HEWLETT—PACKARD

CONFIDENTIAL

DIO BUS SPECIFICATION

VERSION 1.0

APRIL 16, 1984

HP DRAWING NO: 5955—7669

DOCUMENT CONTROL

CHANGES TO THIS DOCUMENT SHOULD BE COORDINATED THROUGH THE RND R&D LAB.

DISTRIBUTION

THIS DOCUMENT IS DISTRIBUTED BY THE RND R&D LAB.

PRINTING HISTORY

First draft: December 14, 1982
Second draft: March 27, 1983

Version 1.0: April 16, 1984

TABLE OF CONTENTS

FOREWARD — CHAPTER 1

This standard is the result of a cooperative effort involving engineers at FSD, RND, CSY and DSD during the period from November 1982 to the present. The following people have contributed to this document; their assistance and the assistance of others is greatly appreciated:

FSD: Doug Buhler
John Byrnes
Steve Chorak
Greg Herman
Bill Hirth
Greg Lawson
Nick Mati
Tim Mikkelsen
Shaw Moldauer
Jon Rubinstein
Martin Speer
Dan Swanson
Tom Thrasher
Keith Weeks
Jerry Wick
Steve Wolf

RND: Vince Cavanna
Steve Haddock

CSY: Paul Zimmer

DSD: Tom Szolyga

In addition, the assistance of our managers is acknowledged:

FSD: Ken Watts

RND: Doug Boliere

Again, thank you for your cooperation and assistance.

Dave Sweetser, FSD

2.1 OBJECTIVES OF THE DIO BUS SPECIFICATION

The DIO (Desktop computer I/O) BUS was first introduced in 1981 with FSD's 9826A computer along with several I/O cards. Since then, several other DIO BUS mainframes and I/O cards have been introduced. Because several divisions will be designing or using hardware based on the DIO BUS, a decision was made to document the DIO BUS as an HP standard. The objectives of this effort are two-fold:

1. The primary objective is to provide sufficient information to design a DIO BUS slave device, such as an I/O card. However, while the DIO BUS is the system bus for several mainframes, future products may utilize the DIO BUS solely as an I/O bus. A DIO-based I/O bus does not require all of the functions presently defined by the DIO BUS. Therefore, the Chapter REQUIRED DIO BUS FUNCTIONS FOR AN I/O BUS lists those signals which would not supported on a DIO-based I/O bus. I/O card designers must adhere to the limitations specified in this chapter.

2. A secondary objective is to support the design of Bus Masters. While Bus Master design is not covered explicitly, the DIO BUS specifications must be followed by Bus Master designers to guarantee that Bus Slave timing requirements are met. Additional Bus Master guidelines are available in the document SERIES 200 SYSTEM SPECIFICATION, available from the FSD Hardware R&D Lab. The purpose of that document is to support Series 200 system design activities.

2.2 PROCEDURE FOR RESOLVING CHANGES

The specifications provided in this document must be followed. In certain cases, however, a designer may not be able to comply with one or more specifications. In such a case, the designer must either: (1) seek to change the specification or (2) obtain approval for area(s) of non-compliance.

Changes to this document require a consensus from FSD, RND and any other division currently using the DIO BUS. Change requests will be coordinated through the RND R&D Lab, with RND reviewing all change requests relative to their necessity and implementation. If an agreement is reached between RND and the requesting division, RND will submit in writing the proposed change to FSD and other using divisions for their concurrence. If all divisions concur, the change will be made. If concurrence is lacking, the change will not be made and the requesting division has the option of pursuing the change through management channels.

In a similar manner, a designer that finds it necessary to violate a specification should notify RND R&D of his intent. RND, if unable to find a solution, will notify all affected divisions in writing of the proposed violation. Each division must respond in writing, either agreeing to the non-compliance or disagreeing. In the case of a disagreement, an effort will be made to resolve it at the lowest possible level; if this is not possible, successive levels of management will become involved as necessary.

2.3 WHAT IS NOT COVERED IN THIS DOCUMENT

The following information is not contained within this document:

1. As stated above, Bus Master design is not covered in this document. Refer to the SERIES 200 SYSTEM SPECIFICATION available from the FSD Hardware R&D Lab for information on Bus Master design and other system design issues.

2. Previous releases of this document contained a card ID Table and a card SELECT CODE Table. However, because of the changing nature of this information, the ID and SELECT CODE Tables have been removed from this document; they can be obtained from the RND R&D Lab. Designers should obtain ID and SELECT CODE assignments from RND before beginning a new design.

3. Several of the DIO BUS timing and electrical specifications were derived from characteristics of the 74LS244 and 74LS245 drivers. Characterization of other logic families (e.g. ALS), however, has not been done. This is an area requiring future work.

2.4 RECOMMENDED DESIGN METHODOLOGY

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

1. Prior to beginning a project, the designer should contact RND to obtain ID and SELECT CODE assignments as well as to ensure that he has the latest version of this specification.

2. Designers should review the design of existing DIO BUS products. The products listed in the separately-available ID and SELECT CODE Tables provide a good reference of existing DIO BUS products. The divisions with ER for these products can be contacted for more information, i.e. the schematic, theory of operation, etc. Designers should also reference the SERIES 200 SYSTEM SPECIFICATION available from the FSD Hardware R&D Lab.

3. An alternative that may expedite I/O card development is the 98630A breadboard interface card available from FSD. This card contains the essential circuitry to support data transfers, interrupt requests and DMA operation and provides approximately 15 square inches for breadboarding.

4. Even though a designer follows this document rigorously, testing of the product in its intended environments is ABSOLUTELY ESSENTIAL. The Chapter I/O CARD QUALIFICATION gives guidelines for this testing.

5. One of the keys to ensuring timely completion of a product is successful interdivisional cooperation. Marketing plans, product data sheets, hardware ERS's, software specifications, etc. should be prepared by the designing division(s) and distributed to those who are involved with the development, manufacturing and marketing of these products or their host systems.

This chapter provides basic background information on the DIO BUS, including terminology, a description of bus interface elements and timing information.

3.1 SPECIFICATION TERMINOLOGY

The following terminology is used throughout this document:

1. 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:

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

2. When a signal is referenced as 'asserted', 'true', 'false', etc., it is relative to the signal FUNCTION. For example, to say that BAS is asserted means the Buffered Address Strobe is active, i.e. performing its function of strobing the address. Whether it exists as BAS or BAS* on the backplane is irrelevant.

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

   HIGH >= 2.0V

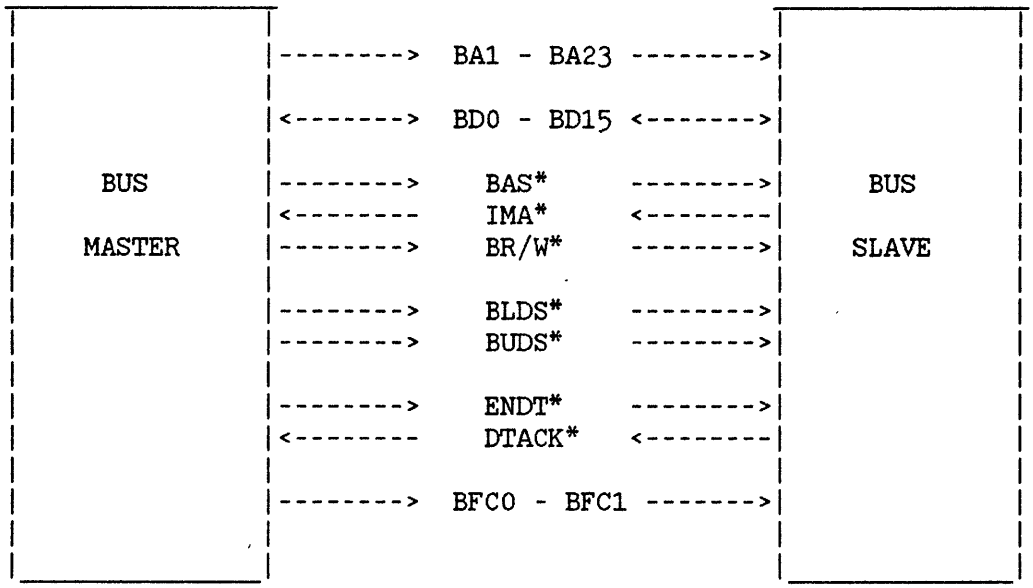
   LOW <= .8V

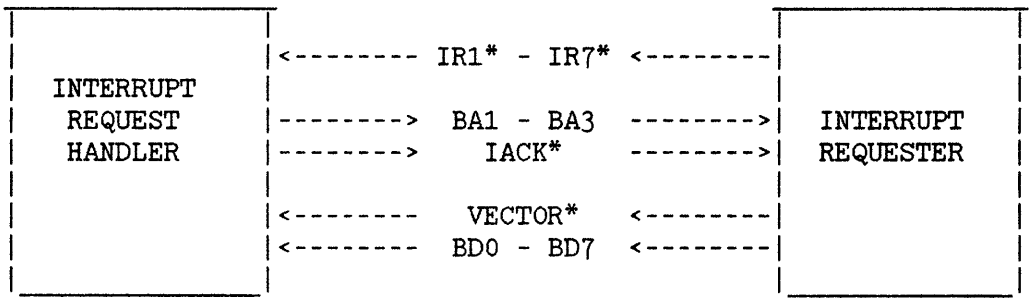
4. The composite signal BR/$\overline{W}$ is denoted as BR/W*. BR/W* high indicates a read operation; BR/W* low indicates a write operation.

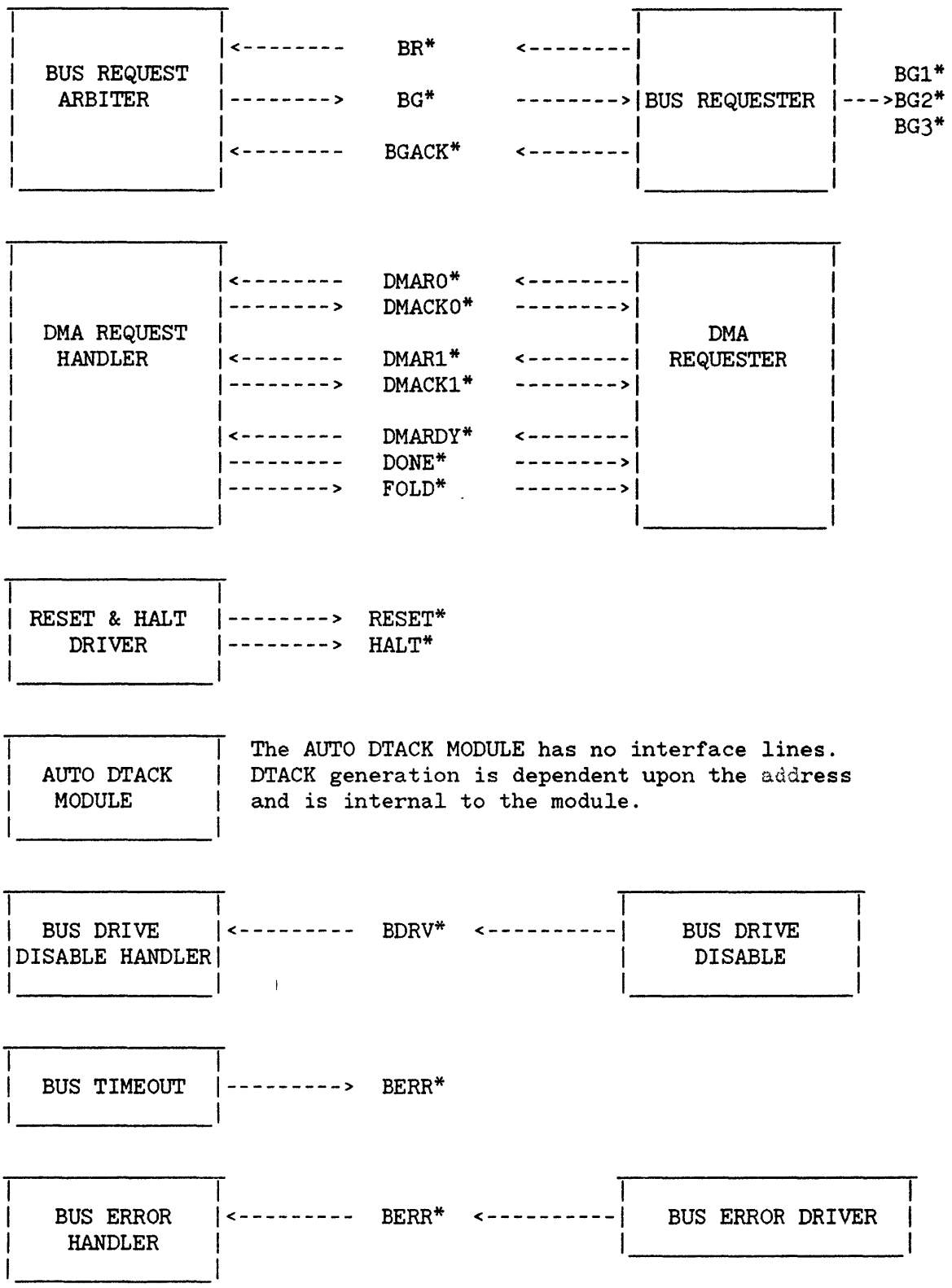
5. The definition of 'byte' is agreed upon by all -- 8 bits. A 'word' is defined to be 16 bits. The least significant bit of a byte or word is defined as Bit 0.

3.2 INTERFACE SYSTEM ELEMENTS

The functional modules of the DIO BUS are shown below. Where signals go specifically from one functional module to another functional module, the two modules are shown side-by-side for clarity. Modules that drive many other modules (e.g. RESET) are shown as stand-alone.

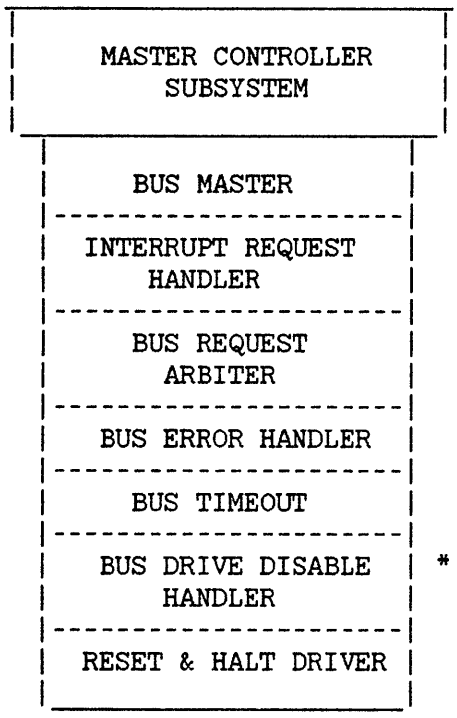





BUS REQUEST ARBITER
BR*
BG*
BGACK*
BUS REQUESTER
BG1*
BG2*
BG3*
DMA REQUEST HANDLER
DMAR0*
DMACK0*
DMAR1*
DMACK1*
DMARDY*
DONE*
FOLD*
DMA REQUESTER
RESET & HALT DRIVER
RESET*
HALT*
AUTO DTACK MODULE
The AUTO DTACK MODULE has no interface lines.
DTACK generation is dependent upon the address
and is internal to the module.
BUS DRIVE DISABLE HANDLER
BDRV*
BUS DRIVE DISABLE
BUS TIMEOUT
BERR*
BUS ERROR HANDLER
BERR*
BUS ERROR DRIVER

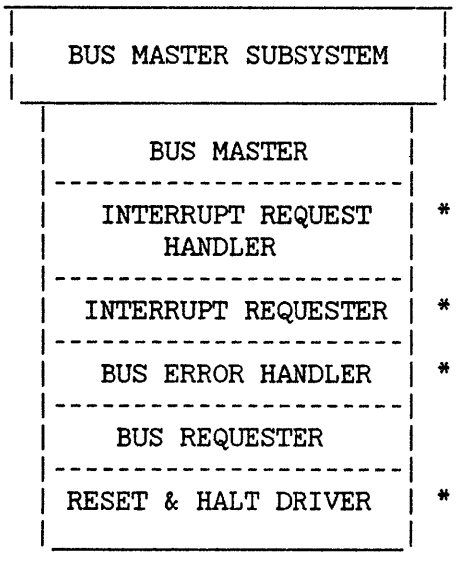
3.3 BUS SUBSYSTEMS

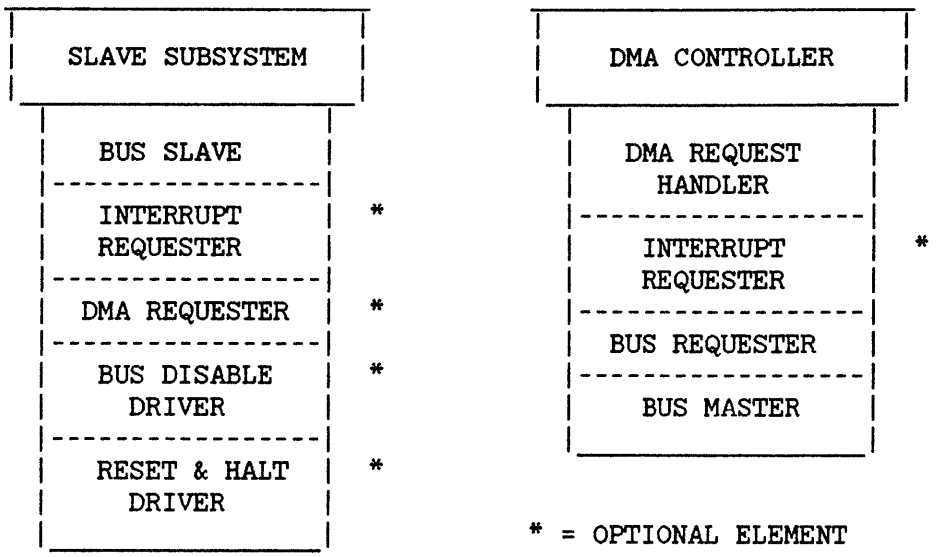
Bus subsystems that are defined are shown below:



There will always be one, and only one, MASTER CONTROLLER SUBSYSTEM. The MASTER CONTROLLER includes functional elements that can be on several boards, e.g. the POWERFAIL DRIVER signals might originate from a power supply board.

* = OPTIONAL ELEMENT





3.4 BUS TIMING BACKGROUND

Before discussing DIO BUS timing, it should be pointed out that in most mainframes (e.g. the 9836A), all slots are identical. There is no ordering or prioritizing of address, interrupt capability, etc. by location in the backplane. All timing specifications apply equally to all slots. It is possible, however, that future mainframes may have slot-dependent features (e.g. interrupt prioritizing). Thus, designers must be aware of features of each mainframe that their products will operate in.

Note also that timing specifications for the DIO BUS were developed using the 8 Mhz 68000. Designs incorporating other processors must ensure that the DIO BUS specifications are met.

The DIO BUS is ASYNCHRONOUS; that is, there is no clock on the backplane to reference signals to. While address and data generation are related to the CPU clock, the actual clock does not appear on the bus. The presence of address or data is indicated by various control lines which execute interlocked 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 68000 is guaranteed to drive the address bus lines 30 nsec prior to asserting Address Strobe. Most receiving devices require at least 15 nsec of address setup time prior to Address Strobe. To guarantee 15 nsec of address setup time, the following rules were developed to control gate delays and bus loading (these are expanded on in more detail in later chapters).

1. Each board is limited to one LS load on the address bus, data bus, the address strobe, the data strobes and the read/write signal.

2. The PC board trace length on bus signals should be as short as possible and, in any case, must not exceed 3 inches.

3. The SN74LS245 (or equivalent SN74LS244) is used to drive the above signals.

NOTE: IN SOME CASES, EXISTING DEVICES VIOLATE THESE RULES; HOWEVER, NEW DESIGNS MUST ADHERE TO THESE RULES.

At this point, a historical perspective is worthwhile to show how the above guidelines originated. During the early development of the 9826, it became clear that controlling bus capacitance was essential, hence Rules 1 and 2 above. Also, to minimize bus skew, it was decided to specify a 'standard' bus driver, hence Rule 3. Next, to develop detailed timing specifications, further analysis was performed on the 74LS245.

To define bus skew, two efforts were made: (1) model the 74LS245 and determine formulas for worst-case minimum and maximum gate delays as a function of bus capacitance and (2) measure delays for a number of 74LS245 parts with different date codes. The formulas for gate delay, if used with worst-case conditions (fully loaded bus, fast address strobe driver, slow bus driver, etc.), yield unworkable numbers (negative setup times). It was felt that such a worst-case scenario is a low probability and not the appropriate design center. Therefore, effort 2 (measuring delays) was investigated.

Using a sample of 74LS245 parts with different date codes, the worst-case difference in gate delays was measured for a 500 pf load. The difference between the fastest gate and the slowest gate (in different packages) was 9.5 nsec. This was derated by 50% to 15 nsec for margin as well as to cover skew on the bus itself caused by different signal loading. Returning to the 68000 example, if the address precedes the address strobe by 30 nsec at the 68000 output, then (using 74LS245's to drive the bus) all receiving devices are guaranteed to have 15 nsec (30 - 15) of address setup time prior to the address strobe.

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

WITH PULL UP RESISTOR: (1 kohm)	47.5 ns high to low 37.5 ns low to high
WITHOUT PULL UP RESISTOR:	39.5 ns high to low 39.5 ns low to high

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

The original DESIGNER'S GUIDE TO THE 9826 CARDCAGE had several timing diagrams labeled 'Bus Master'. These diagrams were basically the timing of the 68000 and did not represent DIO BUS signals. Such diagrams are not included in this document.

In summary, the following points can be made:

1. The skews due to the bus drivers and the bus itself are not specified separately. Instead, a 'lumped' skew specification of 15 ns is provided. Between any two signals driven by 74LS245 drivers, 15 ns of skew can develop between their inputs and outputs, where the outputs are measured at the receiving device on the bus.

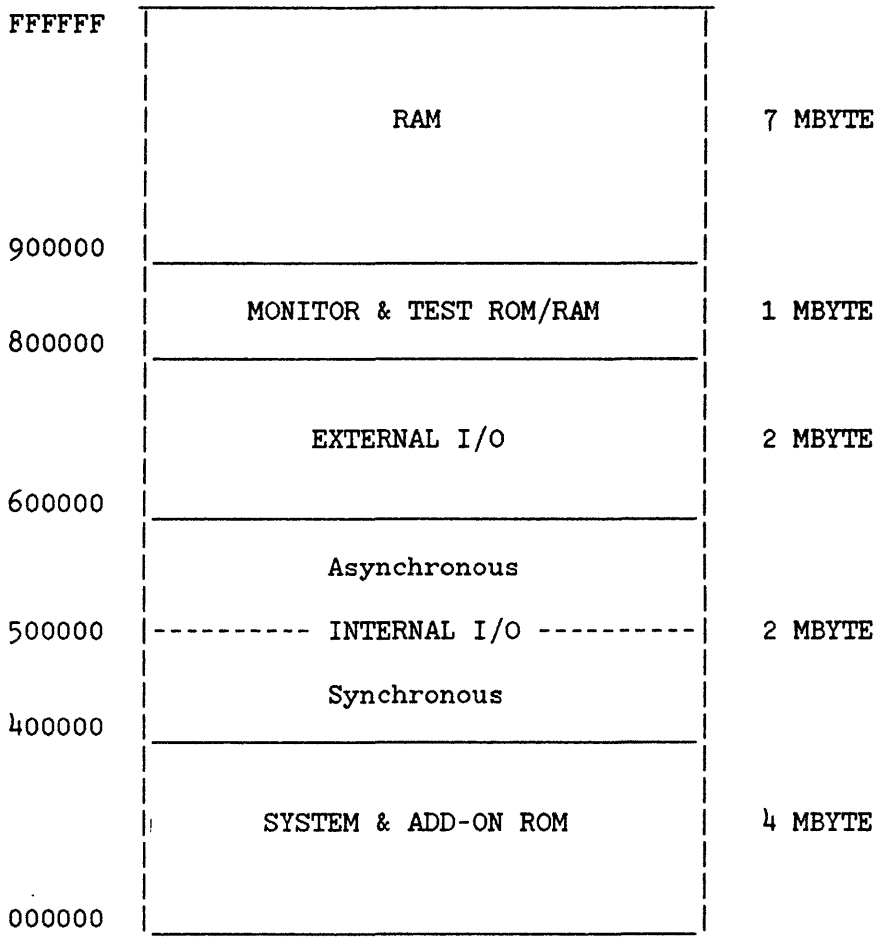
2. In many cases, the DIO BUS timing specifications have been derived using the minimum or maximum 68000 timing specifications plus or minus the 15 ns skew, whichever is appropriate.

3. The guideline does not take into account signals driven by devices other than the 74LS245. Such guidelines may be added by RND in a future release of this document.

The terms CYCLE and STATE need to be explained as these terms are sometimes used erroneously. CYCLE refers to a complete clock cycle, e.g. 125 ns for an 8 Mhz clock. STATE refers to one half of a clock cycle and is based on Motorola's nomenclature; STATES are numbered from S0 to S7, representing 8 states, or 4 cycles. In a bus transaction, the 68000 inserts wait states between states S4 and S5 as necessary. An example of erroneous usage of these terms is in reference to RAM access times: it is often referred to as a 5-state access when the correct description is a 5-cycle access.

4.1 SERIES 200 MEMORY MAP

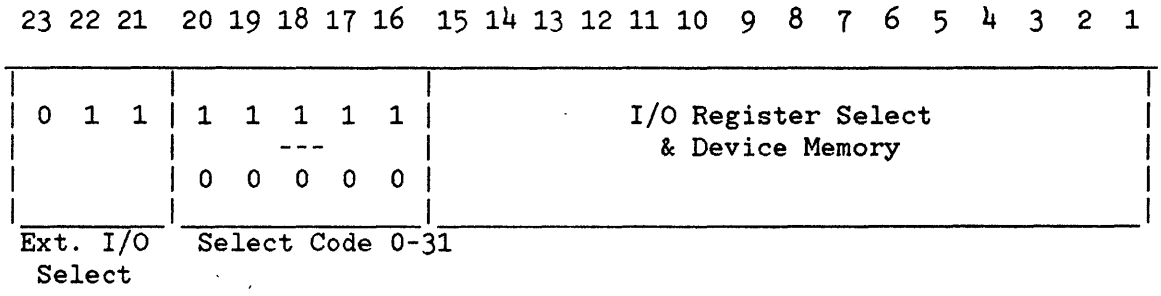
The DIO BUS SPECIFICATION is not intended to document in detail the Series 200 memory map. Instead, memory map documentation is limited to the External I/O memory map. For reference only, the Series 200 Memory Map is shown below.



This Series 200 memory map is shown above; addresses are in HEX. The 68000's 24 bit address can address 16 Mbytes of memory. The External I/O occupies 2 Mbytes (600000 - 7FFFFF). The Internal I/O address space (400000 - 5FFFFF) is used for internal peripherals (e.g. the graphics board) and is discussed in detail in Appendix A.

4.2 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. 5 switches permit the user to set 32 Select Codes, from 0-31, to determine which 64 Kbyte memory space the card resides in. The address format is shown below:



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

With Pascal and BASIC, all Select Codes from 0-7 reference internal I/O devices only. It is important to realize that, ELECTRICALLY speaking, I/O cards can be set to Select Codes 0-7 but that the OPERATING SYSTEMS map Select Codes 0-7 to addresses in the Internal I/O space. Thus, with BASIC and Pascal, I/O cards set from 0-7 are inaccessible; only I/O cards set from 8-31 can be accessed. At the assembly language level, however, I/O cards with Select Codes over the entire range of 0-31 can be accessed.

HPL permits setting of I/O cards to Select Codes 1-6 and 8-15; refer to HPL OPERATING MANUAL AND PROGRAMMING UPDATE FOR THE HP 9826 AND 9836 COMPUTERS, 09826-90040, for more information.

4.3 I/O CARD REGISTERS

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

The defined I/O registers are:

ADDRESS 1	7	6	5	4	3	2	1	0
READ ID	R/L*	SECONDARY ID1	SECONDARY ID0	PRIMARY ID				
WRITE RESET	x	x	x	x	x	x	x	x

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

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

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

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

This feature has not been used to date (except for the 98628 card, where the card's firmware monitors this bit). In future data communications cards (e.g. a terminal multiplexer card, Ethernet, etc.), it is recommended that this function be provided, either via a jumper or a switch. Non-communications oriented cards (e.g. an A/D card) should set this bit to 0. NOTE: Except in rare cases, software is lacking to use this feature; if the remote feature is to be used, software development is required.

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

Whereas the Select Code bits are switch-selectable, the ID bits must be hardwired. Designers should obtain their ID assignment from RND. If possible, the ID and default Select Code should be the same.

SECONDARY ID0,1 -- The 2 SECONDARY ID bits are used to provide 4 additional IDs for each PRIMARY ID. Although these bits are located in a higher order position than the PRIMARY ID bits, they should be consider lower order in that they are used to extend the range of each of the PRIMARY ID's defined in Bits 0-4.

Initially, only bits 0-4 defined the device ID; software typically masked off the upper 3 bits and examined the lower 5 bits. However, because future Internal I/O devices as well as I/O cards will have IDs, it became necessary to increase the number of IDs. Accordingly, Bits 5 and 6 have been defined as the SECONDARY ID bits. Formerly these bits were S (Smart Card Identifier) and R (Reserved), respectively.

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

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

As with the Select Code table, the ID Assignment Table has been removed from this document and is available from the R&D Lab at Roseville Network Division.

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

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

IE - Interrupt Request Enabled, set or cleared by a Write Control, is read by a Read Status. IE is cleared by both a bus reset (RESET* = low) and a card reset (write to Register 1).

IR - Card is requesting interrupt, used for software polling to determine interrupt origin. If IR is true and IE is set, one of the DIO BUS interrupt lines (IR3*, IR4*, IR5* or IR6*) is asserted, depending on the Interrupt Level switches. A software-accessible means of clearing IR must be provided on the card. IR should NOT be cleared by a read of this register.

INTERRUPT LEVEL - These bits permit reading of the interrupt level as shown below:

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

Current I/O cards have 2 switches to set the interrupt level; these switches map into the 2 INTERRUPT LEVEL bits. Alternately, a card could permit the interrupt level to be programmable (e.g. using the corresponding bits in the writeable CONTROL REGISTER). Refer to the Chapter DIO BUS SLAVE DESIGN SUMMARY for more details on interrupt levels.

DE0, DE1 - DMA Enabled on Channel 0, DMA Enabled on Channel 1, set or cleared by a Write Control, read by a Read Status. If the card does not implement DMA, these bits can be used for other functions in both the Control and Status Registers. DE0 and DE1 are cleared by both a bus reset (RESET* = low) and a card reset (write to Register 1).

ADDRESS 5	7	6	5	4	3	2	1	0
READ EXTENSION ID	EXTENSION ID (0 - 255)							

The EXTENSION ID register is valid only if the PRIMARY ID in Register 1 is 0.

This chapter discusses the transfer of data between Bus Masters and Bus Slaves. Timing background information is given in Chapter 3 and should be reviewed prior to reading this chapter.

5.1 DATA TRANSFER SIGNALS

The bus signals used in data transfers are shown below. Signal names starting with B (buffered) are derived from 68000 signal names -- the 68000 name is that which follows the B. A brief description of each signal is given; for more detailed information, refer to the 68000 data sheet. Two of the signals, IMA and ENDT, are HP-defined.

1. BA1-BA23 — The 23 bit address bus. Note that BA0 is not on the bus; its value is conveyed in BUDS and BLDS.

2. BAS* — BUFFERED ADDRESS STROBE, defines when the address is valid, used to delimit bus cycle.

3. BD0-BD15 — BUFFERED DATA 0 - 15, the 16 bit data bus.

4. BR/W* — BUFFERED READ/WRITE, high for read, low for write.

5. BUDS*, BLDS*, BDS* — BUFFERED UPPER DATA STROBE, BUFFERED LOWER DATA STROBE and BUFFER DATA STROBE. BUDS indicates that BD8-BD15 are involved; BLDS indicates that BD0-BD7 are involved. BDS is NOT a bus signal and is used generically to refer to either BLDS or BUDS.

6. DTACK* — DATA TRANSFER ACKNOWLEDGE is issued by the Bus Slave (RAM, I/O card, etc.) to inform the CPU that it can complete the memory cycle. During a read, it indicates that the Bus Slave's data is valid on the bus. During a write operation, it indicates that the Bus Slave has accepted the data.

7. IMA* I'M ADDRESSED, an output from a card that is addressed. This is used by the Bus Expander to reverse its data bus buffers if a card in the Bus Expander is addressed. All DIO cards must generate IMA, including DMA Controllers when registers are read (the 98620 DMA Controller does not generate IMA, which has complicated system design).

8. ENDT* ENABLE DTACK, generated by Processor boards, used by Bus Slaves to control generation of DTACK. Permits a pseudo-synchronous, repeatable access time equivalent to 5 clock cycles.

9. BFC0-BFC2 BUFFERED FUNCTION CODES, defines type of transaction occurring on the bus.

5.2 OVERVIEW OF THE DATA TRANSFER

BAS, which defines when the bus address BA1-BA23 is valid, begins the data transfer operation. BR/W* defines whether the operation is a read or a write. BLDS and BUDS indicate which byte(s) of the 16 bit data bus are involved in the data transfer.

An interlocked handshake is used to transfer data between the Bus Master and the Bus Slave. DTACK (Data Transfer Acknowledge) is the signal that indicates when the Bus Slave has accepted data from the bus or provided data to the bus. The data handshake occurs as follows:

1. Bus Master asserts BA1-BA23, BAS, BR/W, BUDS, BLDS and the data bus BD0-BD15 (if a write operation).

2. Bus Slave asserts DTACK when data accepted from bus or provided to bus.

3. Bus Master negates BAS when DTACK seen (cycle complete).

4. Bus Slave negates DTACK when BAS negated.

DTACK is needed because the response time of devices can vary. For example, the access time of RAM can vary due to refresh. However, other devices, such as the ROM card, are guaranteed to respond to accesses in a fixed time. To simplify design for those devices that respond in a fixed time, certain Processor Boards implement a feature called AUTO DTACK in which the DTACK signal is generated on the Processor Board itself. Processor Boards generate DTACK based on the device address; DTACK is generated for all devices in the address range 0-4FFFFF. This covers ROM cards and devices in the Internal I/O space 400000-4FFFFF. The actual BAS-to-AUTO DTACK timing depends on address bits A14 and A15. Refer to the SERIES 200 SYSTEM SPECIFICATION available from FSD Hardware R&D Lab for more details.

The DIO SPEC permits DTACK* to be driven either by a tri-state buffer or an open collector gate. Assuming a 500 pf bus and a 3.3K pullup resistor (as used on the 09826-66515 CPU board) on DTACK*, an RC time constant of 1.6 usec results. Because this would result in an unacceptably slow rise time on DTACK*, Bus Masters actively pull up DTACK* after a cycle when BAS* goes high. Likewise, Bus Slaves release DTACK* when BAS* goes high but with some inherent delay relative to the active pullup.

This contention (Bus Master pulling DTACK* high, Bus Slave momentarily holding it low) has been observed to cause glitches on DTACK*. This contention problem has been fixed in new processor designs (e.g. by using a delay line to delay the active pullup). However, because of the many machines in the field which could have DTACK* glitches, DTACK* should not be used as a clock (at least not on the rising edge).

Another signal related to DTACK is ENDT (Enable DTACK). ENDT is generated by the Bus Master and may be used to improve the response time of Bus Slaves. On 68000-based Bus Masters, ENDT is basically Address Strobe delayed by 1-1/2 clock cycles. Bus Slaves can optionally use ENDT to generate an 'early' DTACK, permitting 5 cycle read/write accesses. ENDT is discussed in detail in Sections 5.6, 5.7 and 5.8.

5.3 READ CYCLE

Figure 5-1 depicts a read cycle. As discussed in Chapter 3, timing requirements are specified on the bus measured at the signal receiver. The key aspects of a read cycle are:

1. Prior to the beginning of the read cycle, BR/W*, BUDS* and BLDS* are set high by the CPU. A pullup device on the Processor Board actively pulls up DTACK*. The setup time on BR/W* high is 15 nsec 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 BA1-BA23 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 70 ns after BAS is asserted, i.e. the receiver must see IMA within 70 ns of BAS.

4. If a Bus Slave is being addressed, it puts data on the bus after BDS goes true. Note that BDS can precede or follow BAS by up to 75 ns; even though BDS may precede BAS, the Bus Slave cannot drive the bus until it has decoded that it is being addressed. Bus Slaves must not drive the upper/lower data lines unless they are specifically strobed; otherwise, folding by the DMA Controller can be affected.

   NOTE: It is worthwhile to use the above BAS to BDS timing as an example of how some of the timing specifications were derived. Relative to the 68000 clock, BDS or BAS occurs a maximum of 60 ns after the clock. Assuming (very worse case) that one signal occurs 0 ns after the clock and the other occurs 60 ns after the clock, then the signals can be 60 ns apart (in either direction). Adding in 15 ns for bus skew yields the 75 ns specification.

5. The time from BDS to data valid is device-dependent; DTACK is generated by the Bus Slave to qualify this data. There are 3 cases affecting data setup time prior to DTACK:

   A. When using ENDT, data may be placed on the bus after DTACK. Refer to Sections 5.6 and 5.7 for more detail.

   B. If the device does not use ENDT and cannot be a DMA source, then the data setup time prior to DTACK is 15 ns. As usual, this setup time must be met at the receiving device.

C. If the device being read is a DMA source (e.g. memory during a DMA output), then an additional 40 ns of data setup time prior to DTACK must be provided, or 55 ns total. The additional 40 ns is required due to the Fold Buffer delay when transferring data from the upper byte of memory to the low data byte for the I/O card. DMA and Fold Buffer operation is discussed in the Chapter DMA OPERATION.

   For I/O-to-memory DMA (DMA input) transfers, the data setup time from the I/O card is still specified at 15 ns, measured with respect to DMARDY (which the DMA Controller uses to generate BLDS and/or BUDS for the RAM). Unlike DMA output, additional setup time is not required because there are matching delays in both the data (Fold Buffer delay) and in generating BLDS and/or BUDS from DMARDY.

6. The Bus Master detects that DTACK has occurred, accepts the data on the bus and then ends the cycle by negating BAS and BDS. The Bus Master must set BAS false within 350 ns of assertion of DTACK.

7. 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.

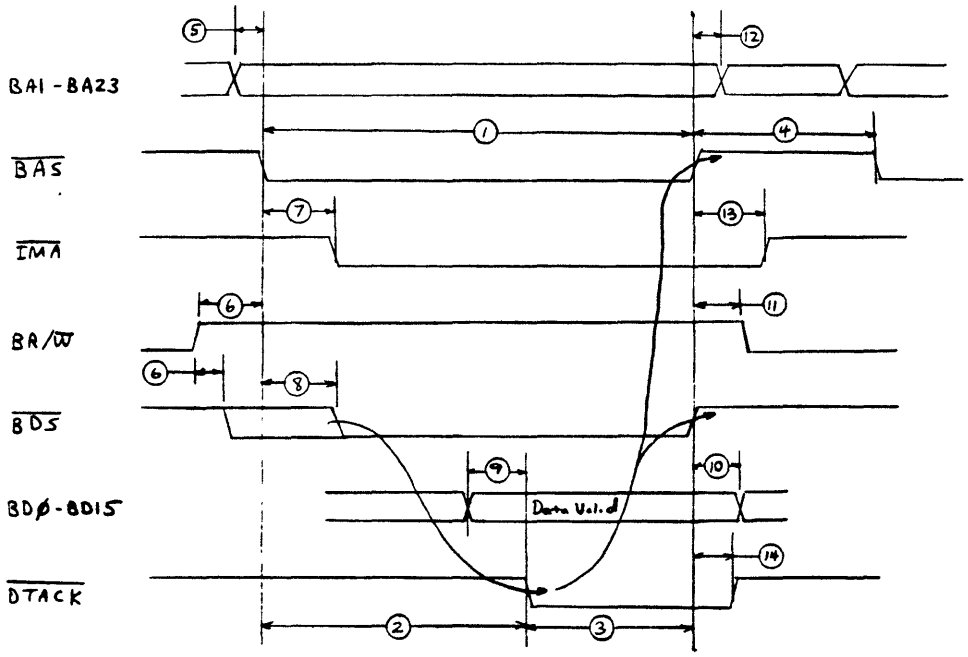
   Bus Slaves drive IMA* and DTACK* with either an open collector gate or a tri-state driver. While the high-to-low transition is controlled by this driver, the low-to-high transition is dependent on 2 factors external to the Bus Slave: (1) A pullup resistor to +5V and (2) an active pullup driver. For DTACK*, the pullup resistor is either on the Processor board or the Bus Expander (depending where the card is installed). For IMA*, the pullup is only in the Bus Expander. To provide a faster rise time than is provided by the pullup resistor, these signals are also actively pulled high. The active pullup devices (either on the Processor board or the Bus Expander) are activated after BAS* goes high. Because the active pullup may be activated before the Bus Slave releases its low-driven signal, a short conflict may occur. To minimize this conflict, the Bus Slave must release IMA* and DTACK* within 50 nsec of BAS* going high.

Because the Bus Expander does not assert IMA* high until 70 ns after BAS* goes high, there should be no conflict. However, problems have been seen with DTACK* since certain Processor boards assert DTACK* high immediately after BAS* goes high. This contention has been observed to cause glitches in DTACK; thus, DTACK* should not be used as a clock. Note that during DMA cycles, the DTACK* pullup is still asserted by the Processor board, not the DMA Controller.

Relative to BAS or BDS, the following signals change with the indicated delay:

A. Address hold: 15ns. min.
B. IMA* release: 50ns max.
C. DTACK* release: 50ns max.
D. Data Bus release: 100ns max.

FIGURE 5-1: DIO BUS READ CYCLE



	READ CYCLE TIMES (ns)	MIN	MAX	NOTES
1a	BAS* low when Bus Error occurs	4000		1
1b	BAS* low to avoid Bus Error		3500	2
2a	BAS* low to DTACK* low w/o bus error, DMA		1500	3
2b	BAS* low to DTACK* low w/o bus error,non-DMA		3000	3
3	DTACK* low to BAS* high		350	4
4	BAS* high	140		
5	Address setup before BAS* low	15		
6	BR/W* high to BAS* low or BDS* low	15		5
7	BAS* low to IMA* low	0	70	6
8	BAS* low to BDS* low	-75	75	7
9a	Data setup before DTACK* low, non-DMA source	15		8
9b	Data setup time before DTACK* low,DMA source	55		8
10	BAS* or BDS* high to data release	0	100	9
11	BR/W* hold after BAS* or BDS* high	65		10
12	Address hold after BAS* or BDS* high	15		10,11
13	BAS* high to IMA* release	0	50	12
14	BAS* high to DTACK* release	0	50	12

NOTES:

1. As discussed in the Chapter BUS ERROR AND STRETCH OPERATION, the BAS* low time is limited in Series 200 mainframes by the Bus Error timer. This timer is set at 4.0 usec on early CPU boards; later CPU boards, however, have increased this time.

2. To provide margin, the maximum BAS* low time is specified as 3500 ns.

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

4. Meeting this timing spec ensures that the maximum BAS* low time of 3500 ns will be achieved, i.e. 3000 ns + 350 ns < 3500 ns.

5. The 68000 specification permits 40 ns; however, the 15 ns limit documented in the DESIGNER'S GUIDE TO THE 9826A CARDCAGE is retained.

6. This time was previously specified at 50 ns; however, because 50 ns is too restrictive and because the Bus Expander (the only user of IMA) allows 70 ns, this time has been increased. Since the IMA* bus driver itself can have up to 50 ns of delay (see Section 3.4), the on-board delay in address decoding must be less than 20 ns.

7. This time was derived from the min/max delays (0/60 ns) for BDS* and BAS* relative to the 68000 clock. Assuming (very worse case) that one signal is delayed 0 ns while the other is delayed 60 ns and adding in 15 ns for bus driver skew yields the +/- 75 ns spec. Because the skew between LDS* and UDS* is not specified by Motorola, one must assume a similar skew is possible between these 2 signals.

8. As usual, these times are measured at the bus receiver.

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

10. Whichever goes high last of BAS* or BDS*. If the 68000 gives up the bus to another bus master at the completion of the cycle, the 65 ns is still guaranteed since the other bus master must wait 100 ns after BAS* and DTACK* are high before assuming control. For a 68000 read cycle followed by a write cycle, the 65 ns is provided by the fact that BR/W* stays high until BAS* goes low at the beginning of the write cycle.

FIGURE 5-1 NOTES (continued)

11. Based on the 68000 and buffer/bus skew, the address hold time is 15 ns. However, the 98256A (256 kbyte RAM board) requires 40 ns of address hold time. This is presently achieved because the address bus is typically driven longer than the specified minimum and is tri-stated after the cycle (i.e. it floats at its current value).

12. The release time specifies when the open collector gate or tri-state driver must stop driving the low signal. The low-to-high transition is then controlled by external pullup resistors and drivers as described above.

5.4 WRITE CYCLE

Figure 5-2 shows the write cycle. As discussed in Chapter 3, timing requirements are specified on the bus measured at the signal receiver. The key aspects of a write cycle are:

1. Prior to the beginning of the write cycle, BR/W*, BUDS* and BLDS* are pulled high by the processor. A pullup device on the Bus Master actively pulls up DTACK*.

2. Address and BFC0-BFC2 are valid with a minimum address setup time of 15ns before BAS is asserted. BR/W* goes low 75ns before to 35ns 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 70 nsec after BAS occurs.

4. Data is valid on the data bus BD0-BD15 a minimum of 15 nsec prior to BDS (BLDS or BUDS, whichever occurs first). The occurrence of BDS indicates that data is valid. Notice that the time from BAS to the data strobes can vary over a wide range (50 ns to 2500 ns). The longer times typically occur during DMA operation because a read must be performed prior to the write operation.

5. When BDS goes true, BR/W* is guaranteed to be 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 35 nsec 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 nsec. The Bus Master then removes the data BD0-BD15 from the data bus. To allow data hold time so that the Bus Slave can clock the data at the same time it asserts DTACK, the minimum Bus Master data hold time after detection of DTACK is 85ns.

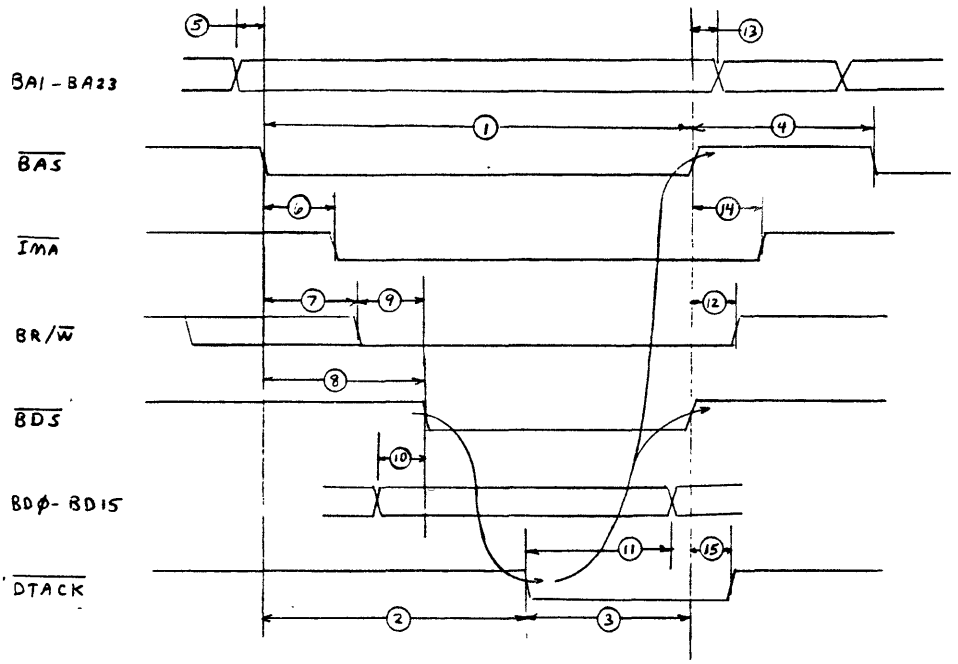
8. The following signals change with the indicated delay after the negation of BDS and BAS (whichever occurs last).

A. Address hold:	15 ns min.
B. BR/W* hold:	25 ns min.
C. IMA* release:	50 ns max.
D. DTACK* release:	50 ns max.
E. DTACK* pull-up asserted on current Processor boards	0 ns min.

5.5 READ-MODIFY-WRITE

The 68000's Read-modify-write cycle is not supported on any current Series 200 hardware. One problem is the requirement for DTACK* to be asserted twice during the cycle. Memory boards do not actively pull DTACK* high; only Processor boards do this (after BAS* goes high). Therefore, between DTACK's, DTACK* is pulled up only by a pullup resistor and is not guaranteed to go high before the second DTACK occurs.

FIGURE 5-2: DIO BUS WRITE CYCLE



	WRITE CYCLE TIMES (ns)	MIN	MAX	NOTES
1a	BAS* low when Bus Error occurs	4000		1
1b	BAS* low to avoid Bus Error		3500	2
2a	BAS* low to DTACK* low w/o bus error, DMA		1500	3
2b	BAS* low to DTACK* low w/o bus error,non-DMA		3000	3
3	DTACK* low to BAS* high w/o bus error		350	
4	BAS* high	140		
5	Address setup before BAS*	15		
6	BAS* low to IMA* low	0	70	
7	BAS* low to BR/W* low	-75	45	
8	BAS* low to BDS* low	50	2500	4
9	BR/W* low to BDS* low	65		5
10	Data setup before BDS* low	15		
11	Data hold after DTACK* low	85		
12	BR/W* hold after BAS* or BDS* high	15		6
13	Address hold after BAS* or BDS*	15		7
14	BAS* high to IMA* release	0	50	8
15	BAS* high to DTACK* release	0	50	8

NOTES:

1. As discussed in the Chapter BUS ERROR AND STRETCH OPERATION, the BAS* low time is limited in Series 200 mainframes by the Bus Error timer. This timer is set at 4.0 usec on early CPU boards; later CPU boards, however, have increased this time.

2. To provide margin, the maximum BAS* low time is specified as 3500 ns.

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

4. The minimum time was listed in previous documentation at 85 ns. However, based on the 68000 timing, a minimum time of 50 ns is required.

5. Specs 7 & 8 imply that BDS* could go low within 5 nsec of BR/W* going low (50-45). In actuality, this cannot happen as guaranteed by Spec 9.

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

7. Based on the 68000 and buffer/bus skew, the address hold time is 15 ns. However, the 98256A (256 kbyte RAM board) requires 40 ns of address hold time. This is presently achieved because the address bus is typically driven longer than the specified minimum and is tri-stated after the cycle (i.e. it floats at its current value).

8. The release time specifies when the open collector gate or tri-state driver must stop driving the signal low. The low-to-high transition is then controlled by external pullup resistors and drivers as described above.

5.6 ENDT OVERVIEW

As discussed previously, Enable DTACK (ENDT) is used to improve the response time of certain DIO BUS devices such as RAM cards. These devices use ENDT (which is basically BAS delayed by 1-1/2 clock cycles) to generate an 'early' DTACK. For example, during a CPU read, ENDT is used to generate DTACK before the data is even on the bus; however, because the 68000 accepts the data up to 90 ns after DTACK, the read cycle will be completed successfully as long as the device provides the data within 90 ns of DTACK. The following comments on ENDT usage are necessary:

1. ENDT can be used to provide 5 cycle read/write accesses. However, during those memory cycles where the card is unable to accept or provide data within the required time (e.g. during a RAM refresh cycle), then the card must inhibit use of ENDT to generate the early DTACK.

2. While ENDT is currently used only for certain cards such as RAM and EPROM, other devices which contain memory, such as new I/O cards, could use ENDT to improve their response time. In general, the RAM on Bus Masters does not use ENDT since the RAM control logic can use the CPU clock.

3. Cards that use ENDT must also be able to work without it. For example, the 98620A DMA Controller does not generate ENDT. Likewise, new Bus Masters are not required to generate ENDT.

4. If a new DMA Controller is designed that generates ENDT during a DMA cycle, ENDT should be used by memory (not the I/O card) during the DMA cycle.

In the timing discussions that follow, the timing margins appear very tight, if not unworkable. This is attributable to using worse-case timing specs for all parameters. In actuality, sufficient margin exists to ensure that 5 cycle accesses are achievable. For example, the worse-case ENDT-DTACK timing on the 09826-66522 board violates the timing specification; however, the production test for 5 cycle timing has yet to detect a failure of this board to achieve 5 cycle accesses (except, of course, during memory refresh cycles).

The maximum time from ENDT-in to DTACK-out is calculated below. This applies to both read and write cycles.

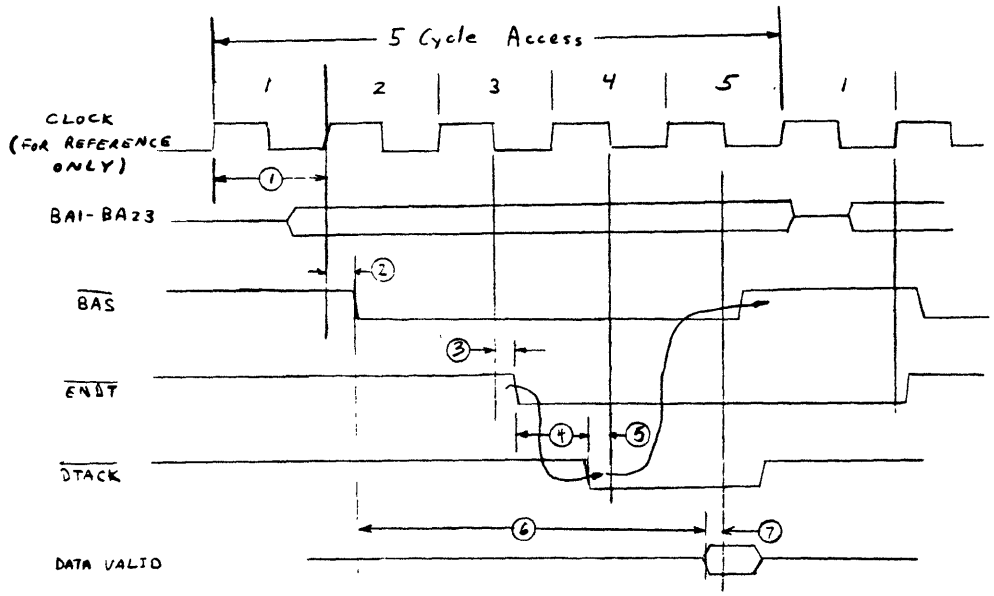
125 ns	Clock cycle period. ENDT is generated on the falling edge of the clock and DTACK must be valid prior to the next falling edge in order to achieve 5 cycle accesses.
-9 ns	Maximum delay in the 74S74 flip-flop that generates ENDT.
-30 ns	Maximum delay in the 74LS245 buffer that drives ENDT. This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

5.7 READ CYCLE USING ENDT

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

437.5 ns	Represents 3-1/2 clock cycles. BAS* starts with the rising edge of the CPU clock; data must be valid just before the clock's falling edge, 3-1/2 cycles later.
-60 ns	Maximum delay from rising edge of clock to AS*.
-47.5 ns	Maximum delay of BAS* bus driver.
-12 ns	Maximum data delay on CPU board due to bus receivers.
-15 ns	CPU data setup time before trailing edge of clock.
303 ns	Thus, the Bus Slave must have data valid within 303 ns of BAS* going low if ENDT is used to generate DTACK as discussed above. Because the Bus Slave's data bus drivers can require up to 39.5 ns, internal delays in accessing data must be less than 303 - 39.5 = 263.5 ns.

FIGURE 5-3: READ CYCLE TIMING WITH ENDT



	READ TIMING (ns)	MIN	MAX	NOTES
1	Clock period @ 8 Mhz	125 ns	nominal	1
2	BAS* delay from clock high		107.5	
3	ENDT* delay from clock low		39	
4	ENDT* low to DTACK* low		54	
5	DTACK* low setup before clock low	32		2
6	BAS* low to read data valid		303	
7	Data setup before clock	27		

NOTES:
1. The clock is on the Bus Master and does not appear on the DIO BUS; it is shown for REFERENCE ONLY.
2. As it should, Spec 3 + Spec 4 + Spec 5 = 125 ns.
3. These times assume ENDT* loading is less than the 500 pf assumed for the data bus, address bus, etc.

5.8 WRITE CYCLE USING ENDT

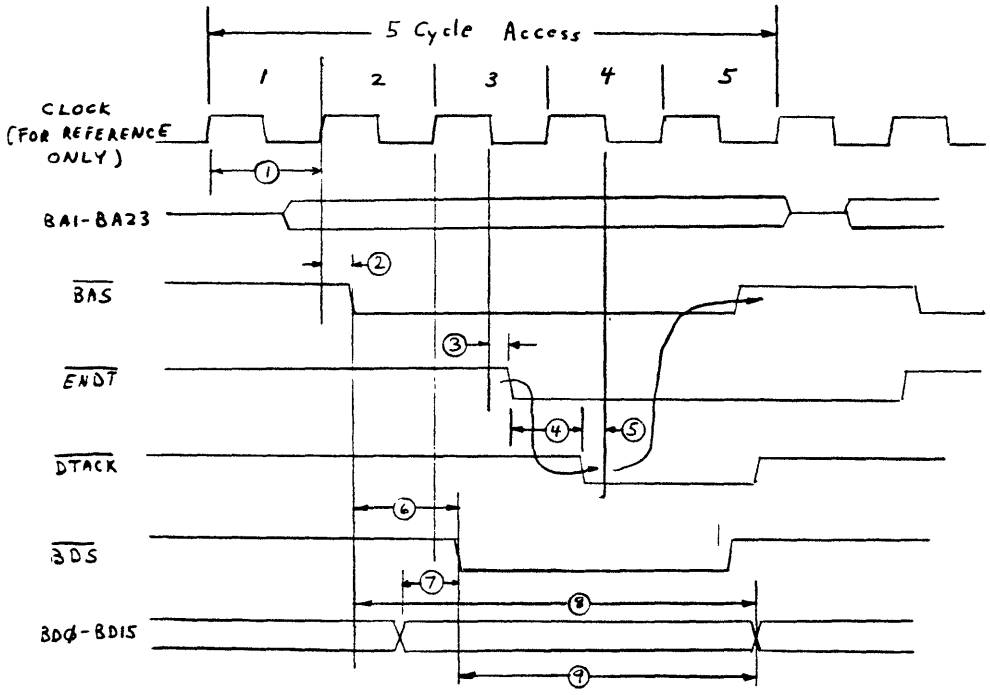
During write cycles, the time from BAS* going low to data becoming invalid (indicating completion of the cycle) is of interest. Likewise, the time from the data strobes BLDS and BUDS going true (indicating valid data on the bus) to the data becoming invalid is also of interest. These times are calculated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

1. MINIMUM BAS* LOW TO DATA INVALID TIME:

437.5 ns	3-1/2 clock cycles
-60 ns	Maximum delay from rising edge of clock to AS*.
-47.5 ns	Maximum delay of BAS* bus driver.
-15 ns	Maximum skew between BAS* and data
0 ns	Minimum delay of data bus drivers. Any delay extends the time from BAS* low to data invalid; however, to continue with the worse-case analysis, 0 must be used.
+30 ns	Minimum data hold after BAS* goes high. Because the data bus buffers continue to drive the bus after BAS*and BDS* go high, this hold time does occur on the bus.
345 ns	

2. MINIMUM DATA STROBES LOW TO DATA INVALID TIME -- The data strobes are delayed from BAS* by one clock cycle (125 ns). Therefore, the time from the data strobes going low to data invalid is 345 - 125 = 220 ns.

FIGURE 5-4: WRITE CYCLE TIMING WITH ENDT



	WRITE TIMING WITH ENDT (ns)	MIN	MAX	NOTES
1	Clock period @ 8 Mhz	125 ns	nominal	1
2	BAS* delay from clock high		107.5	
3	ENDT* delay from clock low		39	
4	ENDT* low to DTACK* low		54	
5	DTACK* low setup before clock low	32		
6	BAS* low to BDS* low (nominal)		125	
7	Data setup before BDS* low	15		
8	BAS* low to data invalid	345		
9	BDS* low to data invalid	220		

NOTES:
1. The clock is on the Bus Master and does not appear on the DIO BUS; it is shown for REFERENCE ONLY.
2. These times assume ENDT* loading is less than the 500 pf assumed for the data bus, address bus, etc.

An exception sequence is generated when the CPU's bus error input signal BERR* is asserted. This signal is open collector, permitting it to be generated by any device (including the Processor board). Applications of the bus error signal are device dependent; several applications that DIO BUS designers should be aware of are discussed in this chapter.

An addition (4-83) to the DIO BUS was the signal STRETCH*, which was assigned to the previously-spare pin 98. The STRETCH* signal, which was to be generated by the 1 Mbyte parity RAM card in case of a parity error, would cause certain processor boards (e.g. the Memory Management Processor board) to insert an extra half cycle in the CPU clock. This would permit BERR* (also generated by the RAM card) to have sufficient setup time at the CPU.

The situation, however, has changed. It was recently (11-83) discovered that even with STRETCH, the BERR signal generated by the RAM card would not have sufficient setup time on the processor board. Therefore, the STRETCH function has been abandoned and will not be generated by RAM cards. However, because Processor boards (09826-66517) have been shipped which respond to STRETCH, it is not possible to free the pin (98) as a spare. The pin, however, has been designated as a 'processor spare' in that future processor boards can use this pin as an OUTPUT since they will not be operating in the same machine as those processor boards which respond to this signal as an input.

6.1 SIGNALS

BERR*	BUS ERROR, when asserted, causes the Processor to terminate the bus cycle. When BERR is negated, the Processor begins its exception processing.
Pin 98	This signal was called STRETCH*, is now a 'processor spare' as discussed above.

6.2 BERR TIMING

BERR timing is different for 68000 and 68010 processors. The 68000 BERR input can be asynchronous; the only requirement is that BERR arrive before DTACK. The 68010 permits BERR to arrive after DTACK; however, BERR must meet a setup time before the CPU clock. According to Motorola, "this setup time is critical to proper operation, and the MC68010 may exhibit erratic behavior if it is violated". Because BERR timing varies for different mainframes and different Processor boards, timing diagrams are not included in this document. The reader is referred to Section 6.6 which provides guidelines for utilizing BERR.

6.3 BUS TIMEOUT

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

A 4-bit BERR counter is located on Series 200 Processor boards and is cleared when BAS is false. When BAS is true, it begins counting (4 Mhz clock) and generates a BERR when it overflows (after 16 counts); 16 x 250 nsec = 4.0 usec. To provide margin, the maximum width of BAS is spec'd at 3.5 usec. To ensure that DTACK has time to reset BAS prior to 3.5 usec, DTACK must arrive 2 CPU clocks (250 nsec, synchronizing time) earlier. Because of this time, other delays and to provide ample margin, the maximum time from BAS* going low to DTACK* going low has been spec'd at 3.0 usec.

From the above, it appears that devices have 3.0 usec to 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 BAS cycle. Thus, the combined RAM read time and I/O card write time must be less than 3.0 usec.

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

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

Because the Master Controller implements the BUS TIMEOUT function, other Bus Masters do not have to generate it; however, other Bus Masters do have to respond to it if the Master Controller generates it while another Bus Master has control of the bus. Acceptable responses are: (1) handle the bus timeout problem or (2) give up the bus to the Master Controller and let it resolve the problem. Refer to the SERIES 200 SYSTEM SPECIFICATION for more details.

6.4 AUTO LOCATE OF PROCESSOR RAM

The BERR signal generated by the BAS timeout counter is used to auto-locate RAM on 9826/36 Processor boards (but not on the 9816 Processor board). The Processor RAM auto-locates itself at the bottom (lower addresses) of RAM memory, eliminating the need for the user to set switches on an internal board. This works as follows: at power-on, the processor writes to and reads from each block of memory to determine how much memory is available. When there is no response from a memory card (indicating that the access is below all plug-in memory), the BERR counter times out and generates BERR. A latch then latches the bus-error address as the Processor board's RAM address. Auto location is a Series 200 implementation detail, not a Bus Master requirement.

6.5 BERR FOR PAGE FAULTING

Memory management Processor boards such as the 09826-66517 generate BERR upon detection of a page fault. DTACK is not generated so that a normal BERR trap occurs. Note: due to the unavailability of the 68010, early 09826-66517 boards have the 68000 and thus do not support page faulting.

6.6 GUIDELINES FOR UTILIZING BERR

This section gives guidelines for using BERR in new designs. It should be noted that presently (4-84) BERR is generated only by Processor boards. The only card which uses BERR (in addition to the Processor boards themselves) is the 98620 DMA Controller.

1. Because of the critical nature of BERR timing and the differences in operation with different mainframes and Processor boards, the FSD & RND R&D Labs should be consulted prior to designing a card which generates BERR. In addition, the following information should be studied:

   A. BERR specifications for the different 680xx Processors.

   B. Hardware and software documentation on BERR operation and limitations in current mainframes.

2. Even though a Processor board or another board may generate BERR, a Bus Slave may not 'see' it. For example, BERR is an input only from the 9888 Bus Expander; thus, a Processor generated BERR is not seen by cards in the expander. Therefore, cards may utilize BERR if it is present, but they must not depend on it being available unless the stipulation is made that the card must reside in the mainframe. Such a stipulation is acceptable for a one-per-system type of card (e.g. the 98620 DMA Controller) but is not acceptable for generic I/O cards.

3. If a Bus Slave does respond to BERR, it should terminate all DIO BUS transactions, i.e. 'get off of the bus'.

The DIO BUS supports seven interrupt levels and 2 methods of responding to interrupts: (1) External Vectored and (2) Autovectored. External vectoring requires the interrupting device to put an 8-bit vector on the bus and assert the VECTOR signal. With autovectoring, the interrupting device does not provide a vector and the Processor generates its own default vector.

CAUTION: External vectored interrupts do not operate as described in the predecessor to this document, the DESIGNERS GUIDE TO THE 9826A CARDCAGE. Existing Processor boards do not generate several of the control signals described in that document.

However, as discussed in the Chapter REQUIRED DIO BUS FUNCTIONS FOR AN I/O BUS, I/O card designers should NOT design cards that implement externally vectored interrupts, i.e. only autovectored interrupts should be used. Therefore, the description of external vectored interrupts in this document (section 7.2) should be viewed as primarily historical in nature, to describe an existing (but never used) DIO BUS feature. While external vectored interrupts should not be used in products, they may be used in special, one-of-a-kind projects as long as the designer accepts the risks involved.

7.1 INTERRUPT SIGNALS

The interrupt signals on the DIO BUS are shown below. Interrupts 3-6 are for external I/O cards. Interrupts 1,2 and 7 are for internal I/O. The assignment of these interrupt levels for the 9826/36 is shown below as an example; however, the SERIES 200 SYSTEM SPECIFICATION should be referenced for the most current information.

1. IR1*	Interrupt 1, Keyboard/real time clock	LOWEST LEVEL
2. IR2*	Interrupt 2, 9826/36 internal floppy controller	

3. IR3* Interrupt 3, External I/O .
4. IR4* Interrupt 4, External I/O .
5. IR5* Interrupt 5, External I/O .
6. IR6* Interrupt 6, External I/O .

7. IR7* Interrupt 7, Reset key, powerfail HIGHEST LEVEL

8. IACK* Interrupt Acknowledge, output from the Bus Master

9. VECTOR* Output of interrupting device if it has an interrupt vector to put on BD0-BD7.

There is no IR0* signal; level 0 is the quiescent (non-interrupting) state. For 68000-based Bus Masters, logic is needed to encode these interrupt signals into the 3 processor inputs, IPL0, IPL1 and IPL2. For example, Series 200 Bus Masters use an 74LS148 8-to-3 priority encoder to encode INT1 - INT7 to generate IPL0, IPL1 and IPL2.

The following interrupt levels are the only I/O interrupt levels which have been 'hardwired'; all plug-in I/O cards have a 2-bit switch to select levels 3-6.

98620A DMA Controller - Level 3
98620B DMA Controller - Programmable from 3-7
Internal HP-IB (9816/26/36) - Level 3
Internal RS-232 (9816) - Level 4

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

Level 1 and 7 interrupts are driven by open collector gates in the 9826/36. However, Level 2, used by the 09826-66561 and 09826-66562 floppy controller boards, is driven by a standard LS-TTL gate and thus cannot be used by other devices.

7.2 EXTERNAL VECTORED INTERRUPT ACKNOWLEDGE CYCLE

If the priority of the pending interrupt is greater than the current processor priority, the following interrupt sequence begins in which the Bus Master attempts to read an 8-bit interrupt vector from the interrupting I/O card. This operation is the same as a normal read, except IACK serves in lieu of BAS; BAS remains false. Interrupt timing is shown in Figure 5-1.

1. Prior to the beginning of the cycle, the lowest three address bits (BA1- BA3) are set equal to the interrupt level being acknowledged. Likewise, the 68000 Function Code bits FC0, FC1 and FC2 are set to 1. The Bus Master decodes the Function Code bits and gates with Address Strobe to generate IACK*.

2. When FC0, FC1 and FC2 are 1, the bus driver for BLDS*, ENDT*, BUDS*, BR/W*, BAS*, BFC0, BFC1 and BFC2 is disabled on these CPU boards: 09826-66514, -66515, -66516 and the 09816-66511. These signals should be considered undefined.

3. In response to IACK, each interrupting card detects: (1) if it is interrupting and (2) if its interrupt level is being acknowledged (by looking at BA1-BA3). If so, the card asserts the VECTOR signal within 100 nsec after IACK occurs to indicate to the processor that it has an 8-bit interrupt vector to put on the lower byte of the data bus. If the interrupting card does not respond within 100 ns, Bus Master circuitry will initiate an Autovector interrupt response, as discussed in the next section.

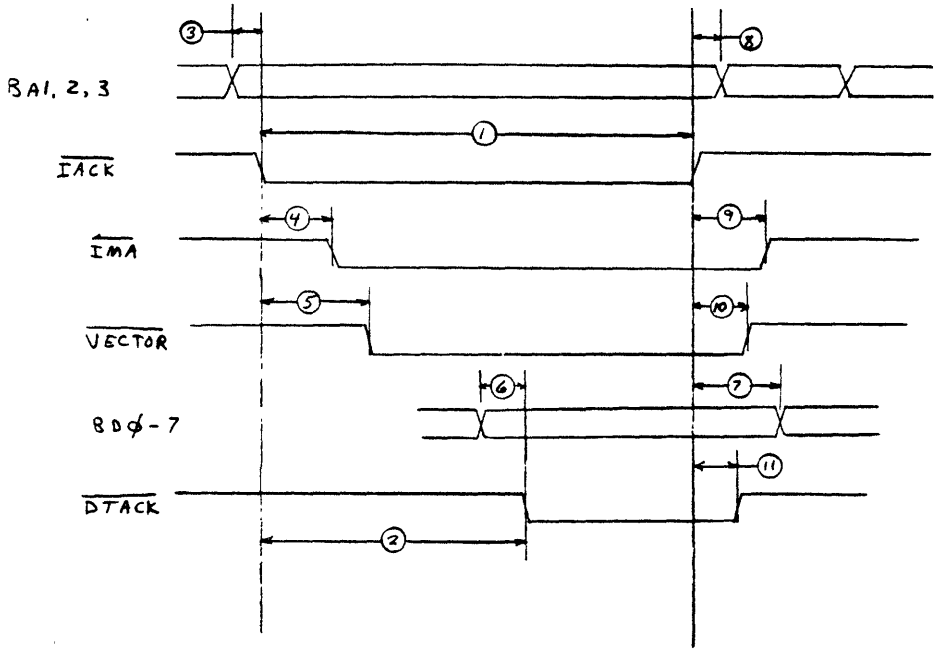
4. If the VECTOR is asserted within 100 ns, reading of the 8-bit vector occurs. Because the data strobes are disabled, the interrupting card provides its vector in response to IACK, as described:

   A. After detecting IACK, the interrupting card drives its vector on BD0-BD7. Data is setup a minimum of 15 ns prior to assertion of DTACK (as measured at the receiver).

   B. The CPU detects that DTACK has occurred, accepts the vector and ends the interrupt acknowledge by setting IACK false.

   C. The interrupting card then releases the VECTOR signal within 50 nsec after IACK goes false and releases the drive on the data bus within 100 nsec after IACK is false.

FIGURE 7-1: EXTERNAL VECTORED INTERRUPT ACKNOWLEDGE CYCLE



	INTERRUPT ACKNOWLEDGE CYCLE TIMING	MIN	MAX
1	IACK* low		4500
2	IACK* low to DTACK* low w/o bus error		3000
3	Address setup before IACK* low	15	
4	IACK* low to IMA* low	0	70
5	IACK* low to VECTOR* low	0	100
6	Data setup before DTACK* low	15	
7	Data hold after IACK* high	0	100
8	Address hold after IACK* high	15	
9	IACK* high to IMA* high	0	50
10	IACK* high to VECTOR* high	0	50
11	IACK* high to DTACK* high	0	50

External Vectoring has not been implemented on any I/O card to date (4-84) and thus its operation has not been tested. If External Vectoring is implemented, extensive qualification will be required to verify that it works with the different mainframes and processor boards. Also, designers should be aware that only one vectoring card per interrupt level is allowed unless a hardware arbitration scheme is included. Multiple autovectoring I/O cards may reside on the same level as a single vectoring card since the vectoring card will override autovectoring cards.

Refer to the Chapter OPERATION IN THE 9888A BUS EXPANDER for problems relative to vectored interrupt operation in the Bus Expander.

7.3 AUTOVECTORED INTERRUPT CYCLE

If an interrupting card does not generate VECTOR, the Bus Master automatically generates its own vector as follows:

1. Steps 1-3 outlined above occur (the Bus Master does not 'know' yet that it is an Autovector cycle).

2. With Autovectoring, the interrupting card does not assert VECTOR in response to IACK, nor does it provide the vector itself. Logic on the Processor Board detects that VECTOR* remains high and, within 100 ns of IACK, asserts VPA (Valid Peripheral Address), an input to the 68000.

3. The processor recognizes that, if VPA is asserted during the interrupt acknowledge cycle, it is an Autovector cycle. The processor uses an internally generated vector that is a function of the interrupt level being serviced. The 7 interrupt vectors, corresponding to interrupt levels 1-7, are 25 through 31 decimal. These are used to access 4-byte addresses from 100 - 124 which vector operation to RAM. Refer to the 68000 Data Sheet for more information.

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 machine level service poll routine must be used to determine which card to service. The two most significant bits of the status register contain interrupt information. The most significant bit indicates if the card is enabled to interrupt. The next most significant bit indicates if the card is in an interrupt state.

The DIO BUS design permits other Bus Masters to acquire the DIO BUS and perform memory read/write operations. To support multiple Bus Masters (up to 4 in addition to the Master Controller), a bus arbitration scheme is used.

NOTE: DESIGNERS CONSIDERING DESIGNING BUS MASTER I/O CARDS MUST CONTACT FSD TO DETERMINE THE SYSTEM IMPLICATIONS OF SUCH PLANS. FOR EXAMPLE, FUTURE MAINFRAMES MAY SEPARATE THE SYSTEM BUS AND THE I/O BUS SUCH THAT THE SYSTEM BUS IS NOT CONTROLLABLE FROM THE I/O BUS.

8.1 BUS ARBITRATION SIGNALS

The signals used to arbitrate control of the bus are:

1. BR* Bus Request, asserted by the device(s) requesting control of the bus, is wire-ORed among all Bus Masters.

2. BG* Bus Grant, Master Controller output, goes to Bus Grant input of Bus Master A.

3. BG1* Bus Grant 1, Bus Grant output from Bus Master A, connected to Bus Grant input of Bus Master B.

4. BG2* Bus Grant 2, Bus Grant output from Bus Master B, connected to Bus Grant input of Bus Master C.

5. BG3* Bus Grant 3, Bus Grant output from Bus Master C, connected to Bus Grant input of Bus Master D. Note that Bus Master D's Bus Grant output is not connected to another device.

6. BGACK* Bus Grant Acknowledge, a tri-state signal generated by the the Bus Master accepting the bus to verify that the bus is now controlled by the new bus master.

8.2 BUS ARBITRATION OVERVIEW

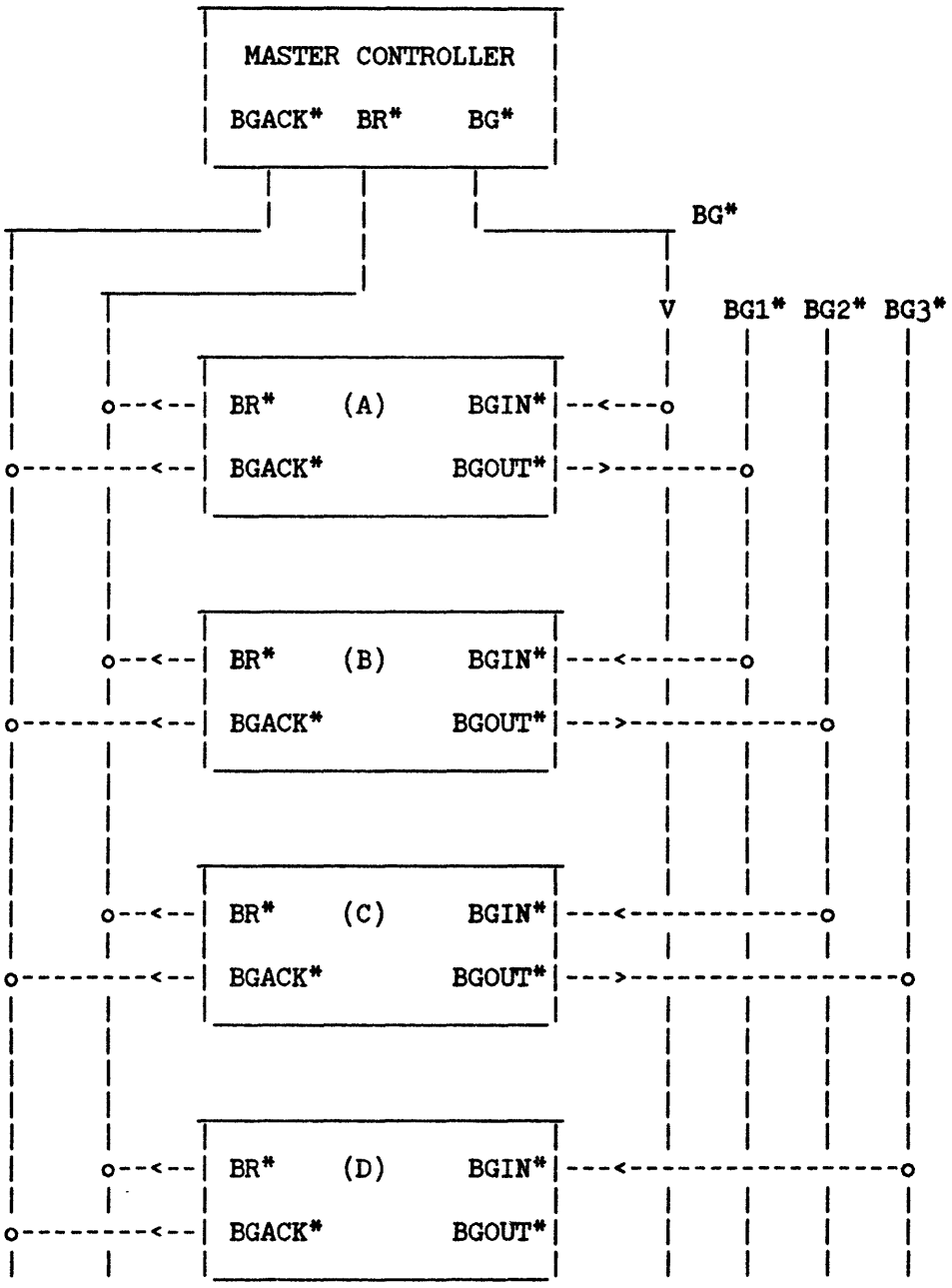
The bus arbitration scheme permits other Bus Masters to request and receive control of the DIO BUS. Certain bus grant signals are daisy chained from the Master Controller to each Bus Master. While it may appear that the daisy chain supports a priority scheme, in actuality it does not. It is basically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

Arbitration is handled via the Bus Grant signals (BG, BG1, BG2 and BG3) which are daisy-chained from the Master Controller to Bus Masters A, B, C and D (see Figure 8-1 on the next page). Connections to the desired Bus Grant signals are made by jumpers on each Bus Master; alternatively, switches could be used. Thus, the daisy chain is NOT hardwired on the backplane itself but requires the user to configure it.

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

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

FIGURE 8-1: CONFIGURATION OF BUS ARBITRATION SIGNALS



NOTE: The BR* output is open-collector. The BGACK* output is tri-state.

8.3 BUS ARBITRATION SEQUENCE

The bus arbitration sequence is discussed below:

1. A Bus Master desiring the bus assert BR to the Master Controller. The Master Controller eventually responds with BG (response time is 0 ns to infinity).

2. The assertion of BG by the Master Controller begins the arbitration among the Bus Masters. If BG reaches a device that is not requesting the bus, it is simply passed along (from BGIN to BGOUT). When BG reaches a device that is requesting the bus, the signal is blocked from going further.

3. The device that 'blocks' BG then awaits the bus's availability as indicated by BAS, DTACK and BGACK going false while BG remains true (again, the response time can be infinity). When these conditions are met, the new Bus Master must then wait a minimum of 100 ns to allow the old Bus Master to tri-state its signals before asserting BGACK.

   NOTE: Current processor boards do not wait 100 ns before enabling their drivers after regaining control of the bus. Therefore, there will be bus contention while the former Bus Master tri-states its drivers.

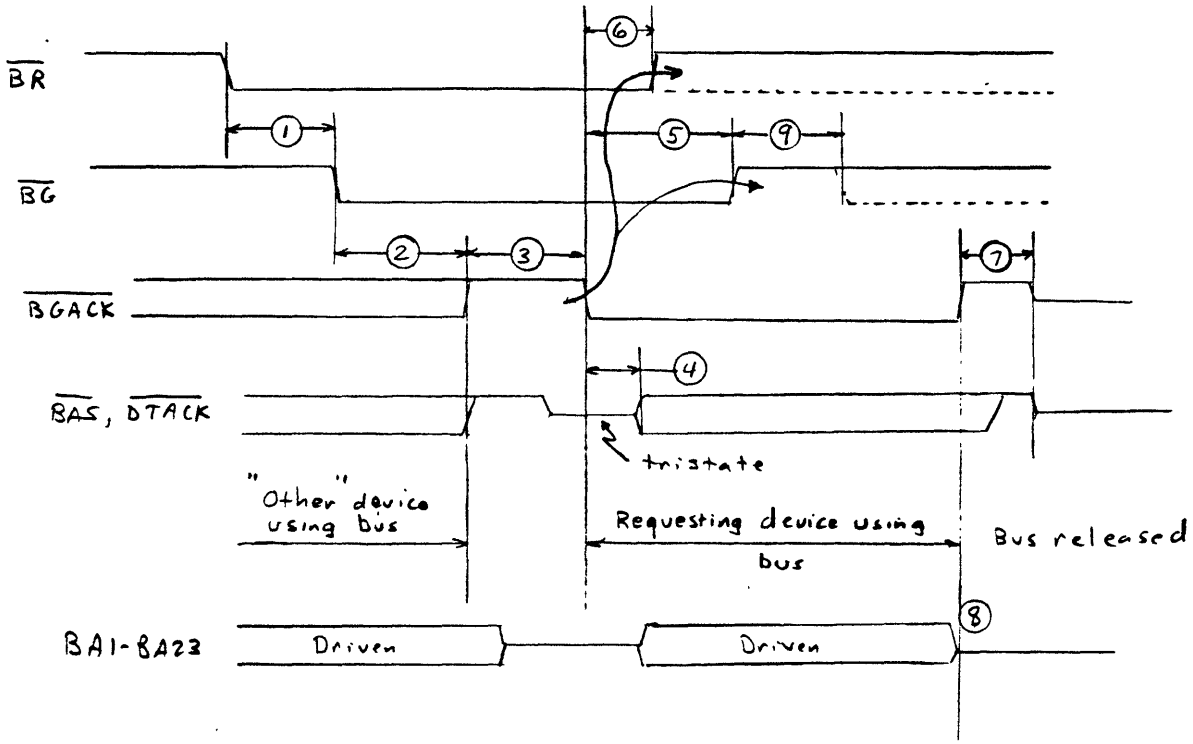
4. After asserting BGACK, the device then releases BR within 100 ns. BR must be removed within 100 ns after assertion of BGACK so that the Master Controller does not decide that another Bus Master is also requesting the bus.

5. When BGACK is asserted, the Master Controller negates BG which ripples through the arbitration daisy-chain and prepares the arbitration logic for the next arbitration.

6. When the current Bus Master is finished, it drives BAS, DTACK and BGACK false prior to tri-stating the bus drivers. The Bus Master must tri-state its drivers within 100 ns of negating BGACK so the next Bus Master can take over the bus. However, certain CPU boards, when regaining control of the bus, drive the address bus immediately upon negation of BGACK. Therefore, the device relinquishing the bus should tri-state its address bus simultaneously with negating BGACK.

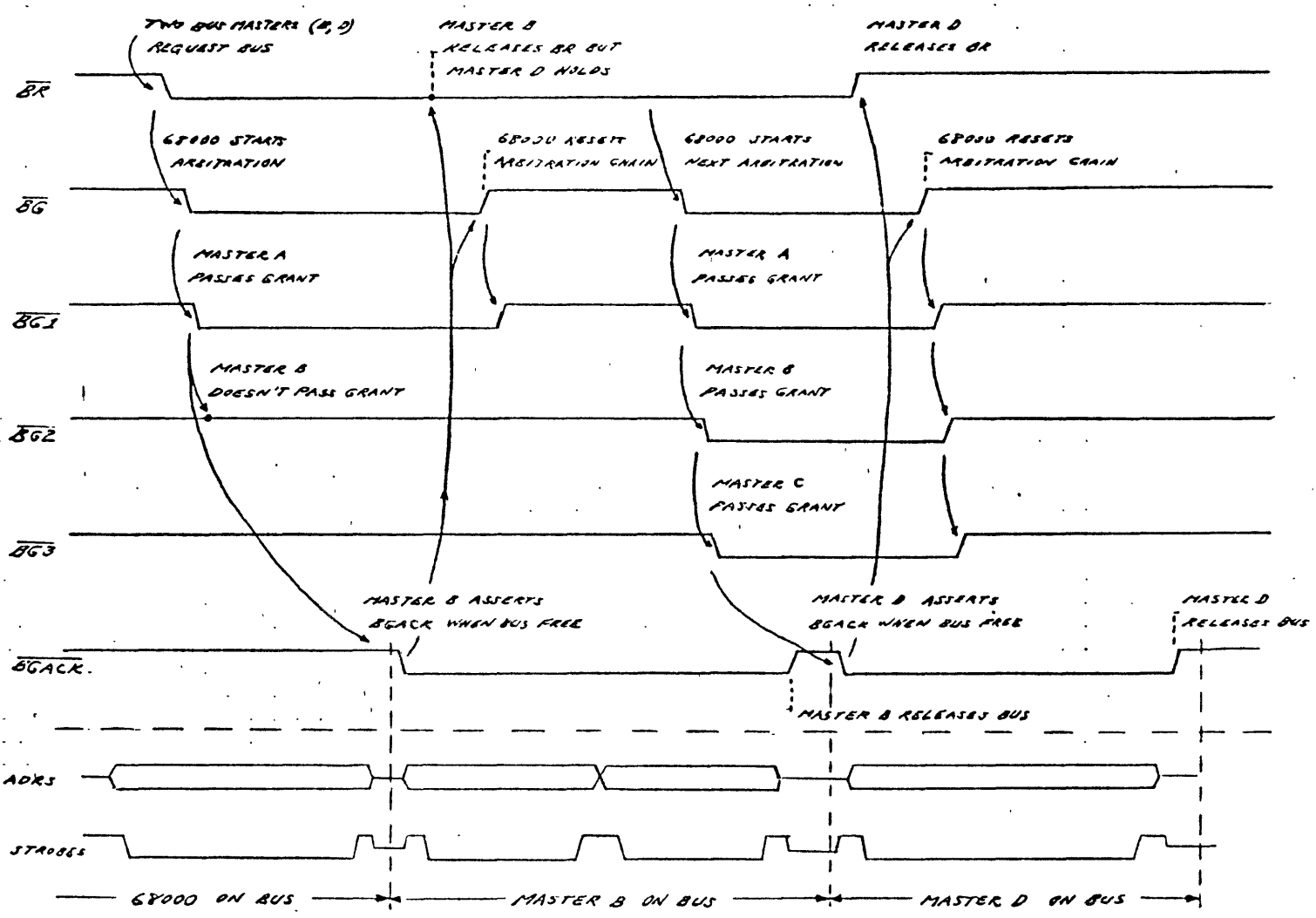
Shown on the next two pages is the bus handoff timing and an example of a bus handoff involving multiple Bus Masters.

FIGURE 8-2: BUS ARBITRATION TIMING



	BUS ARBITRATION TIMING	MIN	MAX
1	BR* low to BG* low	0	infinty
2	BG* low to BGACK*, BAS* and DTACK* high (bus available)	0	infinty
3	Bus available to BGACK* low	100	
4	BGACK* low to bus signals driven	0	
5	BGACK* low to BG* high	190	465
6	BGACK* low to BR* high	0	100
7	BGACK* high to bus signals tri-state except for address bus	0	100
8	BGACK* high to address bus tri-state		0
9	BG* high (arbitration reset time)	187	

FIGURE 8-3: EXAMPLE OF BUS HANDOFF



The DIO BUS supports 2 DMA channels. A DMA Controller (the 98620A card in Series 200 computers) monitors DMA requests from I/O cards, requests control of the bus and orchestrates DMA data transfers. DMA data rates exceeding 2 Mbytes/sec are possible in Word Mode. This chapter gives an overview of DMA operation and discusses DMA input and output operation.

9.1 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.

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

2. DMACK0* / DMACK1* — DMA ACKNOWLEDGE, response from the DMA Controller, acknowledges DMA request on Channel 0 or Channel 1.

3. DMARDY* — DMA READY, indicates that the I/O card has provided the data (DMA input) or accepted the data (output).

4. DONE* — DONE, an output from the DMA Controller to 'tag' the last DMA transfer. DONE can be used at the option of the I/O card designer to determine when DMA is done, e.g. to assert EOI on the last byte in an HP-IB transfer.

5. FOLD* — FOLD, an output from the DMA Controller to indicate that the data byte needs to be folded from the upper byte of the data bus (from memory) to the lower byte (to an I/O card) or from the lower byte (from an I/O card) to the upper byte (to memory).

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

9.2 DMA OVERVIEW

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

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

2. Enables DMA channel 0 or DMA channel 1 on the I/O card by writing to its Control Register 3. When the I/O card is ready to transfer data, it will request a DMA operation on the assigned channel using DMAR0 or DMAR1. In response to this request, the DMA Controller requests and eventually receives control of the bus from the Bus Master and then executes the DMA transfer.

A DMA transfer takes a single bus cycle during which data is both read and stored. The reference to 'a single bus cycle' means that all operations happen during a single BAS* high-low-high cycle. For a DMA output cycle, the data is fetched from memory and written to the I/O card. For a DMA input cycle, the data is read from the I/O card and stored in memory. The I/O card itself is programmed to request the DMA transfer; upon seeing this DMA request, the DMA Controller requests and receives control of the bus and provides the necessary address and control signals to perform the transfer.

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

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. However, most I/O cards are byte oriented and do not even connect to the upper byte of the data bus; Byte mode supports these I/O cards. In Byte Mode, I/O data is transferred on the low byte of the data bus to/from both the upper and lower bytes of memory. The DMA Controller supports this via a 'Fold Buffer' which is used to transfer data between the upper byte of memory and the lower byte of the data bus for I/O cards. During a DMA input operation, the Fold Buffer is alternately used to transfer I/O data to the upper byte of memory (BD8-BD15).

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

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 2 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 is discussed in more detail below.

NOTES: 1. While an I/O card is enabled for a DMA transfer, it must still respond to normal bus cycles between DMA cycles so its Control and Status Registers can be read.

2. The 98620A DMA Controller does not generate ENDT.

3. In Series 200 machines, it is not possible to DMA from alpha or graphics memory.

9.3 98620 DMA CONTROLLER

Because the 98620 DMA Controller is used to implement DMA in Series 200 products, an overview of its performance is given below. Designers implementing DMA on an I/O card should obtain additional documentation on this device. The key features of the 98620 DMA Controller are:

1. Provides 2 independent DMA channels with programmable priorities.

2. Capable of 1.2 Mtransfers/sec, providing 1.2 Mbytes/sec in byte mode or 2.4 Mbytes/sec in word mode.

3. Supports I/O-memory and memory-I/O transfers, but not memory-memory transfers.

4. Provides a full 16 Mbyte address range on both channels and a maximum transfer length of 64k transfers, i.e. 64k bytes (in byte mode) or 128k bytes (in word mode).

5. Memory mapped to asynchronous internal I/O address space 500000 (Channel 0) and 500008 (Channel 1).

6. 'Hardwired' to be Bus Master A (see the Chapter BUS ARBITRATION).

7. Available as a 98620A (interrupt hardwired to level 3) or a 98620B (interrupt programmable from 3-7 for UNIX systems).

9.4 DMA OUTPUT CYCLE

Figure 9-1 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 55 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 in burst mode 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, BLDS/BUDS and DMACK0. Once the I/O card detects that DMACK0 is gone, it removes DMARDY.

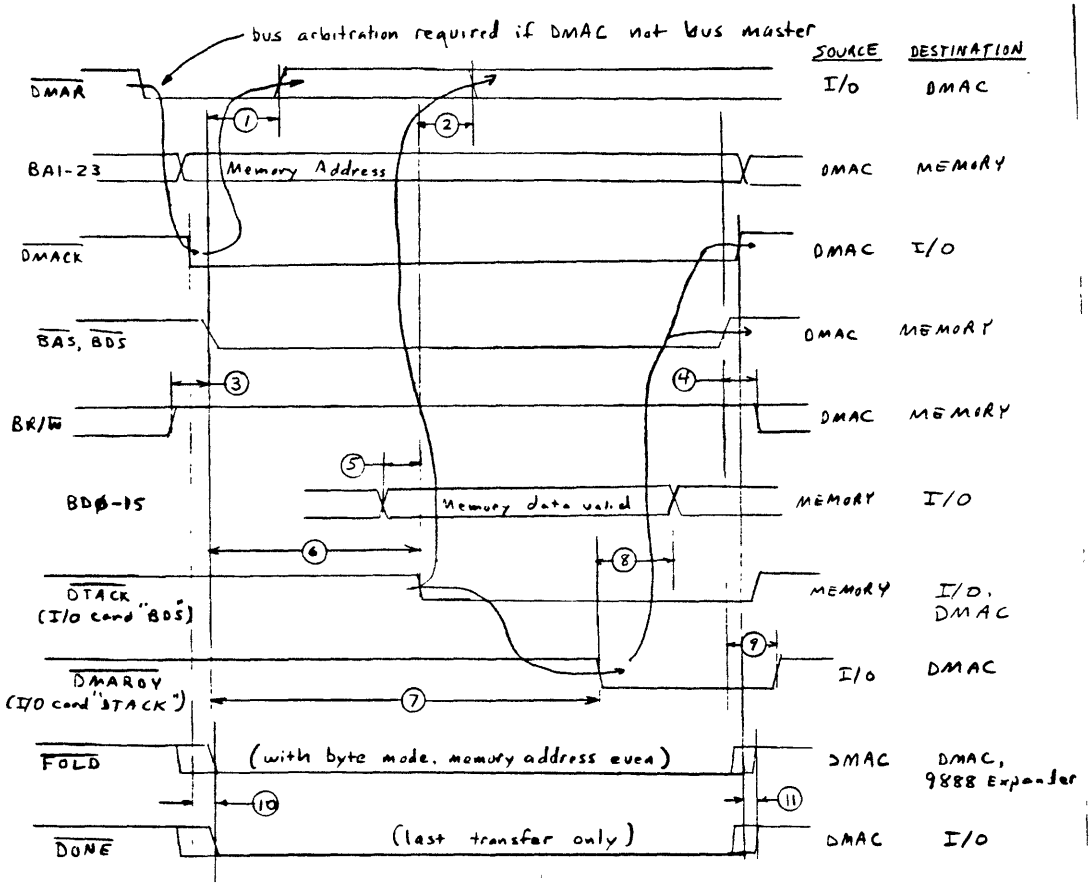
7. On the last transfer, the DMA Controller generates the DONE signal to tell the I/O card that 'this is the last byte'. An I/O card can, at its option, use this bit to inhibit further DMA Requests. Once the transfer count is satisfied, the DMA controller ignores further DMA Requests and relinquishes the bus.

   The DONE signal works as follows: DONE is asserted by the DMA Controller on the last DMA transfer with the same timing as DMACK0. 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. Also, the I/O card can use DONE to tag the last byte with an indicator (e.g. EOI on the 98625 card).

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

FIGURE 9-1: DMA OUTPUT CYCLE



	DMA OUTPUT TIMING	MIN	MAX	NOTES
1	DMAR* release after DMACK*	0		1
2a	DMAR* low after DTACK* low, Priority = 0		65	2,3
2b	DMAR* low after DTACK*, Priority = 1		1600	2,3
3	BR/W* setup before DMACK* low	15		
4	BR/W* hold after DMACK* high	15		
5	Data setup before DTACK* low	55		4
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	
10	DMACK* low to FOLD*, DONE* low	-15	53	
11	DMACK* high to FOLD*, DONE* high	-15	53	

NOTES: 1. The I/O card may keep DMAR* low after its request is acknowledged if it intends to do multiple DMA cycles.

2. To request another DMA cycle, the I/O card must assert assert DMAR* within the specified time after the memory device's DTACK*.

3. After satisfying Specification 2, the 98625A Disc Interface card can set DMAR* high. Thus, during the next cycle, DMAR* may still be high when DMACK* goes low.

4. Memory must provide a minimum of 55 ns data setup time prior to DTACK in order to provide 15 ns of data setup time for the I/O card (due to 40 ns delay through the Fold Buffer).

9.5 DMA INPUT CYCLE

Figure 9-2 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 asserts DMARDY with a minimum data setup time of 15 ns. Because this setup time is measured at the RECIEVER end of the bus, the setup time (for data and DMARDY) at the inputs to the DRIVERS on the I/O card must be 30 ns. DMARDY indicates to the DMA Controller that the bus data is valid.

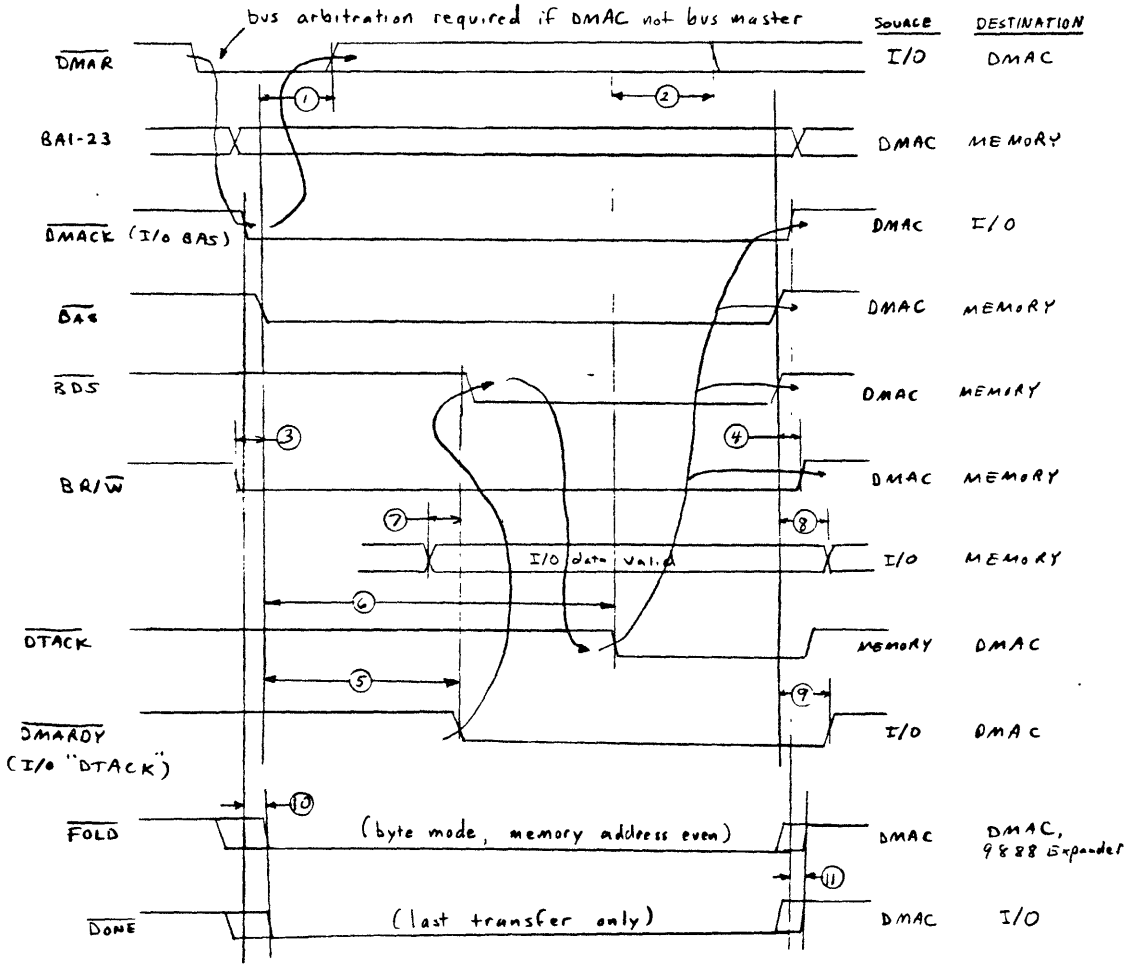
5. IF the DMA transfer is a byte transfer and the data is to be written to the upper byte of memory, the DMA Controller uses its Fold Buffer to move the byte from the lower data byte to the upper data byte. In either case, the DMA Controller detects that DMARDY has occurred and asserts the BLDS and/or BUDS to indicate to memory that data is valid on the bus.

6. The memory then stores the data and asserts DTACK to indicate that data has been accepted. The DMA Controller detects that DTACK has occurred and ends the cycle by removing BAS, BLDS/BUDS and DMACK0. In response to the removal of BAS, the memory card removes DTACK; likewise, in response to the removal of DMACK0, the I/O card removes DMARDY.

7. On the last transfer, the DMA Controller generates the DONE signal to tell the I/O card that 'this is the last byte'. An I/O card can, at its option, use this bit to inhibit further DMA Requests. Once the transfer count is satisfied, the DMA controller ignores further DMA Requests and relinquishes the bus.

8. A bus error also causes the DMA Controller to terminate the DMA transfer and relinquish the bus. A bus error occurs if a DMARDY does not occur within 2.5 usec of DMACK0 going true.

FIGURE 9-2: DMA INPUT CYCLE



	DMA INPUT CYCLE TIMING	MIN	MAX
1	DMAR* release after DMACK*	0	
2a	DMAR* low after DTACK* low for burst mode, Priority = 0		65
2b	DMAR* low after DTACK*, 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	15	
8	Data hold after DMACK* high	0	100
9	DMARDY* release after DMACK* high	0	50
10	DMACK* low to FOLD*, DONE* low	-15	53
11	DMACK* high to FOLD*, DONE* high	-15	53

9.6 DMA SPEED CONSIDERATIONS

To optimize the speed of DMA transfers, the transfer (data read, data write) must be completed during a single bus cycle. Also, the overhead time of the DMA Controller must be minimal and the device connected to the I/O card must be able to provide or accept the data immediately. The time for the existing DMA Controller (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. This time is either 65 ns (DMA Priority bit = 0) or 1.6 usec (Priority bit = 1). See specification 2 in Figures 9-1 and 9-2. Burst mode requires a 65 ns response; if the I/O card does not assert DMAR 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 usec DTACK-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; this works as follows:

1. In Burst Mode, the Priority bit is 0 and 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.

2. If the Priority signal for the channel is 1, then the bus is not relinquished until 1.6 usec after the last transfer is complete.

9.7 TERMINATING DMA TRANSFERS

DMA transfers can be terminated in several ways:

1. The DMA Controller can be programmed to interrupt the Master Controller after the transfer is complete and the bus relinquished.

2. The Bus Master can monitor the ARM bit in the DMA Controller between DMA cycles (assuming that the bus is released between DMA cycles). When the ARM bit is 0 after completion of the DMA transfer, the Bus Master can react accordingly.

3. The I/O card can use the DONE signal from the DMA Controller to interrupt the Bus Master. The 98625A Disc Interface card uses this technique.

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

1. BUS DRIVE DISABLE (BDRV*)
2. RESET*
3. HALT*
4. FUNCTION CODES (BFC0, BFC1, BFC2)

10.1 BUS DRIVE DISABLE

The 09826-66516, 09826-66517 and the 09816-66511 Processor boards respond to the BDRV* signal (this signal is called BMON* in the 9816). The 98206A Test Stimulus Board generates BDRV to disable the CPU board boot ROM from responding and to enable the test code on the Test Stimulus board to respond. This permits testing of the computer without depending on a working boot ROM.

Because BDRV* is primarily a diagnostic/development tool and is not used by I/O cards, detailed information is not provided here. For more information on the functions of BDRV* (BMON*), refer to the appropriate processor board documentation and the 98206A documentation.

10.2 RESET OPERATION

In the 9826 and 9836, RESET and HALT are asserted by the power supply at power-on and remain true until 120 msec after +5V reaches 4.5 volts. When RESET goes false, the 12V supply will have been within its 11.5V limit for at least 55 msec. At power-down, RESET is re-asserted within 15 msec after 5V drops to approximately 4 volts. Note that RESET comes too late to properly reset devices at power-down; RESET would ideally be asserted before 5V reaches its TTL limit (4.75). Refer to the SERIES 200 SYSTEM SPECIFICATION for more information on power-on RESET.

In Series 200 mainframes, activating RESET on the DIO BUS does not reset the Processor board unless HALT is simultaneously asserted; thus, RESET by itself only resets other bus devices. When RESET and HALT are both asserted, the Processor board is reset to its power-on state. In response to RESET and HALT, certain Processor boards refloat their RAM in preparation for auto locate while others keep the address fixed; refer to individual Processor board documentation.

In addition to responding to an external RESET, 68000-based Processor boards can use the RESET instruction to strobe RESET* low on the DIO BUS.

Thus, in specifying RESET* timing, there are 3 areas that need to be covered:

1. The duration of RESET* at power-on to properly reset the Processor -- In new designs, RESET* and HALT* should be low at power-on and remain low until all supplies have stabilized within their operating limits for at least 150 ms. At power-down, RESET* should go low before any supply exceeds its operating limits.

2. The duration of RESET* (and HALT*) after power-on to reset the Processor -- RESET* (and HALT*) should be 50 usec. minimum.

3. The width of the RESET* output pulse generated by the Processor in executing the RESET instruction -- The minimum RESET* pulse width is 8 usec.

10.3 HALT OPERATION

HALT is asserted on the DIO BUS at power-up in Series 200 mainframes with the same timing as RESET. In addition, HALT can be asserted by an DIO BUS device to halt the CPU. HALT cannot be asserted on the DIO BUS by existing Processor boards because their is no output driver from the CPU to the HALT pin; there is, however, a receiving buffer which permits a DIO BUS device to halt the processor in Series 200 mainframes.

Processor boards containing 68000 CPU's support re-run if BERR* and HALT* are asserted per the CPU re-run timing requirements. However, re-run has never been tested in Series 200 mainframes so it is not known whether it functions correctly.

10.4 FUNCTION CODE SIGNALS

The Function Codes (FC0, FC1 and FC2) generated by the 68000 are buffered (BFC0, BFC1 and BFC2) and brought out on the bus for general purpose use and for future expandability. As would be expected, the Function Code buffer is disabled when bus control is passed; however, this buffer is also disabled (on certain Processor boards) during an interrupt acknowledge cycle to inhibit certain control signals which happen to share the same buffer. Therefore, when FC0 = FC1 = FC2 = 1 (interrupt acknowledge), the buffered Function Codes on the bus will float (no pullups) and are undefined. Therefore, to use Function Codes, these guidelines should be followed:

1. When BAS occurs, the Buffered Function Codes on the DIO BUS are valid. BAS* is one of the control signals disabled during an interrupt acknowledge cycle; however, since BAS* is pulled up by a pullup resistor, it does not float (in contrast to BFC0-2, which do float).

2. During an interrupt acknowledge cycle, BFC0, BFC1 and BFC2 on the bus are undefined and cannot be used.

This chapter defines the non-timing electrical specifications for the DIO BUS. Included in this chapter are the power supply tolerances, the I/O card power dissipation specifications and the signal loading. Pinouts of the DIO BUS are discussed in the Chapter MECHANICAL SPECIFICATIONS.

Note that all slots in the I/O backplane are identical in the 9826, 9836A/C, 9816, 9888 and the 9920; there is no ordering or prioritizing of address, interrupt capability, etc. by location in the backplane. Future products, however, may have slot-dependent features.

11.1 POWER DISTRIBUTION AND GROUNDING

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

+5 Vdc -- Main logic supply

+12 Vdc -- Provided for I/O circuitry requiring multiple voltages. Can also be used for analog applications. NOTE: This supply is used for the floppy drive motor in 9826A's; designers should be aware of potential noise on the line.

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

The DIO BUS (see the next Chapter) provides 14 logic ground lines (GND). In addition, 2 lines are used for DGND (dirty ground) to be used for future cards which generate excessive ground noise (e.g. a card with relays). The DGND lines run separately from the I/O backplane thru the motherboard to the power supply. Because they are not presently used by I/O cards, they may be redefined to be 'clean ground', i.e. for use by analog I/O cards; this will be discussed further by the DIO BUS committee.

11.2 POWER SUPPLY TOLERANCES

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

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

11.3 I/O CARD POWER AND CURRENT REQUIREMENTS

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

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

NOTES: 1. The 'average' DC current represents just that, the average current over some small period of time. Thus, peak currents (such as might occur during a memory refresh) are not a concern.

2. Maximum power and current data is for reference only. The designer should ensure that typical power and current specs are not exceeded.

3. Any combination of currents may be used as long as current and power specs are not exceeded. For example, to stay within 4.4 watts for a low-power card, it is not possible to use the typical currents on all supplies, which would dissipate 5.92 watts. If the typical current on +5V is .8 amps, then .4 watts (4.4 - 5 X .8) can be drawn from +12V and -12V.

4. Expression of I/O card power requirements in terms of a standard deviation about a mean is not necessary because: (1) sampling of hardware in production demonstrated that the spread is very small and (2) the power supply has sufficient capacity to handle heavier loads.

11.4 ON-CARD FUSE REQUIREMENTS

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

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

However, FSD's Product Regulation Department states that because of the possibility that I/O cards can operate in mainframes with +/- 12V supplies capable of supplying more than 8 amps, these supplies should also be fused. Because the fuses are inexpensive and auto insertable, this is not seen as a serious disadvantage.

11.5 SIGNAL LOADING

The table below shows the receiver loading and recommended driver for each of the signals.

	Receive Loading	Signal Driver
BAS*, BR/W*, BUDS* and BLDS*	1 LS load max	SN74LS245 buffer
DTACK*, IACK*, BG*, DONE*, BERR*, DMACK0 - DMACK1 INT3* - INT6*, VECTOR*, BR*, DMARDY*, IMA*, DMAR0*-DMAR1*	1 LS load max	Any 3S/OC LS gate
BGACK*, ENDT*	1 LS load max	Any 3 state LS gate
BG1* - BG3*	2 LS loads max.	Any LS gate
BFC0* - BFC2*	2 LS loads max.	SN74LS245
RESET*	5 LS loads max.	SN7417 OC Buffer
HALT*	.8 ma TOTAL for card cage	Any 3S/OC LS gate
BA0 - BA23	1 LS load max.	SN74LS245 Buffer
BD0 - BD15	SN74LS245 Buffer	SN74LS245 Buffer

NOTES: 1. 3S/OC = 3-State driver or Open Collector

2. 'Any 3S/OC LS gate' was indicated as the signal driver for BGACK* in previous versions of this document. However, problems were encountered with the slow rise times of an open-collector driver in a recent project; thus, the spec has been changed to required a 3 state driver.

3. ALS gates are acceptable as receiving devices. Also, Schmitt trigger input devices should be used where possible as receivers.

4. Until ALS drivers are characterized on the bus, only 74LS244/245 drivers should be used.

MECHANICAL SPECIFICATIONS	CHAPTER 12

This chapter presents sufficient mechanical information to ensure that DIO BUS backplanes, card cages and PC boards are mechanically compatible.

12.1 DIO BUS CARD SPECIFICATIONS

The drawings on the following page shows the size requirement for I/O cards, non-I/O cards and the metal coverplate. The main points are:

1. I/O cards fit in every other slot of the I/O backplane and are secured by thumbscrews thru a metal backplate. The slots in between hold non-I/O cards such as RAM and ROM cards.

2. Non-I/O cards have a recess at the rear to allow clearance for the connector of the next-lower I/O card.

3. I/O cards have a keep-out area in the rear where traces and parts are not allowed to prevent them from shorting to the metal backplate.

4. I/O cards also have a keep-out area along each side to keep traces (specifically pop-thrus) away from the cardguides. Solder can bead up through pop-thrus, causing jamming with the I/O card guide. Refer to Section 12.2, item 1, for more information.

5. For both types of cards, space must be left on either side of the board to prevent components from interfering with the card cage guides.

6. I/O connectors are left-justified and extended to the right as needed for the size of the connector.

7. Note that keying (to prevent inserting a card upside down) is provided by the 2 different length notches on each side of the edge connector.

FIGURE 12-1: I/O PC BOARD SIZE

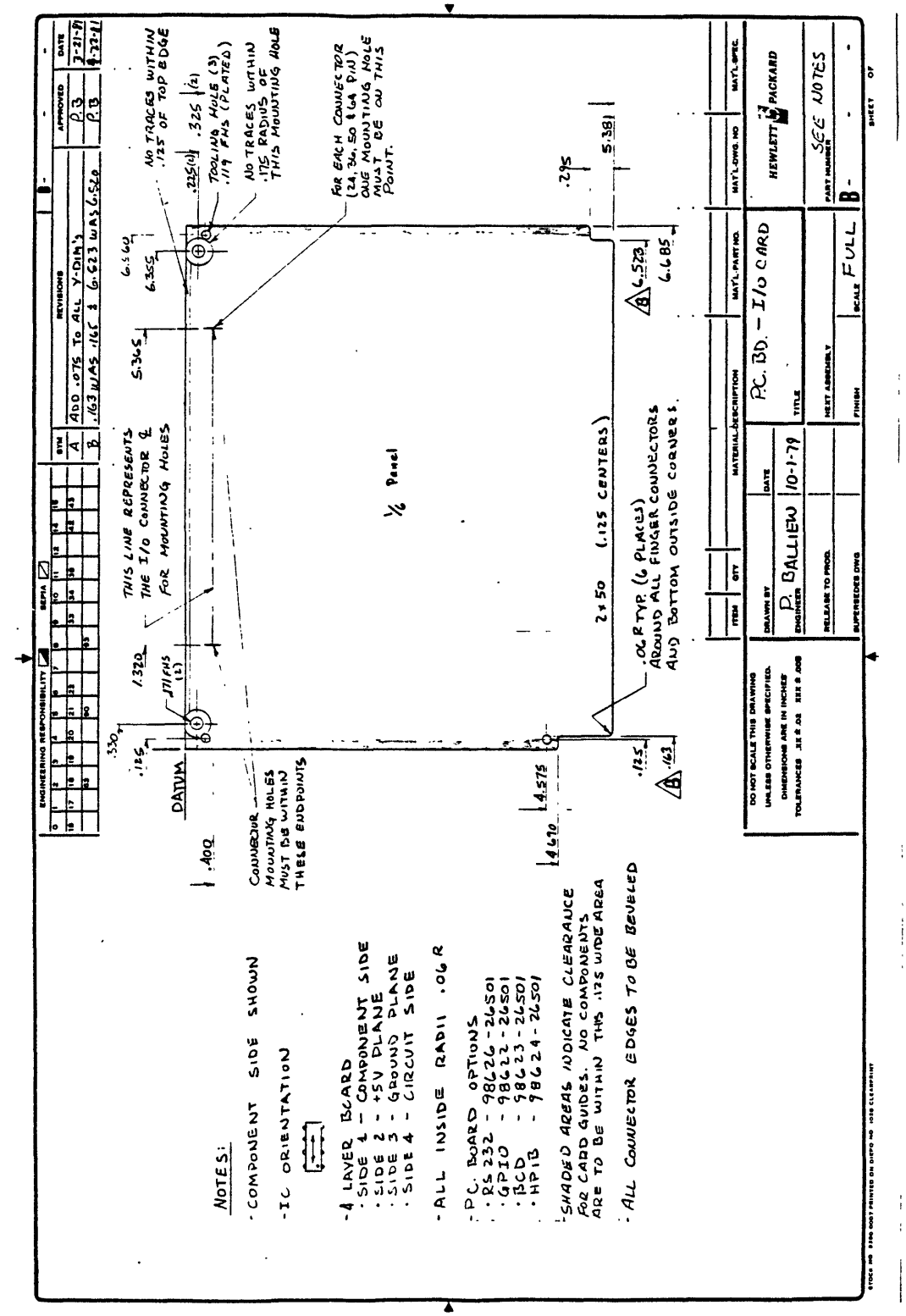


FIGURE 12-2: NON-I/O PC BOARD SIZE

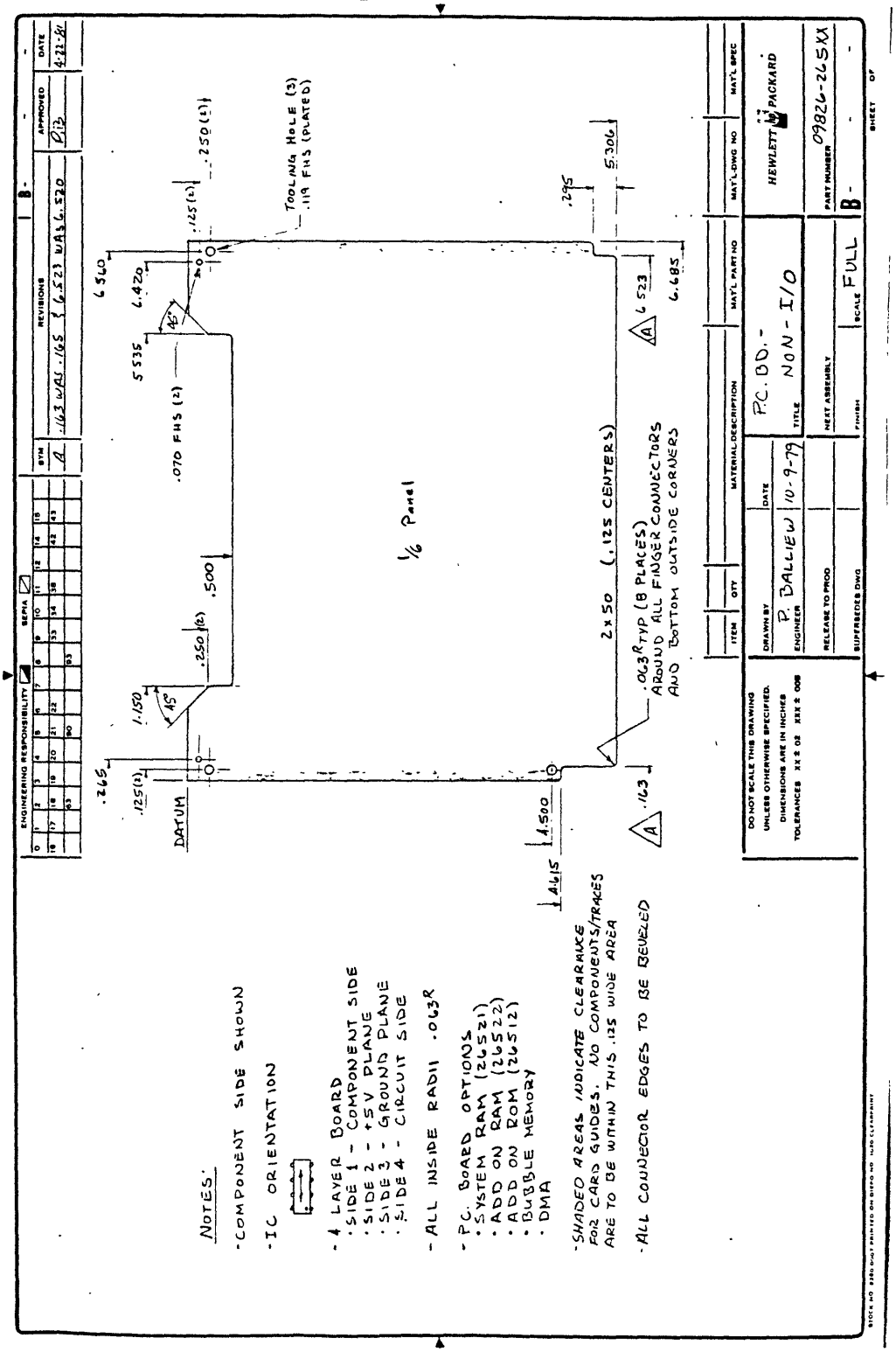
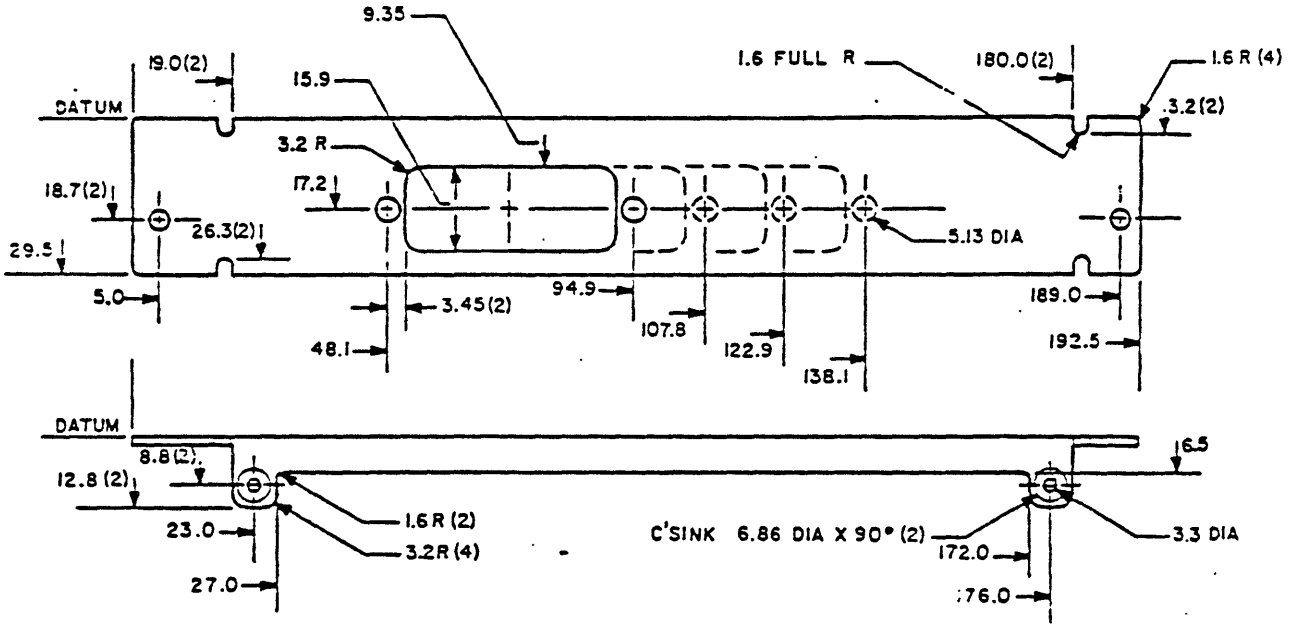


FIGURE 12-3: I/O COVERPLATE DRAWING



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

12.2 CARDCAGE SPECIFICATIONS

Shown on the following 2 pages are a drawing of the 9826/36 cardcage and the I/O backplane. The following points are worth noting:

1. While a component keepout area is specified for cards, PC traces can (and do) run inside of this keepout area. Therefore, to prevent shorts, metal cardguides must not be used.

2. Even though I/O cards only user every other connector on the backplane shown in Figure 12-5, new designs should implement this same connector scheme. In addition to providing slots for RAM cards, ROM cards, etc., this scheme also supports 'double-high' I/O cards that require both boards to connect to the bus.

3. The center-to-center board spacing is 15 mm (.59 inches).

12.3 MINIMIZING RFI

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

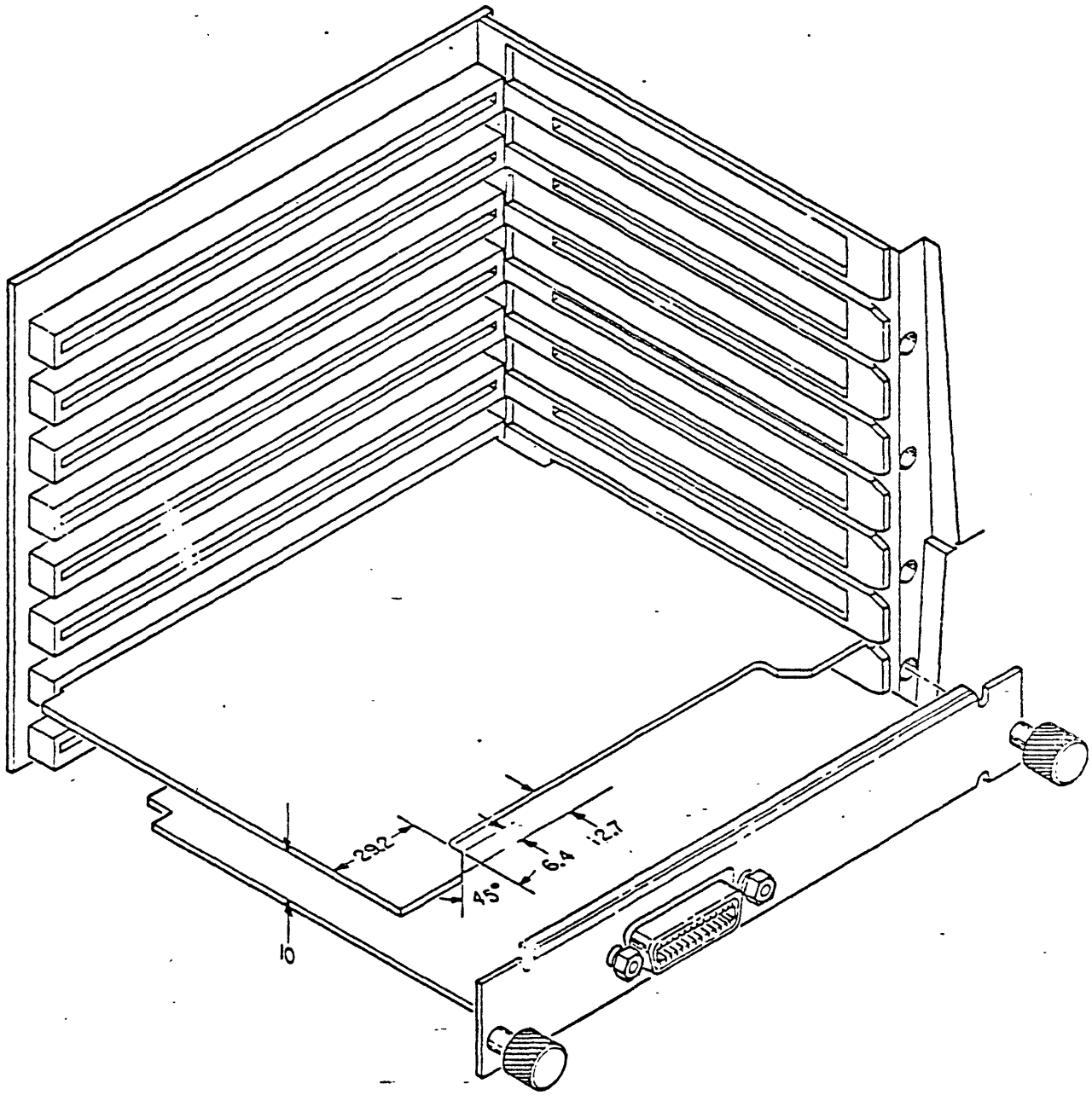
1. PC boards should be a minimum of 4 layers, with planes 2 and 3 reserved for power and ground, respectively. Boards greater than 4 layers should maintain power and ground on the middle layers so that a good high frequency bypass capacitor is formed between the power and ground planes. This minimizes the current loop area.

   It has been suggested that the power and ground layers should be placed nearer the outer layers to act as shielding for inner layer traces. In general, it has been found that placing power and ground on the middle layers (to form the high frequency bypass capacitor) is more effective in minimizing RFI.

2. I/O cards must contain a sheet metal coverplate with thumbscrews to securely mount the board in the computer and provide good electrical connection to safety ground.

3. Grounding of I/O cable housings or shields is done through the I/O coverplate to the mainframe rear panel. Appropriate techniques must be used to ensure solid contact between the connector housing and the coverplate. Also, the thumbscrews must provide good contact between the coverplate and the rear panel. If the shield ground appears on a pin of the connector (as with HP-IB), a generously-sized PC trace must go between the connector pin to one of the screw pads which secure the coverplate to the PC board.

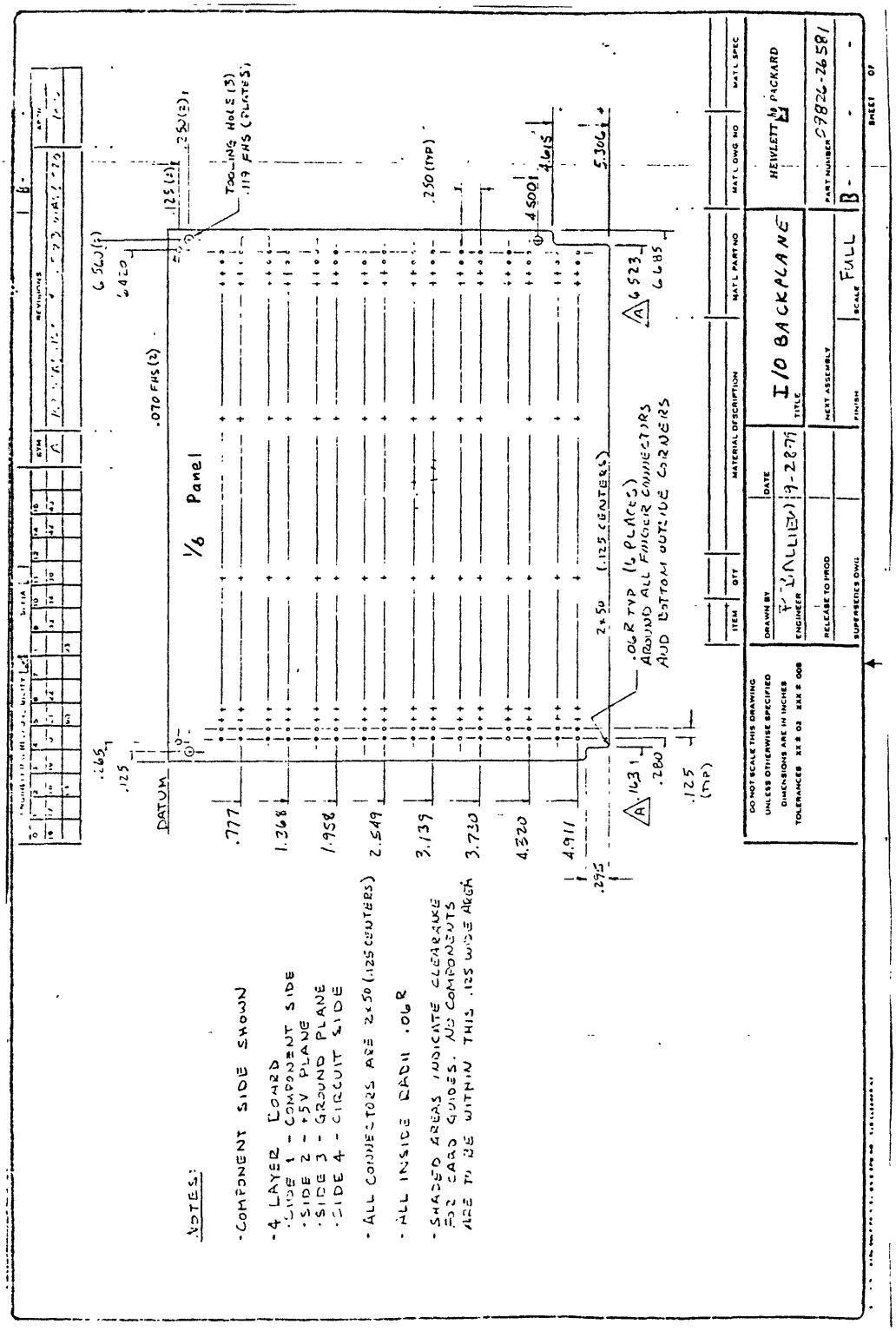
FIGURE 12-4: 9826/36 I/O CARDCAGE



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.

FIGURE 12-5: 9826/36 I/O BACKPLANE



12.4 PC CARD LAYOUT RULES

The following PC layout rules should be met:

1. Each card must limit loading to one LS or ALS load on the following signals: BA1-BA23, BAS, BD0-BD15, BUDS, BLDS and BR/W.

2. 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. Therefore, for 6-8 layer boards containing power and ground on the middle layers, the bus signals should be in the outer layers.

3. If the I/O cable connector contains a drain or shield wire, it should be connected by a generously-sized trace to the screw which secures the I/O backplate to the PC board. The connection to safety ground is then completed by the backplate, thumbscrew and rear panel assembly. The drain or shield wire should not be connected to logic ground.

4. PC boards should be a minimum of 4 layers, with planes 2 and 3 being power and ground, respectively. For >4 layers, the middle layers should be power and ground for a good RF bypass.

5. IC's are mounted parallel to the connector with pin 1 of the IC oriented to the lower left when viewing the board from the component side with connector pin 1 also oriented to the lower left. NOTE: This should be treated as a recommendation since some devices (e.g. the 98629 card) violate this.

6. The maximum component height is 10 mm (.39 inches) to prevent components from shorting to the leads of the next-higher board. As discussed previously, the board center-to-center spacing is 15 mm (.59 inches).

7. The PC edge connector is a standard S100 connector with 100 pins on .125" centers.

8. The edge connector finger width is .060". This dimension was carefully chosen and ensures that adjacent traces won't short in the connector.

12.5 DIO BUS PIN ASSIGNMENTS

Pin assignments for the DIO BUS backplane are shown on the next page. Odd pins are on the component side, even pins on the circuit side. Relative to viewing the board from the component side with the connector pointing down, the pins are numbered from left to right. The following Series 200 pinout considerations are worth noting:

1. The 2 DGND lines run from the I/O backplane thru the motherboard (connecting to the OEM slot) to the power supply; they are not bused to each connector on the 9826/36 motherboard. The OEM slot is an empty slot internal to the 9826/36 for potential use by HP OEMs.

2. Spare lines 0, 1 and 2 (pins 5, 27 and 28) are bussed from the I/O backplane to the CPU, Disc Controller and Expansion slots.

3. The 2 spare lines which were formally +/-15V (pins 97 and 98) are bussed from the I/O backplane connector to the CPU, OEM and Disc Controller slots.

4. IMA* and FOLD* (pins 41 and 42) are bussed to all I/O slots on the I/O backplane in Series 200 products; however, they are NOT connected to the I/O backplane PC edge connector. The I/O backplane female edge connector mounted on the motherboard has pins 41 and 42 tied to +5V.

5. The internal OEM slot connector is the same as the I/O backplane connector except pins 41 and 42 are no-connects. Because the OEM slot does not have the FOLD signal, neither the DMA Controller nor the Bus Expander card are electrically compatible with the OEM slot. Mechanically, the OEM slot accepts a board that is larger than the standard I/O board.

6. Spare pins MUST remain open until they are defined in this document. They are NOT subject to designer definition.

7. The 9826 and 36 reference Pin 29 as BDRV*; the 9816 references it as BMON*.

The DIO BUS pinouts are shown below. These are provided on the I/O backplane only; connectors internal to the 9826/36 do NOT have the same pinouts.

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	+5v	86	+5v	
	87	BD8	88	BD9	
	89	BD10	90	BD11	
	91	BD12	92	BD13	
	93	BD14	94	BD15	
	95	DGND	96	DGND	
	97	spare	98	Processor Spare (was STRETCH*)	
	99	-12v	100	+12v	

FSD manufactures the 9888A Bus Expander for the Series 200 computers. Devices for the Bus Expander are required to meet the same electrical and mechanical constraints as devices which plug directly into a mainframe. The Bus Expander employs delay lines and latches to ensure that all DIO BUS timing requirements are met (except as noted in section 13.2). This chapter discusses features of the Bus Expander and several limitations affecting operation of DIO BUS devices in the expander.

13.1 FEATURES OF THE BUS EXPANDER

The features of the Bus Expander are:

1. The Bus Expander plugs into an I/O slot in the computer; up to 4 Bus Expanders can be plugged into a computer. An Expander may NOT be attached to another Expander.

2. The Bus Expander is totally self-powered.

3. The Bus Expander has 16 connectors which will support 8 I/O cards and 8 RAM cards or 16 RAM cards.

4. A 5.2 foot cable connects the Bus Expander to the computer. Signals are buffered at each end of this cable (on the board that plugs into the computer and on the board in the expander). The IMA (I'm Addressed) signal discussed previously is used to turn these buffers around if the addressed card is in the Expander.

5. I/O cards that implement DMA (e.g. the 98622A GPIO card) can operate in the Bus Expander. If both the memory card and the I/O card involved in a DMA transaction are in the same Expander, then the Expander itself performs any necessary data folding operation in response to the DMA Controller's FOLD signal (if the DMA Controller did the folding, signal delays would be excessive).

13.2 OPERATING LIMITATIONS WITHIN THE BUS EXPANDER

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

1. Bus Masters cannot operate in the Bus Expander; there is no provision for 'turning around' certain signals such as the address bus. Thus, the 98620A DMA Controller must reside in the mainframe.

2. ROM boards are not guaranteed to work in the Bus Expander due to Auto DTACKing. With Auto DTACKing, ROM cards are required to have data present within a certain length of time. Because of signal delays between the Bus Expander and the mainframe, data setup time prior to the Auto DTACK is not guaranteed.

3. RAM boards will operate in the Bus Expander; however, RAM boards typically require 6-cycle accesses as opposed to 5-cycle access for boards installed in the mainframe. This is due to signal delays. Software being executed from the Bus Expander will thus run proportionately slower and timing loops will be altered.

4. Interrupt cards which implement external vectored interrupts will not work in the Bus Expander for 2 reasons:

   A. The VECTOR signal from the interrupting card will not (worse case) reach the Processor board in time to inhibit auto vectoring.

   B. Even if the VECTOR signal reaches the Processor board in time, the interrupt vector cannot be read by the Processor board. This is due to the fact that BAS is not generated during an interrupt vector cycle; without BAS, the card does not generate IMA, so the Bus Expander's data bus buffers do not 'turn around'.

5. The BDRV* signal discussed in the Chapter DIO BUS UTILITIES is not supported in the Bus Expander; cards which use BDRV* must be installed in the mainframe.

6. The BERR* signal is an input from the Bus Expander to the mainframe. A BERR* signal generated by the mainframe (e.g. the CPU board) cannot be seen by an I/O card.

7. The 9826/36 Powerfail option (which is installed internally in the mainframe) is not supported with the Bus Expander. The Bus Expander resets the computer when the power fails and again when power returns. This will destroy any data or programs in memory. For correct operation, the Expander must be turned on before the computer is powered up and not powered down while the computer is operating.

8. A timing error was recently (9-83) discovered in the Bus Expander. The error is on the 09888-66502 card which is the buffering electronics on the 'expander side' of the cable between the mainframe and the expander. This card uses BAS to latch BR/W. The purpose of this latch is to guarantee sufficient BR/W hold time on the trailing edge of BAS. The problem is that the latch's data (BR/W) setup requirement is not guaranteed prior to the clock (BAS) even though the card does delay BAS by 30 ns. A Production Change Order is currently (2-84) being implemented at FSD that will add an additional 35 ns of delay to BAS (65 ns total). At the same time, the board number will be changed from 09888-66502 to 09888-66504. Note that the likelihood of this problem occuring is low due to the number of timing parameters that must be 'biased' towards worst-case.

   The position of FSD is that units in the field were adequately tested in production and no re-work is required. However, the possibility still exists that a unit might be marginal and could (over time, temperature, etc.) exhibit intermittent (and typically undiagnosable) errors. Therefore, it is recommended that internal users upgrade their Bus Expanders with the 09888-66504 board if any inexplicable problems are encountered.

The subject matter of this chapter is now covered in the SERIES 200 SYSTEM SPECIFICATION available from the FSD Hardware R&D Lab; accordingly, the contents of this chapter have been deleted. The reader is encouraged to obtain the above document.

Although many details covering Bus Slave design are covered in other chapters, the key requirements are summarized below. This chapter covers primarily External I/O cards (e.g. the GPIO interface). Examples are provided; however, the reader is strongly encouraged to review the design of existing cards before attempting a new design.

15.1 EXTERNAL 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.

1. I/O cards may be either 8 bit or 16 bit devices. 8 bit devices should reside on the lower (odd) byte of the 16 bit data bus.

2. Designing I/O cards that can perform DMA transfers is left to the option of the designer. I/O cards that implement DMA can optionally use DONE from the DMA Controller to determine when the last transfer is occurring.

3. While a card is enabled for DMA, it must still respond to normal bus cycles so Status and Control registers can be accessed.

4. I/O cards must provide 5 Select Code switches to provide Select Codes up to 31.

5. I/O cards must implement Registers 1 and 3 as defined in the Chapter I/O MEMORY MAP AND I/O CARD REGISTERS. Register 5 may be implemented if an Extension ID is necessary. Additional registers and bits may be defined as needed.

6. Implementing interrupt capability is left to the option of the designer; however, it is STRONGLY recommended that interrupt capability be included. An I/O card without interrupts will not work well with UNIX; if the operating system has to poll an I/O card to determine its need for servicing, extended (e.g. 1 second) dropouts may occur. Because of the previously discussed problems with vectored interrupts, they should not be used.

   It is highly recommended that I/O cards provide switches to select interrupt levels 3 to 6. However, other options are possible, e.g.:

   A. Make the interrupt level programmable (e.g. use undefined bits in the card's CONTROL REGISTER to permit programming of the interrupt level bits).

   B. Permit interrupt levels to be set (either via switches or software) over a broader level than from 3 to 6 (e.g. from 1 to 7). This is NOT recommended, however, since special care is required in system configuration to avoid conflicts. For example, the 09826-66562 floppy control board used in certain Series 200 products is hardwired to interrupt level 2 and drives the interrupt signal with an LS04, i.e. it is not open collector. An I/O card on interrupt level 2 would fight with this LS04. Designers that want to implement interrupts outside of the range from 3 to 6 should consult with hardware and software people at FSD to evaluate potential problems.

7. If I/O cards are designed containing more than one interface (e.g. an RS-232 card containing four channels), the card can be designed in two ways:

   A. If it is desired that existing software work without modification, then the card must appear exactly like multiple single-interface cards, e.g. each interface should occupy its own 64K address slot.

   B. If the design goals do not permit the hardware to be identical with existing I/O cards, then the software drivers will have to be re-written.

8. For testability reasons, it is advised that no write-only registers be implemented; all registers should be read/write. Also, no read-then-clear registers should be used.

9. I/O card designers should design to the DIO BUS functions outlined in the Chapter REQUIRED DIO BUS FUNCTIONS FOR AN I/O BUS.

15.2 EXTERNAL I/O CARD DESIGN EXAMPLE

This section presents an example 8-bit I/O card design and describes the following interface elements:

1. BUS SLAVE INTERFACE (data transfer interface)

2. INTERRUPT REQUESTER

3. DMA REQUESTER

DATA TRANSFER INTERFACE

Figure 15-1 shows a typical circuit used for transferring data between the I/O device and the Processor. The key elements are:

1. U7, a 74LS688 comparator, compares the upper 8 address bits to the bit pattern for the External I/O memory space and the user-set Select Code switches. It is enabled by BAS*; because the address precedes BAS* by 15 ns, the output (CS*) does not have glitches.

2. CS* low generates IMA* low and enables the DTACK* buffer (U8). CS* low is not used to enable the bus driver, U1, because of a possible driver conflict between the I/O card and the CPU board. This would occur during a write cycle when BR/W* goes low after BAS* goes low; BR/W* low causes the CPU drivers to drive the data bus while the I/O card would also drive the data bus (until it recognized that BR/W* is low, within 15-20 ns). To avoid this, the I/O card does not enable its bus driver until BLDS* is low.

3. U6, pin 3 (U6-3) goes low (indicating the start of a cycle) when CS* is low and BLDS* is low. U6-3 low generates DTACK* low with some delay, regardless of whether the operation is a read or a write. The delay time is set to the longest time required for either reading a register (plus 15 ns setup on the bus) or setting up a register for clocking when DTACK is asserted. U6-3 also enables the data bus buffer U1.

4. Reading and writing are then determined by BR/W* as follows:

   A. READING (BR/W* high)

      (1) BR/W* high enables U1 to drive the data bus BD0-BD15.

      (2) U6-3 low and BR/W* high enables the read decoder (U5, 1Y0-3) which enables the device at the address set by BA1-BA2 to drive the internal data bus (and thus BD0-BD15). As mentioned above, the DTACK delay circuit ensures 15 ns of setup time prior to setting DTACK low.

      (3) The card remains in this state until BLDS* or BAS* goes high, disabling U1 and the read decoder.

   B. WRITING (BR/W* low)

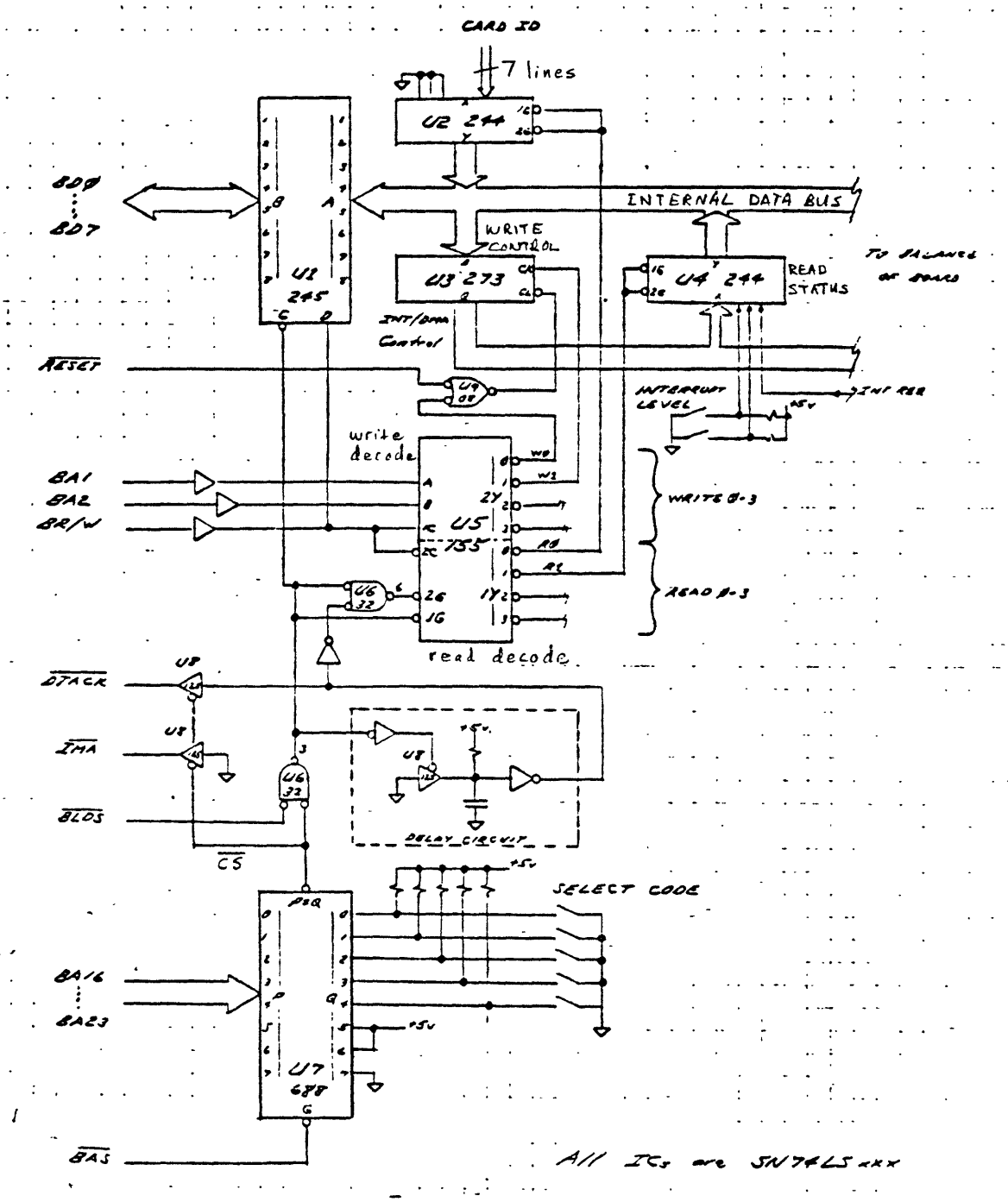
      (1) BR/W* causes U1 to drive the internal data bus.

      (2) U6-3 low and BR/W* low enables the write decoder (U5, 2Y0-3) which generates write strobes depending on the address BA1-BA2.

      (3) 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).

      (4) The write cycle ends when BLDS* or BAS goes high.

FIGURE 15-1: DATA TRANSFER INTERFACE



INTERRUPT REQUESTER

Figure 15-2 shows a typical interrupt request circuit. The circuit operates as follows:

1. If INTERRUPT REQUEST and INTERRUPT ENABLE are true, one of the interrupt request signals (IR3*, IR4*, IR5* or IR6*) will be low, as determined by the 2-bit INTERRUPT LEVEL setting.

2. A software poll is used to determine which card is interrupting. Bit 6 of the Status Register indicates INTERRUPT REQUEST; bit 7 indicates INTERRUPT ENABLE.

NOTE: BECAUSE EXTERNAL INTERRUPT VECTORING DOES NOT WORK IN THE BUS EXPANDER, THE CIRCUITRY FOR IMPLEMENTING EXTERNAL VECTORED INTERRUPTS IS NOT DISCUSSED.

DMA REQUESTER

Figure 15-3 shows a typical DMA interface; it operates as follows:

1. A DMA REQUEST signal from the I/O card causes DMAR0* or DMAR1* to go low if the DMA channel is enabled as determined by DE0 or DE1 in the Control Register.

2. DMACK0 or DMACK1 enables the data bus buffer U1. 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 15-2: INTERRUPT INTERFACE

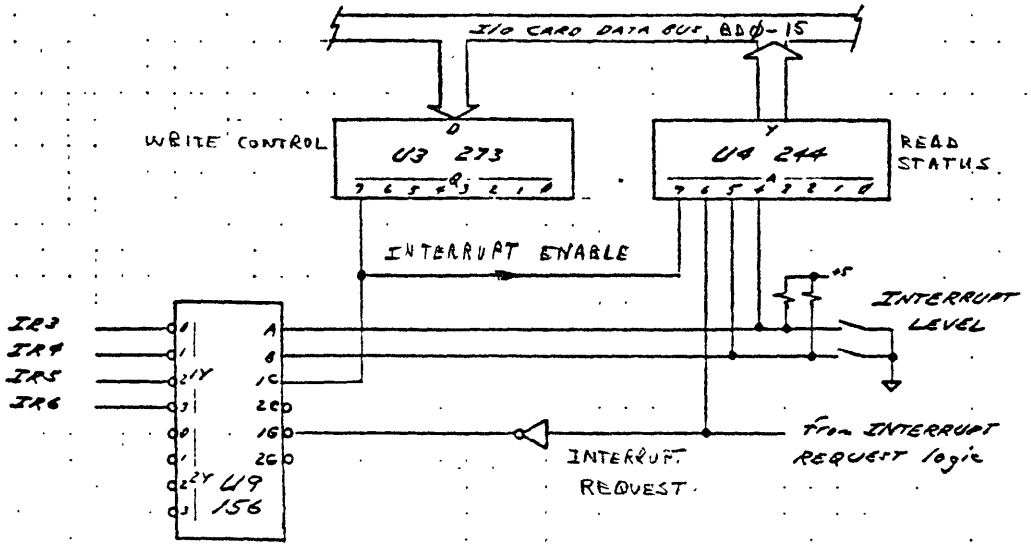
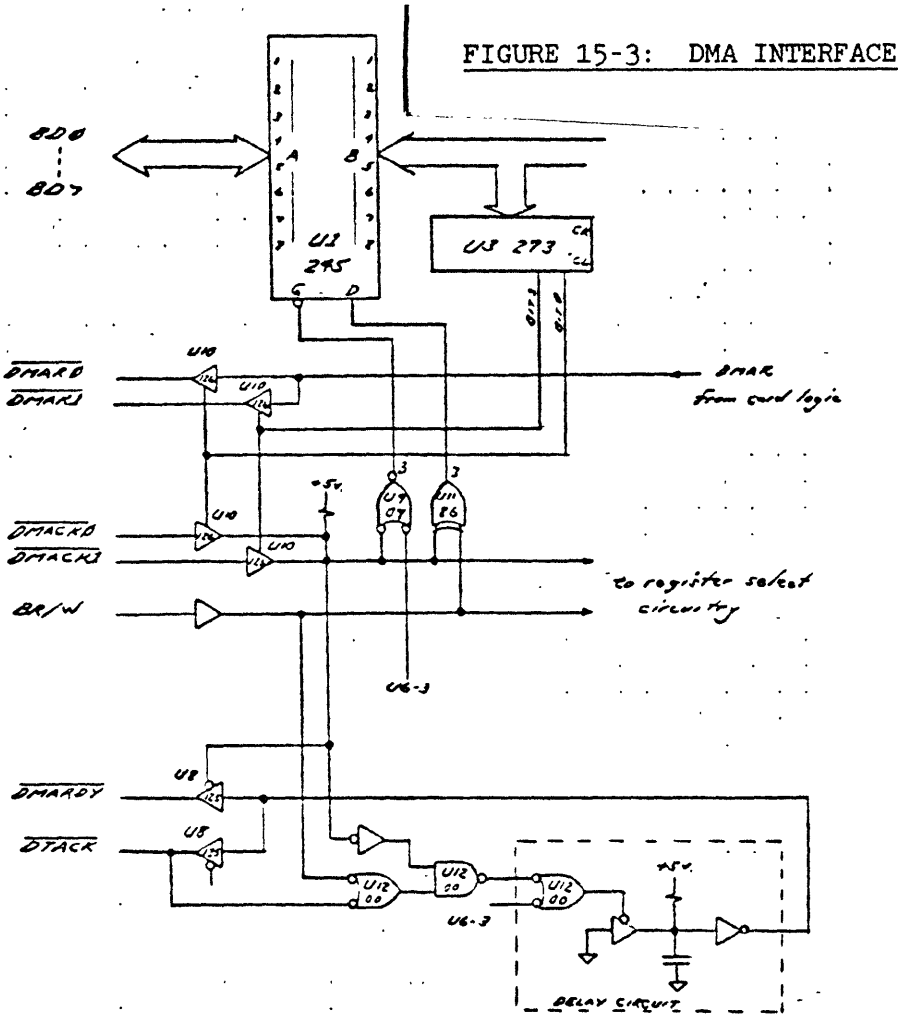


FIGURE 15-3: DMA INTERFACE



Product plans at FSD defined a product (which has since been cancelled) which used the DIO BUS strictly as an I/O bus (i.e. not a memory/processor bus). Accordingly, an investigation was made of what DIO BUS signals would be required on a DIO-based I/O bus. Because the subject may arise again with other future mainframes, the outcome of this investigation is covered in this chapter. This chapter discusses those signals that are required on a DIO-based I/O bus and those signal which are not supported (and thus which must not be used by I/O cards).

16.1 SIGNALS NOT SUPPORTED ON A DIO-BASED I/O BUS

As an I/O bus, not all of the functions provided by the DIO BUS are needed. In addition, there are other functions, such as vectored interrupts, which are not usable and can be eliminated. Listed below are those functions which may not be supported in mainframes which have a DIO-based I/O bus. Therefore, I/O card designers should not (repeat: not) expect these functions and signals to be available. The unsupported signals must be kept open on the I/O backplane and must not be used for other functions.

1. FUNCTION CODES (BFC0, BFC1 and BFC2): These are provided by 68xxx processors and are not available from other processors (e.g. SPECTRUM); therefore, I/O card designers should not use these signals.

2. VECTORED INTERRUPTS (IACK*, VECTOR*): Vectored interrupts are not supported for several reasons:

   A. No vectoring cards have ever been built, either on the bench or in production. Different BOOT ROMs in different mainframes support vectored interrupts differently. Designing a vectoring card would require an extensive qualification effort and would introduce uncertainty as to whether a vectoring card would work in machines already in the field (since this capability has not been tested at the factory).

B. The DIO BUS does not provide a priority scheme to arbitrate among different vectoring cards on the same interrupt level. Therefore, only one vectoring card per interrupt level would be permitted, limiting its usefulness.

C. Vectoring cards, if designed, will not work in the 9888A Bus Expander.

NOTE: Even though I/O cards do not implement vectored interrupts, mainframes themselves may provide this capability for I/O cards. For example, an 'intelligent backplane' can detect that an interrupt request line from an I/O slot is being pulled and provide a vector during the Processor's interrupt acknowledge cycle.

3. BDRV* (BMON*): This is for testing and permits test card code to 'replace' the BOOT ROM on certain processor boards. It is not supported on a DIO-based I/O bus.

4. FOLD*: In Series 200 mainframes, this signal is provided by the 98620A DMA Controller and is used for byte packing from an 8 bit I/O card to 16 bit wide memory. It is presently used only by the 9888A Bus Expander and the 98620A itself and is not used by I/O cards. Thus, it may not supported on a DIO-based I/O bus and should not be used.

5. BERR* as an input: DIO cards may generate BERR but must not expect BERR to be available as an input (for example, the BERR generated by a Processor board in response to a transfer timeout). While a card plugged into a mainframe will see a processor-generated BERR, a card plugged into the 9888 Bus Expander will not see it. This is because the expander drives BERR from the expander into the mainframe, not from the mainframe to the expander.

6. ENDT*: An existing guideline is that cards that work with ENDT* (Enable DTACK) must also be able to work without it (albeit at a lower speed). Therefore, I/O card designers may optionally design cards that use ENDT* (as long as the card will work without it).

16.2 SIGNALS REQUIRED ON A DIO-BASED I/O BUS

Those DIO BUS signals not discussed above are required to be implemented on an I/O bus. Most of these required signals need no further explanation: BD0-BD15, BA1-BA23, BUDS, BLDS, BR/W*, DTACK, IMA*, IR3-IR6*, DONE*, RESET*, HALT*, DMAR0*, DMACK0*, DMAR1*, DMACK1* and DMARDY*. However, several other signals deserve some discussion as to why they are required:

1. IR1*, IR2*, IR7*: This document specifies that I/O cards generate interrupts on IR3-IR6*. However, to preserve generality and expandability, a DIO-based I/O bus should respond to the remaining 3 interrupt levels, IR1*, IR2* and IR7*.

Because I/O cards will be used with different mainframes, operating systems, etc. a thorough I/O card qualification program is absolutely necessary to ensure that all hardware, software and safety issues are resolved. This is an area that is constantly changing as new hardware and software is released, as regulatory requirements change and as our interpretation of these requirements change. Designers are responsible for ensuring that their products are properly qualified per the latest operating environments and regulations.

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

1. SOFTWARE QUALIFICATION
2. HARDWARE QUALIFICATION
3. SAFETY COMPLIANCE

17.1 SOFTWARE QUALIFICATION

The key concern with software is, of course, its reliability. Sufficient QA should be performed to ensure that the card operates with the different operating systems. When operating systems are revised, new operating systems are released or when changes are made to an I/O card which affect its operation, additional software QA should be performed.

17.2 HARDWARE QUALIFICATION

Hardware testing can be divided into 2 areas:

1. CONFIGURATION TESTING
2. ENVIRONMENTAL TESTING

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

CONFIGURATION TESTING -- New I/O cards must be tested with certain mainframe configurations. For example, listed below are several mainframe/CPU board combinations that exist or are planned. This is not to say that a new I/O card should be tested with each combination or that it should be tested with only these combinations. For example, as new CPU boards are developed for new mainframes, the plan for configuration testing should be modified accordingly.

CPU BOARD	9816A	9826A	9836A	9836C	9920A
09816-66512	x				
09826-66514		x			
09826-66515		x	x		
09826-66516		x	x	x	x
09826-66517		x	x	x	x

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

ENVIRONMENTAL TESTING -- In performing environmental tests at FSD, a subset of possible configurations are selected for testing in the following areas:

1. Class B testing.

2. TAF testing (prototypes, small sample size, short duration, high stress). NOTE: TAF = Test-Analyze-Fix.

3. STRIFE testing (production boards, larger sample size, many test cycles). NOTE: STRIFE = Stress-Life.

Class B requires RFI testing. FSD Product Assurance has established the following guidelines for RFI testing of new I/O cards:

1. VDE level B-2 and FCC Class B RFI specifications must be met.

2. The new I/O card must be qualified with each product (or groups of products) that is licensed separately. I/O cards must currently qualify with the following mainframes.

    1. 9816A
    2. 9826A or 9836A or 9836C (all under one license)
    3. 9920A.

3. A system test is required. The system must consist of a mainframe, an input device (e.g. mass memory) and an output device (e.g. a printer). Where it makes sense, the new I/O card may be used to interface to the input or output device. To establish your exact test configuration, you need to contact FSD's Product Regulation Dept.

4. The system should actively exercise the new I/O card during testing.

5. Where a CRT is part of the system, an alternating pattern of dots on the CRT should be displayed.

ASIDE: IT IS HIGHLY RECOMMENDED THAT A DOCUMENT BE PREPARED GIVING DETAILED (AND CURRENT) REQUIREMENTS FOR HARDWARE AND SOFTWARE QUALIFICATION. SUCH A DOCUMENT IS THE BEST MEANS OF COORDINATING INTER-CORPORATE DEVELOPMENT AND QUALIFICATION OF I/O CARDS.

17.3 SAFETY COMPLIANCE

I/O cards should be designed to meet the normal UL/CSA/IEC safety requirements. In addition, the following guidelines have been developed within the Computer Group.

1. Ground current carrying capacity: The conductive path between the I/O cable shield and the safety ground pin of the power receptacle should be capable of carrying 30 amps for 120 seconds. This can also be expressed in resistance for the following 2 cases:

A. For cables less than 4 meters, the DC resistance between the end of the cable and the safety ground pin of the power receptacle should be less than 100 milliohms, measured after 120 seconds.

B. For cable lengths over 4 meters, the DC resistance between the I/O connector on the card itself and the safety ground pin of the power receptacle should be less than 100 milliohms, measured after 120 seconds.

2. Additional clarification is needed for the above specifications:

A. Product Assurance states that the mechanical connection between the connector shroud and the connector on the I/O coverplate cannot be relied upon to carry this current. Therefore, the I/O connector pin(s) alone must be capable of carrying this current. During testing, the connector shroud must be isolated from the connector on the I/O coverplate. Since the safety ground path from the I/O card to the power receptacle is via the thumbscrews (and not the PC edge connector), it is implied that the thumbscrew connection to the rear panel is a reliable connection.

B. Specifications that refer to measuring resistance between an I/O card and the power receptacle are obviously mainframe dependent; to achieve a mainframe independent specification for the designer, Product Assurance states that the designer may substitute 'I/O coverplate at the thumbscrews' for 'safety ground pin of the power receptacle' in specifications 1A and 1B above, while substituting 70 milliohms in lieu of 100 milliohms.

3. Because the +5V supply in the Series 200 computers has over an 8 amp fault current capability, an on-board fuse is required. Refer to Section 11.4 for the recommended fuse. Even though current Series 200 mainframes cannot supply 8 amps on the +/- 12V supply, fuses are still required in these supplies to ensure compliance in future mainframes that are capable of supplying 8 amps.