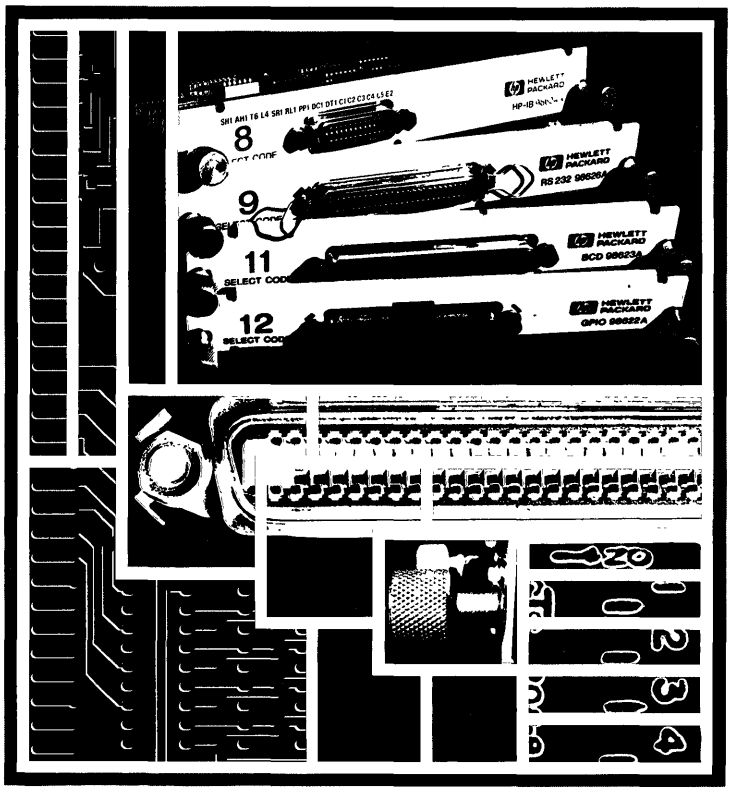
HP 98630
Breadboard Interface
Installation and Programming



HEWLETT PACKARD

Warranty Statement

The HP 98630 Breadboard Interface is warranted against defects in materials and workmanship for a period of ninety (90) days. If HP receives notice of such defects during the warranty period, HP shall, at its option, either repair or replace the defective product.

If the customer or any third party adds to or modifies the product as supplied by HP, then the product is considered a non-qualified device. Any damage caused by a non-qualified device is excluded from the warranty. However, HP will repair or replace only the HP portion on a time and materials basis. Equipment returned to HP for repair must be shipped freight prepaid.

HP 98630
Breadboard Interface
Installation and Programming

Manual Part No. 98630-90000

Hewlett-Packard Desktop Computer Division
3404 East Harmony Road, Fort Collins, Colorado 80525

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.

March 1983...First Edition

Table of Contents

Chapter 1
General Description

The HP 98630 Breadboard Interface allows experienced hardware designers to design custom interfaces. The interface consists of a printed-circuit (PC) board with a minimum of buffer components. These buffers connect user-supplied circuits to HP Series 200 Computers through the computers' backplanes.

There are approximately 15 square inches of unused space on the board intended for use as a prototyping area. This area, which is tinned for easy soldering, has a hole pattern on 100-mil centers for conventional integrated-circuit (IC) packages as well as discrete components.

Applications Supported

The Breadboard Interface provides circuitry provides access to many of the signals on the Series 200 Computer backplane. The circuitry provided by this interface can be used in the following applications:

- data-transfer of bytes and words to and from peripheral devices
- memory registers on the board
- control lines to peripheral devices, and status lines from peripherals
- interrupt requests and acknowledgement
- direct-memory access operations

What This Manual Contains

This manual shows how to use the supplied components and unused space for your own circuits. The rest of this manual consists of the following chapters:

- Chapter 2 describes the interface from a mechanical standpoint. The dimensions of both PC board and rear-panel coverplate are given. Connector dimensions and vertical clearance are also provided.

- Chapter 3 describes how the interface works with HP Series 200 backplane signals. As you read this chapter, keep in mind that the signals on the backplane are fully discussed in the "DIO Bus" chapter of the *Pascal System Internals Documentation*, part number 09826-90074. A copy of the chapter is included with this documentation.

- Chapter 4 describes programming the Breadboard Interface in BASIC, Pascal, and MC68000 Assembler language.

Where to Begin

Before attempting to put any circuits on the PC board, you should become familiar with the physical layout of the interface, how it operates, and specific requirements of any circuitry you add to the board. Read Chapters 2 and 3 for this information.

The DIO Bus chapter fully discusses the Series 200 backplane signals; you will definitely need to refer to that document while reading this manual. You may also want to refer to the *MC68000 User's Manual*, HP part number 09826-90073, for a description of the MC68000 microprocessor signals and operation.

After you are generally familiar with the interface and how it operates, you can begin to formulate and implement design ideas. Chapter 4 provides suggestions for programming the interface.

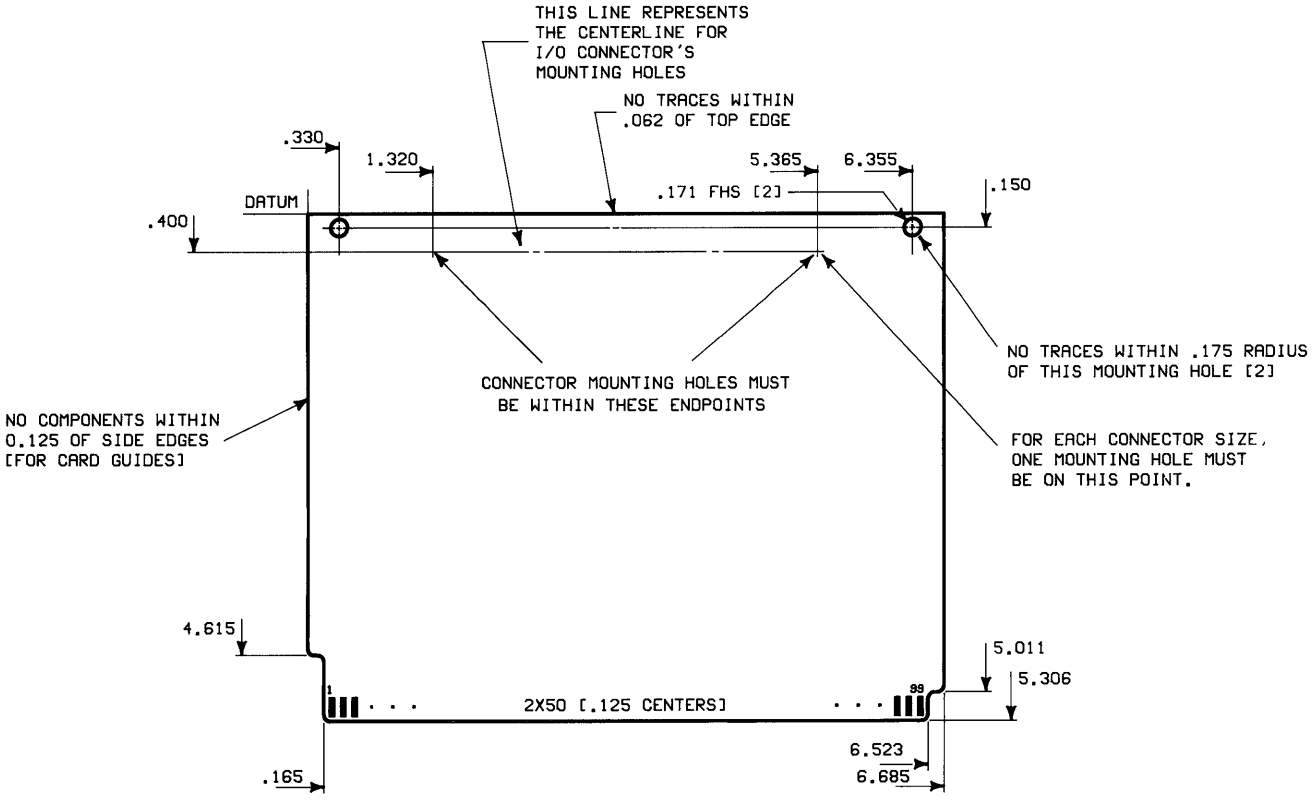
After completing your design, you will need to thoroughly test it; the DIO Bus chapter provides safety requirements and suggestions for test criteria.

Chapter 2
Mechanical Description

This chapter describes the mechanical characteristics of the Breadboard Interface. Drawings of the interface card, the card cage into which it plugs, and rear-panel coverplate are given.

The Breadboard Card

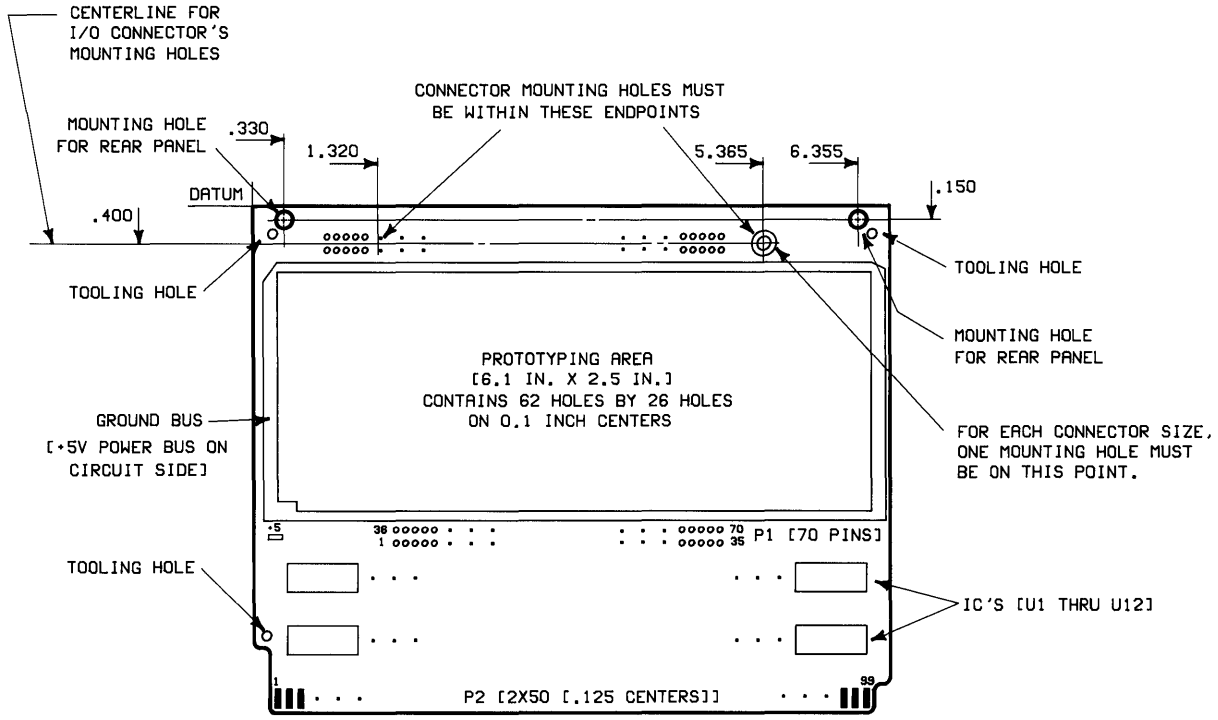
The Breadboard Interface's printed-circuit (PC) board, or card, is the same size as other input/output (I/O) cards. The following drawing shows the dimensions required of all I/O cards, as seen from the component side of the card.



Blank I/O Card

Note that the board is notched on the sides so that it cannot be inserted into the backplane connector upside-down. The rear coverplate, shown in a later drawing, is keyed for the same reason. The backplane PC-edge connector is a standard S-100 connector with 100 pins on 0.125-in centers and 0.060-in fingers.

The following drawing shows the component side of a Breadboard Interface card with relevant dimensions. At the rear of the card are pre-drilled, plated-through holes into which the pins of a connector may be inserted. One mounting hole for the connector has also been drilled in the card. The position of the second mounting hole depends on the size of connector you will use.



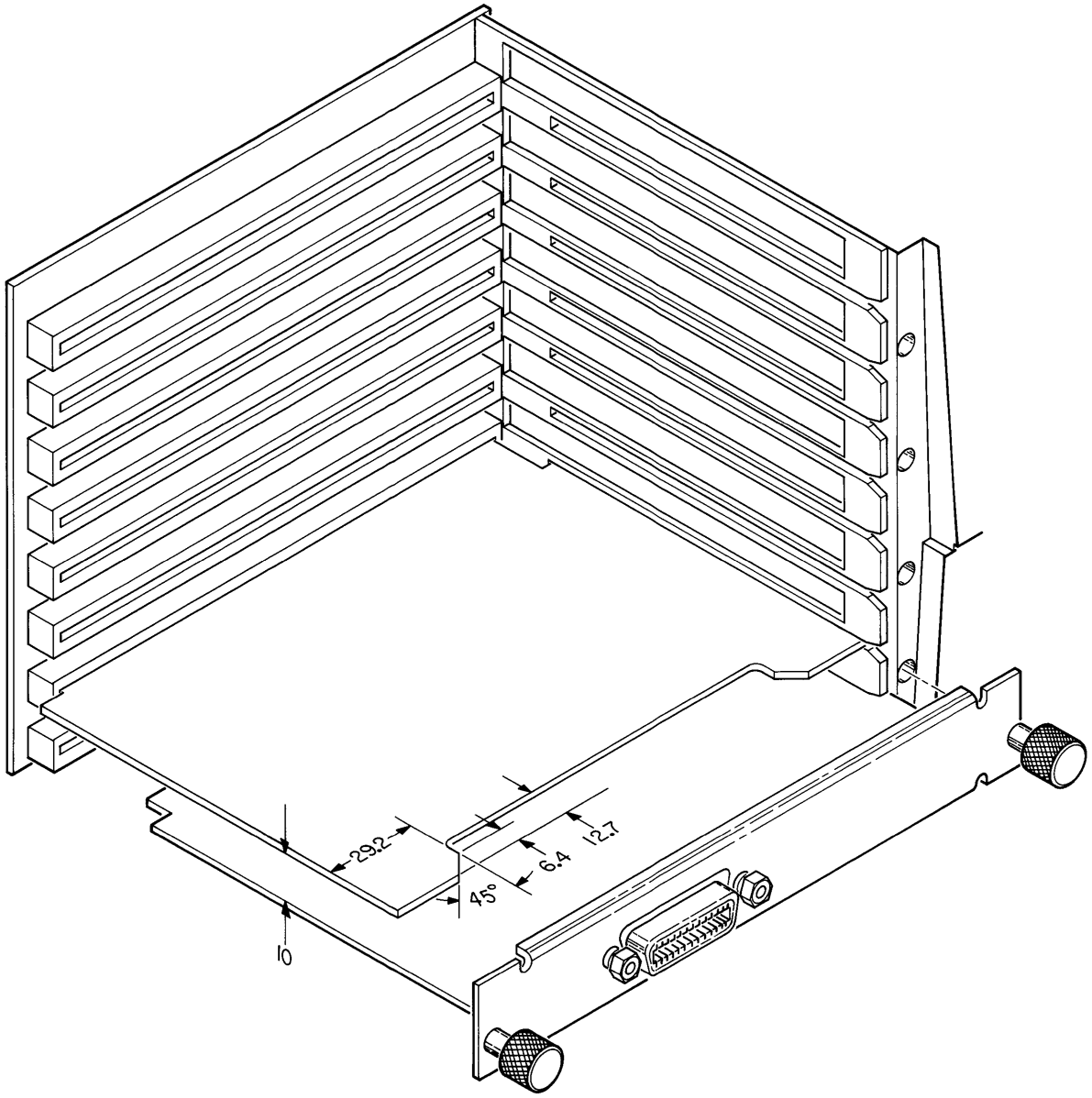
The Breadboard Interface Card

You can place prototype (or final) circuits on the center portion of the card. This area consists of a matrix of holes on 0.1-inch centers, which will accomodate standard integrated-circuit (IC) packages and sockets. There is a ground plane on the component side of the card and a power plane (+5 volts) on the circuit side. Supply voltages of +12 and -12 volts are also available. See Chapter 3, "Theory of Operation," for further details.

Note that there are certain portions of the card on which no components or traces can be placed. The area at the rear of the board is reserved for the rear-coverplate connector, if required for the application. Components cannot be placed within approximately 4 mm of the side edges of the card, because that part of the card fits into the guides in the card cage. Note also that there is a maximum height of 10 mm for any components placed on the card as well as a clearance of 3 mm on the circuit side of the board, as shown in the next section.

Card Cages

Series 200 optional interface and memory cards reside in a card cage which is entirely within the computer (or expander) case. The cards, positioned with the aid of card guides, plug into connectors on a common backplane which contains the system's Desktop-computer Input/Output (DIO) Bus.



The HP 9826/9836 Card Cage

There are eight card slots in Model 26 (9826) and 36 (9836) Computers. The Model 16 Computer only has two slots, while the HP 9920 Computer and HP 9888 Bus Expander each have 16 slots.

Types of Cards and Card Placement

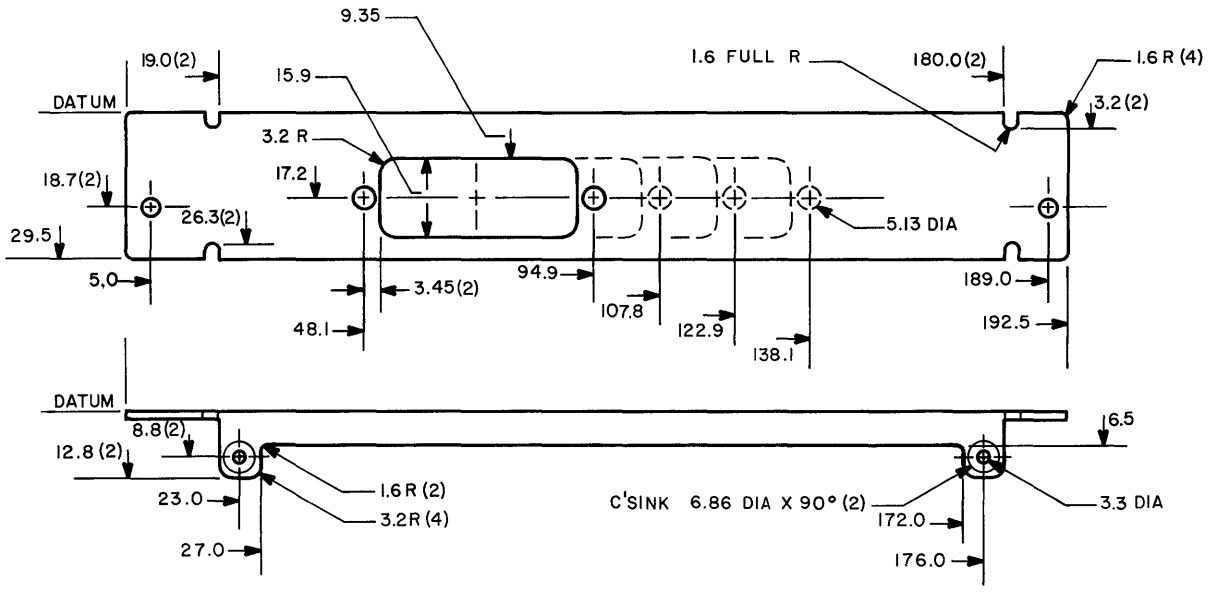
There are two types of option cards: interface (or I/O) cards and non-I/O cards. I/O cards are designated as such because they communicate with peripheral devices through a cable connected to the card's coverplate. Non-I/O cards do not communicate with peripheral devices. Non-I/O cards include such types as memory and Direct Memory Access (DMA) Controller cards.

The size and placement of the coverplate on an I/O card is such I/O cards can only be inserted in every other slot in the card cage, beginning with the bottom slot (see preceding drawing). The coverplate covers the I/O card and the non-I/O card above it.

Only four cards in an eight-card cage can be interface cards because of the rear coverplate. Non-interface cards can be placed in any slot. If eight non-interface cards are placed in the card cage, four blank coverplates should be placed over the cards to enclose the card cage and provide shielding.

I/O Card Coverplate

The following drawing shows the dimensions of the blank coverplate supplied with the HP 98630. The drawing shows the placement of different sizes of connectors which can be used with the Breadboard Interface.



I/O Card Coverplate

The drawing shows cut-outs and drilled holes for 24-, 36-, 50-, and 64-pin connectors. Amphenol manufactures connectors which can be used with the Breadboard Interface, as shown in the following table.

Connector Size	Amphenol Part Number
24-pin	57-92245-12
50-pin	57-92505-12
64-pin	57-92645-12

The coverplate also provides an electrical ground for the connector, as described in the next section.

Grounding

In order to ensure safe, reliable operation with Series 200 Computer products, *all* specifications in this document must be strictly followed when designing Bus Slave devices. Keep in mind that *you* are responsible for any circuitry that you design and use with HP products, both in terms of personal safety and proper operation with the equipment.

I/O cards must be designed to meet *all* UL, CSA, and IEC safety requirements. The connector (both shell and pins) and I/O cable must be grounded through the rear coverplate, which is electrically and mechanically connected to the chassis through the dog-bolts which secure the card in the card cage. This grounding scheme must meet the following requirements:

1. Ground-current carrying capacity: The conductive path between the I/O cable (cable shield, connector ground pins, and connector shell) and the safety ground pin of the power receptacle must be capable of carrying 30 amperes for 120 seconds. This requirement can be expressed in ohmic resistance for the following two cases:

   a. For cable lengths less than 4 metres, the dc resistance between the end of the cable and the safety ground pin of the power receptacle should be less than 100 milliohms.

   b. For cable lengths greater than 4 metres, the dc resistance between the I/O connector on the card's coverplate (cable shield, connector ground pins, and connector shell) and the safety ground pin of the power receptacle should be less than 100 milliohms.

WARNING

I/O CABLES AND CONNECTORS WHICH ARE NOT GROUNDED AS STATED ABOVE PRESENT A POTENTIAL SHOCK HAZARD TO THE USER OF THE EQUIPMENT.

---

2. Fault-current carrying capacity: A UL/CSA/IEC requirement is that any device operating from a supply capable of supplying more than 8 amps be fused. Therefore, any board plugged into the backplane **must** have a 4-amp maximum fuse in series with the +5V bus.

CAUTION

AN I/O CARD NOT EQUIPPED WITH THE PROPER FUSE IS CONSIDERED TO BE A MISUSE OF THE EQUIPMENT AND MAY RESULT IN A PERSONAL HAZARD TO THE OPERATOR AND/OR EQUIPMENT DAMAGE.

Installation

Use the following procedure to install interface card(s) in the computer.

1. Turn computer power off. If the interface is connected to a peripheral, disconnect it from the peripheral.

CAUTION

BEFORE INSTALLING ANY INTERFACE IN THE COMPUTER, BE SURE THAT POWER IS OFF. FAILURE TO DO SO MAY DAMAGE THE COMPUTER, VOIDING ANY WARRANTY IN EFFECT.

---

2. Slide the card into one of the slots in the card cage. Remember that I/O boards will only fit in every other slot, beginning with the lowest slot, because of the coverplate.

3. Make sure that the edge of the card is aligned properly with the backplane connector, and then tighten the dog bolts finger tight. This is a *very important* step, since these bolts secure the card in the cage and provide a path to the ground terminal of the computer's power cord.

4. Connect the I/O cable to the coverplate.

5. Connect the I/O cable to the peripheral.

Chapter 3
Theory of Operation

This chapter describes the signals provided by the Breadboard Interface. A block diagram shows the different parts of the circuits provided with the board. A brief description of operation describes how the circuits operate. A complete schematic diagram shows circuit components, signal names, and connector pins.

Brief Overview

The Series 200 Computer backplane is an implementation of the Desktop-computer I/O (DIO) Bus. The DIO Bus is basically an electronically buffered version of the asynchronous input and output (I/O) bus of the MC68000 microprocessor. You may want to refer to the *MC68000 User's Manual*, HP part number 09826-90073, as you read this discussion.

Note

Keep in mind that the timing information given in the DIO Bus chapter refers to the signals *at the inputs of the drivers on the Processor Board.* The signals that arrive at the backplane may already have been skewed by variations in the 74LS245 buffer devices on the processor board and capacitances of the bus lines. See "Bus Timing Background" in the first section of the DIO Bus chapter for a diagram showing the originations of bus skew.

The Breadboard Interface can be used to implement three general types of I/O operations:

- data transfers
- interrupt requests and acknowledgements
- direct-memory access (DMA) operations

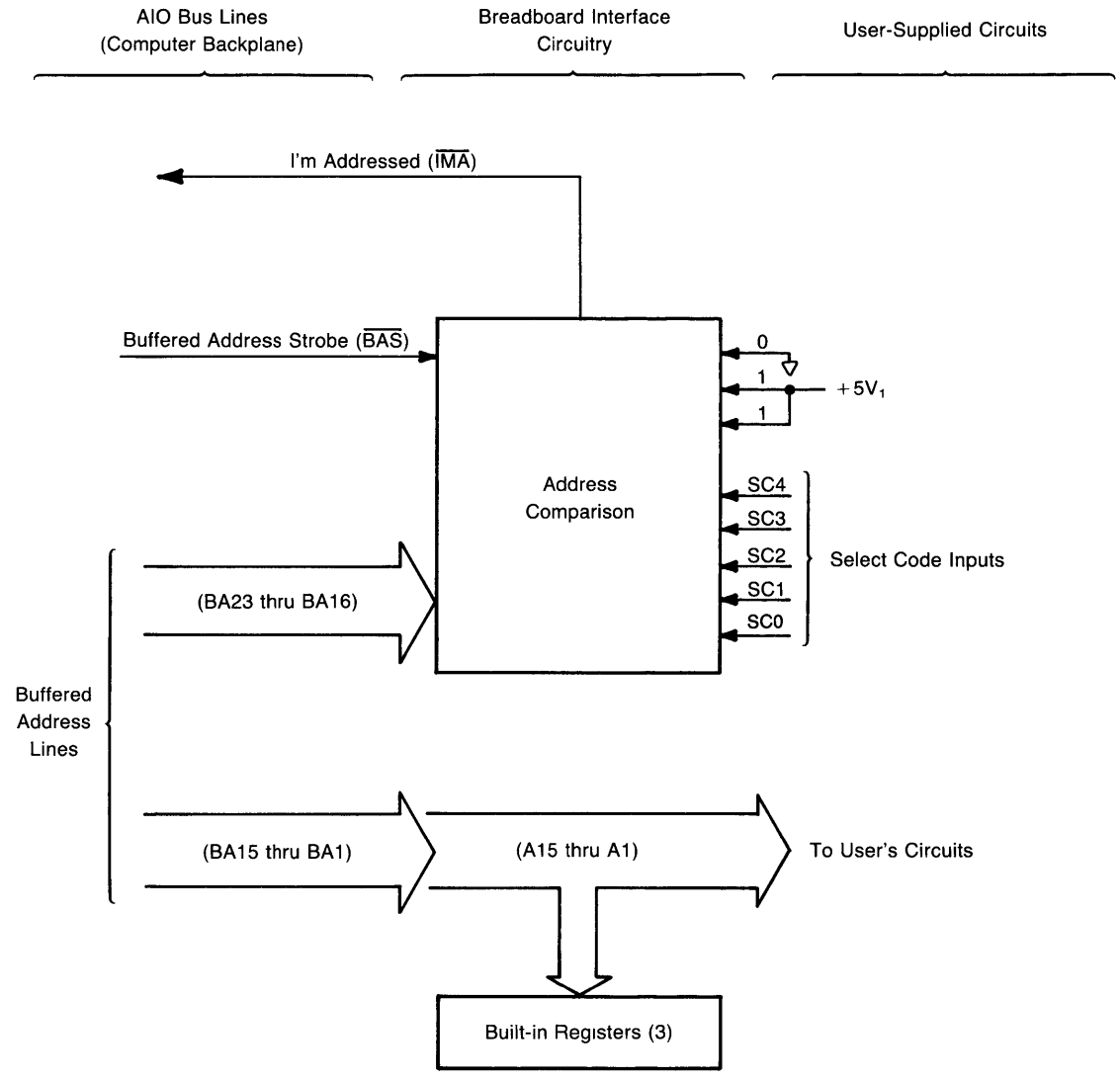
The interface provides part of the data-transfer circuitry. You must provide the remainder of the circuitry required for your application. Interrupt and DMA applications also require additional circuits.

Data Transfers

The data-transfer portion of the interface consists of address-decoding and data-bus management circuits. The standard interface registers on the board are considered part of this circuitry. See Chapter 4 for the standard interface registers' offset addresses and definitions.

Address Decoding

The twenty-three Buffered Address lines of the backplane (BA23 through BA1) are available to the Breadboard Interface. The most significant eight bits (BA23 through BA16) are used to select the interfaces and are not accessible to the circuitry added to the board. The following block diagram shows this portion of the circuitry.



Address-Decoding Circuitry

The address-decoding circuit consists of a digital comparator (U11) which compares the address placed on the address bus to the address setting of the interface (SC4 through SC0). The three most significant bits of the interface's address setting (which are compared to BA23 through BA21) are hard-wired to "011," respectively. The next five bits (which are compared to BA20 through BA16) are determined by the logic levels placed on SC4 through SC0, respectively. These five signals are used to set the interface to select codes 0 through 31.

The select code can be set by tying the signals called SC4 through SC0 either to logic High (to match a "1" on that address bit) or logic Low (for a "0"). For instance, the interface is assigned a select code of 31 when all lines are tied High.

Note that the interfaces occupy computer memory space from hexadecimal addresses 600 000 to 7FF FFF. This space is divided into thirty-two equal sections by BA20 through BA16, each of which occupies 65 536 (hexadecimal F FFF) memory addresses. Each interface can potentially have this number of memory locations.

During access cycles, the address is placed on BA23 through BA1 and allowed to settle (for at least 15 nanoseconds) before Buffered Address Strobe (BAS*) is asserted (Low). On the Breadboard Interface, this strobe enables the comparison of the address to the select code (pin 1 of U11).

The interface detects that it is addressed when the comparator (U11) asserts its "P=Q" output (pin 19), which gates the I'm Addressed signal (IMA*, pin 3 of U1) onto the backplane. This response is automatically made by hardware on the interface in the required 40-nanosecond response time.

The interface (or peripheral) must complete the transaction by acknowledging with the Data Transfer Acknowledge signal (DTACK*, pin 6 of U1), as described in the next section.

Data Direction and Upper/Lower Data Strobes

The DIO Bus defines the direction of the data with the Buffered Read/Write signal (BR/W*, pin 11 of U9). When BR/W* is negated (logic High), the Bus Master (processor board) is reading data from the card; when BR/W* is asserted (logic Low), the master is writing data to the card.

The bus also defines which of the data lines are to be used in the transfer: Buffered Upper Data Strobe (BUDS*, pin 13 of U9) indicates that the upper data byte (D15 through D8) is to be used, while Buffered Lower Data Strobe (BLDS*, pin 17 of U9) indicates that the lower byte (D7 through D0) is to be used. If both are asserted (Low), then all 16 data lines are to be used in the transfer.

These data strobes are directly from the processor board. There are also two other available data strobe signals (UDCS* and LDCS*) which are qualified by Card Select (CS*).

Data-Transfer Acknowledgement

The acknowledgement on DTACK* (made by the interface or peripheral device) must get back to the processor within 3.0 microseconds of BAS* being asserted.

During read operations, this acknowledgement indicates that the interface has placed the data on the appropriate data lines and that the data can be read. During write operations, it indicates that the data have been accepted by the interface. If this acknowledgement is not made within 3 microseconds, a bus error occurs. Bus errors are discussed in the DIO Bus chapter.

General Data-Transfer Timing

For data transfer, the general I/O protocol is as follows. Refer to the Breadboard schematic diagram as you read through this description. Details of data-transfer timing are fully discussed in the DIO Bus chapter.

1. The R/W* signal is used to indicate data direction (High=Read from interface, Low=Write to interface)

2. The address is placed on A23 thru A1

3. The Buffered Address Strobe (BAS*) is asserted to indicate that the address is valid

4. If A23 thru A21 are "011" and A20 thru A16 match the select code switches, the card recognizes that it is selected

5. The Card Select signal (CS*) is generated by Breadboard Interface hardware

6. The CS* signal asserts the I'm Addressed signal (IMA*), which signals that an interface is present at that select code

7. The Upper Data Strobe (UDS*) and/or Lower Data Strobe (LDS*) are asserted to indicate which data lines (D15 thru D8 and/or D7 thru D0, respectively) are to be used in the transfer

8. The data are read or written (after settling)

9. The CS* signal has already enabled the DTACK* buffer (U1 pin 4) to drive the DTACK* line on the bus (U1 pin 6) with the current logic level of DTACK* on the card (U1 pin 5)

10. When DTACK* is detected within 3 microseconds, the data transfer is complete. If it is not detected within this time interval, a bus error occurs. See the DIO Bus chapter for further details on bus error processing.

Register Access

The Breadboard Interface has four built-in registers: Read Registers 1 and 3, and Write Registers 1 and 3. These are standard interface registers whose definitions are given in the DIO Bus chapter and in Chapter 4 of this manual.

These registers are accessed by performing reads and writes of the registers' memory locations, which are offsets of 1 and 3, respectively, to the I/O card's base address (or select code). For instance, to access register 7 on the interface at select code 24, use address 780 007 (hexadecimal).

Address lines A15 and A14, the LDCS* line, and the logical OR of DTACK* with BR/W* enable U2 to select these registers. When U2 is enabled, address lines A2 and A1, along with BR/W*, are decoded to produce one of eight register-select signals: four signals are used to select these standard registers, and four are available for custom applications.

Note that A13 through A3 and UDCS* have not been used to select the standard registers. This register-selection scheme results in the following memory map on the interface. Note that 56 Kbytes (plus Registers 5 and 7) of address space are available for your applications.

UDCS* (See Note 1)	LDCS*	Address Lines 15 14 . . . . . 2 1	Register Selected
	1	0 0 X X X X X X X X X X 0 0	Register 1
	1	0 0 X X X X X X X X X X 0 1	Register 3
	1	0 0 X X X X X X X X X X 1 0	Register 5
	1	0 0 X X X X X X X X X X 1 1	Register 7
	1	X 1 X X X X X X X X X X X X	8 Kbytes
	1	1 0 X X X X X X X X X X X X	8 Kbytes
	1	1 1 X X X X X X X X X X X X	8 Kbytes
1		X X X X X X X X X X X X X X	32 Kbytes

Notes

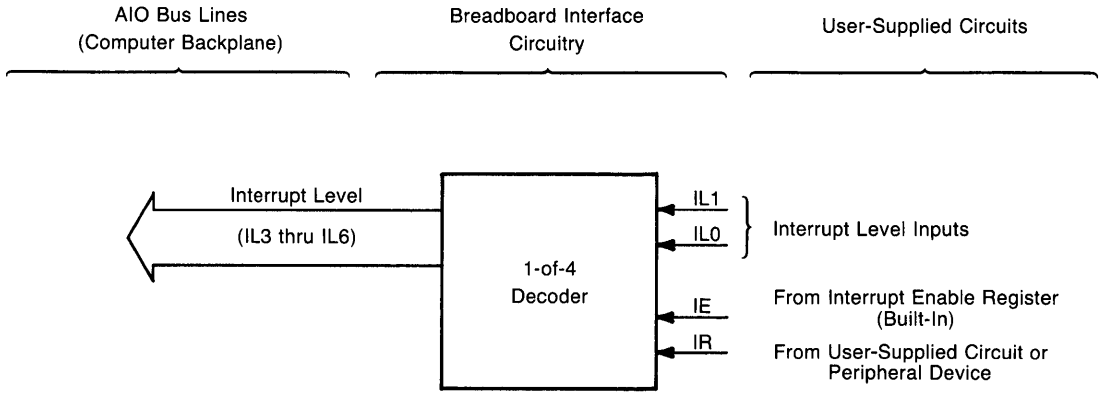
1. Note that no entry exists in one of the UDCS* and LDCS* columns while the other shows a "1." This convention indicates that the table shows the addresses of bytes, not words. Showing these signals asserted at the same time would obscure the information shown in the table.

2. "X" signifies irrelevant (don't care).

Interrupts

The interface accepts interrupt requests from external source(s), or from sources added to the board. These interrupts can be given one of four different hardware priorities. Hardware provided on the Breadboard Interface takes care of generating an interrupt request on the appropriate hardware interrupt level. The program can assign a software priority to the interrupt's service routine.

The following diagram describes the interrupt circuitry provided with the Breadboard Interface.



Interrupt Circuitry

General Interrupt Timing

The general interrupt request and acknowledge cycle is as follows:

1. The Breadboard Interface or an external peripheral asserts the Interrupt Request signal (IR, pin 6 of P1)

2. If interrupts are enabled (IE, pin 8 of U4, is High), the outputs of the 1-of-4 decoder (U8) are enabled. Decoder inputs IL1 and IL0 (pins 3 and 13 of U8, respectively) determine which output is asserted (Low).

3. It is then up to the language system to process the interrupt properly. Programming interrupt service routines is discussed in Chapter 4.

4. The software may decide to disable interrupts temporarily (until the pending interrupt is serviced).

Note

Externally vectored interrupts are not supported with the DIO Bus.

Direct Memory Access

The Breadboard Interface has no built-in circuitry for handling DMA applications. All DMA applications must be implemented by your own circuit designs. See the DIO Bus chapter for a complete description of DMA operations.

Signal Loading

The DIO Bus chapter provides the loads which can be each placed on each backplane signal. You can use these figures for backplane signals which are not buffered by the Breadboard Interface (such as DMAR0*, DMAR1*, and BA13 -- BA3).

Signals buffered by the Breadboard Interface can be loaded as limited by the characteristics of either the device which drives the signal or the supply currents, whichever is *more* restrictive.

Schematic Diagram and Parts List

The schematic diagram and component locator of the HP 98630 Breadboard Interface are located at the rear of this manual.

This is the list of components supplied with the 98630 interface.

Component Designator	Device Number	Description
U1	74LS125	Quad 3-State Buffer
U2	74LS138	1-of-8 Decoder
U3	74LS32	Quad 2-Input Nor Gate
U4,U5,U9	74LS244	Octal Buffer
U6,U12	74LS245	Octal Transceiver
U7	74LS00	Quad 2-Input Nand Gate
U8	74LS156	Dual 1-of-4 Decoder (OC)
U10	74LS273	8-Bit Register
U11	74LS688	Octal Comparator
C1	--	0.1 uF Capacitor
C2 thru C7	--	0.1 uF Capacitor (DIP)
R1	--	1.0 kOhm Resistor
F1	--	1.5 A Fuse
F3	--	4 A Fuse

Chapter 4
Programming Notes

This chapter provides brief notes regarding programming the Breadboard Interface using BASIC, Pascal, and MC68000 Assembler languages. Programming the card in HPL is very similar to BASIC.

The following topics are discussed in this chapter:

- Select codes
- Memory/register addresses
- Interface reset
- Data transfers
- Bus errors
- Interrupts
- DMA operations

Programming techniques used to access the features of the Breadboard Interface are only briefly outlined in this manual. For an in-depth treatment of the topics, see the manual(s) referred to in the text. The next section describes the general content of these manuals.

Relevant Documentation

This section lists the manuals that you will need to consult to write programs for the Breadboard Interface, because actual techniques for programming the interface are only outlined in this manual.

BASIC

BASIC programming techniques for HP Series 200 interfaces are described in the following manuals:

- *BASIC Interfacing Techniques*, part number 09826-90025 (or 09826-90020, which is an earlier edition)
- *BASIC Language Reference*, part number 09826-90056 (or 09826-90055, which is an earlier version).

The only *directly* relevant information regarding the Breadboard Interface is the brief discussion of the READIO and WRITEIO registers in Chapter 6 of *BASIC Interfacing Techniques*. However, you may want to read Chapters 8 through the end of the book to find out about how the other interfaces operate and are programmed to get some design ideas. The installation manuals for each interface may also provide valuable hardware-design ideas. Syntax and semantics for the READIO and WRITEIO keywords are described in the *BASIC Language Reference.*

Pascal

Pascal programming techniques applicable to the Breadboard Interface are described in the following manuals:

- *Pascal 2.0 User's Manual*, part number 98615-90020 (or the 09826-90070 for Pascal 1.0)
- *Pascal Procedure Library User's Manual*, part number 09826-90075
- *Pascal System Internals Documentation*, part number 09826-90074.

General use of the Pascal system is described in the *Pascal User's Manual.* Programming techniques for using the Breadboard Interface with Pascal involve using procedures and functions described in the *Pascal Procedure Library User's Manual.* Programming techniques for advanced topics such as interrupts and DMA operations are described in the *Pascal 2.0 System Internals Documentation.*

Assembler Language

Assembler language programming techniques applicable to the Breadboard Interface are described in the following manuals:

- *Pascal 2.0 User's Manual*, part number 98615-90020 (or the 09826-90070 for Pascal 1.0)
- *MC68000 User's Manual*, part number 09826-90073

The Assembler is accessed from the Pascal system as described in the *Pascal User's Manual.* The *MC68000 User's Manual* describes Assembler language programming for the 68000 microprocessor. The Pascal manual describes Assembler directives.

Select Codes

The select code inputs (SC4 through SC0) can be used to set the card to select codes 0 through 31; however, only select codes 8 through 31 are usable with BASIC and Pascal, since these systems map select codes 0 through 7 into internal I/O devices. With Assembler language programs, all select code settings can be used; resultant base addresses are 600 000 through 7F0 000 (hexadecimal). Select codes and resultant addresses are further described in a subsequent section of this chapter.

Note

The select code used for this interface must *not* be used by any other interface currently in the computer or in an HP 9888 Bus Expander currently connected to the machine.

Interface Reset

There are two ways to reset the Breadboard Interface: press the [RESET] key ([SHIFT] - [PAUSE]), or write to register 1. Pressing [RESET] pulses the RESET* and SRESET* signals low, while writing to register 1 pulses SRESET* only. Thus, the result of resetting the card depends on what you have connected to these signal lines; however, in general, Interface Reset should be a method of bringing a card to some known state.

The data written to register 1 is irrelevant -- the mere operation of writing to that register resets the card. Note also that the contents of register 3 (U10 and bits 0 and 1 of U4) are cleared.

With BASIC, use the WRITEIO statement to write to register 1. An example might be as follows:

```
  Select_code=15
Register=1
Data=An_integer
WRITEIO Select_code,Register;Data
```

With Pascal, use the procedure called "iowrite_byte." An example Pascal statement, which analogous to the preceding BASIC-language WRITEIO statement, is as follows:

```
iowrite_byte(24,1,Anydata);
```

It is also suggested that you use the "ioinitialize" and "iouninitialize" procedures before and after performing I/O operations, respectively. Keep in mind that the program must IMPORT the GENERAL_0 and GENERAL_1 modules to use these and the "iowrite_byte" procedures. See the Pascal procedure library manual for further details.

With Assembler language, use a MOVE instruction that writes one byte of data to the proper memory location. For an interface at select code 15, the base address is 6F0 000 (hexadecimal) and the offset is 1. Register addressing is further described in the next section.

I/O Memory/Register Addresses

The Breadboard Interface implements all "standard" I/O registers. These registers are described later in this chapter.

The preceding section showed an example of writing into a register on the Breadboard Interface by using the WRITEIO statement. The first argument of the statement was the select code; the second was the register number (or "offset address" of the register, from the card's base address). The third argument was the integer data written into the register; this argument's INTEGER value is obtained and written into the register by the WRITEIO statement.

Similarly, registers can be written by using the READIO numeric function. An example is as follows.

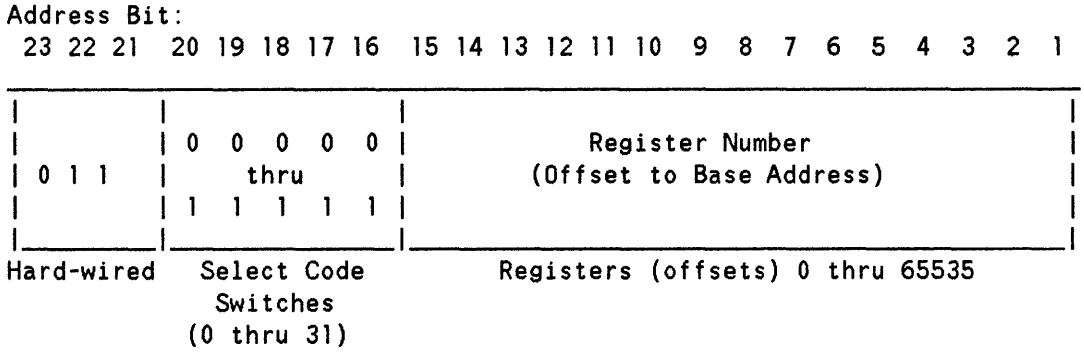
```
Enable_reg=READIO(24,3)
```

The preceding statement reads the value of register 3 on the card at select code 24. The register's contents are placed in the numeric variable `Enable_reg`.

The preceding examples do not take into account whether or not the register is an eight- or sixteen-bit register. If the register being read is a sixteen-bit register, the data must be read with *two* READIO statements: one READIO will be used to read the upper eight bits (which are at an even offset address), and another READIO will be used to read the lower eight bits (on the next-higher odd address). With Pascal and Assembler languages, you can use procedures (or instructions) that access a word of data in one operation.

External I/O Memory Map

The 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. Five switches permit the user to choose one of 32 select codes, ranging from 0 through 31, to determine which 64 Kbyte address space the card resides in. Switches should be implemented in the I/O card design for flexibility reasons. The address format, shown below, locates I/O devices in memory locations 600 000 through 7FF FFF. Note that all registers or memory locations on an I/O card are offsets to the card's "base address."



The external I/O memory map is shown below. The default factory select code settings are shown for the interfaces currently available. Note that select codes 15 and 16 are reserved for use with custom I/O cards.

	SELECT CODE	BASE ADDRESS	STANDARD ASSIGNMENT	
7FFFFF	31	7F0000	Reserved	
	30	7E0000	Reserved	
	29	7D0000	98627 (continued)	
	28	7C0000	98627 Color Output	
	27	7B0000	Reserved	
	26	7A0000	Reserved	
	25	790000	Reserved	
	24	780000	Reserved	
	23	770000	Reserved	
	22	760000	Reserved	
	21	750000	98629A SRM	
	20	740000	98628A Datacomm	
	19	730000	Reserved	
	18	720000	Reserved	
	17	710000	Reserved	
	16	700000	Custom I/O Card 2	
	15	6F0000	Custom I/O Card 1	
	14	6E0000	98625 Disc	
	13	6D0000	Reserved	
	12	6C0000	98622 GPIO	
	11	6B0000	98623 BCD	
	10	6A0000	Reserved	
	9	690000	98626 RS-232	Note 1
	8	680000	98624 Ext. HP-IB	
	7	670000		
	6	660000		
	5	650000		
	4	640000		
	3	630000		
	2	620000		
	1	610000		
	0	600000		
600000				

NOTE 1. The 98626A interface built into the 9816A is "hardwired" to this Select Code.

The function of certain registers within I/O cards are pre-assigned. Note that because most I/O cards are byte-oriented and these registers are connected to the lower byte of the data bus, their memory addresses are odd (1, 3, 5, and so forth) relative to the card's base address.

The designer is free to implement registers in addition to (but not instead of) the ones listed below. Also, the designer is not required to uniquely map each register to a location within the card's address space (i.e. several offset addresses may access the same register, which simplifies address decoding, as long as the addresses are not outside the card's 64 Kbyte address space).

Standard I/O Registers

The standard I/O card registers are defined as follows; the register number is the offset (added to the base address of the card) which is used to access the register.

Read Register 1: ID Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	ID4	ID3	ID2	ID1	ID0
Value =128	Value =64	Value =32	Value =16	Value =8	Value =4	Value =2	Value =1

ID4 thru ID0 -- Contain the card ID, which uniquely identifies each type of I/O card.

Currently Defined ID Numbers

0 - Reserved	16 - Custom I/O Card 2
1 - 98624	17 - Reserved
2 - 98626	18 - Reserved
3 - 98622	19 - Reserved
4 - 98623	20 - 98628A/98629A
5 - Reserved	21 - Reserved
6 - Reserved	22 - Reserved
7 - Reserved	23 - Reserved
8 - 98625	24 - Reserved
9 - Reserved	25 - Reserved
10 - Reserved	26 - Reserved
11 - Reserved	27 - Reserved
12 - Reserved	28 - 98627
13 - Reserved	29 - Reserved
14 - Reserved	30 - Reserved
15 - Custom I/O Card 1	31 - Reserved

Note that two ID numbers, 15 and 16, have been reserved for custom I/O cards designed and implemented outside of Hewlett-Packard.

Write Register 1: Interface Reset

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
The value written into this register is irrelevant.							
Value =128	Value =64	Value =32	Value =16	Value =8	Value =4	Value =2	Value =1

Writing any value into this register performs an Interface Reset of the card. The card's actual response to this action depends on how it is designed.

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

1. If the card contains LSI chips, one or more commands may be defined which can be sent to gracefully return the card to its power-on state.

2. If the card does not have such a sequence, the card may be capable of being reset to its power-on state by writing to register 1.

Read Register 3: Interrupt and DMA Status

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IE	IR	INT LVL Switches		Undefined		DE1	DE0
Value =128	Value =64	Value =32	Value =16	Value =8	Value =4	Value =2	Value =1

IE -- Interrupts Enabled: If this bit is set, interrupts are enabled.

IR -- Interrupt Request: If this bit is set, the card is requesting an interrupt. This bit is used during software polling to determine interrupt origin.

INT LVL Switches -- Interrupt Level: The interrupt level is typically set by two switches on the I/O card; these switches map into the 2 Interrupt Level bits.

Switch Setting	Interrupt Level
0 0	3
0 1	4
1 0	5
1 1	6

Undefined -- These bits have no standard definition and are thus are available for user-defined functions.

DE1 -- When this bit is set, DMA is enabled on channel 1.

DE0 -- When this bit is set, DMA is enabled on channel 0.

Write Register 3: Interrupt and DMA Enable

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EI	Undefined					DE1	DE0
Value =128	Value =64	Value =32	Value =16	Value =8	Value =4	Value =2	Value =1

EI -- Setting this bit enables interrupts.

Undefined -- These bits have no standard definition and are thus are available for user-defined functions.

DE1 -- Setting this bit enables DMA operations on Channel 1.

DE0 -- Setting this bit enables DMA operations on Channel 0.

Data Transfers

Data transfers are generally implemented with handshakes or some other similar synchronization method. Pascal and Assembler languages support DMA and interrupt operations for data transfer; interrupts and DMA operations are not possible with BASIC.

The BASIC-language OUTPUT, ENTER, and TRANSFER statements *cannot* be used with the Breadboard Interface, since the operating system cannot call firmware drivers for this card. This section briefly describes implementing a handshake.

Let's suppose that your Breadboard Interface has additional read and write registers which are available for general-purpose use. Let's also suppose that your peripheral device has the following four handshake lines:

- a Peripheral Status (output) line which indicates whether the peripheral is "OK" or "not OK"
- a Peripheral Control (input) line which it monitors as a "transmit request" from the computer
- a Data Direction (input) line which indicates whether the computer is to send or receive data
- a Peripheral Flag (output) line that indicates whether it is Ready or Not Ready to receive (or transmit) data

Let's, yes, even further assume that you just happen to have four registers on your Breadboard Interface, two of which can be read to determine the state of the two output lines from the peripheral and two of which can be written to set the two inputs to the peripheral. For simplicity, assume that bit 0 of all four registers is connected to the relevant signal line.

The following program segment shows an example of a BASIC-language data-transfer driver that can be used to send data from computer to peripheral. The handshake timing is performed entirely in software. Pascal and Assembler program segments for this type of data transfer would be analogous.

```
100   Select_code=15
110   !
120   ! Define register numbers.
130   Dir_reg=9    ! To define data direction.
140   Psts_reg=7   ! To read Peripheral Status line.
150   Data_reg=7   ! To write Data (8 bits).
160   Pflg_reg=5   ! To read Peripheral Flag line.
170   Pctl_reg=5   ! To write Peripheral Control line.
180   !
190   Data$="ASCII Characters"
200   !
210   Ok=READIO(Select_code,Psts_reg)
220   IF Ok THEN    ! Peripheral is OK, so send data.
230     WRITEIO Select_code,Dir_reg;1   ! Set direction=out.
240     FOR Char=1 TO LEN(Data$)
250 Ready_chk:Ready=READIO(Select_code,Pflg_reg)   ! Check for PFLG=Ready.
260           IF NOT Ready THEN Ready_chk
270           WRITEIO Select_code,Pctl_reg;0                    ! Clear PCTL.
280           WRITEIO Select_code,Data_reg;NUM(Data$[Char;1]) ! Place byte.
290           WRITEIO Select_code,Pctl_reg;1                    ! Set PCTL.
300     NEXT Char
310   ELSE  ! Peripheral is NOT OK, so quit.
320     PRINT "Peripheral Error on select code";Select_code
330   END IF
340   !
350   END
```

Note that this example assumes that the program is fast enough to detect transitions on the Peripheral Flag line. This assumption may *not* be valid for many types of handshakes -- you may want to add circuitry that can detect high-speed pulses on this line. The hardware implementation of this design is left up to your ingenuity. The HP 98622 GPIO Interface implements this type of circuit, which is described in the installation manual, HP part number 98622-90000.

Bus Errors

As shown in the Read and Write Cycle timing diagrams, the peripheral (or Breadboard Interface) must respond to the computer's request for a data transfer by asserting DTACK* within the specified time. When this acknowledgement is not made, the Processor Board generates a "bus error."

The BASIC-language operating system reports `Error 163 I/O interface not present`. You can use ON ERROR to recover from the error, if desired. With Pascal, you may want to use TRY/RECOVER to process any errors encountered. With Assembler language, the processor handles the error as an "exception" and uses exception vector number 2. See the *MC68000 User's Manual* for further details.

Interrupts and DMA Operations

Interrupt and DMA operations must be programmed using Pascal or Assembler language programs, as described in the *Pascal 2.0 System Internals Documentation.* BASIC does not support these type of operations on the Breadboard Interface.

The DIO Bus

Reprint of Chapter 20 of the *Pascal System Internals Documentation,* (HP part no. 09826-90074)

Table of Contents

The DIO Bus

Introduction

The Series 200 Desktop-computer Input/Output (DIO) Bus standard defines both mechanical and electrical requirements of cards which are to be used as optional input/output (I/O) devices with HP Series 200 Computers. The DIO Bus was first implemented in the Model 26 Computer (HP 9826A), followed shortly thereafter by the Model 36 (9836A) and the Model 16 (9816A). The 9888A Bus Expander, which can be used with Series 200 Computers, also implements the DIO Bus.

The DIO Bus is designed around the MC68000 series of microprocessors. If you want further information regarding MC68000 operation, refer to the *MC68000 User's Manual*, HP part number 09826-90073.

Objectives

The purpose of this document is to provide sufficient documentation to permit experienced digital-hardware designers to develop devices for use with the DIO Bus. The goal is to provide enough information to design Bus Slaves -- in particular, I/O cards. Bus Masters, such as Processor and DMA Controller boards, cannot be designed by using the information in this document. You may want to use the HP 98630 Breadboard Interface as the beginning of your own custom interface.

Any questions you may have regarding the information in this document should be brought to the attention of your local HP Desktop Computer Systems Engineer.

Designer's Responsibilities

In order to ensure safe, reliable operation with Series 200 Computer products, the specifications in this document must be strictly followed when designing Bus Slave devices. The "Electrical Specifications" and "Mechanical Specifications" sections describe topics such as available power-supply current and size requirements of I/O cards. The section called "Design Qualification" provides safety and operating requirements that your I/O card design must meet to qualify as a usable device.

Keep in mind that you are responsible for any circuitry that you design and use with HP products, both in terms of personal safety and proper operation with the equipment.

CAUTION

HEWLETT-PACKARD SHALL NOT BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL. REPAIRS NECESSITATED BY MISUSE OF THE EQUIPMENT, OR BY HARDWARE, SOFTWARE, OR INTERFACING NOT PROVIDED BY HEWLETT-PACKARD ARE NOT COVERED BY THE WARRANTY.

Signal Terminology

The following convention is used throughout this document: Active-low signals are denoted with a * following the name. This is equivalent to a bar over the signal name which is often used for active-low signals. Thus, the following are equivalent:

$$\text{BAS*} = \overline{\text{BAS}}$$

When a signal is referenced as "asserted" or "true," "negated" or "false," and so forth, it is relative to the signal's *function.* For example, to say that BAS is asserted means that it is active (performing its function). Whether it exists as BAS (active-high) or BAS* (active-low) on the backplane is irrelevant.

References to "high" and "low" refer directly to TTL logic voltage levels. When referring to high and low signals, the actual name of the signal is used. For example, when the signal BAS* is described as being low, the signal entitled BAS* has a TTL logic-low voltage level. The TTL logic levels are defined as follows:

Logic High: >= 2.0 volts

Logic Low: <= 0.8 volts

System Elements

The functional modules of the DIO Bus are shown below. Where signals go specifically from one functional module to another, the two modules are shown side-by-side for clarity.

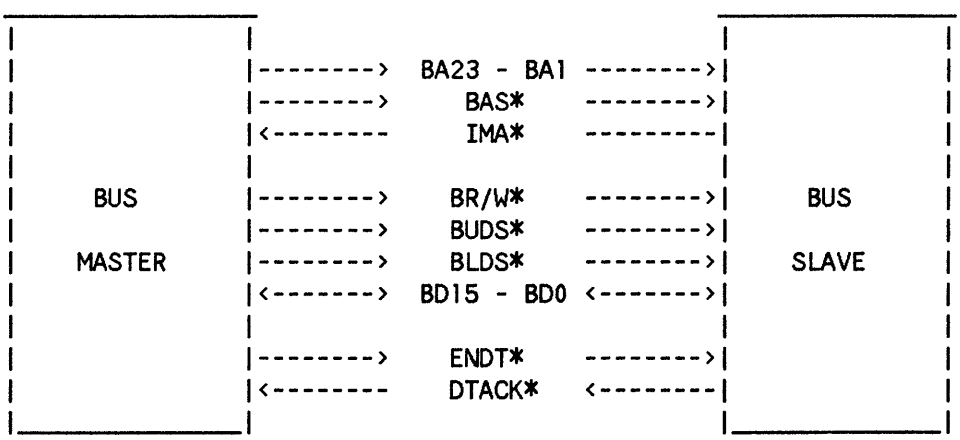


Figure 1a. Data-Transfer System Elements

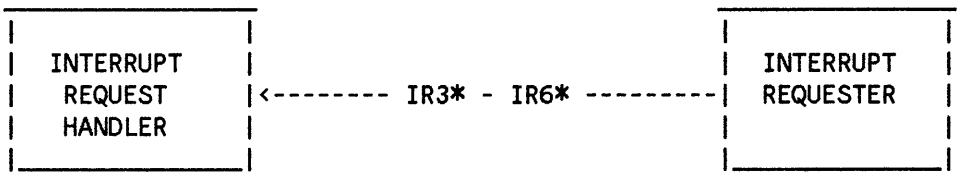


Figure 1b. Interrupt System Elements

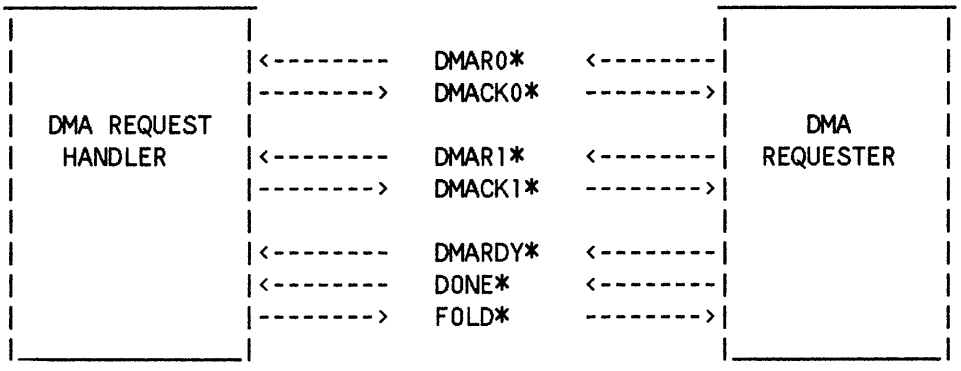
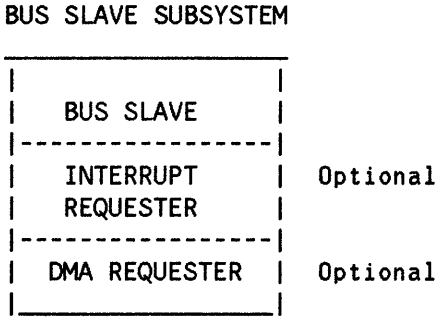


Figure 1c. Direct-Memory Access System Elements

Bus Slave Subsystem

A diagram of a Bus Slave subsystem is shown in the following drawing.



The Bus Slave subsystem consists of the system interface elements listed on the right hand side of Figures 1a, 1b, and 1c.

Bus Timing Background

The key feature of the DIO Bus is that it is *asynchronous* -- in other words, there is no clock signal on the backplane to which other signals are referenced. 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 handshakes to convey address and data. 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 microprocessor is guaranteed to drive the bus address lines 30 ns prior to asserting Address Strobe. Most receiving devices require at least 15 ns of address setup time prior to Address Strobe. To guarantee 15 ns of address setup time, the following rules were developed to control gate delays and bus loading (these are expanded in greater detail in later sections).

- Each board is limited to one LS TTL load on the address bus, data bus, the address strobe, the data strobes, and read/write signal.
- The PC board trace length on bus signals should be as short as possible and, in any case, must not exceed 3 inches.
- An SN74LS245 (or equivalent SN74LS244) is used to buffer the above signals.

Thus the designer is given the guideline that, from the input of the 74LS245 bus driver to the input of the slave's bus receivers, 15 ns of skew are possible (see Figure 2). Signal skews are due to *differences* in device delays and physical properties of bus lines (such as capacitance). Therefore, skews due to the bus drivers and the bus are not specified separately. Detailed signal-loading specifications are discussed in the "Electrical Specifications" section.

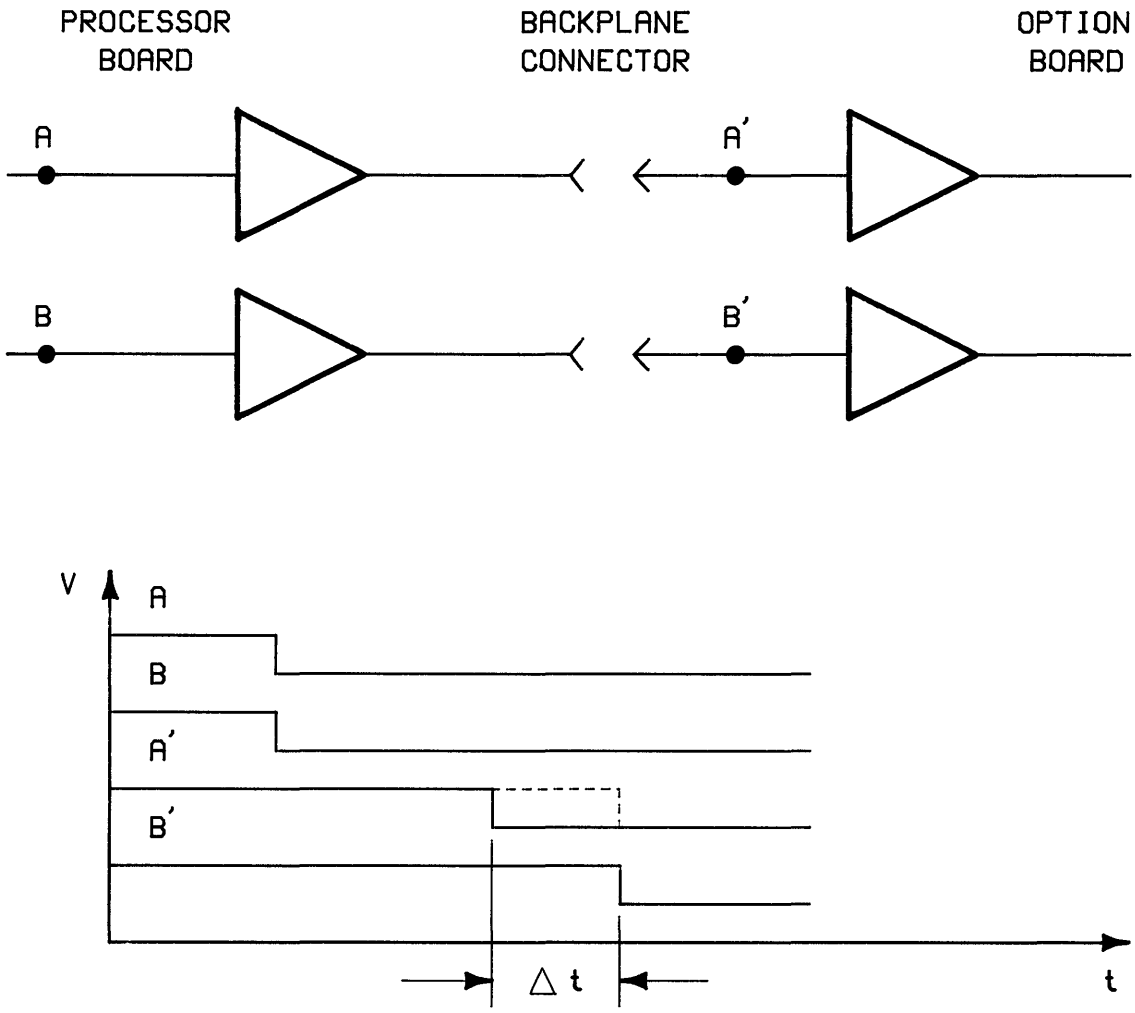


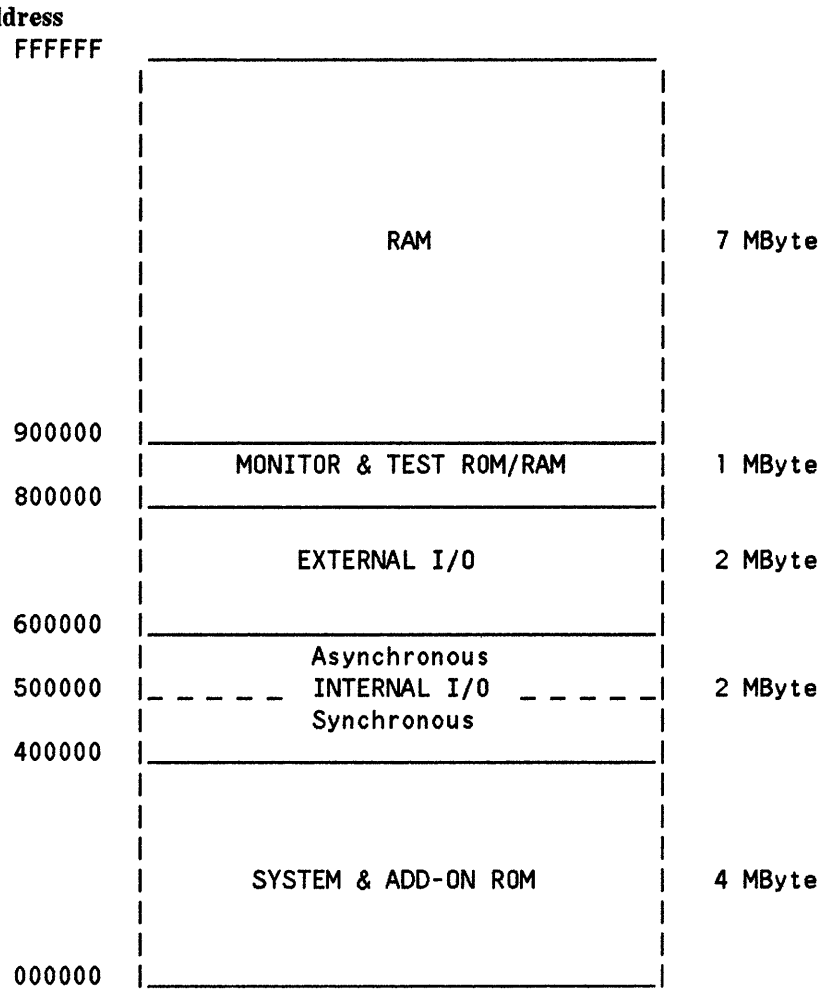
Figure 2. Origins of Signal Skew

Memory Map

This Bus specification is not intended to document in detail all Series 200 Computers' memory maps. Instead, documentation of the memory map is limited to the External I/O memory map and standard I/O register assignments.

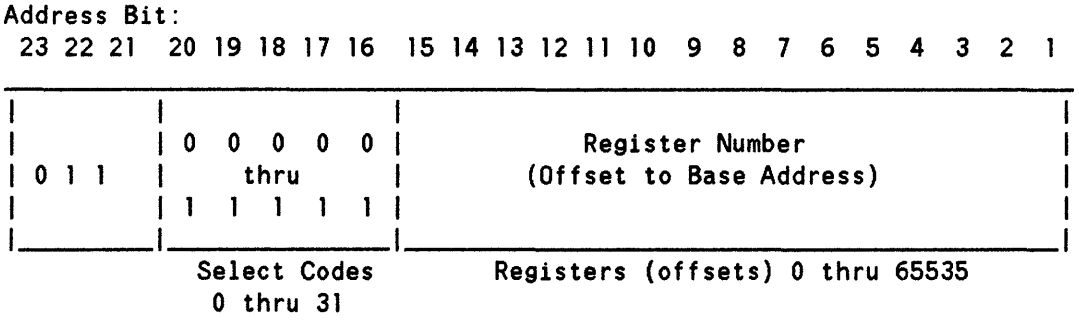
Series 200 Memory Map

The Series 200 memory map is shown below. The 68000's 24-bit address bus can directly address 16 Mbytes of memory. The External I/O occupies 2 Mbytes of address space (hexadecimal addresses 600000 through 7FFFFF).



External I/O Memory Map

The 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. Five switches permit the user to choose one of 32 select codes, ranging from 0 through 31, to determine which 64 Kbyte memory space the card resides in. Switches should be implemented in the I/O card design for flexibility reasons. The address format, shown below, locates I/O devices in memory locations 600 000 through 7FF FFF. Note that all registers or memory locations on an I/O card are offsets to the card's "base address."



Not all external I/O select codes can be used with the Pascal operating system. With Pascal I/O procedures, all select codes from 0 through 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 through 7. However, the Pascal and BASIC language systems map select codes 1 through 7 to other addresses in the internal I/O address space. Thus, I/O cards set from 1 through 7 are inaccessible with these languages; only I/O cards set from 8 through 31 can be accessed. At the assembly language level, however, I/O cards with select codes over the entire range of 0 through 31 can be accessed.

The external I/O memory map is shown below. The default factory select code settings are shown for the interfaces currently available. Note that select codes 15 and 16 are reserved for use with custom I/O cards.

	SELECT CODE	BASE ADDRESS	STANDARD ASSIGNMENT	
7FFFFF	31	7F0000	Reserved	
	30	7E0000	Reserved	
	29	7D0000	98627 (continued)	
	28	7C0000	98627 Color Output	
	27	7B0000	Reserved	
	26	7A0000	Reserved	
	25	790000	Reserved	
	24	780000	Reserved	
	23	770000	Reserved	
	22	760000	Reserved	
	21	750000	98629A SRM	
	20	740000	98628A Datacomm	
	19	730000	Reserved	
	18	720000	Reserved	
	17	710000	Reserved	
	16	700000	Custom I/O Card 2	
	15	6F0000	Custom I/O Card 1	
	14	6E0000	98625 Disc	
	13	6D0000	Reserved	
	12	6C0000	98622 GPIO	
	11	6B0000	98623 BCD	
	10	6A0000	Reserved	
	9	690000	98626 RS-232	Note 1
	8	680000	98624 Ext. HP-IB	
	7	670000		
	6	660000		
	5	650000		
	4	640000		
	3	630000		
	2	620000		
	1	610000		
	0	600000		
600000				

NOTE 1. The 98626A interface built into the 9816A is "hardwired" to this Select Code.

Registers

The function of certain registers within I/O cards are pre-assigned. Note that because most I/O cards are byte-oriented and these registers are connected to the lower byte of the data bus, their memory addresses are odd (1, 3, 5, and so forth) relative to the card's base address.

The designer is free to implement registers in addition to (but not instead of) the ones listed below. Also, the designer is not required to uniquely map each register to a location within the card's address space (i.e. several offset addresses may access the same register, which simplifies address decoding, as long as the addresses are not outside the card's 64 Kbyte address space).

Standard I/O Registers

The standard I/O card registers are defined as follows; the register number is the offset (added to the base address of the card) which is used to access the register.

Read Register 1: ID Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	ID4	ID3	ID2	ID1	ID0
Value =128	Value =64	Value =32	Value =16	Value =8	Value =4	Value =2	Value =1

ID4 thru ID0 -- Contain the card ID, which uniquely identifies each type of I/O card.

Currently Defined ID Numbers

0 - Reserved	16 - Custom I/O Card 2
1 - 98624	17 - Reserved
2 - 98626	18 - Reserved
3 - 98622	19 - Reserved
4 - 98623	20 - 98628A/98629A
5 - Reserved	21 - Reserved
6 - Reserved	22 - Reserved
7 - Reserved	23 - Reserved
8 - 98625	24 - Reserved
9 - Reserved	25 - Reserved
10 - Reserved	26 - Reserved
11 - Reserved	27 - Reserved
12 - Reserved	28 - 98627
13 - Reserved	29 - Reserved
14 - Reserved	30 - Reserved
15 - Custom I/O Card 1	31 - Reserved

Note that two ID numbers, 15 and 16, have been reserved for custom I/O cards designed and implemented outside of Hewlett-Packard.

Write Register 1: Interface Reset

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
The value written into this register is irrelevant.							
Value =128	Value =64	Value =32	Value =16	Value =8	Value =4	Value =2	Value =1

Writing any value into this register performs an Interface Reset of the card. The card's actual response to this action depends on how it is designed.

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

1. If the card contains LSI chips, one or more commands may be defined which can be sent to gracefully return the card to its power-on state.

2. If the card does not have such a sequence, the card may be capable of being reset to its power-on state by writing to register 1.

Read Register 3: Interrupt and DMA Status

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IE	IR	INT LVL Switches		Undefined		DE1	DE0
Value =128	Value =64	Value =32	Value =16	Value =8	Value =4	Value =2	Value =1

IE -- Interrupts Enabled: If this bit is set, interrupts are enabled.

IR -- Interrupt Request: If this bit is set, the card is requesting an interrupt. This bit is used during software polling to determine interrupt origin.

INT LVL Switches -- Interrupt Level: The interrupt level is typically set by two switches on the I/O card; these switches map into the 2 Interrupt Level bits.

Switch Setting	Interrupt Level
0 0	3
0 1	4
1 0	5
1 1	6

Undefined -- These bits have no standard definition and are thus are available for user-defined functions.

DE1 -- When this bit is set, DMA is enabled on channel 1.

DE0 -- When this bit is set, DMA is enabled on channel 0.

Write Register 3: Interrupt and DMA Enable

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EI	Undefined					DE1	DE0
Value =128	Value =64	Value =32	Value =16	Value =8	Value =4	Value =2	Value =1

EI -- Setting this bit enables interrupts.

Undefined -- These bits have no standard definition and are thus are available for user-defined functions.

DE1 -- Setting this bit enables DMA operations on Channel 1.

DE0 -- Setting this bit enables DMA operations on Channel 0.

Data Transfers

This section discusses the transfer of data between Bus Masters and Bus Slaves. Briefly stated, data transfers on the DIO Bus are made by writing data to a memory location (or reading data from a location). The Bus Master (the CPU, or Processor, board) defines the address, data direction, and whether a byte or word is to be transferred. The Bus Slave device that contains the memory location makes an acknowledgement to the master, which completes the transfer.

The rest of this section describes the signals involved in the transfer and the transfer process.

Data Transfer Signals

The bus signals used in data transfers are shown below. Signal names starting with B (for "buffered") are derived from signal names of the 68000 microprocessor -- the 68000 name is that which follows the "B" in the name. A brief description of each signal is given; for more detailed information on the buffered 68000 signals, refer to the *MC68000 User's Manual*, HP part number 09826-90073. Two of the signals, IMA* and ENDT*, are HP-defined.

BA23--BA1 The 23-bit address bus. Note that BA0 is not on the bus; its meaning is conveyed in BUDS* and BLDS* (see below).

BAS* Buffered Address Strobe, defines when the address is valid.

BD15--BD0 The 16-bit data bus.

BR/W* Buffered Read/Write: High for read, Low for write.

BUDS*, Buffered Upper Data Strobe, Buffered Lower Data BLDS*, Strobe and Buffered Data Strobe: BUDS* indicates BDS* that BD15-BD8 are required; BLDS* indicates that BD7-BD0 are required. BDS* is used generically to refer to either BLDS* or BUDS*. It is not a bus signal; it is used for discussion purposes only.

DTACK* Data Transfer Acknowledge: issued by the currently addressed slave (RAM, I/O card, etc.) to inform the master that the slave can complete the transfer cycle. During a read operation, it indicates that data signals placed on the bus by the slave are now valid. During a write operation, it indicates that the slave has accepted the data.

IMA* I'm Addressed: issued by a card that detects itself being addressed. It is also used by the Bus Expander to reverse its data-bus buffers when a card in the Bus Expander is addressed.

ENDT* Enable DTACK: generated by the CPU board and (optionally) used by Bus Slaves to generate the DTACK* signal automatically, which reduces the access-overhead time and provides a pseudo-synchronous, repeatable access-cycle time. Note that this signal does not have to be implemented in a Bus Slave design.

Data Transfer Overview

Before the transfer is initiated, the R/W* line is used to indicate the direction of data flow. BAS*, which defines when the bus address is valid, begins the data transfer operation. As soon as BAS* is asserted, each Bus Slave should check to see if the address currently on the address bus is contained within its address space. Once a Bus Slave detects that it is being addressed, it must acknowledge with IMA*. BLDS* and BUDS* indicate which data lines are involved in the transfer.

An interlocked handshake is used to transfer data from the Bus Master to the Bus Slave. The steps of the handshake are as follows:

1. Bus Slave asserts DTACK* when data accepted from bus or provided on bus.

2. Bus Master sees DTACK* and enters "terminate cycle" sequence.

3. Bus Master negates BAS* when cycle completes.

4. Bus Slave negates DTACK* when BAS* negated.

As mentioned earlier, ENDT* is available from the Bus Master. It is used to improve the response time of Bus Slaves. ENDT* is basically BAS* delayed by one and one-half clock cycles (of an 8 MHz clock). A Bus Slave can use this signal to pseudo-synchronize an asynchronous process (e.g. dynamic memory) so that access time is fixed (at five clock cycles). ENDT* timing is discussed later in this chapter.

The reference to clock cycles should not be confused with the state designators S0 through S7 shown on 68000 timing diagrams. One clock cycle is equal to two states. For instance, one clock cycle corresponds to states S0 and S1 on the 68000 timing diagrams. The following diagram shows the relationship of the CPU-board clock cycles and 68000 state designators.

```
| S0 | S1 | S2 | S3 | S4 |  w |  w | S5 | S6 | S7 |
|---------|---------|---------|---------|---------|
| Cycle 1 | Cycle 2 | Cycle 3 | Cycle 4 | Cycle 5 |
```

Read Cycle Description

Figure 3 shows the timing of signals during a read cycle. Timing specifications are for Bus Slaves. The key aspects of a read cycle are as follows:

1. Prior to the beginning of the read cycle, BR/W*, BUDS*, BLDS* and DTACK* are actively pulled high. The setup time on BR/W* high is 15 ns before BAS* goes low. Because this is a read cycle, BR/W* remains high during the entire cycle.

2. The Bus Master drives the address bus BA23-BA1 with a minimum address setup time of 15 ns before BAS* is asserted.

3. All Bus Slaves determine if they are being addressed using BAS* as a decode enable; the device being addressed responds with IMA* within 50 ns after BAS* occurs.

4. When a Bus Slave is addressed, it puts data on the bus after BDS* is asserted. (Note that BDS* can precede or follow BAS* by up to 75 ns.) DTACK* is asserted by the Bus Slave to indicate that the data signals are valid and that the transfer can be completed by the Bus

Master reading the data; thus, the time from BDS* to data valid is device-dependent. Bus Slaves must drive the data bus 30 ns prior to asserting DTACK* (except when using ENDT* or during memory-to-I/O DMA transfer). This 30 ns set-up time requirement must be met *at the receiving device*, so it must include all signal skews (see specification 9a of Figure 3).

Note

The MC68000 specification for asynchronous data setup time permits valid data to follow DTACK* by as much as 75 ns, due to delays in synchronizing DTACK*; however, to support DMA and other Bus Masters, the DIO Bus requires data to precede DTACK* (except when using ENDT*; see specification 9 of Figure 3).

For memory-to-I/O DMA transfers, the data must be valid 45 ns prior to DTACK*. This is to accomodate delay through the Fold Buffer (on the HP 98620 DMA controller card) as well as any additional bus delay. Additional bus delay occurs because twice the normal bus load is driven (the bus load *to* the Fold Buffer and the bus load *from* the Fold Buffer to the I/O card). Up to 30 ns of delay can be permitted and still meet the write timing specification of 15 ns data setup time prior to DTACK* (see specification 9b of Figure 3). The DMA section further describes DMA timing.

For I/O-to-memory DMA transfers, the data setup time from the I/O card is still specified at 30 ns. This is because there are matching delays in both the data (Fold Buffer plus 2 times normal bus loading) and the DTACK* signal (I/O card generates DMARDY* which the DMA Controller delays in "converting" to DTACK*). The DMA section further describes DMA timing.

---

5. The Bus Master detects that DTACK* has occurred and ends the cycle by negating BAS* and BDS*. BAS* is set false within 350 ns of assertion of DTACK*.

6. The Bus Slave detects that the cycle has ended when BAS* or BDS* go false (whichever occurs first) and then stops driving IMA*, DTACK*, and the Data Bus. Likewise, the Bus Master stops driving the address bus. Relative to BAS* or BDS* (whichever occurs last), the following signals change with the indicated delay:

   Address hold: 15 ns (min.)
   IMA* high: 50 ns (max.)
   DTACK* high: 50 ns (max.)
   Data Bus hold: 100 ns (max.)

Note

Processor Boards actively drive DTACK* high when BAS* goes high after a read or write cycle. Because the I/O card will keep DTACK* low until it sees BAS* high, there is a brief (<100 ns) DTACK* driver conflict until the I/O card stops driving DTACK*.

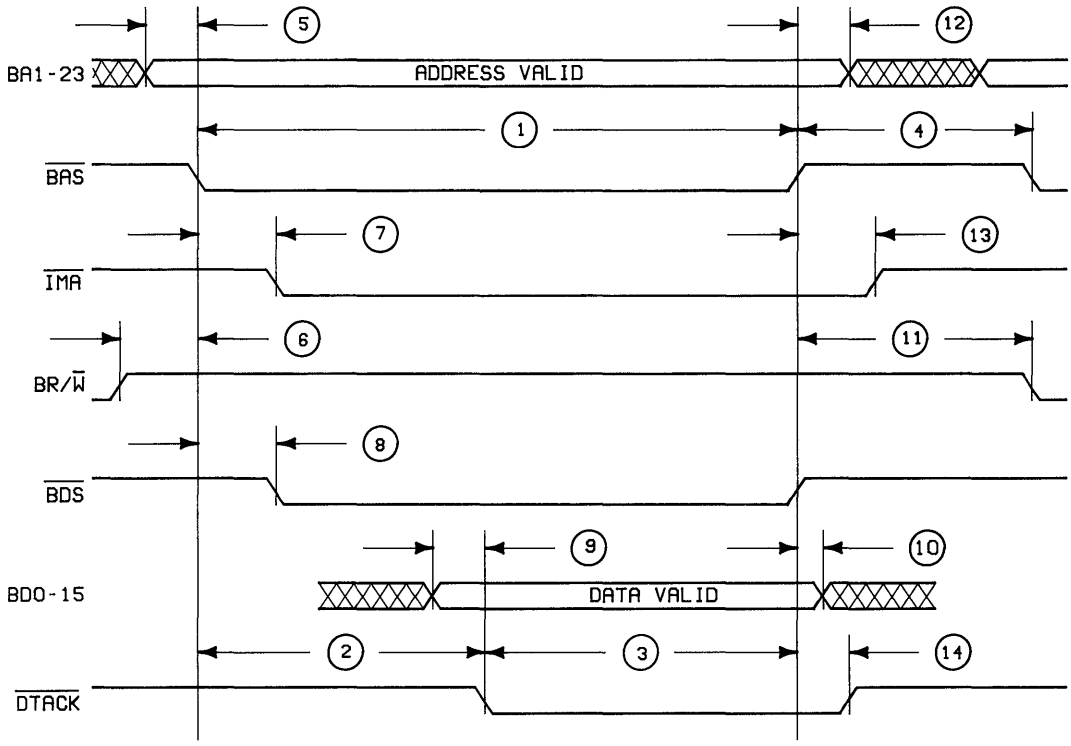


Figure 3. Read Cycle Timing

	Read Cycle Times	Min.	Max.	
1	BAS* low (limited by BERR* on CPU board)		4500	
2	BAS* low to DTACK* low (without Bus Error):			
a	DMA Operation		1500	
b	Non-DMA Operation		3000	
3	DTACK* low to BAS* high (without Bus Error)		350	Note 1
4	BAS* high	140		
5	Address setup before BAS* low	15		
6	BR/W* high to BAS* low or BDS* low	15		
7	BAS* low to IMA* low	0	50	
8	BAS* low to BDS* low	-75	75	
9	Data setup before DTACK* low:			
a	Data transfers without ENDT* or I/O-to-memory DMA transfers	30		
b	Memory-to-I/O DMA transfers	45		
c	Data transfers with ENDT*	-75		
10	Data hold after BAS* or BDS* high	0	100	Note 2
11	BR/W* high after BAS* or BDS* high	15		Note 3
12	Address hold after BAS* or BDS*	15		Notes 3&4
13	BAS* high to IMA* high	0	50	
14	BAS* high to DTACK* high	0	50	

Notes

1. This time must be met if Time 2 is 3000 ns. Otherwise, the requirement is that Time 2 + Time 3 < 3500 ns.

2. Whichever goes high first of BAS* or BDS*.

3. Whichever goes high last of BAS* or BDS*.

4. Based on the 68000 and buffer/bus skew specifications, the address hold time is 15 ns.

Write Cycle Description

Figure 4 shows the timing of signals during a write cycle. Timing specifications are for Bus Slaves. The key aspects of a write cycle are as follows:

1. Prior to the beginning of the write cycle, BR/W*, BUDS*, BLDS* and DTACK* are actively pulled high by the Bus Master.

2. Address is valid on the address bus BA23-BA1 with a minimum address set-up time of 15 ns before BAS* is asserted. The address is valid a minimum of 5 ns before BR/W* goes low (Write). BR/W* goes low sometime in the interval from 75 ns before to 85 ns after the assertion of BAS*.

3. All devices on the bus determine if they are being addressed using BAS as a decode enable; the device being addressed responds with IMA* within 50 ns after BAS* occurs.

4. Data is valid on the data bus BD15-BD0 a minimum of 15 ns prior to the assertion of BDS*. Notice that the time from BAS* to the data strobes can vary over a wide range (50 ns to 2500 ns). Longer times can occur when doing a DMA operation, because a read must be performed prior to the write operation.

5. When BDS* is asserted (low), BR/W* is guaranteed to already be asserted (low); BR/W* should *not* be qualified with BAS* because of the potential for bus conflicts, since BR/W* can still be changing up to 85 ns after BAS* goes true.

6. The Bus Slave stores the data and asserts DTACK* indicating to the Bus Master that the storage operation is complete.

7. The Bus Master detects that DTACK* is true and negates BAS* and BDS* within 350 ns. The Bus Master then removes the data BD15-BD0 from the data bus. To ensure data hold time sufficient to allow the Bus Slave to clock the data at the same time it asserts DTACK*, the minimum Bus Master data hold time after detection of DTACK* is 85 ns.

8. The following signals change with the indicated delay after the negation of BDS* and BAS* (whichever occurs last).

   Address hold: 15 ns (min.)
   BR/W* high: 25 ns (min.)
   DMA* high: 50 ns (max.)
   DTACK* high: 50 ns (max.)
   DTACK* (a pull-up 0 ns (min.) is asserted on the Processor Board)

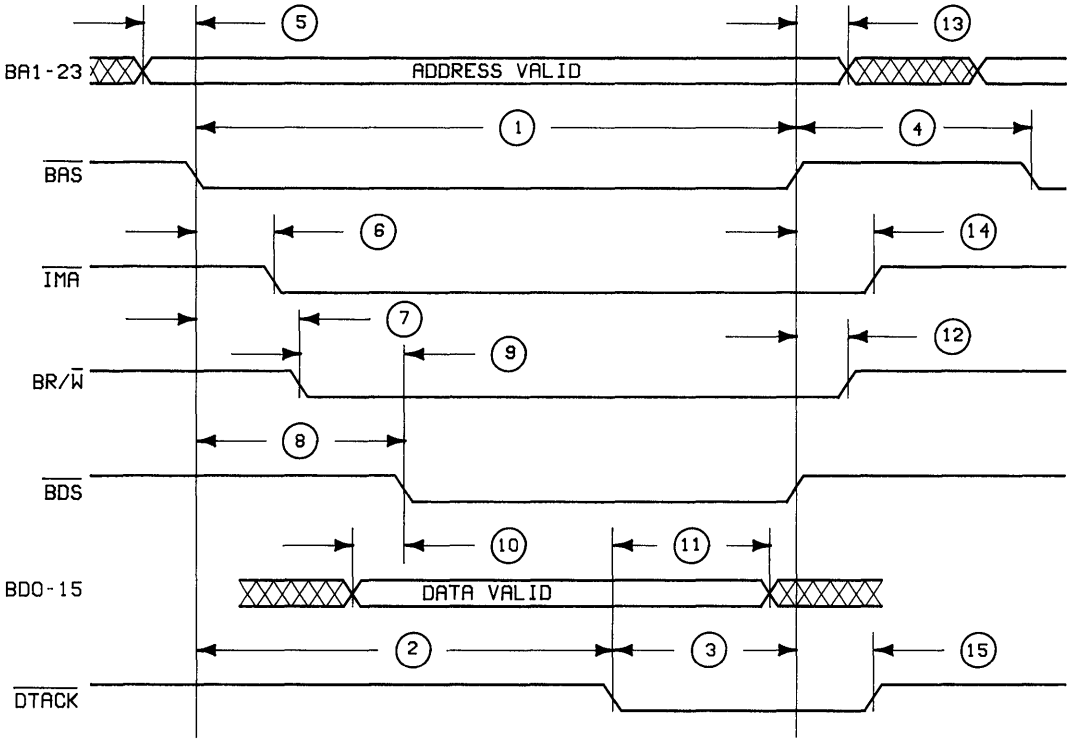


Figure 4. Write Cycle Timing

	Write Cycle Times	Min.	Max.	
1	BAS* low (limited by BERR* on CPU board)		4500	
2	BAS* low to DTACK* low (without Bus Error)		3000	
3	DTACK* low to BAS* high (without Bus Error)		350	Note 1
4	BAS* high	140		
5	Address setup before BAS*	15		
6	BAS* low to IMA* low	0	50	
7	BAS* low to BR/W* low	-75	85	
8	BAS* low to BDS* low	50	2500	
9	BR/W* low to BDS* low	65		Note 2
10	Data setup before BDS* low	15		
11	Data hold after DTACK* low	85		
12	BR/W* hold after BAS* or BDS* high	15		Note 3
13	Address hold after BAS* or BDS*	15		Note 4
14	BAS* high to IMA* high	0	50	
15	BAS* high to DTACK* high	0	50	

Notes

1. This time must be met if Time 2 is 3000 ns. Otherwise, the requirement is that Time 2 + Time 3 < 3500 ns.

2. Times 7 & 8 imply that BR/W* could go low after BDS* goes low; in actuality, this cannot happen as guaranteed by Time 9.

3. Whichever of BAS* or BDS* goes high last.

4. Based on the 68000 and buffer/bus skew specifications, the address hold time is 15 ns.

Read-Modify-Write

No current I/O cards support read-modify-write operations.

Enable DTACK Timing

As discussed previously, Enable DTACK (ENDT*) is used to improve the response time of backplane cards by generating DTACK* pseudo-synchronously, permitting memory accesses to always occur in five clock cycles (of an 8 MHz clock). ENDT* is currently used only by Series 200 RAM cards. However, I/O cards that contain large amounts of memory can use ENDT* to speed up the transfer of data to and from this memory, since the access time of memory is usually fixed. During a read, DTACK* is generated before data is set up on the bus; however, because of synchronization delays, the 68000 permits data to follow DTACK*.

ENDT* is generated by the Processor board and is delayed from BAS* by one and one-half clock cycles, as shown below. The Bus Slave must have a 25 ns "turnaround" time from the assertion of ENDT* to asserting DTACK*. Also, data must be valid within 265 ns of BAS* being asserted during a read cycle.

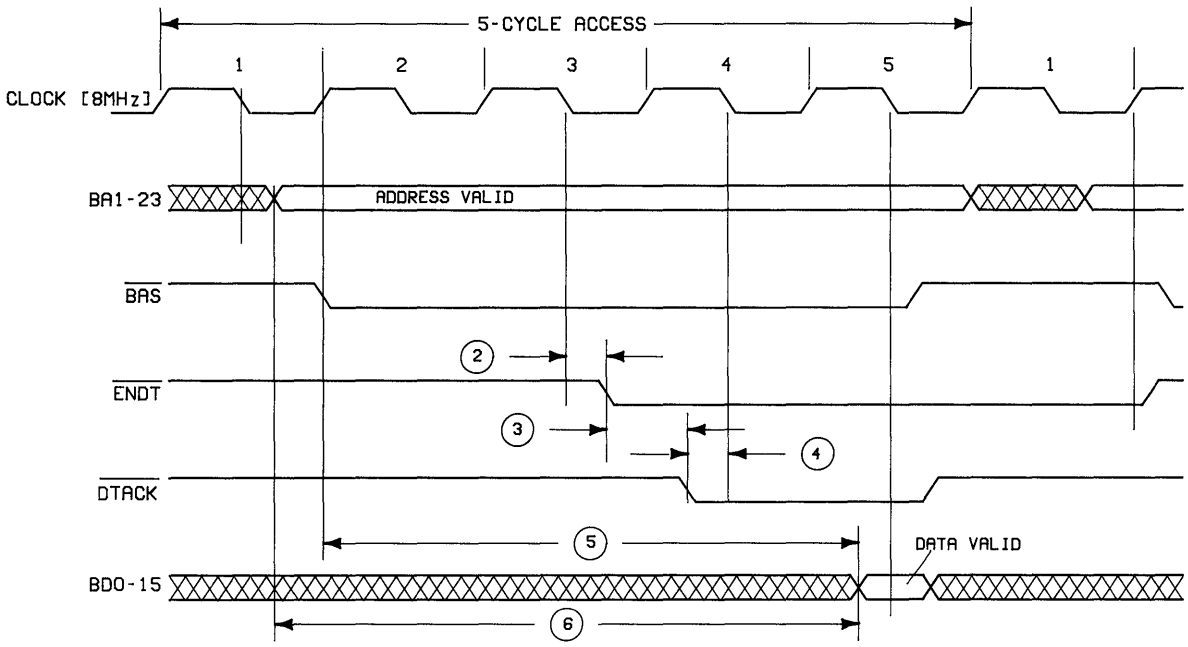


Figure 5. Enable DTACK Timing

	ENDT* Timing	Min.	Max.	
1	Clock cycle time @ 8 Mhz	125 ns nominal		Note 1
2	Clock low to ENDT* low		40	
3	ENDT* low to DTACK* low		50	Note 2
4	DTACK* setup time		35	
5	BAS* low to read data valid		265	
6	Address to read data valid		295	Note 3

Notes

1. The clock is on the Processor board and does not appear on the DIO Bus; it is shown for reference only.

2. To ensure that DTACK* is low at the Processor board input within 50 ns, the maximum ENDT* to DTACK* gate delay on the Bus Slave device *cannot* exceed 25 ns; this must represent the worse-case gate delay. The DTACK* driver must ensure that DTACK* is driven low on the bus within an additional 25 ns.

3. Whichever of these times is longer.

Direct Memory Access

All DMA operations with Series 200 Computers require the use of the HP 98620 Direct Memory Access (DMA) Controller card. This card monitors DMA requests from I/O cards, requests control of the bus, and orchestrates DMA data transfers. The DIO Bus supports two direct memory access channels. DMA transfers between I/O cards and memory are supported; memory-to-memory transfers are not. DMA data rates exceeding 1 million transfers/second are possible in word (or byte) mode. This section gives an overview of DMA operation and discusses DMA input and output operation.

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 (see the previous chapter).

DMAR0*, DMA Request: asserted by an I/O card to request DMAR1* a DMA transfer on DMA Channel 0 or Channel 1.

DMACK0* DMA Acknowledge: a response from the DMA DMACK1* Controller which acknowledges DMA request on Channel 0 or Channel 1.

DMARDY* DMA Ready: indicates that the I/O card has provided the data (DMA input) or accepted the data (DMA output).

DONE* Done: an output from the DMA Controller to indicate that DMA is done. DONE* can be used at the option of the I/O card designer to determine when DMA is done.

FOLD* Fold: an output from the DMA Controller to indicate that a data byte is being folded from the upper byte of the data bus to the lower byte (or vice versa). This folding is performed by the DMA Controller.

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

DMA Overview

Two enable a DMA transfer, the operating system (or program) must perform two operations:

1. Program the DMA Controller with the type of transfer (word/byte, input/output, priority, etc.).

2. Enable DMA channel 0 or DMA channel 1 on the I/O card by writing to Write Register 3. The I/O card will then request a DMA operation on the assigned channel. In response to this request, the DMA Controller requests and eventually receives control of the bus and then begins the DMA transfer.

A DMA transfer occurs during a single bus cycle during which data is both read and stored. 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 for the transfer.

During a DMA operation, the memory device does a normal data transfer using BAS*, BR/W*, BDS*, DTACK*, etc. Therefore, the I/O card must use different signals to handshake 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 responds with DMACK0* which the I/O card treats the same as BAS* in that it begins the actual data transfer cycle. When the I/O card has provided or accepted the data, it responds with DMARDY*, which the DMA Controller interprets as DTACK* and responds accordingly.

Both byte and word DMA transfers are supported. In word mode, data is transferred a word at a time between memory and the I/O device. In byte mode, data is transferred on the lower byte of the I/O card; however, because the data in memory is "packed", both upper and lower bytes of memory must be accessed. The DMA Controller supports this via a "Fold Buffer," which is used to transfer data between the upper byte of the data bus (for memory accesses) 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 (BD15-BD8). Likewise, during a DMA output operation, the Fold Buffer is used to transfer memory data on BD15-BD8 to the lower data byte for the I/O card. The DMA Controller provides the FOLD* signal indicating when folding is to occur; this is used primarily by the Bus Expander. Bus Slave designs should not incorporate this signal.

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. Relinquishing of bus control depends upon two factors: 1) The time for the I/O card to request another DMA transfer and, 2) the channel priority programmed into the DMA Controller. This subject is discussed in more detail below.

DMA Output Cycle Description

Figure 6 shows the DMA Output cycle. 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 DMAR0, indicating that it is ready to begin a DMA output 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 read cycle:

   a. Memory address is put on the bus and BR/W* line is set to Read (high).

   b. BAS* and BDS* are asserted for the memory device and DMA Acknowledge (DMACK0*) is asserted to indicate to the I/O card that a DMA cycle has started. The I/O card responds to DMACK0* 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 DMACK0, it can optionally release DMAR0. This is discussed in more detail below.

5. The memory device fetches the data, places it on the bus with 30 ns of setup 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 priority is not set for channel 0 and DMAR0* was set false after DMACK0*, the I/O card must re-assert DMAR0* 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 to it like Bus Masters normally respond to DTACK*, ends the cycle by removing BAS*, BDS*, and DMACK0*. Once the I/O card detects that DMACK0* is gone, it removes DMARDY*.

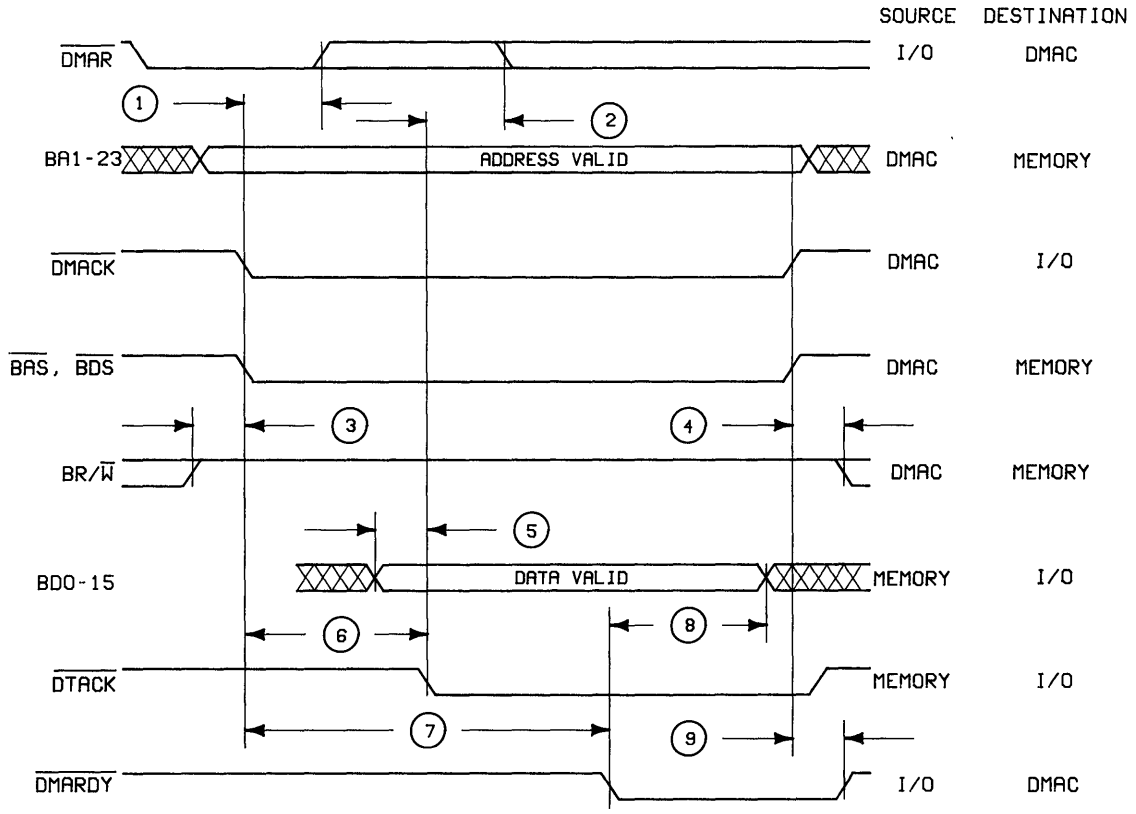


Figure 6. DMA Output Cycle Timing

	DMA Output Timing	Min.	Max.	
1	DMAR* release after DMACK* low	0		Note 1
2	DMAR* low after DTACK* low:			
a	Priority = 0		65	
b	Priority = 1		1600	
3	BR/W* setup before DMACK* low	15		
4	BR/W* hold after DMACK* high	15		
5	Data setup before DTACK* low	15		
6	BAS* low to DTACK* low	0	1500	
7	DMACK* low to DMARDY* low (w/o Bus Error)		3000	
8	Data hold after DMARDY* low	85		
9	DMARDY* release after DMACK* high	0	50	

Notes

1. The I/O card may keep DMAR* low after its request is acknowledged.

DMA Input Cycle Description

Figure 7 shows the DMA Input cycle. To do a DMA input, 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 is put on the bus and BR/W* line is set to write (low). Notice that BR/W* is set low prior to BAS contrary to a normal write cycle in which BR/W* may go low after BAS. Since the DMA Controller knows that a memory write operation is to occur, it can assert BR/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 below. In response to DMACK0*, the I/O card fetches the data, places it on the bus and, after a minimum data setup time of 30 ns, asserts DMARDY* to indicate 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*, BDS*, and DMACK0*. In response to the removal of BAS*, the memory card removes DTACK*; likewise, in response to the removal of DMACK0*, the I/O card removes DMARDY*.

7. On the last byte, 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 control line 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 2.5 us of DMACK0* going true.

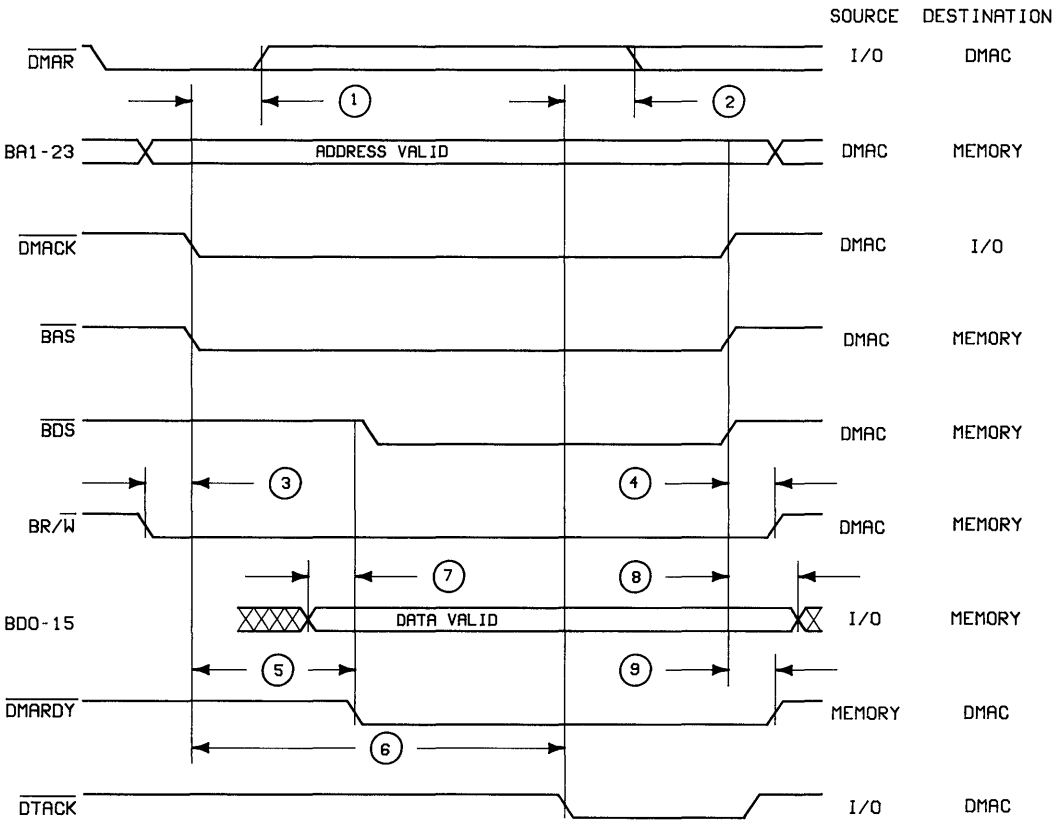


Figure 7. DMA Input Cycle Timing

	DMA Input Cycle Timing	Min.	Max.
1	DMAR* release after DMACK* low	0	
2	DMAR* low after DTACK* low:		
a	Priority = 0		65
b	Priority = 1		1600
3	BR/W* low before DMACK* low	15	
4	BR/W* high after DMACK* high	15	
5	DMACK* low to DMARDY* low (w/o Bus Error)		2500
6	BAS* low to DTACK* low (w/o Bus Error)		3000
7	Data setup before DMARDY* low	30	
8	Data hold after DMACK* high	0	100
9	DMARDY* release after DMACK* high	0	50

DMA Speed Considerations

To optimize the speed of DMA transfers, the transfer (read/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 (98620A) to synchronize the handshake signals is similar to the response of the 68000 and adds minimal overhead.

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*.

The amount of time that the I/O card has after DTACK* depends on the current setting of the Priority bit on the DMA Controller card. This time is either 65 ns (DMA Priority bit = 0) or 1.6 µs (Priority bit = 1). See specification 2 in Figures 6 and 7. Priority bit set to 0 requires a 65 ns response; if the I/O card does not assert DMAR0* within 65 ns, the DMA Controller relinquishes the bus at the end of the cycle to the next highest priority bus master (typically the Master Controller).

Because designing an I/O card to respond to DTACK* with DMAR0* within 65 ns adds complexity to the I/O card, the DMA Controller can be programmed for the 1.6 µs DTACK*-to-DMAR0* response time in the hope that the I/O card will generate another DMA Request. As mentioned above, selection of this time is done with the Priority bit, which operates as follows:

- If the Priority bit is 0 the I/O card must assert DMAR0* within 65 ns of DTACK*. This ensures that the DMA Controller keeps control of the bus and provides 1.2 Mbytes/sec.

- If the Priority signal for the channel is 1, then the bus is not relinquished until 1.6 µs 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 Bus Master 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 (if the bus is released) and when the DMA operation is complete. When ARM is 0, the transfer is complete.

3. The I/O card can use the DONE* signal from the DMA Controller to interrupt the Bus Master.

The DONE* signal works as follows: DONE* is asserted by the DMA Controller on the last byte of a DMA transfer. For an input operation, DONE* can be used by the I/O card to inhibit further acceptance or handshaking 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 can 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 DMAR0* 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. DONE* has sufficent setup and hold time to be qualified off of DMACK*.

Bus Error Operation

An exception sequence is generated when the Processor Board's Bus Error (BERR*) signal is asserted. This is an open-collector signal, permitting it to be generated by any device (including the CPU board). Current applications of the Bus Error signal are discussed in this chapter. This section is for informational purposes; the Bus Error line should not be implemented with Bus Slaves. Refer to the *MC68000 User's Manual* for timing requirements of the 68000 processor.

The Bus Error Signal

One signal is involved in bus error operation: the Bus Error signal (BERR*). When asserted, this signal causes the 68000 processor to terminate the current bus cycle and float the address and data bus. When negated, the processor begins its exception processing.

Bus Timeouts

The Master Controller will generate BERR* when an accessed device fails to respond within a certain time. As discussed previously, a device responds with DTACK*. Thus, the time from asserting BAS* to the arrival of DTACK* is monitored; since DTACK* causes BAS* to go false (with some delay), 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 four-bit BERR* counter, located on Series 200 CPU boards, remains cleared as long as BAS* is false. When BAS* is true, it begins counting (4 Mhz) and generates a BERR* when it overflows after 16 counts (16 × 250 ns = 4.0 µs). To provide margin, the maximum width of BAS* is set at 3.5 µs. To ensure that DTACK* has time to negate BAS* prior to 3.5 µs, DTACK* must arrive two CPU clocks cycles (250 ns) earlier. Because of this time and other delays, and to provide ample margin, the maximum time from BAS* going low to DTACK* going low has been specified at 3.0 µs.

From the above description, it appears that devices have 3.0 µs to assert DTACK*. However, for devices that implement DMA, it is not this simple. For example, during a DMA output cycle, RAM must be read *and* the I/O card written to within one cycle while BAS* is asserted. Thus, the *combined* RAM read time and I/O card write time must be less than 3.0 µs or a bus error will occur. DMA timing is discussed in detail in the DMA chapter.

For a DMA input (I/O read, memory write), DMARDY* must be true (indicating valid I/O card data) within 2.5 µs of DMACK* (which goes true at the same time as BAS*). As usual, DTACK* (indicating the memory card has accepted the data) must be true within 3.0 µs of BAS* to prevent a BERR* timeout.

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

To ensure data transfers without a Bus Error, the following timing requirements must be met.

Description of Time Interval	Maximum Time (in ns)
BAS* low (read or write)	3500
DTACK* low to BAS* high (read or write)	350
Write to Bus Slave:	
BAS* low to DTACK* low (non-DMA write)	3000
DMACK* low to DMARDY* low (DMA write)	3000
Read of Bus Slave:	
BAS* low to DTACK* low (non-DMA source)	3000
BAS* low to DTACK* low (DMA source is memory)	1500
DMACK* low to DMARDY* low (DMA source is I/O)	2500

Interrupt Operation

Even though the MC68000 supports two types of interrupts, only one method of responding to interrupts is currently supported on the DIO Bus: autovectored interrupts. With autovectoring, the interrupting device does not provide a vector to the interrupt service routine -- the Processor Board generates its own default vector. Like the 68000, the DIO Bus supports seven interrupt (hardware) priority levels.

Interrupt Signals

The interrupt signals on the DIO Bus are shown below. Interrupts can occur at levels 1 through 7 (lowest to highest); interrupt levels 3 through 6 are for "external" I/O cards. The following table shows the assignment of interrupt levels for Series 200 computers.

Signal	Hardware Priority (INT LVL)	Device
INT1*	1	Keyboard/Real-Time Clock
INT2*	2	Internal Disc
INT3* - INT6*	3 - 6	External I/O
INT4*	7	Reset Key, Powerfail

Note that there is no INT0* signal: interrupt level 0 is the quiescent (non-interrupting) state. For 68000-based Bus Masters, logic is used to encode these interrupt signals into the three processor inputs, IPL0, IPL1, and IPL2. For example, Series 200 Processor boards use an 74LS148 8-to-3 priority encoder to allow INT1 through INT7 to generate the proper combination of IPL0, IPL1 and IPL2.

The following interrupt levels are the only interrupt levels which have been "hardwired"; all other I/O cards have a two-bit switch to select one of levels 3 through 6.

Internal HP-IB	Level 3
DMA Interface	Level 3
Internal RS-232 (Model 16 only)	Level 4

Note

Even though IR1, IR2, and IR7 are not used for external I/O, the signals have been put on the backplane for expandability and compatibility with future products. *Do not* use IR1, IR2, or IR7 in Bus Slave implementations.

Interrupt Description

The DIO Bus does not support interrupt vectoring from the I/O cards. Instead, autovectoring on the Processor Board is used. The interrupt sequence is described as follows; refer to the subsequent timing diagram as you read about the sequence:

1. When an interrupt request is made on one of the Interrupt Request signal lines (IR1* through IR7*), logic on the Processor board is used to encode the signals into the three inputs IPL0, IPL1, and IPL2 of the 68000 processor. Series 200 processor boards use an 74LS148 8-to-3 priority encoder to allow INT1* through INT7* to generate the proper combination of IPL0, IPL1 and IPL2.

2. If the level of the interrupt is greater than the current processor priority, the processor begins exception processing (after finishing the current instruction).

3. In response to an interrupt request, the Processor Board generates one of 7 vectors that is a function of the interrupt level. These interrupt vectors are mapped in high memory by the Boot ROM. For information about the 7 interrupt vectors, refer to the Boot ROM section in the *Pascal 2.0 System Internals Documentation*, HP part number 09826-90074.

4. 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 software polling routine must be used to determine which card to service. The two most significant bits of read register 1 contain interrupt information: Bit 7 is set if the card is enabled to interrupt, and bit 6 is set if the card is currently requesting an interrupt.

Utility Signals

This section identifies and defines the signal lines which serve utility-type functions on the DIO Bus. These utility lines supply reset and state-decoding capabilities for the bus.

Signals

The utility signals consist of the following lines:

- RESET*
- HALT*
- Function Codes (BFC0, BFC1, BFC2)

Reset Operations

The RESET* signal goes low at system power-up to allow cards to properly initialize. The Processor board can also generate RESET*.

Power-Up and Power-Down Resets

RESET* is automatically generated within the system at power-up and remains true until 120 ms after +5V reaches 4.5 volts. When RESET* is negated, the +12V supply will have been within in its 11.5-volt limit for at least 55 ms. At power-down, RESET* is re-asserted within 15 ms after +5V drops to approximately 4 volts.

Reset by the Processor

The processor can generate a Reset by executing a RESET instruction. See the *MC68000 User's Manual* for further information.

Reset by a Bus Slave

Asserting RESET* by an DIO Bus device does not reset the processor board unless HALT* is simultaneously asserted; thus, RESET* by itself only resets other bus devices.

Note

Bus Slaves should not be allowed to assert RESET* and HALT* signals simultaneously, since such action Resets the entire system.

Function Code Signals

The Function Codes (FC2, FC1, and FC0) generated by the 68000 are buffered (BFC2, BFC1, and BFC0) 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; however, this buffer is also disabled during an interrupt acknowledge cycle to inhibit certain control signals which happen to share the same buffer. Therefore, when FC2=FC1=FC0=1 (interrupt acknowledge), the buffered Function Codes on the bus will float (no pull-ups) and are undefined. Therefore, to use Function Codes, the following guidelines should be followed:

- When BAS* occurs, the Buffered Function Codes are valid. BAS* is one of the control signals inhibited by disabling of the buffer during an interrupt acknowledge cycle; so when BFC2 -- BFC0 are floating, BAS* is pulled high with a pull-up resistor.

- During an interrupt acknowledge cycle, the presence of IACK* indicates that FC2=FC1=FC0=1. However, BFC2, BFC1, and BFC0 are undefined and cannot be used.

Note

It is recommended that Bus Slaves not use the Function Codes, even though they are currently defined, since future implementations may not define them.

Electrical Specifications

This section defines the non-timing electrical specifications for the DIO Bus. Included in this chapter are power supply tolerances, card power dissipation specifications, and signal loading information. Pinouts of the DIO Bus are discussed in the "Mechanical Specifications" section.

Power Distribution and Grounding

Power on the DIO Bus is distributed on the backplane as regulated, dc-voltage 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.

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

Note

This +12V supply is used for the 9826A's disc drive motor. Designers should be aware of potential noise on the line.

Power Supply Tolerances

The specfications shown below allow for the effects of line regulation, load regulation, cross regulation, initial accurracy, temperature stabiblity, and ripple. The tolerances represent the worst-case tolerances for existing DIO Bus devices.

Supply	Tolerance	Range
+5	+5/-4.3%	5.25, 4.78
+12	+6/-4%	12.7, 11.5
-12	+10/-4%	-13.2,-11.5

Power Requirements of Cards

The following table shows the typical and maximum supply power (and current) available for use with cards used in Series 200 computers and bus expanders.

TYPICAL POWER & CURRENT				MAXIMUM POWER & CURRENT			
TYP. POWER PER CARD (WATTS)	TYPICAL AVERAGE CURRENT (Amps dc) +5	+12	-12	MAX. POWER PER CARD (WATTS)	MAXIMUM AVERAGE CURRENT (Amps dc) +5	+12	-12
4.4	.8	.096	.064	5.3	.96	.120	.080

Backplane power is a finite resource which makes its availability dependent upon the configuration of the backplane. The diagram above shows standard power requirements for a typical Bus Slave. If a Bus Slave uses more than the standard current requirements, other calculations are needed to ensure power consumption does not exceed power supply design limits. The following table gives the maximum current (and power) alotted to the backplane.

Maximum Typical Amps

	+5	+12	-12
Model 16	2.2 A	0.33 A	0.20 A
	Not to exceed 17.5 Watts total		
Models 26&36	7.6 A	0.91 A	0.60 A
	Not to exceed 41.6 Watts total		

CAUTION

POWER LIMITS DISCUSSED IN THIS SECTION MUST NOT BE EXECEEDED. FAILURE TO FOLLOW THESE GUIDELINES VOIDS ANY WARRANTIES ON THE AFFECTED EQUIPMENT AND DEVICES.

Current requirements for existing interface and accessory cards.

Interface or Accessory	Typical current (mA) +5V	+12V	-12V
HP 98254 64 Kbyte Memory	590	0	0
HP 98256 256 Kbyte Memory	830	0	0
HP 9888A Bus Expander interface	1000	0	0
HP 98620 DMA Controller	1200	0	0
HP 98622 GPIO Interface	750	0	0
HP 98623 BCD Interface	500	0	0
HP 98624 HP-IB Interface	470	0	0
HP 98625 High-speed Disc Interface	600	8	0
HP 98626 RS-232 Serial Interface	400	50	50
HP 98627 Color Video Interface	1100	0	0
HP 98628 Datacomm Interface	720	37	60
HP 98629 Resource Mgmt. Interface	750	37	37
HP 98691 PDI (without EPROM)	750	37	60
HP 98028 Resource Mgmt. Multiplexer (4 channels connected)	530	530	0
HP 13264 Data Link adapter	30	160	23
HP 13265 300-baud Modem	100	45	45
HP 13266 Current loop adapter	200	90	80

Correct voltages are guaranteed from the mainframe power supply only if the above limits are observed (these limits refer to **all** power supplied to the backplane, including any external devices obtaining power from I/O cards). Even though the system can operate properly for short periods while excedding these limits, local heating and other stresses that result shorten the life and compromise the reliability of the power supply.

The following examples show power calculations for two configurations (all current is in mA):

Example 1

	+5	+12	-12
1 HP 98620 DMA Controller	1200	0	0
1 HP 98256 256 Kbyte memory card	830	0	0
1 HP 98628 Datacomm Interface	710	37	60
1 HP 13265 300-baud Modem	100	45	45
	2840	82	105

Total power: 16.4 Watts

Conclusion: Total currents are well within the acceptable limits for Models 26 and 36.

Example 2

	+5	+12	-12
1 HP 98620 DMA Controller	1200	0	0
3 HP 98256 256 Kbyte memory car	2490	0	0
4 HP 98628 Datacomm Interface	2840	148	240
4 HP 13265 300-baud Modem	400	180	180
	6930	328	420

Total power: 43.6 Watts

Conclusion: The value for total power exceeds the maximum. This configuration should not be attempted.

On-Card Fuse Specifications

A UL/CSA/IEC requirement is that any device operating from a supply capable of supplying more than 8 amps be fused. Therefore, any board plugged into the backplane **must** have a 4-amp maximum fuse in series with the +5V bus.

Signal Loading

The following table shows the drive capabilities of each output signal and recommended drivers for each input signal on the DIO Bus.

Signal Name	Maximum Receive Loading	Recommended Send Device
BAS*, BR/W*, BUDS*, BLDS*	1 LS load max.	SN74LS245 buffer
IMA*, DTACK*, ENDT*	1 LS load max.	Any 3S/OC LS gate
DMAR1*, DMAR0*, DMACK1*, DMACK0*, DMARDY*, DONE*	1 LS load max.	Any 3S/OC LS gate
INT6*-INT3*	1 LS load max.	Any 3S/OC LS gate
BFC2*-BFC0*	2 LS loads max.	SN74LS245
RESET*	5 LS loads max.	SN7417 OC Buffer
HALT*	.8 mA TOTAL for card cage	--
BA23-BA1	1 LS load max.	SN74LS245 Buffer
BD15-BD0	SN74LS245 transceiver	

Notes

1. "3S" signifies a device with 3-state outputs

2. "OC" signifies an device with open-collector outputs

Mechanical Specifications

This section presents sufficient mechanical information to create printed-circuit (PC) boards that fit into the Series 200 backplane.

Specifications for Cards

The drawing on the following page shows the size requirement for I/O or non-I/O cards. The points worth noting are:

- I/O cards fit in every other slot of the I/O backplane. The slots in between hold non-I/O cards such as RAM 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 coverplate.
- For both types of cards, space must be left on either side of the board to prevent components from interfering with the card guides in the card cage.
- I/O connectors are left-justified and extended to the right as needed for the size of the connector.

The following drawing shows the outline of an I/O-type board that can be used in the Series 200 card cages.

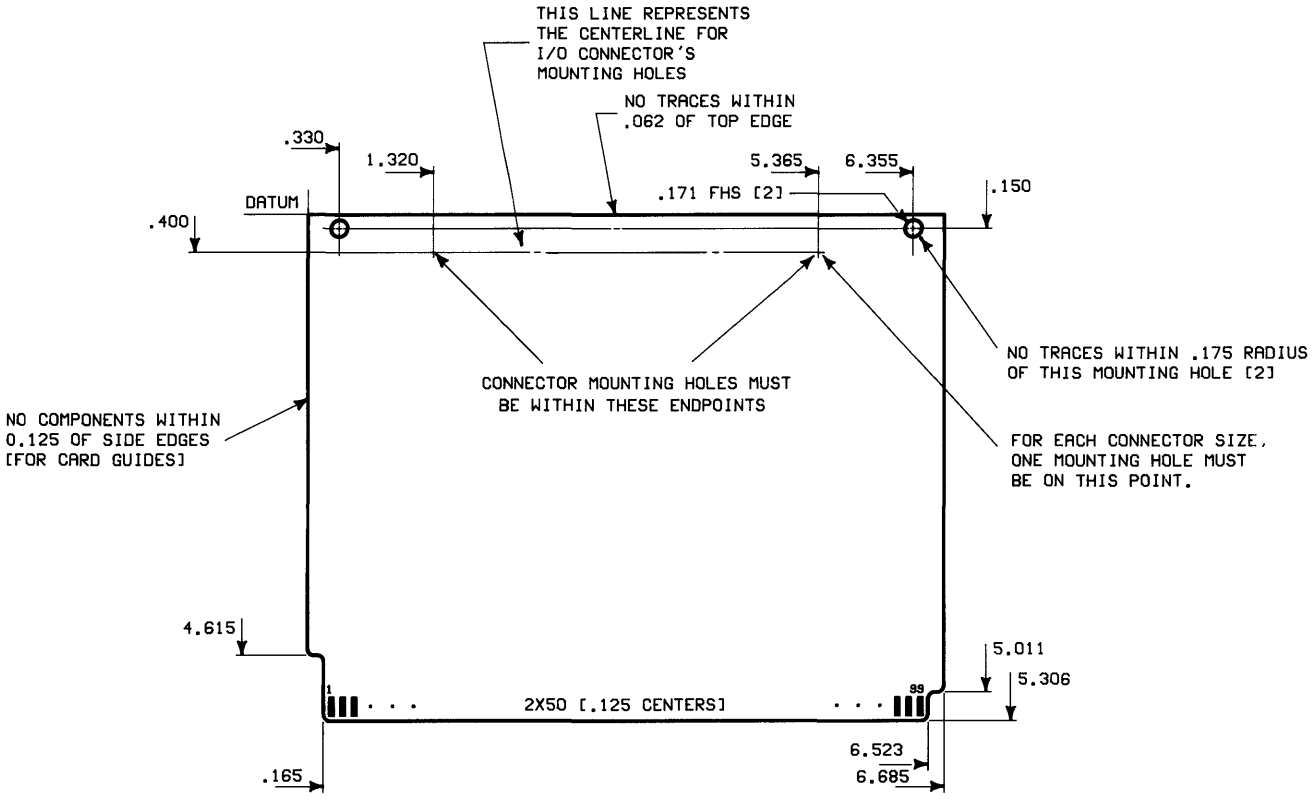


Figure 8. Blank PC Board Outline

Non-I/O-type PC boards must have a cutout to allow room for the connector of the next lower I/O board's connector, as shown in the next drawing.

Card Cage Specifications

Shown below are the specifications of the 9826A/9836A card cage. The drawing also shows the maximum vertical height of components on the boards. The rear panel is further described in the next drawing.

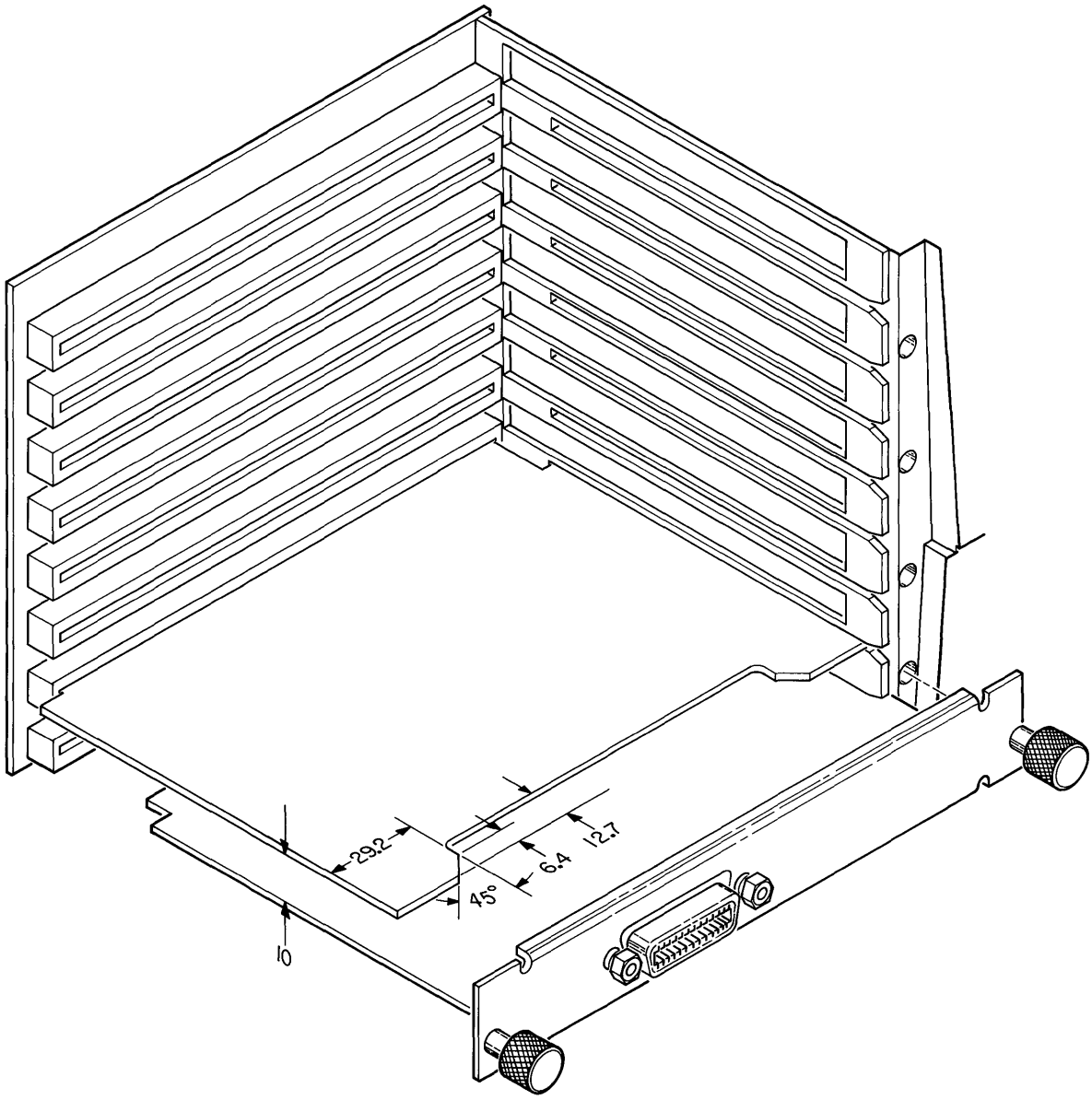


Figure 9. HP Series 200 Card Cage

I/O Card Coverplate

The coverplate for I/O cards is shown in the following drawing. Note the placement of different sizes of connectors.

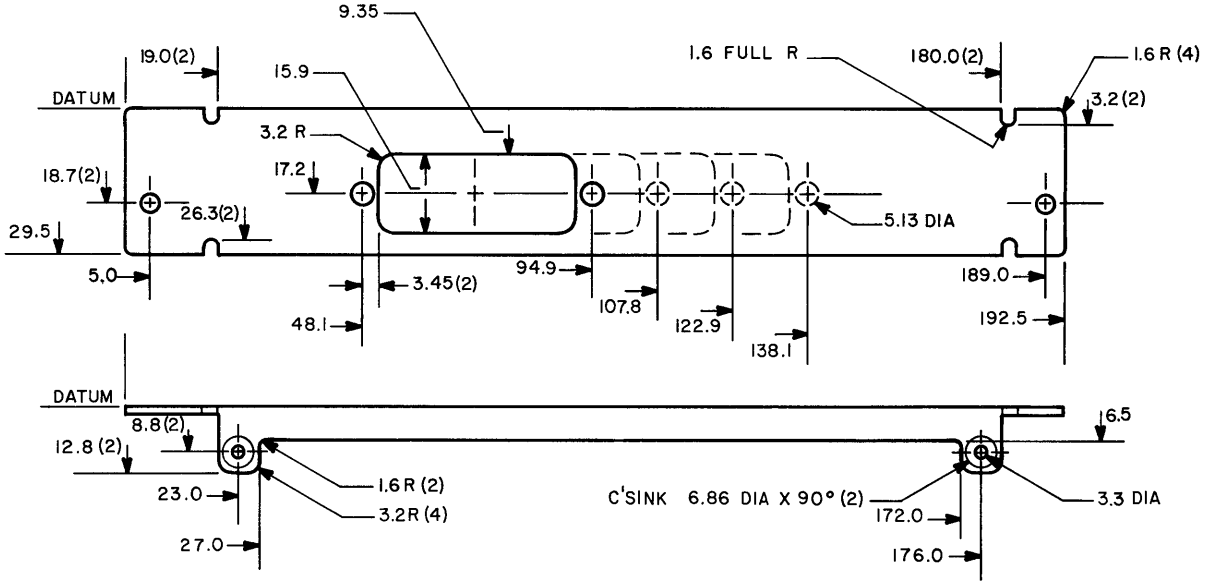


Figure 10. I/O Card Coverplate

Minimizing Electromagnetic Noise

The following rules should help to minimize electromagnetic interference (EMI) problems:

- 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 by the planes.
- I/O cards must have a sheet metal coverplate with screws which 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 coverplate. Appropriate techniques must be used to ensure solid contact between the connector housing and the coverplate. If the shield ground appears on a pin of the connector (as with HP-IB), termination can be done with a generously sized PC trace from the connector to the screwpad which secure the coverplate to the PC board.

PC-Board Layout Rules

The following PC-board layout rules should be met:

- Each card must limit loading to one LS load on the following signals: BA23-BA1, BAS*, BD15-BD0, BUDS*, BLDS*, and BR/W*.
- The PC board trace length for the above signals 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.
- PC boards should be a minimum of four layers, with planes 2 and 3 being power and ground, respectively. For boards with more than four layers, the middle layers should be power and ground.
- IC's are mounted parallel to the connector with pin 1 being in the lower left when viewing the board from the component side with connector pin 1 in the lower left corner.
- Adequate bypass capacitors are highly recommended.
- The PC edge connector is a standard S-100 connector with 100 pins on 0.125-in centers and 0.060-in fingers.

Pin Assignments

The DIO Bus pinout is shown in the following table. 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 numbers increase from left to right.

The following conventions are used in the table:

1. A "-" in front of the pin number indicates should *not* be implemented by a Bus Slave. "Spare" pins should not be used at all.
2. A "#" in front of the pin number indicates an the line is *optional* in a particular subsystem.

Component Side		Circuit Side	
1	DMAR0*	2	DMAR1*
3	DMACK0*	4	DMACK*
-5	Spare 0	-6	IR7*
-7	IR2*	-8	IR1*
9	DMARDY*	-10	BG1*
-11	BG2*	-12	BG3*
13	GND	14	GND
15	IR4*	16	IR3*
17	IR6*	18	IR5*
-19	VECTOR*	-20	IACK*
21	GND	22	GND
-23	BG*	-24	BR*
25	#DONE*	-26	BGACK*
-27	Spare 1	-28	Spare 2
-29	BDRV* (BMON*)	30	#ENDT*
-31	BFC0	-32	BFC1
-33	BFC2	34	DTACK*
35	GND	36	GND
37	RESET*	-38	BERR*
39	GND	40	GND
41	#IMA*	-42	FOLD*
43	BLDS*	44	BUDS*
45	BR/W*	46	BAS*
47	GND	48	GND
49	HALT*	50	BA1
51	BA2	52	BA3
53	BA4	54	BA5
55	BA6	56	BA7
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	+5 V	86	+5 V
87	BD8	88	BD9
89	BD10	90	BD11
91	BD12	92	BD13
93	BD14	94	BD15
-95	DGND	-96	DGND
-97	Spare 3	-98	Spare 4
99	-12 V	100	+12 V

Operation in the Bus Expander

Devices for the HP 9888 Bus Expander are required to meet the same electrical and mechanical constraints as devices which plug directly into the I/O cardcage. The Bus Expander employs delay lines and latches to ensure that all timing requirements are met. This chapter discusses features of the Bus Expander and several limitations affecting operation of DIO Bus devices in the expander.

Features of the Bus Expander

The features of the Bus Expander are as follows:

- The Bus Expander card plugs into an I/O slot in the computer. Up to 4 Bus Expanders can be plugged into a computer; however, an expander may not be attached to another expander.
- The Bus Expander is totally self-powered.
- The Bus Expander has sixteen slots which will support eight I/O cards and eight non-I/O cards, or sixteen non-I/O cards.
- 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 I'm Addressed signal (IMA*) discussed previously is used to "turn the buffers around" if the addressed card is in the expander.

Operating Limitations With the Expander

In designing DIO Bus devices, designers should be aware of the following limitations when operating in the 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.
2. RAM boards will operate in the Bus Expander; however, RAM boards require six-state accesses as opposed to five-cycle access for boards installed in the computer. This is due to signal delays. Software being executed from the Bus Expander will thus run proportionately slower and timing loops will be altered.
3. For information only, the 9826/36 Powerfail option (which is installed internal to 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.

Design Summary

Many details covering Bus Slave design have been covered in other sections. The key requirements are summarized below.

I/O Card Design Guidelines

The following design guidelines are for external I/O cards only. Following these guidelines is a sample design highlighting key features.

- I/O cards should provide five select code switches to allow select codes settings of 0 through 31.
- I/O cards must implement the four standard I/O registers (and bits thereof) described in the Registers section. Additional registers may be defined as needed.
- I/O cards may be either 8-bit or 16-bit devices.
- Implementing interrupt capability is left to the option of the designer; however, it is strongly recommended that interrupt capability be included. I/O cards that implement interrupt capability must have switches to select interrupt levels 3 to 6.
- Implementing DMA capability 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 DMA is done.

I/O Card Design Example

This section presents an example 8-bit I/O card design and describes the following interface elements: data-transfer, interrupt, and DMA.

Data-Transfer Interface

Figure 11 shows a typical circuit used for transferring data between the I/O device and the Processor. The key steps that occur while accessing the card are as follows:

1. An Address Comparator, the 74LS688, compares the upper 8 address bits (BA23-BA16) to the bit pattern for the External I/O memory space (011) and the user-set select code switches (00000 through 11111). This comparison is enabled by BAS*. The Card Select output (CS*) is generated when the addresses match. CS* does not have glitches, because the address precedes BAS* by 15 ns.
2. CS* generates IMA* and enables the DTACK* buffer. CS* is not used to enable the Data Bus Buffer, the 74LS245, because of a possible driver conflict between the I/O card and the Processor Board. This conflict would otherwise occur during a write cycle when BR/W* is asserted while BAS* is low; BR/W* low would cause the Processor Board buffers to drive the data bus while the I/O card was still driving the data bus (until it recognizes that BR/W* is low, which takes 15-20 ns). To avoid this, the I/O card does not enable its Data Bus Buffer until BLDS* is low.

3. When CS* and BLDS* are both asserted, LDCS* is generated (indicating the start of an access cycle). LDCS* generates DTACK* (after some delay), regardless of whether the operation is a read or a write. The delay time should be set to the longest time required for either reading a register (plus 30 ns setup on the bus) or setting up a register for clocking when DTACK* is asserted. LDCS* also enables the Data Bus Buffer.

4. Reading and writing are then determined by BR/W*, and the steps of both processes are as follows:

   a. While Reading, BR/W* high enables the Data Bus Buffer to drive the DIO Bus data lines (BD7 through BD0).

   b. LDCS* low and BR/W* high enables the Register Select chip, the 74LS138, which generates a read strobe for the device at the address selected by BA2 and BA1. The addressed device then drives the I/O card's data bus. As mentioned above, the DTACK* delay circuit should be designed to ensure 30 ns of setup time prior to driving DTACK* low.

   c. The card remains in this state until BLDS* or BAS* goes high, disabling the Data Bus Buffer and the Register Select chip.

   a. While Writing, BR/W* causes the Data Bus Buffer to drive the I/O card's data bus.

   b. LDCS* low and BR/W* low enables the Register Select chip, which generates a write strobe at the device selected by BA2 and BA1.

   c. Generation of DTACK* ends the write strobe. The data hold time for the register is guaranteed by the minimum response time of the Bus Master to DTACK* low (85 ns).

   d. The write cycle ends when BLDS* or BAS* goes high.

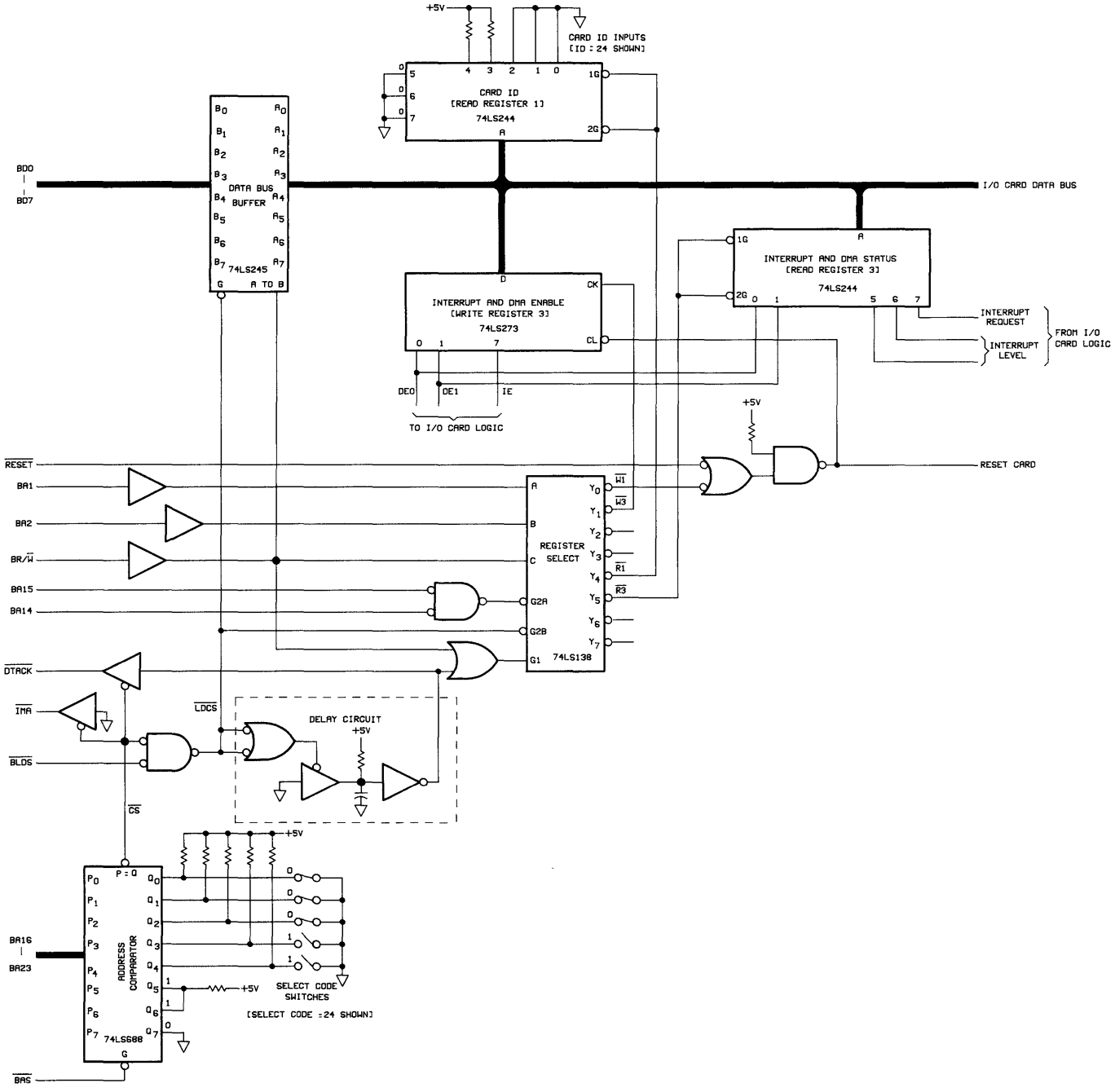


Figure 11. Example Data-Transfer Interface Design

Interrupt Interface

Figure 12 shows a typical interrupt request circuit. The circuit operates as follows:

1. If Interrupt Request (IR) and Interrupt Enable (IE) are true, one of the interrupt request signals (IR3*, IR4*, IR5* or IR6*) will be low, as determined by the two Interrupt Level lines (IL0 and IL1).

2. I/O cards are polled by software to determine which card is interrupting. When bit 7 of the Interrupt and DMA Status Register (Read Register 3) is set (to 1), interrupts are enabled; if bit 6 is set (1), an interrupt is being requested by the card (or peripheral connected to the card).

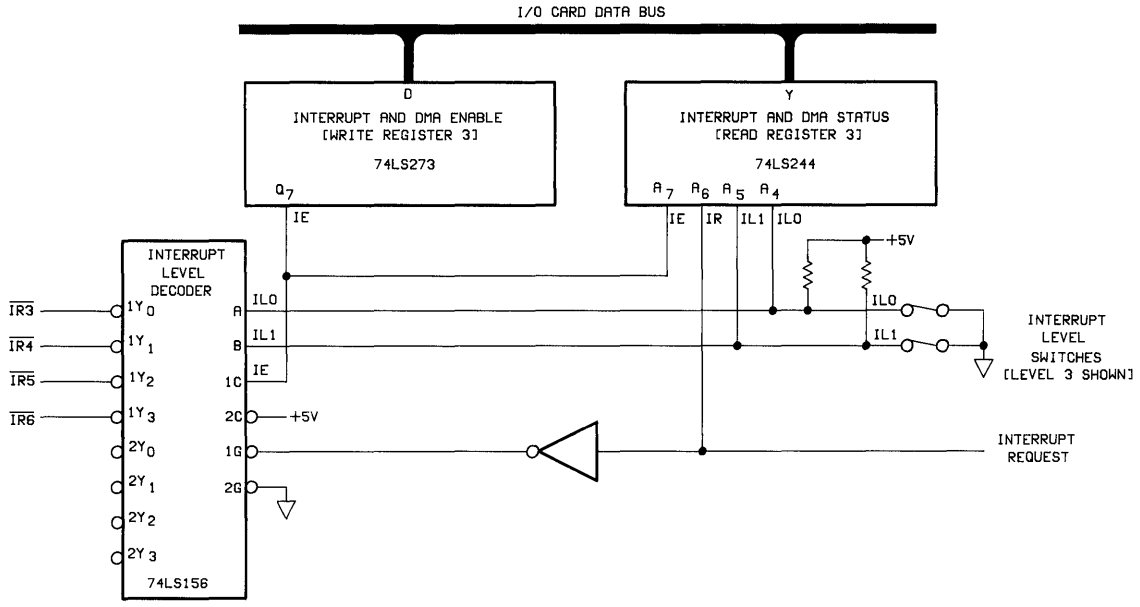


Figure 12. Example Interrupt Interface Design

Note

External interrupt vectoring is not supported with the DIO Bus.

DMA Interface

Figure 13 shows a typical DMA interface. It operates as follows:

1. If a DMA channel is enabled (by setting bit DE0* or DE1* in the Interrupt and DMA Enable Register, Write Register 3), a DMA Request signal from the I/O card drives DMAR0* (or DMAR1*) low.

2. DMACK0* (or DMACK1*) enables the Data Bus Buffer. During a normal R/W operation, the direction of this buffer is determined by BR/W*. During a DMA operation, the exclusive-OR gate inverts BR/W* to control the direction; this is because BR/W* is intended for memory so the I/O card uses it in the opposite way.

3. The I/O card generates DMARDY* just as it normally generates DTACK* during a R/W cycle. For a DMA input cycle, the delay timing starts immediately with BR/W* high. For a DMA output cycle, the delay timing is not started until valid data is on the bus as indicated by DTACK* from the memory device.

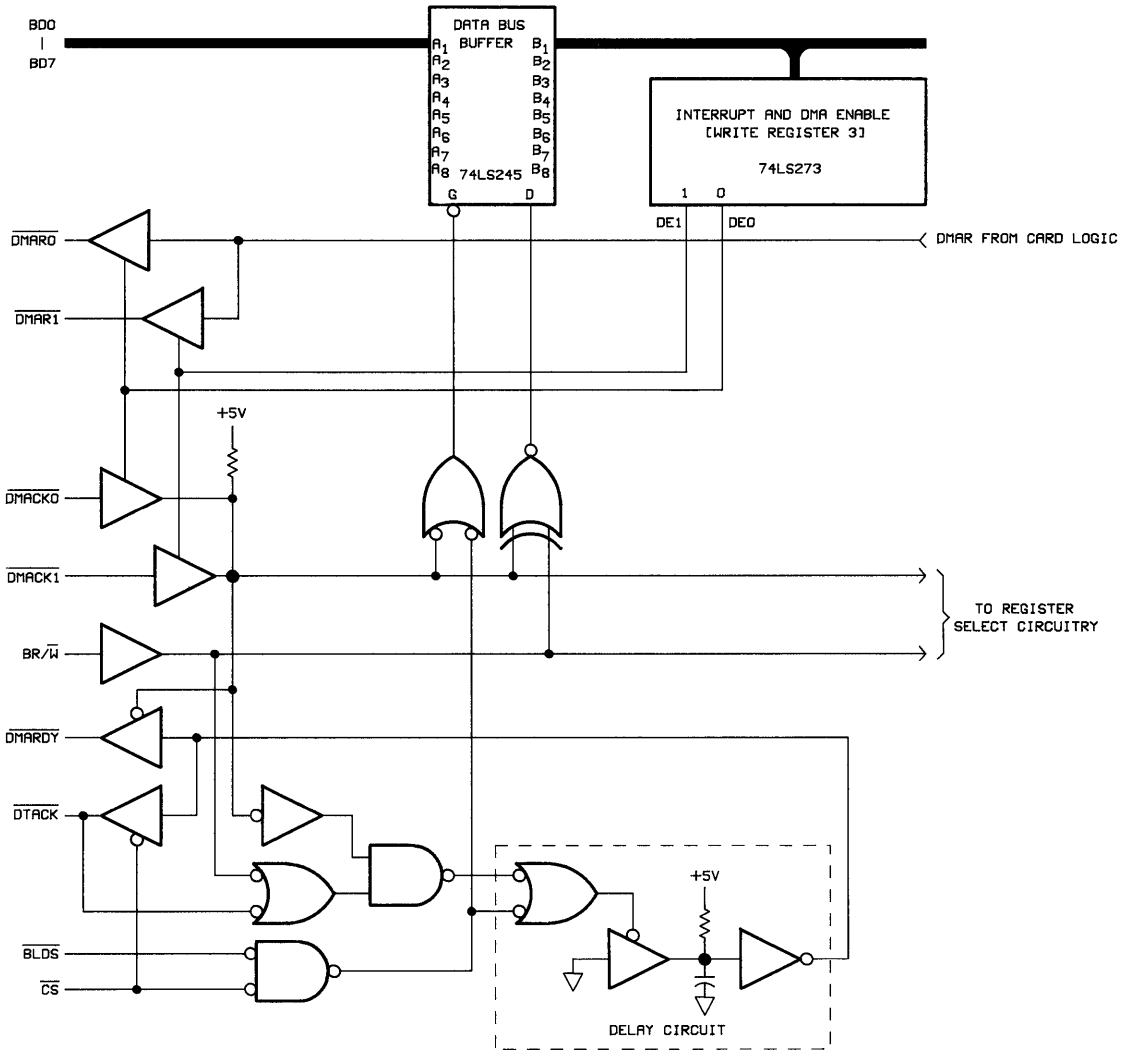


Figure 13. Example DMA Interface Design

Design Qualification

All HP Computers and supporting equipment are designed according to rigorous standards and thoroughly tested to ensure that they meet these high standards. Every new design for an I/O card to be used with Series 200 Computers should likewise be carefully qualified for adherence to acceptable design standards. Qualification for new cards can be broken down into the following areas:

- Safety compliance
- Hardware qualification
- Software qualification

This section outlines qualifications that should be made before finalizing any Bus Slave design.

Safety Compliance

I/O cards must be designed to meet *all* UL, CSA, and IEC safety requirements. This requires that the I/O card's coverplate *always* be secured to the computer chassis with dog bolts and that the I/O connector and cable be adequately grounded to the coverplate. This configuration must meet the following requirements:

1. Ground-current carrying capacity: The conductive path between the I/O cable (cable shield, connector ground pins, and connector shell) and the safety ground pin of the power receptacle must be capable of carrying 30 amperes for 120 seconds. This requirement can be expressed in ohmic resistance for the following two cases:

   a. For cable lengths 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.

   b. For cable lengths greater than 4 meters, the dc resistance between the I/O connector on the card's coverplate (cable shield, connector ground pins, and connector shell) and the safety ground pin of the power receptacle should be less than 100 milliohms.

WARNING

I/O CABLES WHICH ARE NOT GROUNDED AS STATED ABOVE PRESENT A POTENTIAL SHOCK HAZARD TO THE USER OF THE EQUIPMENT.

---

2. Fault-current carrying capacity: Because the +5V supply in the Series 200 computers has over an 8-ampere fault-current capability, an on-card fuse is required. Refer to the "Electrical Specifications" section for the recommended fuse.

CAUTION

AN I/O CARD NOT EQUIPPED WITH THE PROPER FUSE IS CONSIDERED TO BE A MISUSE OF THE EQUIPMENT AND MAY RESULT IN A PERSONAL HAZARD TO THE OPERATOR AND/OR EQUIPMENT DAMAGE.

Hardware Qualification

Hardware testing can be divided into two areas: environmental testing and configuration testing. Environmental testing involves testing the I/O card throughout the range of operating environments. Configuration testing involves testing several different I/O operations with all significant mainframe configurations.

In performing environmental tests, a subset of possible configurations are selected for testing in the following areas:

- Initial testing of a small sample of prototypes: Testing at this stage involves high electrical, thermal, and mechanical stresses of short duration, often to the level of inducing failures so as to identify weak points of the hardware.

- Strife (stress+life) testing of a larger sample of production devices: Testing at this stage involves thermal cycling and vibration tests derived primarily from MIL-Std-810B. These tests also involve forcing failures, determining the cause, and implementing a solution. However, the emphasis is on determining failure modes and assessing margins. It is also the first opportunity to search for production-process related failure mechanisms.

  Approximately 50 thermal cycles are performed with the range of temperatures varying from -20 to 65 °C. The time for each cycle is adjusted so that the units reach thermal equilibrium during the dwell portion of each cycle.

  Random vibration tests are performed periodically with a range of accelerations from 1-2 g (9.8-19.6 m/s/s) rms. Total accumulated time is on the order of 25 minutes.

- Environmental testing of a small sample of units: This stage of testing involves a complete set of tests known as HP Class B (Industrial and Commercial) Environmental tests. Testing is performed in the following areas for adherence to the stated standards:

Type of Test	Description
Temperature:	
Non-operating (storage)	-40 to +75 degrees Celsius
Operating (survival)	-20 to +65 degrees Celsius
Normal Operating	0 to +55 degrees Celsius
Humidity:	
Operating	40 degrees Celsius at 5 to 95% Relative Humidity
Non-operating	65 degrees Celsius at 90% Relative Humidity
Condensation	Operates without damage and recovers within specified limits.
Vibration:	
Cycle Range	5-55-5 Hz.
Amplitude (p-p)	0.38 mm
Sweeptime	15 min. (1 min./octave)
Dwell @ Resonances	10 min. at each resonance
Ampl. @ Resonances	3.17 mm @ 5-10 Hz.
	1.52 mm @ 10-25 Hz.
	0.38 mm @ 25-55 Hz.
Shock:	
Magnitude	30 g (approx. 294 m/s/s)
Duration	11 ms
No. Shocks	18 (3 on ea. of 6 surfaces)
Waveform	Half-sine
Bench handling	102 mm tilt drop
Altitude:	
Non-operating	15 300 m
Operating	4 600 m

- RFI testing: This type of testing requires compliance with VDE Level B (with a 2 db margin) and FCC Class B standards.

Software Qualification

The key concern with software is, of course, its reliability. Sufficient testing should be performed to ensure that the card operates properly with the desired operating system(s). When operating systems are revised, new operating systems are released, and when changes are made to an I/O card which affect its operation, additional software testing should be performed.

Part No. 98630-90000
E0383
Microfiche No. 98630-99000

Printed in U.S.A.
First Edition, March 1983