from left to right. The following considerations are worth noting:

- The 2 DGND lines run from the I/O backplane thru the motherboard (connecting to the OEM slot) to the power supply. They are not bused to each connector on the 9826/36 motherboard. The OEM slot is an empty slot internal to the 9826/36 for potential use by HP OEMs. The Series 300 ties these lines directly to the ground plane.
- In the Series 200 (and Series 300 Models 310/320), spare lines x0, x1 and x2 (pins 5, 27 and 28) are bused from the I/O backplane to the CPU, Disc Controller and Expansion slots. Series 300 Models 330 and 350 do not bus these signals to the motherboard along with four others: $\overline{\text{VECTOR}}$, $\overline{\text{IACK}}$, $\overline{\text{IMA}}$ or $\overline{\text{BG3}}$.
- The 2 spare lines which were formerly +15V and -15V (pins 97 and 98) are bused from the I/O backplane connector to the DIO II backplane on the Series 300 models 330 and 350, and to the CPU OEM and Disc Controller slots in the Series 200 and Series 300 Models 310 and 320.
- The signals $\overline{\text{IMA}}$ and $\overline{\text{FOLD}}$ (pins 41 and 42) are bused to all I/O slots on the DIO I/O backplane in Series 200 products and Series 300 Models 310 and 320. However, they are NOT connected to the DIO I/O backplane PC edge connector. The DIO I/O backplane female edge connector mounted on the motherboard has pins 41 and 42 tied to +5V.
- The internal OEM slot connector in the 9826/36 is the same as the I/O backplane connector except pins 41 and 42 are no-connects. Because the OEM slot does not have the FOLD signal, neither the DMA Controller nor the Bus Expander card are electrically compatible with the OEM slot. Mechanically, the OEM slot accepts a board that is larger than the standard I/O board.
- Spare pins MUST remain open until they are defined by Hewlett-Packard in future specification changes. They are **NOT** subject to designer definition.
- The 9826 and 9836 reference Pin 29 as $\overline{\text{BDRV}}$; the 9816 references it as $\overline{\text{BMON}}$.

The DIO Bus pinouts are shown in Table 11-3. These are provided on the DIO I/O backplane only. Connectors internal to the 9826/36 do NOT have the same pinouts.

Table 11-3 DIO PC Edge Connector Pinout

PIN NO.	ROW	PIN NO.	ROW
COMPONENT SIDE		CIRCUIT SIDE	
1	$\overline{\text{DMAR0}}$	2	$\overline{\text{DMAR1}}$
3	$\overline{\text{DMACK0}}$	4	$\overline{\text{DMACK}}$
5	spare x0	6	IR7'
7	$\overline{\text{IR2}}$	8	IR1'
9	$\overline{\text{DMARDY}}$	10	$\overline{\text{BG1}}$
11	$\overline{\text{BG2}}$	12	$\overline{\text{BG3}}$
13	GND	14	GND
15	$\overline{\text{IR4}}$	16	IR3'
17	$\overline{\text{IR6}}$	18	IR5'
19	$\overline{\text{VECTOR}}$	20	$\overline{\text{IACK}}$
21	GND	22	GND
23	$\overline{\text{BG}}$	24	$\overline{\text{BR}}$
25	$\overline{\text{DONE}}$	26	$\overline{\text{BGACK}}$
27	spare x1	28	spare x2
29	$\overline{\text{BDRV}}$	30	$\overline{\text{ENDT}}$
31	BFC0	32	BFC1
33	BFC2	34	$\overline{\text{DTACK16}}$
35	GND	36	GND
37	$\overline{\text{RESET}}$	38	$\overline{\text{BERR}}$
39	GND	40	GND
41	$\overline{\text{IMA}}$	42	$\overline{\text{FOLD}}$
43	$\overline{\text{BLDS}}$	44	$\overline{\text{BUDS}}$
45	BR/ $\overline{\text{W}}$	46	$\overline{\text{BAS24}}$
47	GND	48	GND
49	$\overline{\text{HALT}}$	50	BA1
51	BA2	52	BA3
53	BA4	54	BA5
55	BA6	56	BA7
continued on the next page			

Table 11-3 DIO PC Edge Connector Pinout (Continued)

PIN NO.	ROW	PIN NO.	ROW
COMPONENT SIDE		CIRCUIT SIDE	
57	BA8	58	BA9
59	BA10	60	BA11
61	GND	62	GND
63	BA12	64	BA13
65	BA14	66	BA15
67	BA16	68	BA17
69	BA18	70	BA19
71	BA20	72	BA21
73	BA22	74	BA23
75	GND	76	GND
77	BD0	78	BD1
79	BD2	80	BD3
81	BD4	82	BD5
83	BD6	84	BD7
85	+5V	86	+5V
87	BD8	88	BD9
89	BD10	90	BD11
91	BD12	92	BD13
93	BD14	94	BD15
95	DGND	96	DGND
97	SPARE 1	98	PSPARE
99	-12V	100	+12V
spare xN – not connected to system board slot			
pins 12,19,20,41- not connected to system board slot			

Operation In The HP 9888A Bus Expander

12

The HP 9888A Bus Expander is produced for the Series 200 computers and works to a limited extent with the Series 300 computers. Devices for the bus expander are required to meet the DIO electrical and mechanical specifications, **NOT** those of the DIO II bus. The bus expander employs delay lines and latches to ensure that all DIO Bus timing requirements are met (except as noted in *Operating Limitations in the Bus Expander* in this chapter). This chapter discusses features of the bus expander and several limitations affecting operation of DIO Bus devices in the expander.

Note

In the Series 300 Models 310 and 320 computers, DIO RAM is not allowed in the HP 9888 Expander. In the Series 300 Models 330 and 350, and future DIO II products, DIO I/O cards that support DMA are not allowed in the expander. Specifically the HP 9888 Expander does not allow any DMA transfers through it to DIO II machines.

Features Of The Bus Expander

The features of the bus expander are:

1. The bus expander plugs into a DIO I/O slot in the computer. Up to 4 bus expanders can be plugged into a computer. An expander may **NOT** be attached to another expander.
2. The bus expander is totally self-powered.
3. The bus expander has 16 connectors which will support 8 DIO I/O cards and 8 DIO accessory cards or 16 DIO accessory cards.
4. A 5.2 foot cable connects the bus expander to the computer. Signals are buffered at each end of this cable (on the board that plugs into the computer and on the board in the expander). The DIO IMA (I'M Addressed) signal is used to turn these buffers around if the addressed card is in the expander.
5. I/O cards that implement DMA (that is, the HP 98622A General Purpose I/O card) can operate in the bus expander in the Series 200, and Series 300 Models 310 and 320. It can **NOT** operate in the bus expander in DIO II products such as the Series 300 Models 330 and 350. If a Series 200 product is used with the expander and both the DIO memory card and the I/O card involved in a DMA transaction are in the same expander, then the expander itself performs any necessary data folding operation in response to the DMA Controller's FOLD signal (if the DMA Controller did the folding, signal delays would be excessive).

Operating Limitations In The Bus Expander

In designing DIO Bus devices, designers should be aware of certain limitations when operating in the HP 9888A Bus Expander:

1. Bus masters cannot operate in the bus expander. There is no provision for “turning around” certain signals such as the address bus. Thus, the HP 98620 DMA Controller must reside in the mainframe.
2. ROM boards are not guaranteed to work in the bus expander due to automatic data transfer acknowledge (Auto DTACKing). With Auto DTACKing, ROM cards are required to have data present within a certain length of time. Because of signal delays between the bus expander and the mainframe, data setup time prior to the Auto DTACK is not guaranteed.
3. DIO RAM boards will operate in the bus expander in Series 200 products. However, DIO RAM boards typically require 6 cycles to access data as opposed to 5-cycle accesses for boards installed in the Series 200 mainframe. This is due to signal delays. Software being executed from the bus expander will thus run proportionately slower and timing loops will be altered.
4. Interrupt cards which implement external vectored interrupts will not work in the bus expander for 2 reasons:
 - a. The VECTOR signal from the interrupting card will not (worst-case) reach the processor board in time to inhibit auto vectoring.
 - b. Even if the VECTOR signal reaches the processor board in time, the interrupt vector cannot be read by the processor board. This is due to the fact that BAS is not generated during an interrupt vector cycle. Without BAS, the card does not generate IMA, so the bus expander’s data bus buffers do not “turn around”.
5. The $\overline{\text{BDRV}}$ signal discussed in the Chapter 9, *DIO II Bus Utilities* is not supported in the bus expander. Cards which use $\overline{\text{BDRV}}$ must be installed in the mainframe.
6. The $\overline{\text{BERR}}$ signal is an input from the bus expander to the mainframe. A $\overline{\text{BERR}}$ signal generated by the mainframe (that is, the CPU board) cannot be seen by an I/O card.
7. The 9826/36 Powerfail option (which is installed internally in the mainframe) is not supported with the bus expander. The bus expander resets the computer when the power fails and again when power returns. This will destroy any data or programs in memory. For correct operation, the expander must be turned on before the computer is powered up and not powered down while the computer is operating.

Although many details covering Bus Slave design are covered in other chapters, the key requirements are summarized below. This chapter covers primarily External I/O cards (for example: the GPIO interface). Example circuits are not provided and the reader is strongly encouraged to review the design of existing cards before attempting a new design.

External I/O Card Design Guidelines

The following design guidelines are for external I/O cards only.

1. I/O cards may be either 8 bit, 16 bit or 32-bit devices. 8 bit devices should reside on the lower (odd) byte of the 16 bit data bus.
2. Designing I/O cards that can perform DMA transfers is left to the option of the designer. I/O cards that implement DMA can optionally use DONE from the DMA Controller to determine when the last transfer is occurring.
3. While a card is enabled for DMA, it must still respond to normal bus cycles so Status and Control registers can be accessed.
4. DIO I/O cards must provide 5 Select Code switches to provide Select Codes up to 31. DIO II cards have two option on switches depending on their address map location. Refer to Chapter 3, *DIO II Memory Map* for more information.
5. DIO I/O cards must implement Registers 1 and 3 as defined in Chapter 3, *DIO II Memory Map*. DIO II devices in the “Uncached Space must implement Register 101. DIO II devices that reside in the Cached Space (other than RAM) must reside in DIO space with Registers 1 and 3 defined. They must also have a pointer to DIO II Uncached Space so the software knows where to find the rest of the device. The pointer register should exist at an offset of 8 from the base address. In DIO Space, Register 5 may be implemented if an Extension ID is necessary. Additional registers and bits may be defined as needed.
6. Implementing interrupt capability is left to the option of the designer. However, it is **STRONGLY** recommended that interrupt capability be included. An I/O card without interrupts will not work well with UNIX. If the operating system has to poll an I/O card to determine its need for servicing, extended (for example: 1 second) dropouts may occur. Because of the previously discussed problems with vectored interrupts, they should not be used.
7. It is highly recommended that I/O cards provide switches to select interrupt levels 3 to 6. However, other options are possible. Such as:
 - a. Make the interrupt level programmable (for example: use undefined bits in the card’s CONTROL REGISTER to permit programming of the interrupt level bits).

- b. Permit interrupt levels to be set (either via switches or software) over a broader level than from 3 to 6 (such as, from 1 to 7). This is NOT recommended, however, since special care is required in system configuration to avoid conflicts. For example: the 09826-66562 floppy disc control board used in certain Series 200 products is hardwired to interrupt level 2 and drives the interrupt signal with an LS04 (that is: it is not an open collector device). An I/O card on interrupt level 2 would fight with this LS04. Designers that want to implement interrupts outside of the range from 3 to 6 should consult with Hewlett-Packard's hardware and software support personnel to evaluate potential problems.
8. If I/O cards are designed containing more than one interface (for example: an RS-232 card containing four channels), the card can be designed in two ways:
 - a. If it is desired that existing software work without modification, then the card must appear exactly like multiple single-interface cards (that is, each interface should occupy its own 64K address slot).
 - b. If the design goals do not permit the hardware to be identical with existing I/O cards, then the software drivers will have to be re-written.
9. For testability reasons, it is advised that no write-only registers be implemented. All registers should be read/write. Also, no read-then-clear registers should be used.

External DIO II Design Example

Since all cards will be different in complexity and interfacing needs, design examples for DIO II cards are not given. It is recommended that designers get existing (recent) DIO II designs for examples of interfacing techniques. For information on DIO design examples refer to the *HP 9000 Series 200/300 Computers Access Development Guide* (Manual Reorder No. 09800-90011).

Note

Adequate test data must be available to demonstrate that a device complies with regulations. Whenever a regulatory agencies or other parties require it, the test data must be produced for inspection. This includes test data demonstrating compliance with safety regulations. It is the responsibility of the designer and manufacturer of the device to ensure these regulations are met and the data is available.

Because many cards may be used with different mainframes, operating systems, etc. a thorough card qualification program is absolutely necessary to ensure that all hardware, software and safety issues are resolved. This is an area that is constantly changing as new hardware and software is released, as regulatory requirements change, and as our interpretation of these requirements change. Designers are responsible for ensuring that their products are properly qualified according to the latest operating environments and regulations. It is also the responsibility of the designer and manufacturer of cards and accessories to adequately test new designs so customers are ensured of reliable and correct operation.

In this chapter information is presented concerning portions of Hewlett-Packard's quality assurance program, as examples of the card qualification process.

Qualification of new cards can be broken down into three areas:

- Software Qualification
- Hardware Qualification
- Safety Compliance

Software Qualification

The key concern with software is its reliability. Sufficient testing should be performed to ensure that the card operates with the different operating systems. When operating systems are revised, new operating systems are released, or changes are made to any card which affects its operation, additional software testing should be performed to ensure compatibility between software and card.

Hardware Qualification

Hardware testing can be divided into 2 areas:

- Configuration Testing
- Environmental Testing

Configuration Testing involves testing the new cards with different mainframe configurations. Environmental testing involves testing the card in different operating environments.

Configuration Testing

New cards must be tested with the mainframe configurations that the card will be used with. Likewise, new mainframes must be tested to ensure I/O card configurations are compatible. New cards should be tried in as many different mainframes as possible to determine if design errors may be found under some unsupported configurations which may be used in the future. As many different configurations should be tried as possible. This is **not** to say that a new card should be tested with every possible combination, but a good matrix of testing should be set up to reduce possible problems. It is up to the designer to determine what this matrix should be.

I/O cards should also be tested in the HP 9888A Bus Expander attached to a subset of mainframes/CPU boards.

Environmental Testing

All new designs should be tested sufficiently to ensure that the card will operate reliably over its entire operating range. It should also be tested for reliability under the most severe conditions that can be expected during normal operation.

In performing environmental tests at Hewlett-Packard's Fort Collins Systems Division, a subset of possible configurations are selected for testing in the following areas:

- Test-Analyze-Fix (TAF) testing
- Stress-Life (STRIFE) testing
- Class B testing.

TAF testing

This is the testing of a small sample of prototypes involving high levels of electrical, thermal, and mechanical stress for short periods of time. Often these stresses are applied until failure occurs to identify weak points in the design.

STRIFE Testing

STRIFE testing consists of applying many thermal and vibration tests to a large sample of production devices. While these tests also involve forcing failures, the emphasis is on determining failure modes and assessing margins.

All sample devices are subjected to extensive thermal cycling, beyond the normal specified operating range, to ensure that an ample margin of safety exists. The duration of each cycle is selected so that the units reach thermal equilibrium during the dwell portion of each cycle.

Vibration tests are performed with sufficient acceleration and duration to ensure that all components can safely withstand any shock or vibration encountered during normal use.

Class B Testing

Class B testing includes RFI testing. The following guidelines for RFI testing of new I/O cards are used by Hewlett-Packard at Fort Collins.

- VDE level B-2 and FCC Class B RFI specifications must be met.
- The new card must be qualified with each product (or groups of products) that may include the card and is licensed separately. I/O cards must currently qualify with the following Series 200/300 Computer Models.
 - HP 9816A
 - HP 9826A or HP 9836A or HP 9836C (all under one license)
 - HP 9920A
 - HP 9817
 - HP 9837
 - Series 300 Model 310
 - Series 300 Model 320
 - Series 300 Model 330
 - Series 300 Model 350
- A system test is required. The system must consist of a mainframe, an input device (for example: mass memory) and an output device (for example: a printer). Where it makes sense, the new card may be used to interface to the input or output device.
- The system should actively exercise the new card during testing.
- Where a CRT is part of the system, an alternating pattern of dots on the CRT should be displayed.

Safety Compliance

I/O cards should be designed to meet the normal UL/CSA/IEC safety requirements. Additional requirements may also apply, depending on which country has jurisdiction over the equipment. In addition, the following guidelines have been developed within the Hewlett-Packard Fort Collins Computer Group.

1. Ground current carrying capacity: The conductive path between the I/O cable shield and the safety ground pin of the power receptacle should be capable of carrying 30 amps for 120 seconds. This can also be expressed in resistance for the following 2 cases:
 - a. For cables less than 4 meters, the DC resistance between the end of the cable and the safety ground pin of the power receptacle should be less than 100 milliohms, measured after 120 seconds.
 - b. For cable lengths over 4 meters, the DC resistance between the I/O connector on the card itself and the safety ground pin of the power receptacle should be less than 100 milliohms, measured after 120 seconds.
2. Additional clarification is needed for the above specifications:
 - a. Hewlett-Packard's product assurance guidelines state that the mechanical connection between the connector shroud and the connector on the I/O coverplate cannot be relied upon to carry this current. Therefore, the I/O connector pin(s) alone must be capable of carrying this current. During testing, the connector shroud must be isolated from the connector on the I/O coverplate. Since the safety ground path from the I/O card to the power receptacle is via the thumbscrews (and not the PC edge connector), it is implied that the thumbscrew connection to the rear panel is a reliable connection.
 - b. Specifications that refer to measuring resistance between an I/O card and the power receptacle are obviously mainframe dependent. To achieve a mainframe independent specification for the designer, Hewlett-Packard's product assurance guidelines state that the designer may substitute "I/O coverplate at the thumbscrews" for "safety ground pin of the power receptacle" in specifications 1a and 1b above, while substituting 70 milliohms in lieu of 100 milliohms.
3. Because the +5V supply in the Series 200 and Series 300 computers has more than an 8 amp fault current capability, a fuse is required on each board that connects to this supply. Refer to Chapter 10, *Electrical Specifications, On Card Fuse Specification* section for the recommended fuse. Even though current Series 200 and Series 300 mainframes cannot supply 8 amps on the +12V and - 12V supply, fuses are still required on boards that connect to these supplies. This ensures compliance for future mainframes that are capable of supplying 8 amps.

CAUTION

An improperly fused card is considered to be a misuse of equipment and may result in a hazard to operators and/or installers. It may also cause equipment damage. Any accessory that does not meet fusing requirements must not be installed in any computer system that is supported by HP field support organizations.

4. Accessory cover plates must be adequately secured to the computer chassis with thumb-screws and any I/O connectors and cables must be adequately grounded to the coverplate.

WARNING

I/O cables that do not meet the above stated grounding requirements presents a potential shock hazard to equipment installers and users. They cannot be used in any system that is supported by HP field support organizations.

:

MANUAL COMMENT SHEET

DIO II

Accessory Development Guide

HP 9000 Series 300 Computers

Manual Reorder No. 98562-90010

Name: _____

Company: _____

Address: _____

Phone No: _____

Please note the latest printing date from the Printing History (page iii) of this manual and any applicable update(s);
so we know which material you are commenting on _____.



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 37 LOVELAND, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

Hewlett-Packard Company
Attn: Customer Documentation
3404 East Harmony Road
Fort Collins, Colorado 80525





HP Part Number
98562-90010

Microfiche No. 98562-99010
Printed in U.S.A. 2/87



98562-90010

For Internal Use Only