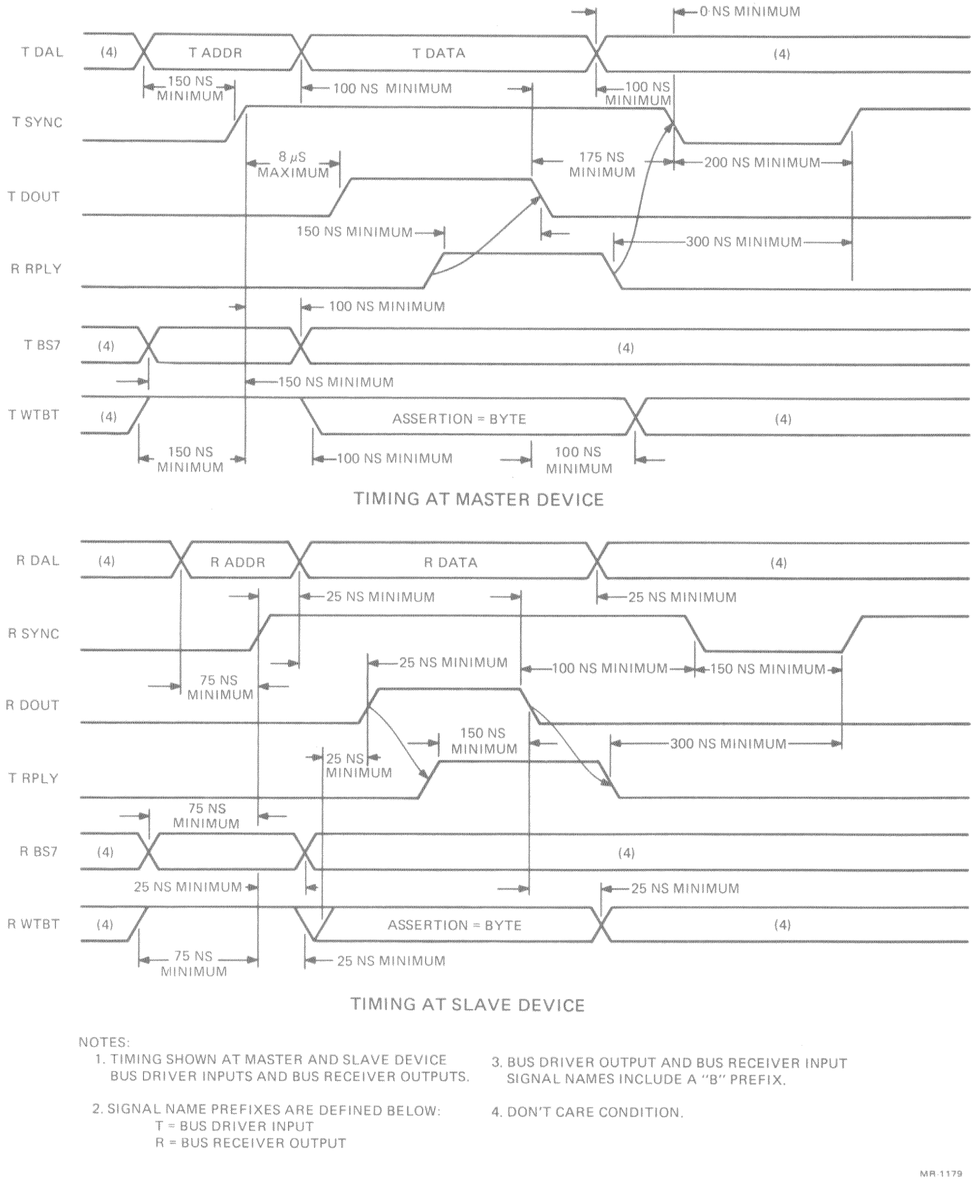


Figure A-4 DATO or DATOB Bus Cycle Timing

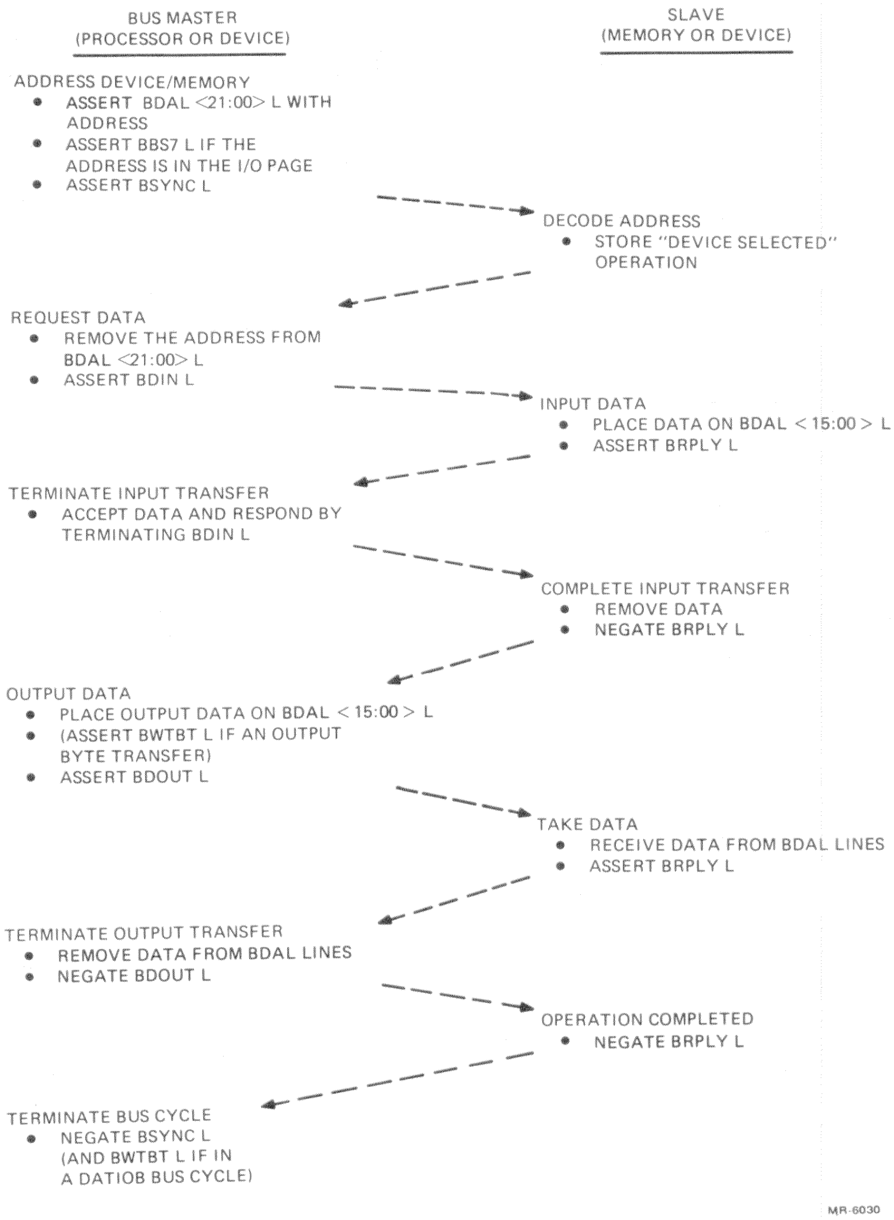
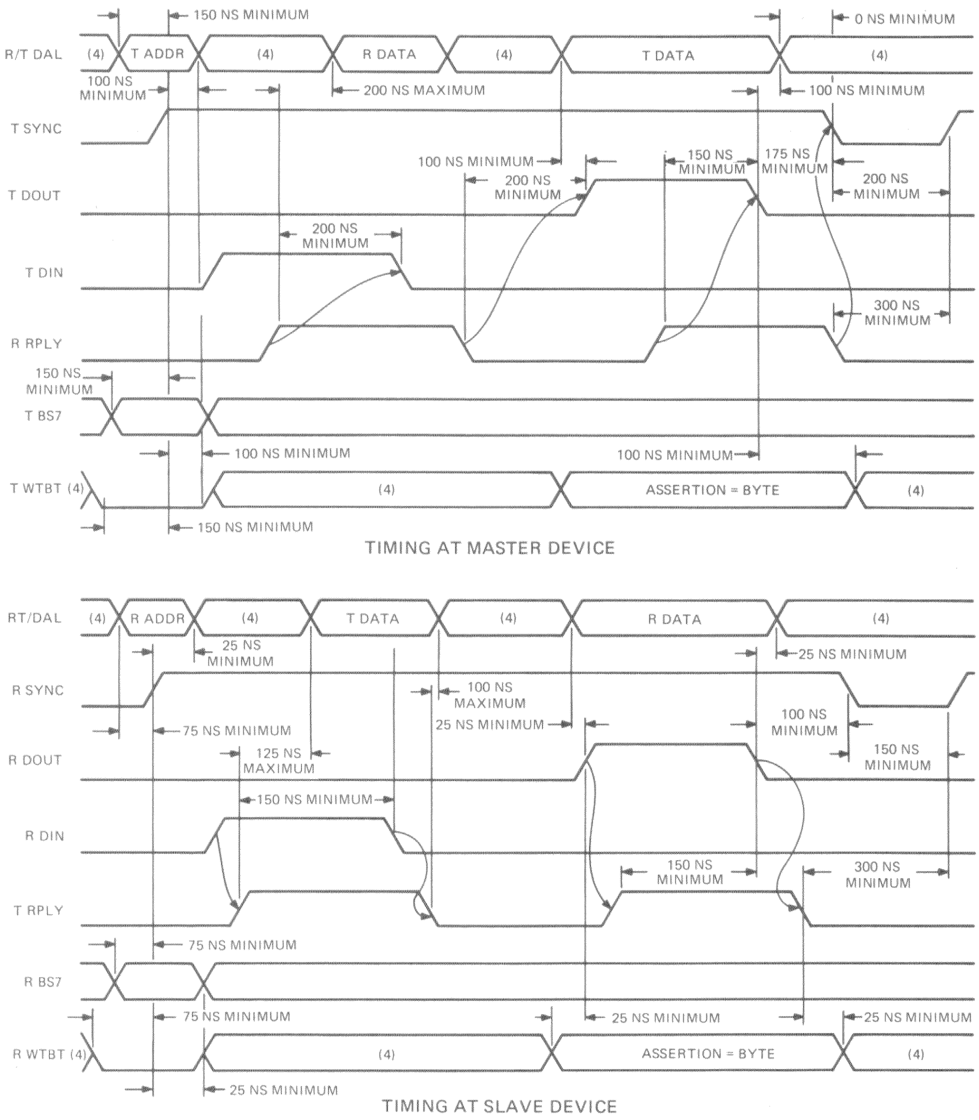


Figure A-5 DATIO or DATIOB Bus Cycle



NOTES:

1. TIMING SHOWN AT REQUESTING DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS
2. SIGNAL NAME PREFIXES ARE DEFINED BELOW:
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT
3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX.
4. DON'T CARE CONDITION.

MR-6036

Figure A-6 DATIO or DATIOB Bus Cycle Timing

A.4 DIRECT MEMORY ACCESS

The direct memory access (DMA) capability allows direct data transfer between I/O devices and memory. This is useful when using mass storage devices (for example, disks) that move large blocks of data to and from memory. A DMA device needs to know only the starting address in memory, the starting address in mass storage, the length of the transfer, and whether the operation is read or write. When this information is available, the DMA device can transfer data directly to or from memory. Since most DMA devices must perform data transfers in rapid succession or lose data, DMA devices are provided the highest priority.

DMA is accomplished after the processor (normally bus master) has passed bus mastership to the highest-priority DMA device that is requesting the bus. The processor arbitrates all requests and grants the bus to the DMA device electrically closest to it. A DMA device remains bus master until it relinquishes its mastership. The following control signals are used during bus arbitration.

BDMGI L	DMA grant input
BDMGO L	DMA grant output
BDMR L	DMA request line
BSACK L	Bus grant acknowledge

A.4.1 DMA Protocol

A DMA transaction can be divided into three phases:

1. Bus mastership acquisition phase
2. Data transfer phase
3. Bus mastership relinquishment phase.

During the bus mastership acquisition phase, a DMA device requests the bus by asserting BDMR L. The processor arbitrates the request and initiates the transfer of bus mastership by asserting BDMGO L.

The maximum time between BDMR L assertion and BDMGO L assertion is DMA latency. This time is processor-dependent. BDMGO L/BDMGI L is one signal that is daisy-chained through each module in the backplane. It is driven out of the processor on the BDMGO L pin, enters each module on the BDMGI L pin, and exits on the BDMGO L pin. This signal passes through the modules in descending order of priority until it is stopped by the requesting device. The requesting device blocks the output of BMDGO L and asserts BSACK L. If BDMR L is continuously asserted, the bus hangs.

During the data transfer phase, the DMA device continues asserting BSACK L. The actual data transfer is performed as described earlier.

The DMA device can assert BSYNC L for a data transfer 250 ns (minimum) after it received BDMGI L and its BSYNC L bus receiver becomes negated.

During the bus mastership relinquishment phase, the DMA device gives up the bus by negating BSACK L. This occurs after completing (or aborting) the last data transfer cycle (BRPLY L negated). BSACK L may be negated up to a maximum of 300 ns before negating BSYNC L. Figure A-7 shows the DMA protocol, and Figure A-8 shows DMA request/grant timing.

NOTE

If multiple data transfers are performed during this phase, consideration must be given to the use of the bus for other system functions, such as memory refresh (if required).

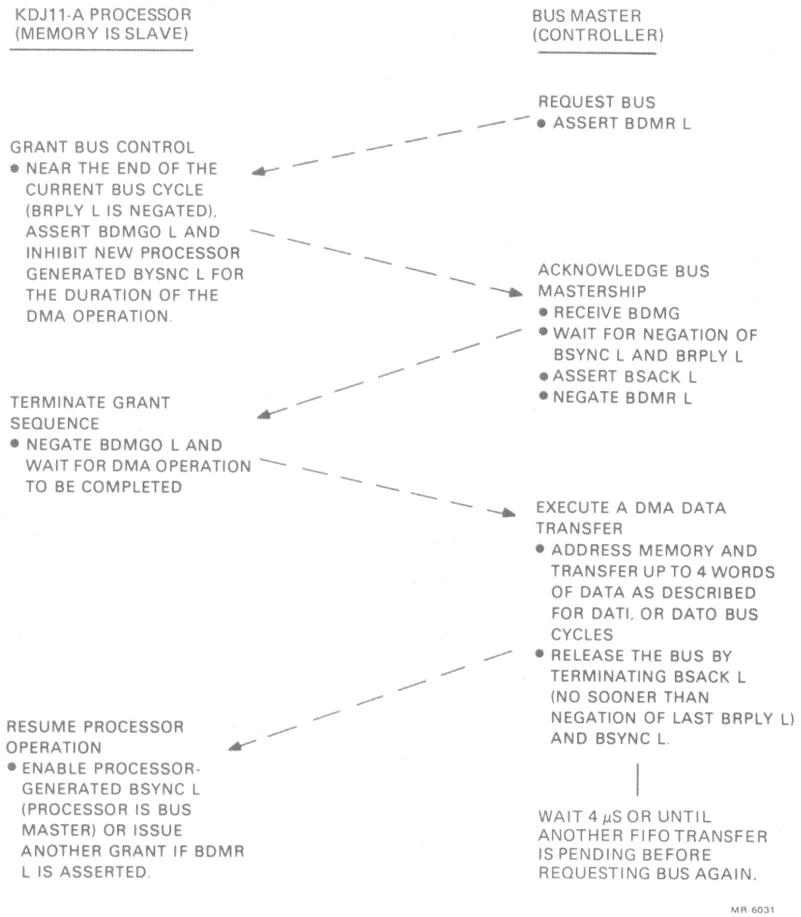


Figure A-7 DMA Protocol

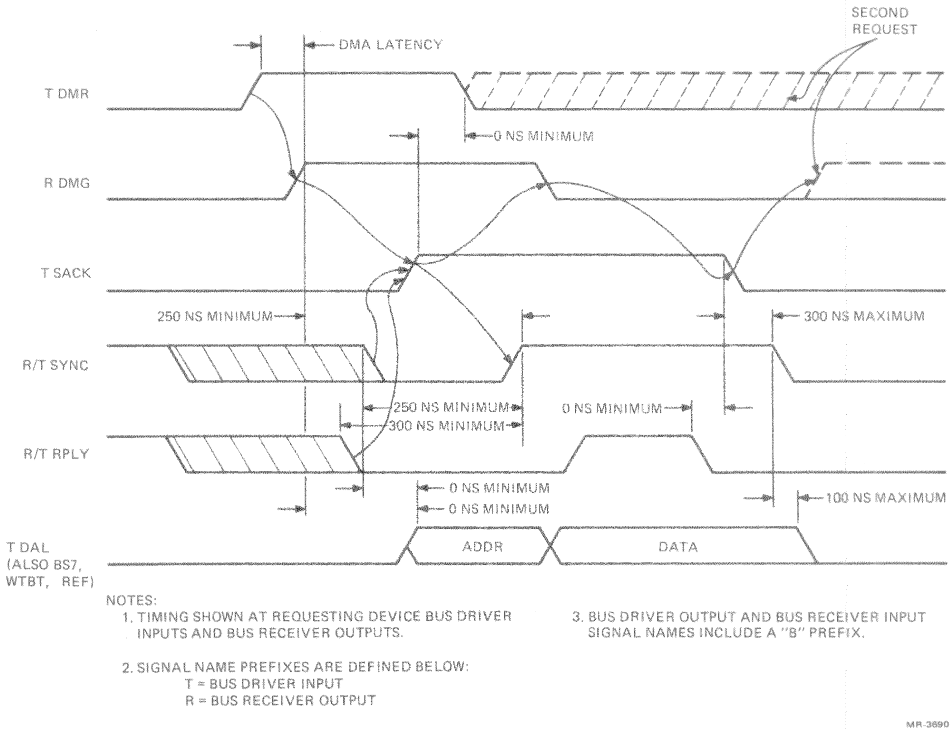


Figure A-8 DMA Request/Grant Timing

A.4.2 Block Mode DMA

For increased throughput, block mode DMA may be implemented on a device for use with memories that support this type of transfer. In a block mode transaction, the starting memory address is asserted, followed by data for that address, and data for consecutive addresses.

By eliminating the assertion of the address for each data word, the transfer rate is almost doubled. The DATBI and DATBO bus cycles are described below.

A.4.2.1 DATBI -- The device addressing portion of the cycle is the same as described earlier for other bus cycles. (See Figure A-9.) The bus master gates BDAL<21:00>, BBS7, and the negation of BWTBT onto the bus.

The master asserts the first BDIN 100 ns after BSYNC, and asserts BBS7 a maximum of 50 ns after asserting BDIN for the first time. BBS7 is a request to the slave for a block mode transfer. BBS7 remains asserted until a maximum of 50 ns after the assertion of BDIN for the last time. BBS7 may be gated as soon as the conditions for asserting BDIN are met.

The slave asserts BRPLY a minimum of 0 ns (8 ns maximum to avoid bus timeout) after receiving BDIN. It asserts BREF concurrently with BRPLY if it is a block mode device capable of supporting another BDIN after the current one. The slave gates BDAL<15:00> onto the bus 0 ns (minimum) after the assertion of BDIN, and 125 ns (maximum) after the assertion of BRPLY.

The master receives the stable data from 200 ns (maximum) after the assertion of BRPLY until 20 ns (minimum) after the negation of BDIN. It negates BDIN 200 ns (minimum) after the assertion of BRPLY.

The slave negates BRPLY 0 ns (minimum) after the negation of BDIN. If BBS7 and BREF are both asserted when BRPLY is negated, the slave prepares for another BDIN cycle. BBS7 is stable from 125 ns after BDIN is asserted until 150 ns after BRPLY is negated. The master asserts BDIN 150 ns (minimum) after BRPLY is negated, and the cycle is continued as before. (BBS7 remains asserted and the slave responds to BDIN with BRPLY and BREF.) BREF is stable from 75 ns after BRPLY is asserted until 20 ns (minimum) after BDIN is negated.

If BBS7 and BREF are not both asserted when BRPLY is negated, the slave removes the data from the bus 0 ns (minimum) and 100 ns (maximum) after negating BRPLY. The master negates BSYNC 250 ns (minimum) after the assertion of the last BRPLY, and 0 ns (minimum) after the negation of that BRPLY.

A.4.2.2 DATBO -- The device addressing portion of the cycle is the same as shown in Figure A-10. The bus master gates BDAL<21:00>, BBS7, and the assertion of BWTBT onto the bus.

A minimum of 100 ns after BSYNC is asserted, data on BDAL<15:00> and the negated BWTBT are put onto the bus. The master then asserts BDOUT a minimum of 100 ns after gating the data.

The slave receives stable data and BWTBT from 25 ns (minimum) before the assertion of BDOUT to 25 ns (minimum) after the negation of BDOUT. The slave asserts BRPLY 0 ns (minimum) after receiving BDOUT. It also asserts BREF concurrently with BRPLY if it is a block mode device capable of supporting another BDOUT after the current one.

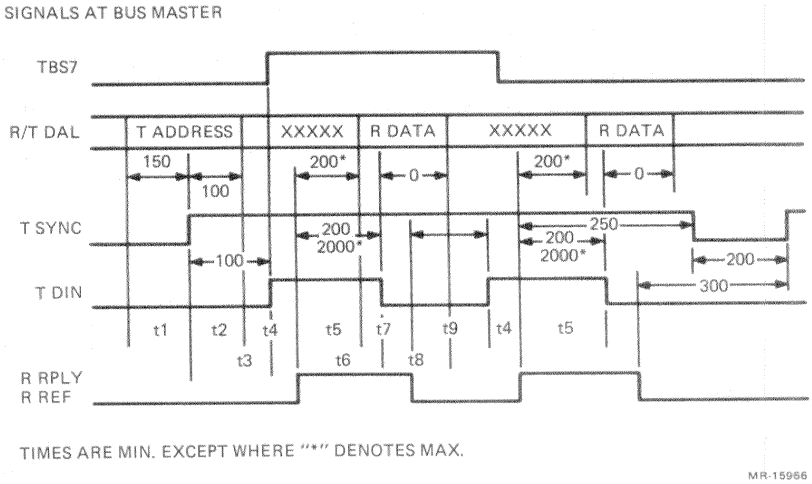


Figure A-9 DATBI Bus Cycle Timing

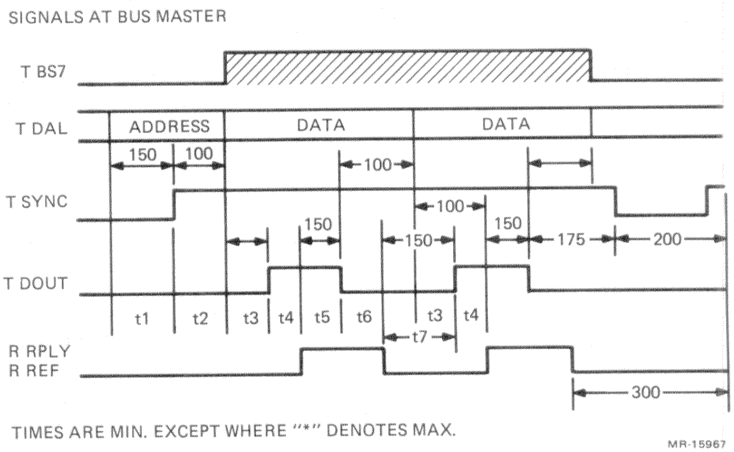


Figure A-10 DATBO Bus Cycle Timing

The master negates BDOUT 150 ns (minimum) after the assertion of BRPLY. If BREF was asserted when BDOUT was negated, and the master wants to transmit more data in this block mode cycle, the new data is gated onto the bus 100 ns (minimum) after BDOUT is negated. BREF is stable from 75 ns (maximum) after BRPLY is asserted until 20 ns (minimum) after BDOUT is negated. The master asserts BDOUT 100 ns (minimum) after gating new data onto the bus and 150 ns minimum after BRPLY negates. The cycle continues as before.

If BREF was not asserted when BDOUT was negated, or if the bus master does not want to transmit more data in this cycle, the master removes data from the bus 100 ns (minimum) after negating BDOUT. The slave negates BRPLY 0 ns (minimum) after negating BDOUT. The bus master negates BSYNC 175 ns (minimum) after negating BDOUT, and 0 ns (minimum) after the negation of BRPLY.

A.4.3 DMA Guidelines

1. Systems with memory refresh over the bus must not include devices that perform more than one transfer per acquisition.

2. Bus masters that do not use block mode are limited to four DATI, four DATO, or two DATIO transfers per acquisition.

3. Block mode bus masters that do not monitor BDMR are limited to eight transfers per acquisition.

4. If BDMR is not asserted after the seventh transfer, block mode bus masters that do monitor BDMR may continue making transfers until the bus slave fails to assert BREF, or until they reach the total maximum of 16 transfers. Otherwise, they stop after eight transfers.

A.5 INTERRUPTS

The interrupt capability of the Q22-Bus allows an I/O device to temporarily suspend (interrupt) current program execution and divert processor operation to service the requesting device. The processor inputs a vector from the device to start the service routine (handler). Like the device register address, hardware fixes the device vector at locations within a designated range below location 001000. The vector indicates the first of a pair of addresses. The processor reads the contents of the first address, the starting address of the interrupt handler. The contents of the second address is a new processor status word (PS).

The new PS can raise the interrupt priority level, thereby preventing lower-level interrupts from breaking into the current interrupt service routine. Control is returned to the interrupted program when the interrupt handler is ended. The original interrupted program's address (PC) and its associated PS are stored on a stack. The original PC and PS are restored by a return from interrupt (RTI or RTT) instruction at the end of the handler. The use of the stack and the Q22-Bus interrupt scheme can allow interrupts to occur within interrupts (nested interrupts), depending on the PS.

Interrupts can be caused by Q22-Bus options or the MicroVAX CPU. Those interrupts that originate from within the processor are called "traps". Traps are caused by programming errors, hardware errors, special instructions, and maintenance features.

The following are Q22-Bus signals used in interrupt transactions.

BIRQ4 L	Interrupt request priority level 4
BIRQ5 L	Interrupt request priority level 5
BIRQ6 L	Interrupt request priority level 6
BIRQ7 L	Interrupt request priority level 7
BIAKI L	Interrupt acknowledge input
BIAKO L	Interrupt acknowledge output
BDAL<21:00>	Data/address lines
BDIN L	Data input strobe
BRPLY L	Reply

A.5.1 Device Priority

The Q22-Bus supports the following two methods of device priority.

1. Distributed Arbitration -- Priority levels are implemented on the hardware. When devices of equal priority level request an interrupt, priority is given to the device electrically closest to the processor.

2. Position-Defined Arbitration -- Priority is determined solely by electrical position on the bus. The closer a device is to the processor, the higher its priority is.

A.5.2 Interrupt Protocol

Interrupt protocol on the Q22-Bus has three phases: the interrupt request phase, interrupt acknowledge and priority arbitration phase, and interrupt vector transfer phase. Figure A-11 shows the interrupt request/acknowledge sequence.

The interrupt request phase begins when a device meets its specific conditions for interrupt requests. For example, the device is ready, done, or an error has occurred. The interrupt enable bit in a device status register must be set. The device then initiates the interrupt by asserting the interrupt request line(s). BIRQ4 L is the lowest hardware priority level and is asserted for all interrupt requests for compatibility with previous Q22 processors. The level at which a device is configured must also be asserted. A special case exists for level 7 devices that must also assert level 6. For an explanation, refer to the discussion below on arbitration involving the 4-level scheme.

Interrupt Level	Lines Asserted by Device
4	BIRQ4 L
5	BIRQ4 L, BIRQ5 L
6	BIRQ4 L, BIRQ6 L
7	BIRQ4 L, BIRQ6 L, BIRQ7 L

Figure A-11 Interrupt Request/Acknowledge Sequence