Firefox Workstation M-Bus Specification

Revision 2.1

Michael Nielsen (DECWSE::NIELSEN)

Workstation Systems Engineering
Digital Equipment Corporation
100 Hamilton Avenue
Palo Alto, CA 94301
415-853-6779

December 29, 1987

RESTRICTED DISTRIBUTION

Blank Page

Table of Contents

Revision History

Date	Version	Changes
26 Dec 87	2.1	Added READU transaction Added MRUN signal Added M-bus state diagram Added slave MBRQ assertion during MWP6 Revised M-bus timing
30 Apr 87	2.0	Arbitration priority changed to LRU Support two simultaneous interlocks MSHARED,MDATINV,MBUSY,MHALT added Local I/O space added MACKOK renamed to MPOK More reserved signals added -12 volt etch added (for future use)
27 Jan 87	1.1	MACOK added; MDCOK updated
10 Jan 87	1.0	First external release
6 Nov 86	0.0	Preliminary draft

Blank Page

4. Firefox M-Bus Specification

This is the design specification of the Firefox M-bus. The M-bus is a synchronous memory interconnect between Firefox modules. The M-bus protocol allows the processor snoopy caches to maintain consistent data in all caches on a cycle-by-cycle basis.

The M-bus supports a maximum of eight modules that arbitrate for the M-bus via a least-recently-used-priority, distributed-resolution scheme. The M-bus can transfer up to 32 bits of address or data in a single M-bus cycle. M-bus memory-space transactions always transfer 4 longwords of data between caches and memory. Memory-space reads and cache victim writes are unmasked transfers. Cache write-throughs are masked transfers. M-bus I/O-space transactions always transfer one masked longword of data between processors and I/O devices. M-bus interrupt-acknowledge transactions transfer an interrupt vector between a processor and I/O device. M-bus transactions nominally complete in 4 to 10 cycles, depending on the number of data longwords transferred. Slave devices may insert additional wait cycles before completing M-bus transactions.

The target M-bus cycle time is 70 ns. The internal module logic need not be synchronous to the M-bus clock. Nevertheless, modules that participate in memory space must meet the timing for indicating shared status. Consequently, some processor modules may require a longer minimum M-bus cycle time. M-bus modules are not required to support M-bus cycle times in excess of 100 ns.

The M-bus time-multiplexes and encodes its control signals to minimize signal count and power consumption. Even though signals are time-multiplexed, the protocol design is such that different modules never drive the same signal on consecutive M-bus cycles, which eliminates problems with overlapping in the backplane tristate driver.

The M-bus supports single-bit error detection on its command and data signals with parity. The M-bus supports detection of single-bit errors on its M-bus arbitration signals with distributed protocol checking. The M-bus does not support hardware error correction.

The following sections describe the M-bus terminology, M-bus operation, M-bus addressing, M-bus signals, M-bus transaction types and sequences, M-bus example transactions, M-bus interface registers, and M-bus electrical specifications.

In all discussions of M-bus signals, values will be described as asserted or deasserted. This refers to their logical value, independent of their physical active-high or active-low signal levels. In all figures, M-bus signals will be shown with a high level for asserted, and a low level for deasserted. All addresses are in hexadecimal.

4.1. Terminology

The following terms describe the operations of the M-bus:

Cycle	One cycle is the period of time between rising edges on the bus A clock. During a given cycle, the M-bus may be idle, arbitrating, transferring an address or data between modules, or waiting for a module to respond to a request.
Transaction	A transaction is the sequence of bus cycles that accomplish a logical operation. An example would be, reading four longwords of data from memory. Transactions are atomic sequences of bus cycles; there are no pending transactions.
Master	The M-bus module that arbitrates for the M-bus and initiates a bus transaction is the master. For read class transactions the master specifies an address and waits for a slave to supply read data. For write class transactions the master specifies an address and supplies write data to a slave.
Slave	The M-bus module that monitors bus transactions and responds to a request initiated by a master is the slave. Some M-bus transaction requests may be completed by more than one slave, in which case the slaves arbitrate for the right to complete the transaction.
Longword	A longword is 32 bits of data, which can be transferred between modules in a single bus cycle.
Octaword	An octaword is 128 bits of data, which can be transferred between modules in four sequential bus cycles.
Line	A line is one entry of a cache. Firefox cache lines are octaword in size.
Victim	A victim is the cache entry that will be removed to make room for a new cache entry.
Shared	A cache line is marked as shared when the same octaword address may be present in more than one cache.
Dirty	A cache line is marked as dirty when it has been modified in the cache since it was read from memory.
Masked	If a data transfer is masked, then only some of the bytes in a longword should be read/written.
Unmasked	If a data transfer is unmasked, then all of the bytes in a longword must be read/written.
Undefined	For masked transfers, the value of nonrequested bytes in the longword is undefined; that is, the value may not correspond to the transaction address. However, the data-bus signals are still driven with some specific value, and this value is used in any parity calculations.

4.2. Operation

The M-bus supports communication between modules for memory-space operations, I/O-space operations, and interrupt operations.

Memory-space references are always cached and only generate M-bus transactions to support:

- Read-miss fills
- Write-miss fills
- Dirty-victim writes
- Shared-cache-line write-throughs
- Interlocked reads
- Unlock writes

When a cache reference misses, the cache entry is reallocated and filled with a M-bus memory read. This applies to either a cache read or write. If the victim entry in the cache line targetted for reallocation is dirty, a M-bus memory write flushes the data out to memory before the reallocation and a M-bus memory read occurs. After cache writes to lines that are shared, the cache generates a M-bus memory write-through to update the other caches and memory. Cache writes to unshared cache lines do not generate M-bus transactions.

Whenever a M-bus memory read or write-through occurs, all caches probe their tag store to determine whether or not they contain the specified octaword. If a cache contains the octaword referenced by a memory read and the octaword is dirty, the cache supplies the read data in place of a memory module. This ensures that the data from dirty cache entries is used rather than stale memory data. All caches that contain the octaword referenced by a memory write-through update their data store with the supplied write data. This ensures that shared cache lines remain consistent. After every M-bus memory read or write-through, caches that contain the octaword update the shared bit of their tag store that indicates whether or not the line is in more than one cache.

The M-bus supports two simultaneous interlocked transactions to different hexaword addresses. Internal interlocked reads and unlock writes always generate M-bus reads and writes. Unlock writes are functionally equivalent to write-through transactions. Every M-bus interface contains a two-entry content-addressable-memory that records addresses that are currently locked. This algorithm allows all M-bus interfaces in the workstation to stall conflicting interlocked reads from their internal logic until the current interlocked transaction is completed. Stalled interlocked reads do not generate M-bus traffic until the current interlocked transaction is completed. Noninterlocked transactions proceed regardless of whether or not an interlocked transaction is in progress.

M-bus memory-space read transactions to nonexistent memory modules abort the M-bus cycle after a memory module normally responds.

Global I/O-space references are never cached and always generate M-bus transactions. M-bus I/O-space references to non-existent modules abort the M-bus cycle after an I/O module normally responds. M-bus I/O-space references to address-space *holes* in I/O modules abort after a timeout of tens of microseconds. M-bus I/O space references may terminate with an indication that the reference should be retried later to avoid deadlocks between the M-bus and busses accessed through adapters.

Global interrupt-acknowledge references always generate M-bus transactions. Since multiple modules may service the same interrupt level, interrupt-acknowledge races are resolved by passive-release termination to all but the first module to issue an M-bus interrupt acknowledge. *Passive release* means that the interrupt is not seen by software.

Table 4-1 lists the peak bandwidth of the M-bus for the various transaction types in Mbytes/second.

Table 4-1: M-Bus Peak Transaction Bandwidth

Transaction	Minimum Cycles	Bytes	BW @ 70 ns	BW @ 80 ns	BW @ 90 ns	BW @ 100 ns
Memory read	10	16	22.9	20.0	17.8	16.0
Memory write	6	16	38.1	33.3	29.6	26.7
I/O read	4	4	14.3	12.5	11.1	10.0
I/O write	4	4	14.3	12.5	11.1	10.0

4.3. Addressing

The M-bus supports a 2-Gbyte memory address space and a separate 2-Gbyte I/O address space. Each module is assigned a 32-Mbyte region of I/O space as a function of its backplane slot. Table 4-1 lists the address-space assignments.

Table 4-2: Address-Space Assignments for the M-Bus Module

M-Bus Address Range	VAX Address Range	Mbytes	Function
00000000..1FFFFFFF	00000000..1FFFFFFF	512	Memory space
20000000..7FFFFFFF		1536	Reserved memory space
80000000..87FFFFFF	20000000..27FFFFFF	128	Global I/O space
88000000..8FFFFFFF	28000000..2FFFFFFF	128	Local I/O space
90000000..91FFFFFF	30000000..31FFFFFF	32	Slot 0 I/O space
92000000..93FFFFFF	32000000..33FFFFFF	32	Slot 1 I/O space
94000000..95FFFFFF	34000000..35FFFFFF	32	Slot 2 I/O space
96000000..97FFFFFF	36000000..37FFFFFF	32	Slot 3 I/O space
98000000..99FFFFFF	38000000..39FFFFFF	32	Slot 4 I/O space
9A000000..9BFFFFFF	3A000000..3BFFFFFF	32	Slot 5 I/O space
9C000000..9DFFFFFF	3C000000..3DFFFFFF	32	Slot 6 I/O space
9E000000..9FFFFFFF	3E000000..3FFFFFFF	32	Slot 7 I/O space
A0000000..FFFFFFFF		1536	Reserved I/O space

Memory modules and processor caches jointly maintain the memory-space region. There are no preconceived ideas about memory-space assignments as a function of backplane slot. Memory modules have programmable memory-space base addresses via a register in their I/O-space assignment. Other modules that might reside in memory space (graphics modules, for example) should have similar functionality. Programmable base addresses need only resolve to 1-Mbyte boundaries. This allows the address range of multiple memory modules to form a single, contiguous region of memory starting at address 00000000.

Current VAX processors cannot access the reserved memory space. Processors that generate 32-bit physical addresses can access the full 2 Gbytes of memory space.

The global I/O space is defined by the implementation; that is, some VLSI I/O devices have hardwired base addresses that fall in this region. I/O modules that require more than the 32 Mbytes associated with their backplane slot can map some of their resources to the global I/O space.

The local I/O space is also defined by the implementation. It is for use by modules that have strictly local resources in I/O space. For example, processor modules could implement special purpose coprocessors in local I/O space so that the coprocessor appears at the same physical address for each processor.

The slot-specific I/O space should contain all of the I/O resources associated with a module. There is one M-bus interface register required of all modules that serves to identify the module class and M-bus interface chip. This register is mapped to the top of the slot-specific I/O-space region. The remainder of the slot-specific I/O-space region is dependent on the implementation. A M-bus module must not respond to the slot-specific region for another backplane slot.

Current VAX processors cannot access the reserved I/O space. Processors that generate 32-bit physical addresses can access the full 2 Gbytes of I/O space.

Current VAX processors only generate 30-bit physical addresses. Table 4-2 lists the connection of their address signals to MDAL signals for cycle P2 of M-bus transactions. For all other M-bus cycles the VAX DAL<31:00> is directly connected to MDAL<31:00>.

Table 4-3: VAX 30-Bit Physical Address to M-Bus 32-Bit Physical Address Mapping

M-Bus Address	VAX Address
MDAL<31>	DAL<29>
MDAL<30>	0
MDAL<29>	0
MDAL<28:00>	DAL<28:00>

4.4. Signals

The M-bus consists of four groups of signals that implement M-bus arbitration, data transfer, workstation control, and clock distribution. Table 4-4 lists the M-bus signals and their functions. The *asserted* column indicates the active assertion state on the backplane.

Table 4-4: Summary of M-Bus Signals

Signal	Count	Type	Asserted	Function
MBRQ	8	TTL	Low	Bus requests
MBUSY	1	OC	Low	Module busy
MCMD	4	TRI	High	Bus cycle command
MSTAT	2	TRI	High	Bus cycle status
MDAL	32	TRI	High	Data and address
MXPAR	3	TRI	High	Parity
MSHARED	1	OC	Low	Shared line
MDATINV	1	OC	Low	Data invalid
MID	3	TTL	High	Module ID
MRESET	1	OC	Low	Workstation reset
MABORT	1	OC	Low	Transaction abort
MIRQ	4	OC	Low	Interrupt requests
MHALT	1	OC	Low	Halt processors
MPOK	1	OC	High	AC power OK
MDCOK	1	OC	High	DC power OK
MRUN	1	OC	Low	System running
MCLKA	1	TTL		Bus clock-A phase
MCLKB	1	TTL		Bus clock-B phase
MCLKI	1	OC		Interval clock
Total	68			

Table 4-5 shows the cycles composing a single minimum-length, read-class, M-bus transaction. The *module* column indicates the module driving the MDAL signals. The wait cycles (P3 through P5) are used for cache probes in processor modules and memory-array access in memory modules. During cycle P6, if a cache *hits* on the address it asserts the MSHARED signal. If a cache hits and the line is dirty, the cache also asserts its MBRQ signal and supplies the read data in place of a memory module. If no cache hits with a dirty line, then a memory module provides the read data. The slave module may insert wait cycles starting with P7.

Table 4-5: Cycles in a M-Bus Read Transaction

Cycle	Module	Action
P1	Master	Bus arbitration
P2	Master	Address
P3		Wait
P4		Wait
P5		Wait
P6		Shared status
P7	Slave	Read data
P8	Slave	Read data
P9	Slave	Read data
P10	Slave	Read data

Table 4-6 shows the cycles composing a single minimum-length, write-class, M-bus transaction. The *module* column indicates the module driving the MDAL signals. During cycles P3 through P6, processor caches that hit on the address, together with the selected memory module, write the data. Any caches that hit indicate this by asserting their MSHARED signal during cycle P6.

Table 4-6: Cycles in a M-Bus Write Transaction

Cycle	Module	Action
P1	Master	Bus arbitration
P2	Master	Address
P3	Master	Write data
P4	Master	Write data
P5	Master	Write data
P6	Master	Write data, shared status

The M-bus cycle boundaries, Pn, are defined by the rising edge of MCLKA. Unless otherwise noted in the following sections, all transitions of M-bus signals are synchronous with respect to MCLKA.

4.4.1. M-Bus Arbitration

The MBRQ signals perform arbitration for the M-bus among the modules within a Firefox workstation. M-bus arbitration serves three functions: determination of the next M-bus master, determination of the M-bus slave for some types of transaction, and indication of the module driving the M-bus. M-bus arbitration employs a least-recently-used-priority, distributed-resolution algorithm.

4.4.1.1. MBRQ

Each module drives exactly one of the eight MBRQ signals and monitors the MBRQ signals from the other 7 backplane slots. The MBRQ signal for a given slot corresponds to the backplane slot number. For example, slot 0 drives MBRQ<0> and monitors MBRQ<1:7>, and slot 4 drives MBRQ<4> and monitors MBRQ<0:3,5:7>.

Each module asserts its MBRQ signal on a standard connector pin during an idle M-bus cycle when it needs to acquire the M-bus. Otherwise, a module deasserts its MBRQ signal. A module may not assert its MBRQ signal during arbitrary M-bus cycles; it may do so only during idle M-bus cycles and when it is driving the M-bus. After winning the arbitration cycle, the M-bus master continues to assert its MBRQ signal during P2 and P3 of all transactions, and during P4 and P5 of memory-space transactions. During the course of some M-bus transactions, multiple potential M-bus slaves arbitrate for the slave role. These potential slaves use the MBRQ signals to arbitrate. After winning the slave-arbitration cycle or being selected by the transaction address, the slave asserts its MBRQ signal during P4 of I/O-space transactions, during P4 and P5 of interrupt-acknowledge transactions, during P6 of memory-write transactions, and during P7, P8, and P9 of memory-read transactions.

Each module monitors the state of the remaining seven MBRQ signals to independently resolve a M-bus arbitration. The seven other MBRQ signals are connected to a fixed set of backplane connector pins in a slot-dependent fashion. These pins are called MBRM<0:6> to identify them as the slot-dependent wiring of the MBRQ signals.

Table 4-7 shows the backplane permutation of the MBRQ signals for each slot. During a P1 arbitration cycle, a module *loses* if any MBRM signal at a higher priority is asserted. The decision as to whether or not a particular MBRM signal is at a higher priority for this particular cycle is implemented as a MBRM mask maintained by each module. If the mask bit for a given MBRM signal is set, that module is at a higher priority. If the mask bit for a given MBRM signal is clear, that module is at a lower priority. The MBRM mask is only updated after P1 M-bus arbitration; slave arbitration does not update the MBRM mask.

Table 4-7: MBRQ Wiring to MBRM Pins for Each Backplane Slot

Slot	MBRM<0>	MBRM<1>	MBRM<2>	MBRM<3>	MBRM<4>	MBRM<5>	MBRM<6>
0	MBRQ<1>	MBRQ<2>	MBRQ<3>	MBRQ<4>	MBRQ<5>	MBRQ<6>	MBRQ<7>
1	MBRQ<0>	MBRQ<2>	MBRQ<3>	MBRQ<4>	MBRQ<5>	MBRQ<6>	MBRQ<7>
2	MBRQ<0>	MBRQ<1>	MBRQ<3>	MBRQ<4>	MBRQ<5>	MBRQ<6>	MBRQ<7>
3	MBRQ<0>	MBRQ<1>	MBRQ<2>	MBRQ<4>	MBRQ<5>	MBRQ<6>	MBRQ<7>
4	MBRQ<0>	MBRQ<1>	MBRQ<2>	MBRQ<3>	MBRQ<5>	MBRQ<6>	MBRQ<7>
5	MBRQ<0>	MBRQ<1>	MBRQ<2>	MBRQ<3>	MBRQ<4>	MBRQ<6>	MBRQ<7>
6	MBRQ<0>	MBRQ<1>	MBRQ<2>	MBRQ<3>	MBRQ<4>	MBRQ<5>	MBRQ<7>
7	MBRQ<0>	MBRQ<1>	MBRQ<2>	MBRQ<3>	MBRQ<4>	MBRQ<5>	MBRQ<6>

The MBRM mask is initialized after MRESET or MABORT from a decoded value of the MID signals that makes slot 0 the highest priority and slot 7 the lowest priority. Table 4-8 shows the initial MBRM-mask values for each module. For example, the module in slot 2 initializes its MBRM<0:6> mask to 1100000#2 so that slots 0 and 1 are initially at higher priority, and slots 2 though 6 are initially at lower priority.

Table 4-8: Initialization Values for MBRM Masks

Slot	MBRM<0>	MBRM<1>	MBRM<2>	MBRM<3>	MBRM<4>	MBRM<5>	MBRM<6>
0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
2	1	1	0	0	0	0	0
3	1	1	1	0	0	0	0
4	1	1	1	1	0	0	0
5	1	1	1	1	1	0	0
6	1	1	1	1	1	1	0
7	1	1	1	1	1	1	1

Whenever a module wins a M-bus arbitration, it sets all its MBRM mask bits; that is, the winner views all other modules as higher priority for the next M-bus arbitration. Whenever a M-bus-arbitration cycle completes, all modules clear the MBRM mask bit of the winner; that is, the winner is viewed as lowest priority for the next M-bus arbitration. Table 4-9 shows the change in the MBMR mask from the initial value after the module in slot 2 wins a P1 arbitration cycle.

Table 4-9: Initialization Values MBRM Masks

Slot	MBRM<0>	MBRM<1>	MBRM<2>	MBRM<3>	MBRM<4>	MBRM<5>	MBRM<6>
0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
2	1	1	1	1	1	1	1
3	1	1	0	0	0	0	0
4	1	1	0	1	0	0	0
5	1	1	0	1	1	0	0
6	1	1	0	1	1	1	0
7	1	1	0	1	1	1	1

When a module loses an arbitration cycle to a higher-priority module, it must deassert its MBRQ signal until the end of the current M-bus transaction. The module that wins an arbitration cycle continues to assert its M-bus request until the M-bus slave takes over the M-bus. Whenever a slave module is driving the MSTAT and MDAL signals, it must assert its MBRQ signal.

When a module loses an arbitration cycle, it must still monitor the M-bus transaction to maintain synchronization with the other modules.

During idle M-bus cycles, all MBRM signals are monitored to determine the start of M-bus transactions initiated by other modules.

Each module should implement M-bus-monitoring logic that detects assertion of other MBRQ signals when it is driving the M-bus. If multiple modules erroneously believe they won a M-bus arbitration, they will observe each other's MBRQ signal as asserted and signal a M-bus error. If multiple slaves erroneously respond to a M-bus transaction, they will observe each other's MBRQ signal as asserted and signal a M-bus error. This serves as a form of single-bit error detection on the MBRQ signals.

Table 4-10 lists cycles in which exactly one M-bus master, *M*, or exactly one M-bus slave, *S*, should be asserting its MBRQ signal.

Table 4-10: M-bus Cycles Requiring MBRQ Checking

Transaction	P1	P2	P3	P4	P5	P6	P7	P8	P9
Memory Read		M	M	M	M		S	S	S
Memory Write		M	M	M	M				
I/O Read		M	M	S					
I/O Write		M	M	S					
Interrupt Ack.		M	M		S				

Modules must also assert their MBRQ signal when MRESET is asserted. This allows M-bus-monitoring logic to determine which backplane slots contain modules, even if the modules have hardware failures that prevent them from responding as M-bus slaves during M-bus transactions. Because of timing restrictions, assertion of MBRQ during reset must be pipelined; that is, MRESET is ORed into the next cycle state for MBRQ. This implies that the MBRQ signals will still be asserted during the first cycle that MRESET becomes deasserted. Consequently, M-bus-interface state machines must not interpret MBRQ assertion during this cycle as M-bus arbitration. This may be implemented by using a one-cycle-delayed copy of MRESET to reset internal logic.

Figure 4-1 shows a workstation with modules in backplane slots 0, 4, and 5. In response to the assertion of MRESET, all modules reset their internal logic and their M-bus-interface logic asserts their MBRQ signals. Backplane termination resistors maintain a deasserted level on the slot 1, 2, 3, 6, and 7 MBRQ signals. During cycle P4, a M-bus-interface register may latch and save the value of MBRQ<0:7> for use by software. Cycle P5 is the first M-bus cycle that a M-bus transaction can start.

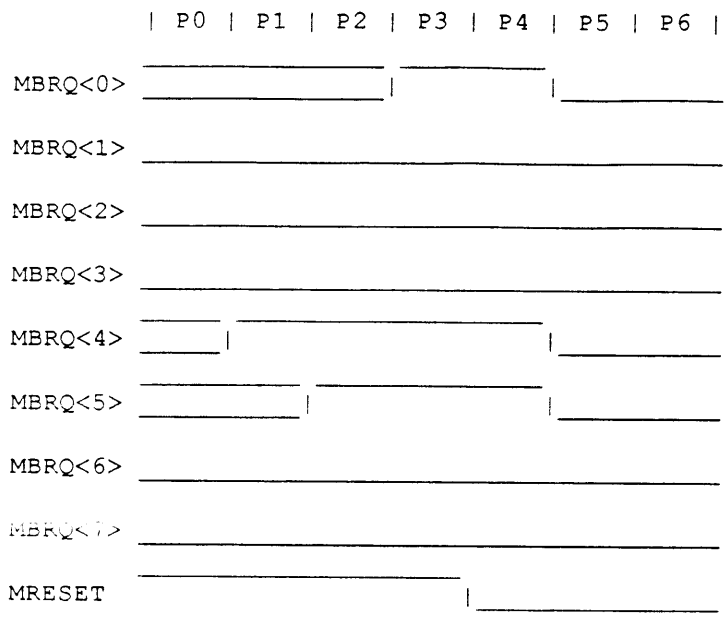


Figure 4-1: MBRQ Assertion During MRESET

It is recommended that M-bus-interface logic monitor its own MBRQ backplane signal and verify that it is driven with the correct state.

4.4.1.2. MBUSY

Commencement of a new M-bus transaction can be stalled by asserting the MBUSY signal. Assertion of MBUSY during a transaction has no effect on the current transaction. Assertion of MBUSY suppresses transition of the M-bus state from P1 to P2. The primary use of MBUSY is to stall commencement of a new transaction until a memory-write transaction completes in all slaves. This is necessary because memory writes are broadcast transactions and may complete in a different number of M-bus cycles in the slaves.

4.4.2. Data Transfer

Data transfer between two or more modules is accomplished via the MDAL signals under the control of the MCMD and MSTAT signals. The MXPAR signals enable single-bit error detection of the MCMD, MSTAT, and MDAL signals.

The MDAL signals are driven by M-bus masters to specify addresses, write data, and interrupt-acknowledge levels. The MDAL signals are driven by M-bus slaves to specify read data and interrupt vectors. The MCMD signals are driven by M-bus masters to specify the transaction type and I/O-space byte masks. The MSTAT signals are driven by M-bus slaves to indicate the status of the current transaction.

4.4.2.1. MCMD

When a M-bus master is driving the MDAL signals, the MCMD signals indicate the contents of the MDAL signals. There are three uses of the MCMD signals: transaction type, memory write-through byte mask, and I/O-read/write byte mask. Table 4-11 lists the interpretation of MCMD during various transaction cycles. All transactions are shown without slave wait cycles.

Table 4-11: Interpretation of MCMD During Transaction Cycles

Transaction	Cycle	Interpretation
All transactions	P1	<Not driven>
All transactions	P2	Transaction type
Memory read	P3	<Not driven>
Memory read	P4	<Not driven>
Memory read	P5	<Not driven>
Memory read	P6	<Not driven>
Memory read	P7	<Not driven>
Memory read	P8	<Not driven>
Memory read	P9	<Not driven>
Memory read	P10	<Not driven>
Memory write	P3	Byte mask
Memory write	P4	Byte mask
Memory write	P5	Byte mask
Memory write	P6	Byte mask
I/O read	P3	Byte mask
I/O read	P4	<Not driven>
I/O write	P3	Byte mask
I/O write	P4	<Not driven>
Interrupt acknowledge	P3	<Not driven>
Interrupt acknowledge	P4	<Not driven>
Interrupt acknowledge	P5	<Not driven>

Table 4-12 lists the encoding of the MCMD field during cycle P2 of all M-bus transactions. A memory-space versus an I/O-space transaction is specified by MDAL<31> of a read or write address; MDAL<31> is 0 for memory space and 1 for I/O space. M-bus masters must drive MDAL<31> to 1 for interrupt-acknowledge transactions.

Table 4-12: MCMD Encoding During Cycle P2 of Transactions

Value	Mnemonic	Function
0000	RESERVED	
0001	RESERVED	
0010	RESERVED	
0011	RESERVED	
0100	RESERVED	
0101	READ	Request read
0110	RESERVED	
0111	WRITET	Request write-through
1000	RESERVED	
1001	READI	Request read interlocked
1010	WRITE	Request victim or I/O write
1011	WRITEU	Request write unlock
1100	RESERVED	
1101	INTACK	Request interrupt acknowledge
1110	READU	Request read unshared
1111	RESERVED	

When the M-bus master issues an I/O-space read or write, the MCMD signals specify the byte mask for the longword address. For both I/O reads and writes, the M-bus master supplies the byte mask during cycle P3 of a M-bus transaction. Table 4-13 lists the correspondence of mask bits to MDAL bytes. If MCMD<n> is asserted, then the corresponding byte of the longword is to be transferred. If MCMD<n> is deasserted, then the corresponding byte of the longword contains undefined data. When some bytes of MDAL are undefined because of byte masks, they still enter into the calculation of MDPAR.

Table 4-13: Correspondence Between Mask Bits and MDAL Bytes

Mask Bit	MDAL Byte
MCMD<3>	MDAL<31:24>
MCMD<2>	MDAL<23:16>
MCMD<1>	MDAL<15:08>
MCMD<0>	MDAL<07:00>

A M-bus master should only drive the MCMD signals during a cycle in which it is specifying a transaction type or byte mask. When a M-bus master is driving the MCMD signals, it must specify valid parity on MCPAR.

4.4.2.2. MSTAT

When a M-bus slave responds to a M-bus transaction, it asserts its MBRQ signal, specifies transaction status on the MSTAT signals, and supplies data on the MDAL signals for read-class transactions. Table 4-14 lists the interpretation of MSTAT during each cycle of the various M-bus transactions. All transactions are shown without slave wait cycles.

Table 4-14: Interpretation of MSTAT During Transaction Cycles

Transaction	Cycle	Interpretation
All transactions	P1	<Not driven>
All transactions	P2	<Not driven>
All transactions	P3	<Not driven>
Memory read	P4	<Not driven>
Memory read	P5	<Not driven>
Memory read	P6	<Not driven>
Memory read	P7	Data status
Memory read	P8	Data status
Memory read	P9	Data status
Memory read	P10	Data status
Memory write	P4	<Not driven>
Memory write	P5	<Not driven>
Memory write	P6	<Not driven>
I/O read	P4	Transaction status
I/O write	P4	Transaction status
Interrupt ack	P4	<Not driven>
Interrupt ack	P5	Transaction status

If no MBRQ signal is asserted during the *first* M-bus slave cycle of a transaction, then no slave responded and the transaction is immediately terminated. Table 4-15 lists the first M-bus slave cycle for each transaction type.

Table 4-15: First Slave Cycle of Transactions

Transaction	First Slave Cycle
Memory read	P7
I/O read	P4
I/O write	P4
Interrupt acknowledge	P4

When a M-bus slave does respond, it specifies cycle status on the MSTAT signals. Table 4-16 lists the MSTAT encodings.

Table 4-16: MSTAT Encoding

Value	Mnemonic	Function
00	WAIT	Stall transaction
01	GOOD	I/O write complete/good read data
10	CORRECTED/RETRY	Corrected memory-read-data/retry-I/O transaction
11	ERROR	Transaction error

If a M-bus slave requires additional cycles to complete the current transaction, it specifies WAIT status. A M-bus slave must not specify WAIT status after returning the first longword of a memory-space read. If WAIT is specified during P8, P9, or P10 of a memory read, the M-bus master should treat it as GOOD status.

A M-bus slave specifies GOOD status when it completes an I/O write, and also when it returns memory-read data, I/O-read data, or an interrupt vector.

A M-bus slave specifies CORRECTED status while it returns memory-read data that has a corrected single-bit error. A M-bus slave specifies RETRY status when I/O-space transactions must be retried or when a deadlock with another resource would result.

A M-bus slave specifies ERROR status for I/O-space transactions that reference non-existent resources. Non-existent-resource references should be minimized, as they take up to 256 M-bus cycles to time out.

When a M-bus slave is driving the MSTAT signals, it must specify valid parity on MSPAR.

4.4.2.3. MDAL

The MDAL signals transfer information between modules during M-bus transactions. Table 4-17 lists the interpretation of MDAL during the various M-bus transaction cycles. All transactions are shown without slave wait cycles. Modules may only drive MDAL when acting as a M-bus master or slave. Whenever a module is driving the MDAL signals, it must specify valid parity on MDPAR.

Table 4-17: Interpretation of MDAL During Transaction Cycles

Transaction	Cycle	Signals	Interpretation
All transactions	P1	<Not driven>	
Memory read	P2	MDAL<31:04>	Octaword address
Memory read	P3	<Not driven>	
Memory read	P4	<Not driven>	
Memory read	P5	<Not driven>	
Memory read	P6	<Not driven>	
Memory read	P7	MDAL<31:00>	Data
Memory read	P8	MDAL<31:00>	Data
Memory read	P9	MDAL<31:00>	Data
Memory read	P10	MDAL<31:00>	Data
Memory write	P2	MDAL<31:04>	Octaword address
Memory write	P3	MDAL<31:00>	Data
Memory write	P4	MDAL<31:00>	Data
Memory write	P5	MDAL<31:00>	Data
Memory write	P6	MDAL<31:00>	Data
I/O read	P2	MDAL<31:02>	Longword address
I/O read	P3	<Not driven>	
I/O read	P4	MDAL<31:00>	Data
I/O write	P2	MDAL<31:02>	Longword address
I/O write	P3	MDAL<31:00>	Data
I/O write	P4	<Not driven>	
Interrupt acknowledge	P2	MDAL<06:02>	Level
Interrupt acknowledge	P3	<Not driven>	
Interrupt acknowledge	P4	<Not driven>	
Interrupt acknowledge	P5	MDAL<15:00>	Vector

For read and write transactions, a memory-space versus an I/O-space transaction is specified by MDAL<31> of the address; MDAL<31> is 0 for memory space, or 1 for I/O-space or interrupt-acknowledge transactions.

4.4.2.4. MXPAR

The MCPAR signal specifies even parity for the MCMD signals. The MSPAR signal specifies even parity for the MSTAT signals. The MDPAR signal specifies even parity for the MDAL signals. Even parity means that there is an even number of 1s in a signal group and its corresponding parity bit.

Whenever a module drives any signal of MCMD, MSTAT, or MDAL, it must drive all signals of the group, plus the corresponding MXPAR signal.

Table 4-18 lists the M-bus cycles for which modules must check M-bus parity. The five types of M-bus transactions are shown, together with an indication of which cycle *C* , *S* , or *D* checking is required for the MCMD/MCPAR, MSTAT/MSPAR, and MDAL/MDPAR signals. All transactions are shown without slave wait cycles. MDAL parity errors must be ignored during slave wait cycles.

Figure 4-2: M-bus Cycles Requiring Parity Checking

Transaction	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Memory read		CD					SD	SD	SD	SD
Memory write		CD	CD	CD	CD	CD				
I/O read		CD	C	SD						
I/O write		CD	CD	S						
Interrupt acknowledge		CD			SD					

4.4.2.5. MSHARED

When memory-read, memory-read-unshared, memory-read-interlocked, memory-write-through, and memory-write-unlock transactions start on the M-bus, all caches probe their tag store to determine whether or not they contain the octaword. If a cache does contain the octaword, it asserts MSHARED during P6. Caches that contain the octaword update their tag-shared bit with the value of MSHARED during P7.

4.4.2.6. MDATINV

Whenever a module drives data onto MDAL that is known to contain a parity error, it asserts MDATINV. Local parity errors occur when cache data stores have parity errors, memory modules have uncorrectable ECC errors, or devices have hardware failures. When modules receive data with MDATINV asserted, either they must indicate an error to the transaction request and not use the data, or they must retain an indication that the data is invalid along with the data. For example, when caches receive invalid data during fill operations, they could intentionally write the data store with invalid parity. This prevents the undetected spread of invalid data.

4.4.3. Workstation Control

The MID, MRESET, MPOK, MDCOK, MRUN, MIRQ, MHALT, and MABORT signals initialize and coordinate the various modules in a Firefox workstation.

4.4.3.1. MID

The MID signals uniquely identify each M-bus backplane slot with a value from 0 to 7. The MID signals are connected to the backplane +5-volt and ground planes to achieve the value for a given slot. Table 4-18 lists the MID connections for each slot. The MID value is used in the M-bus-arbitration logic and I/O-space-address decoding to eliminate the need for switches or jumpers on M-bus modules. The MID value must be available to each module via a CSR.

Table 4-18: MID Slot Connections

Slot	MID<2>	MID<1>	MID<0>
0	GND	GND	GND
1	GND	GND	+5V
2	GND	+5V	GND
3	GND	+5V	+5V
4	+5V	GND	GND
5	+5V	GND	+5V
6	+5V	+5V	GND
7	+5V	+5V	+5V

Since the MID signals are tied directly to power planes, modules must provide a series resistor for each MID signal as appropriate for their interface logic to avoid device damage.

4.4.3.2. MRESET

The MRESET signal is asserted to initiate a workstation-wide reinitialization. While MRESET is asserted, all logic on M-bus modules must be set to a known state and the current M-bus transaction must be aborted. On the asserted to deasserted transition of MRESET, modules that perform self-testing or that require more extensive internal restart processing should commence their tests or processing. Such modules should provide a status bit accessible at all times via a CSR in their M-bus interface that other modules can use to ascertain whether or not their services are available for use.

The preceding paragraph does not imply that modules must perform self-testing when MRESET first becomes deasserted. Some modules may require software coordination or direction of self-testing. For example, if all memory modules perform self-testing in parallel, the power-distribution capacity of the workstation might be exceeded.

MRESET has a minimum assertion width of eight M-bus cycles. MRESET must be asserted for a minimum of 70 milliseconds after DC power is available on powerup. MRESET must be asserted whenever the MDCOK signal is deasserted. Transitions of the MPOK signal have no effect on MRESET.

Any module or any package-switch logic may be used to assert MRESET.

4.4.3.3. MPOK

The MPOK signal is asserted by the workstation power supplies when the AC line power is within specification. When MPOK becomes deasserted, modules should initiate power-failure actions. Deassertion of MPOK does not reset or abort M-bus transactions; it is a higher-level indication that a shutdown is imminent.

The transitions of MPOK are asynchronous with respect to the M-bus clocks.

Since Firefox workstations do not support battery backup of memory, the only activity expected, when MPOK transitions from asserted to deasserted, is completion of disk sector writes in progress. Modules must not stop responding as M-bus slaves when MPOK is deasserted.

4.4.3.4. MDCOK

The MDCOK signal is asserted by the workstation power supplies when they are supplying DC power within specification. When MDCOK becomes deasserted, modules may freeze internal resources as appropriate and stop responding to M-bus transactions.

The transitions of MDCOK are asynchronous with respect to the M-bus clocks.

4.4.3.5. MRUN

The MRUN signal may be asserted by a workstation module to indicate that the workstation is *running*. The MRUN signal is typically connected to a front-panel LED as an indication that the workstation has passed module self-test and is either booting or running the operating system. The MRUN signal may be driven by any module, though typically is only driven by the workstation I/O module.

The transitions of MRUN are asynchronous with respect to the M-bus clocks.

4.4.3.6. MIRQ

The MIRQ signals are asserted to indicate a pending interrupt. When internal logic on a module has a pending interrupt, it asserts its interrupt-request signal. The M-bus interface propagates this interrupt request onto one of the MIRQ signals. The M-bus interface of a module that is servicing interrupts propagates the MIRQ signal onto its local interrupt request.

M-bus-interface logic should provide a mechanism that both masks propagation of internal interrupt requests onto MIRQ signals and propagation of MIRQ signals onto internal interrupt requests. This scheme is based upon the assumption that a module either requests or services interrupts for a given level but never does both.

Transitions of the MIRQ signals are asynchronous to the M-bus clocks.

4.4.3.7. MHALT

When the MHALT signal is asserted, workstation processors should halt. Any module or any package-switch logic may assert MHALT.

Transitions of the MHALT signal are asynchronous to the M-bus clocks.

4.4.3.8. MABORT

The MABORT signal is asserted by any module detecting an error condition during a M-bus transaction. Transaction errors are the following:

- M-bus-arbitration errors (multiple masters or slaves)
- Parity errors on the MCMD, MSTAT, or MDAL signals
- Reserved values of the MCMD codes
- Too many slave wait/busy cycles
- Interlock violations
- Cache tag-parity errors

M-bus-arbitration errors occur when multiple MBRQ signals are asserted when only one module, either the M-bus master or the M-bus slave, should be in control of the M-bus. When a master or slave detects a MBRQ signal from another module asserted while it is driving the M-bus, it should assert its MABORT signal. M-bus errors should also be generated if the M-bus master/slave MBRQ signal is prematurely deasserted during a transaction.

Parity errors on the MCMD, MSTAT, and MDAL signals occur when there are M-bus transceiver failures, connector failures, logic failures that cause no module to drive the M-bus, logic failures that cause multiple modules to drive the M-bus, and failures in the parity-checking logic. Parity errors can also result when AC timing is marginal, in which case only some of the modules may detect the error. When any module

detects a parity error on MCMD, MSTAT, or MDAL during one of the M-bus cycles specified by the table in the MXPAR section, that module should assert its MABORT signal.

Only 7 of the possible 16 MCMD encodings are valid during P2 of M-bus transactions. Invalid encodings result when there is a hardware failure in the M-bus master's M-bus-interface MCMD logic, when there is a hardware failure in the monitoring module's M-bus-interface MCMD logic, or when the monitoring module is *out of sync* in a different transaction phase than the M-bus master. When any module detects an invalid encoding during P2 of a M-bus transaction, it should assert its MABORT signal.

If a M-bus slave specifies WAIT status for more than 256 M-bus cycles, MABORT should be asserted. M-bus slave interfaces should implement their own timeout logic for M-bus-initiated, internal-bus transactions. If an internal timeout occurs, the slave module's M-bus interface should specify ERROR status to terminate the M-bus transaction. To minimize the length of time that the M-bus is stalled, the internal timeout should be the minimum necessary for the specific module. Indicating ERROR status also limits the failure status to only the module that initiated the M-bus transaction. The M-bus slave wait timer is intended to catch failures of slave M-bus-interface logic, which affects the entire workstation. When any module detects too many slave wait-status cycles, it should assert its MABORT signal.

When a read-interlocked transaction is initiated, all M-bus interfaces update their interlocked unit. The address is interlocked against other interlocked reads to that same address until a write-unlock transaction is initiated. While an address is interlocked, M-bus interfaces must stall internal requests for read-interlocked transactions to that same address. If a M-bus interface observes a read-interlocked transaction on the M-bus for an address it considers interlocked, this means that a hardware failure has caused the interlock units to become inconsistent between the M-bus interfaces of the modules. As a result, the interface should assert its MABORT signal. Similarly, if a M-bus interface observes a write-unlock transaction for an address it does not consider interlocked, it should also assert its MABORT signal. Modules that never act as M-bus masters need not implement the interlock unit and corresponding checking logic.

During memory-space transactions, all caches probe their tag store to determine whether or not the octaword is present in their cache. If a parity error is detected in the tag for the specified octaword, the cache probe cannot be completed. When such a tag-parity error occurs, the module must assert its MABORT signal.

Once a module asserts its MABORT signal, it must remain asserted for eight cycles to ensure the current M-bus transaction has been completed and that all M-bus interfaces have returned to the idle state. The current M-bus master and/or M-bus slave should abort the transaction as soon as practical. At the end of the current transaction, a workstation-wide machine check should be initiated. Inherent pipelining of the M-bus may result in some errors not being detected until the cycle after the one in which the transaction was completed on the M-bus.

Lack of slave response is immediately indicated during P7 of memory-space read transactions, during P4 of I/O-space transactions, and during P4 of interrupt-acknowledge transactions, because none of the MBRQ signals are asserted. This results in immediate termination of the M-bus transaction. Modules must not assert MABORT in this situation; instead, the M-bus interface of the M-bus master should indicate an error to its internal logic, and the M-bus should immediately return to the idle state.

M-bus interfaces must clear their interlocked-sequence-in-progress flags whenever MABORT is asserted on the M-bus, as part of returning to an idle state. The state of the interlocked-sequence-in-progress flags is unchanged by cycles that terminate because of no slave response. Read interlocked transactions must only be issued at addresses to which a slave is known to respond, as the interlock flag is set as soon as the M-bus master acquires the M-bus.

4.4.4. Clock Distribution

The MCLKA and MCLKB signals are the master clocks for all of the M-bus interface logic. The MCLKI signal functions as an interval-timer interrupt.

4.4.4.1. MCLKA

MCLKA is the master clock for the M-bus. All signal transitions and M-bus-interface state machines are referenced to MCLKA. The M-bus cycles, Pn, are defined by the rising edge of MCLKA. M-bus signals transition after the rising edge of MCLKA; that is, MCLKA should be used to clock registers and enable latches driving the M-bus. MCLKA is radially distributed to each module to minimize skew between modules.

4.4.4.2. MCLKB

MCLKB is the slave clock for the M-bus. M-bus receiver registers and latches should be clocked by MCLKB. MCLKB will be positioned with respect to MCLKA to meet receiver hold-time requirements in the presence of clock skew. MCLKB is radially distributed to each module to minimize skew between modules.

4.4.4.3. MCLKI

The MCLKI signal is a 100-Hz square wave for use as an interval-clock interrupt. Transitions of the MCLKI signal are asynchronous with respect to the M-bus clocks. Multiple modules may have circuitry to generate the MCLKI signal. Consequently, modules that do implement MCLKI circuitry must default to not driving the signal until enabled by software. This implies that no modules drive the MCLKI signal after workstation reset (MRESET asserted). Software must enable driving MCLKI on one of the modules before processors receive interval clock interrupts.

4.5. Transactions

There are five categories of M-bus transactions:

- Memory read
- Memory write
- I/O read
- I/O write
- Interrupt acknowledge

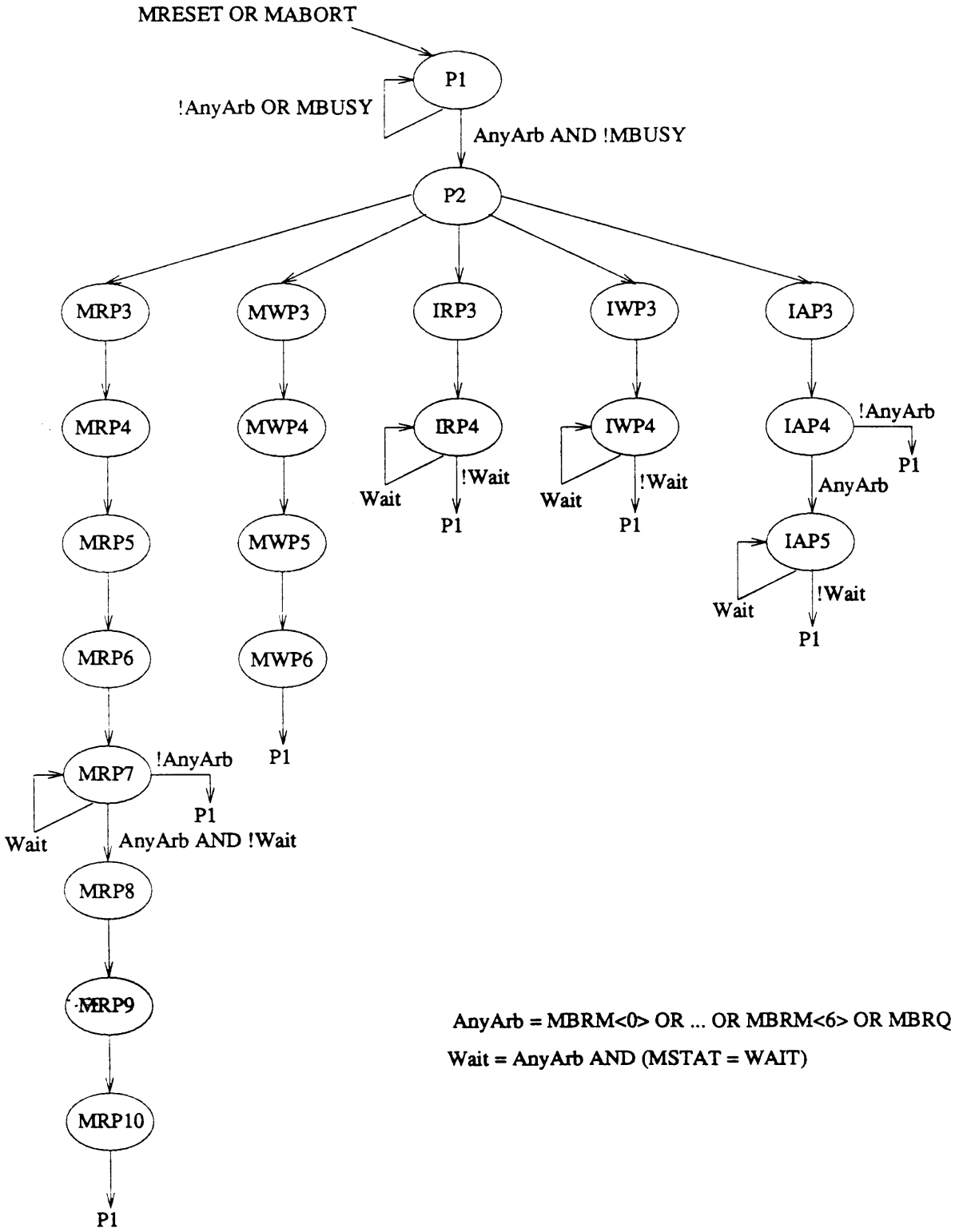
Memory-space operations transfer data between two or more modules and maintain data consistency between all caches. Memory-space transactions always transfer four longwords (one cache line) between modules. When a memory-space read or write-through transaction is initiated, all modules probe their cache to determine whether or not the line is shared. For memory-space reads, a cache supplies the read data for shared dirty lines, and a memory module supplies read data for unshared lines or shared clean lines. For memory-space write-throughs, the memory module, as well as all caches that contain the line, update the line. For memory-space victim writes, only the memory module updates the line.

I/O-space transactions transfer data between exactly two modules. I/O-space transactions transfer one masked longword between modules. I/O-space data is never cached.

Interrupt-acknowledge transactions transfer one interrupt vector between exactly two modules.

Figure 4-3 shows the M-bus states for the various phases of transactions. The MR phase prefix stands for *memory read*. The MW phase prefix stands for *memory write*. The IR phase prefix stands for *I/O read*. The IW phase prefix stands for *I/O write*. The IA phase prefix stands for *interrupt acknowledge*. The *AnyArb* and *Wait* terms represent derived signals within M-bus interface logic. For the I/O read and write states, no slave response termination of the transaction is implicit in the *!Wait* term which contains *AnyArb*.

Table 4-19: M-Bus State Diagram



4.5.1. Transaction Notation

The following sections contain sample M-bus transactions in the format shown in Table 4-20 *cycle* column lists the sequential transaction phase, for example, P3. The *MBRQ* column lists the module asserting its MBRQ signal, with arbitration assertions in parentheses. The *MCMD* column lists the cycle-type code present on the MCMD signals. The *MSTAT* column lists the cycle-status code present on the MSTAT signals. The *MDAL* column indicates the interpretation of the value present on the MDAL signals. The *function* column describes the operation that occurs during the cycle.

Table 4-20: M-Bus Cycle Nomenclature

Cycle	MBRQ	MCMD	MSTAT	MDAL	Function
Pn	WHO	OP/MASK	STATUS	VALUE	WHAT

4.5.2. Memory-Space Reads

Table 4-21 shows the cycles that compose a memory-space read transaction. During cycle P1, the initiating module arbitrates for the M-bus. During cycle P2, the M-bus master indicates the type of transaction and the read address. The octaword address is specified on MDAL<30:4>; MDAL<31> and MDAL<3:0> must be 0. During cycles P3 through P6, the master waits for a slave to respond. A memory module or cache transmits read data during cycles P7 through P10.

Table 4-21: M-Bus Memory-Read Transaction

Cycle	MBRQ	MCMD	MSTAT	MDAL	Function
P1	(M)				Bus arbitration
P2	M	READ		Address	Read address
P3	M				Wait
P4	M				Wait
P5	M				Wait
P6	(S)				Wait
P7	S		GOOD	Data 0	First longword
P8	S		GOOD	Data 1	Second longword
P9	S		GOOD	Data 2	Third longword
P10			GOOD	Data 3	Fourth longword

The M-bus cycles P3 through P6 provide dynamic RAM-access time for memory modules. Memory modules must not provide read data before cycle P7. Memory modules requiring more RAM-access time may insert additional WAIT cycles after cycle P6. Starting in cycle P7, memory modules must either specify WAIT or read data.

During a memory-space read transaction, all caches in Firefox workstations have M-bus cycles P3 through P5 to complete a probe of their tag store. If a cache hit to a dirty line results, the modules' M-bus interface asserts its MBRQ signal during cycle P6 and drives the MBRQ, MSTAT, and MDAL signals during cycles P7 through P10. This applies only to shared dirty cache lines; memory modules supply data for shared clean lines.

There are two types of memory-read transactions, READ and READU. Both transactions have the same format, require cache modules to probe their tag store, assert the MSHARED signal if a hit results, and supply data if a hit to a dirty line results. If no cache arbitrates to supply read data, then the selected memory module supplies read data. After a READ transaction, tag stores with hits set their SHARED bit. After a READU transaction, tag stores do not modify their SHARED bit.

The READ transaction must be used by processor modules and I/O modules that require a coherent memory-space. I/O module that do not need require write-through updates of previously read data should

use READU. This allows such I/O modules to read communication blocks cached in processor modules without the processor modules incurring the overhead of unnecessary write-throughs.

If a memory module detects any MBRQ signal asserted during cycle P6, it aborts its read cycle.

When modules present read data on the MDAL signals, the MSTAT signals specify either GOOD, indicating that no data errors occurred, or CORRECTED, indicating that a single-bit error was detected and corrected. If the module detects a parity error or double-bit ECC error it asserts MDATINV while driving that longword onto MDAL. If memory modules implement ECC on words wider than 32 bits, they may specify CORRECTED for each longword transfer of memory words whether or not that longword is the one that actually contains the corrected single-bit error. Table 4-22 shows an example of a memory read of corrected data.

Table 4-22: Memory-Read Transaction with Corrected Data

Cycle	MBRQ	MCMD	MSTAT	MDAL	Function
P1	(M)				Bus arbitration
P2	M	READ		Address	Read address
P3	M				Wait
P4	M				Wait
P5	M				Wait
P6					Wait
P7	S		GOOD	Data 0	First longword
P8	S		GOOD	Data 1	Second longword
P9	S		CORRECTED	Data 2	Third longword
P10			CORRECTED	Data 3	Fourth longword

Once the first longword of data is supplied, no additional slave wait cycles are allowed. The four longwords of data must be transferred in four consecutive M-bus cycles. M-bus-interface logic must interpret additional WAIT status as ERROR status.

Whenever a M-bus interface detects CORRECTED status in response to a memory-space read, it should generate an interrupt to its local processor.

4.5.3. Memory-Space Writes

Table 4-23 shows the cycles that compose a memory-space write transaction. During cycle P1, the initiating module arbitrates for the M-bus. During cycle P2, the M-bus master indicates the type of transaction and the write address. The octaword address is specified on MDAL<30:4>; MDAL<31> and MDAL<3:0> must be 0. During cycles P3 through P6, the M-bus master transfers write data.

Table 4-23: M-bus Memory-Write Transaction

Cycle	MBRQ	MCMD	MSTAT	MDAL	Function
P1	(M)				Bus arbitration
P2	M	WRITE		Address	Write address
P3	M	MASK		Data 0	First longword
P4	M	MASK		Data 1	Second longword
P5	M	MASK		Data 2	Third longword
P6	S	MASK		Data 3	Fourth longword

During each M-bus cycle in which a master transfers write data on MDAL, MCMD functions as a byte mask. If MCMD<n> is asserted, then the corresponding byte of MDAL must be written into a shared cache line. If MCMD<n> is deasserted, then the corresponding byte of MDAL should not be written into a shared cache line.

Memory modules always write the entire octaword, regardless of the byte mask. This implies that memory modules have invalid contents for shared dirty lines until the victim write occurs. However, since caches supply read data for M-bus reads of shared dirty lines, the memory module value is irrelevant.

If any byte of a longword has an internal module-parity error, the module must assert MDATINV while it drives the longword containing that byte onto MDAL. Modules that write the invalid data into their internal storage must store an indication that the data is invalid along with the data. For example, memory modules force a bad check-bit-field for the affected memory address; caches force bad parity for the affected byte.

During a memory-space write transaction, all caches in Firefox workstations have M-bus cycles P3 through P5 to complete a probe of their tag store. If a cache detects a hit, it must assert MSHARED during cycle P6.

Memory modules or caches that are referenced by the memory address assert their MBRQ signal during P6. The M-bus master should treat the lack of any MBRQ as a no-slave response. Memory modules or caches that require more cycles to complete the write transaction may assert MBUSY to stall subsequent transactions.

4.5.4. I/O-Space Reads

I/O-space read transactions adhere to the same M-bus protocol as memory-space read transactions except that I/O-space references are uncached and transfer at most one longword of data. Consequently, caches need not monitor I/O-space transactions. Since caches are not involved, the mandatory wait cycles have been eliminated. Table 4-24 shows an I/O-read transaction.

Table 4-24: M-Bus I/O-Read Transaction

Cycle	MBRQ	MCMD	MSTAT	MDAL	Function
P1	(M)				Bus arbitration
P2	M	READ		Address	Read address
P3	M	MASK			Byte mask
P4	S		GOOD	Data	Read data

During cycle P1, the initiating module arbitrates for the M-bus. During cycle P2, the M-bus master indicates the type of transaction and the read address. The longword address is specified on MDAL<30:2>; MDAL<31> must be 1; MDAL<1:0> is undefined. During cycle P3, the M-bus master specifies the byte mask for the longword address.

The M-bus slave may stall the return of read data, starting in cycle P4, by specifying WAIT on MSTAT. Most I/O devices will require several WAIT cycles.

If the M-bus slave specifies GOOD status, the M-bus transaction terminates and the M-bus master returns the read data to its internal logic.

If the M-bus slave specifies RETRY status, the M-bus transaction terminates and the M-bus master instructs its internal logic to retry the transaction at a later time.

If the M-bus slave specifies ERROR status, the M-bus transaction terminates and the M-bus master informs its internal logic that the read failed. The M-bus interface of the master should implement a status register that allows internal logic to determine whether no slave responded or a slave responded with an error.

4.5.5. I/O-Space Writes

I/O-space write transactions adhere to the same M-bus protocol as memory-space write transactions except that I/O-space references are uncached and transfer at most one longword of data. Consequently, caches need not monitor I/O-space transactions. Since caches are not involved, the mandatory wait cycles have been eliminated. Table 4-25 shows an I/O-write transaction.

Table 4-25: M-bus I/O-Write Transaction

Cycle	MBRQ	MCMD	MSTAT	MDAL	Function
P1	(M)				Bus arbitration
P2	M	WRITE		Address	Write address
P3	M	MASK		Data	Write data
P4	S		GOOD		Write completed

During cycle P1, the initiating module arbitrates for the M-bus. During cycle P2, the M-bus master indicates the type of transaction and the write address. The longword address is specified on MDAL<30:2>; MDAL<31> must be 1; MDAL<1:0> is undefined. During cycle P3, the M-bus master specifies the byte mask for the longword address and the write data.

The M-bus slave may stall the completion of the write transaction starting in cycle P4 by specifying WAIT on MSTAT. Most I/O devices will require several WAIT cycles.

If the M-bus slave specifies GOOD status, the M-bus transaction terminates and the M-bus master indicates successful completion of the write to its internal logic.

If the M-bus slave specifies RETRY status, the M-bus transaction terminates and the M-bus master instructs its internal logic to retry the transaction at a later time.

If the M-bus slave specifies ERROR status, the M-bus transaction terminates and the M-bus master informs its internal logic that the write failed. The M-bus interface of the master should implement a status register that allows internal logic to determine whether no slave responded or a slave responded with an error.

4.5.6. Interlocked Transactions

Interlocked transactions are initiated by an interlocked-read transaction. Interlocked-read transactions are identical to normal read transactions except that they also record the interlocked address and set the interlocked-sequence-in-progress flag in one of the two interlock-unit slots. Write-unlock transactions are identical to normal write transactions except that they also deassert the interlocked-sequence-in-progress flag for the locked address.

When a processor generates an interlocked read, its cache must force a miss. This guarantees that the interlocked read generates a M-bus transaction. If a processor requests an interlocked-read transaction of its M-bus interface, and that address is locked or the interlock unit is full, the processor must be stalled until the address is unlocked.

Regardless of the state of the interlocked-sequence-in-progress flags, noninterlocked M-bus transactions proceed. In other words, the interlock unit only blocks interlocked-read transactions from arbitrating for the M-bus until the write-unlock transaction for the interlocked address is completed.

Assertion of MABORT unconditionally clears the interlocked-sequence-in-progress flags.

Refer to the sections discussing memory-space transactions for detailed descriptions of the M-bus protocol.

4.5.7. Interrupt-Acknowledge Transactions

An interrupt-acknowledge transaction follows the same protocol as a memory space read transaction. In place of a read address, the M-bus master provides an interrupt level. All modules with a pending interrupt at that level assert their M-bus-arbitration request (MBRQ) signal during cycle P4. The module with the highest M-bus priority supplies an interrupt vector during cycle P5. The highest-priority module may insert additional WAIT cycles before providing the vector. Most I/O devices will require multiple WAIT cycles to produce a vector. Lower-priority modules abort the transaction after cycle P4. Table 4-26 shows an example of an interrupt-acknowledge transaction.

Table 4-26: M-Bus Interrupt-Acknowledge Transaction

Cycle	MBRQ	MCMD	MSTAT	MDAL	Function
P1	(M)				Bus arbitration
P2	M	INTACK		Level	Interrupt level
P3	M				Wait
P4	(S)				Slave arbitration
P5	S		GOOD	Vector	Interrupt vector

During cycle P1, the initiating module arbitrates for the M-bus. During cycle P2, the M-bus master indicates the type of transaction and the interrupt level. The interrupt level is specified on MDAL<6:2>; MDAL<31> must be 1; MDAL<30:7> and MDAL<1:0> are undefined.

If no MBRQ signals are asserted during cycle P4, indicating that no M-bus interfaces are arbitrating to provide an interrupt vector, the M-bus interface of the M-bus master should initiate passive-release processing (*passive release* means that software is uninterrupted). If multiple processors simultaneously initiate an interrupt-acknowledge transaction, the highest-priority processor receives the interrupt vector, and all other processors receive passive releases. This allows an interrupt level to be serviced by multiple processors.

If the M-bus slave specifies GOOD status, the M-bus transaction terminates and the M-bus master returns the interrupt vector to its internal logic. The interrupt vector is encoded on MDAL<15:0>; MDAL<31:16> is undefined.

If the M-bus slave specifies RETRY status, the M-bus transaction terminates and the M-bus master instructs its internal logic to retry the transaction at a later time.

If the M-bus slave specifies ERROR status, the M-bus transaction terminates and the M-bus master informs its internal logic that the interrupt acknowledge failed. This is equivalent to a no-slave-response passive release.

4.6. Example Transactions

The following sections show sample memory, I/O, and interrupt-acknowledge transactions.

All signals in pictorial diagrams are shown with active-high assertion state for clarity. This may not correspond to the assertion state on the backplane.

4.6.1. Memory Read to Unshared Line

Figure 4-3 shows a no-wait-state memory read for an unshared cache line with two modules arbitrating for the M-bus.

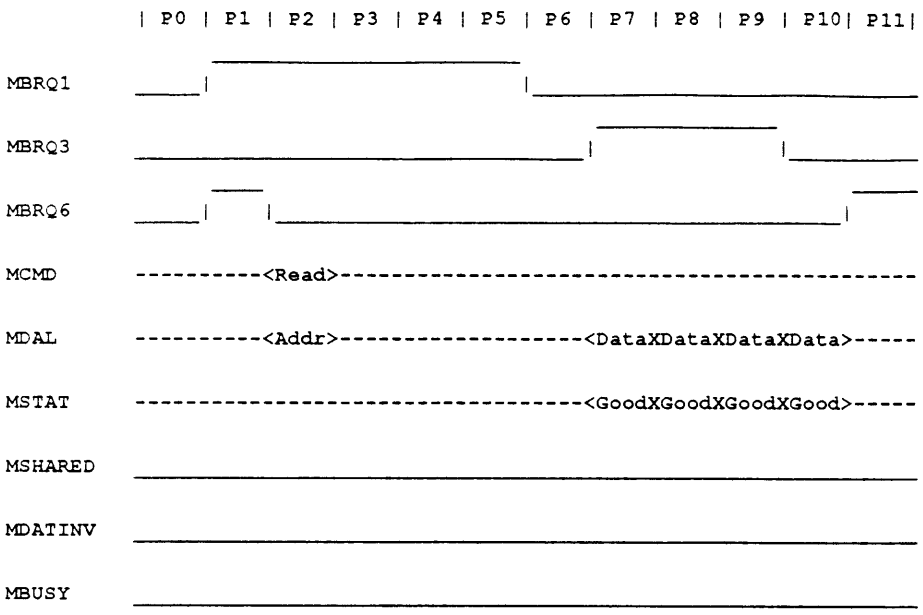


Figure 4-3: Memory Read Transaction to Unshared Line

P0 The M-bus is idle.

P1 The modules in slots 1 and 6 arbitrate for the M-bus.

P2 Slot 1 has higher priority, wins the M-bus arbitration, and continues to assert its MBRQ signal to confirm this, at the same time that it drives MCMD with READ and MDAL with the memory address. Slot 6 deasserts its MBRQ, since it lost the M-bus arbitration.

P3 Modules monitoring the M-bus transaction start decoding the address.

P4 Modules monitoring the M-bus transaction continue servicing the request.

P5 Modules monitoring the M-bus transaction continue servicing the request.

P6 No caches contain the referenced line, so MSHARED remains deasserted.

P7 The memory module in slot 3 contains the referenced line. It asserts its MBRQ, indicates good data, and supplies the first longword.

P8 Slot 3 supplies the second longword.

P9 Slot 3 supplies the third longword.

P10 Slot 3 supplies the fourth longword, ending the transaction.

P11 Slot 6 rearbitrates for its pending transaction. This is the earliest cycle that a new transaction may start.

4.6.2. Memory Read to Shared Clean Line

Figure 4-4 shows a memory read for a shared, clean cache line.

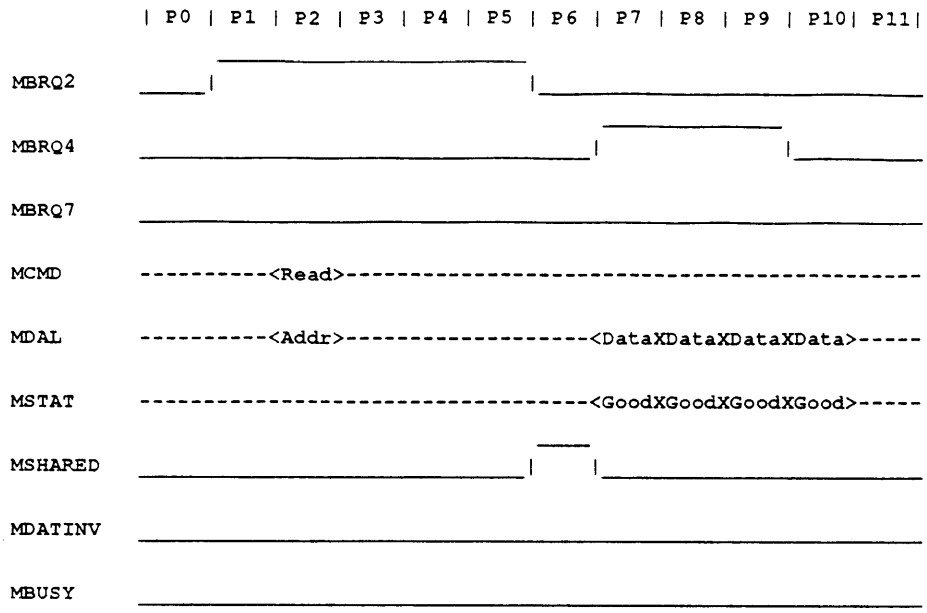


Figure 4-4: Memory Read Transaction to Shared Clean Line

P0 The M-bus is idle.

P1 The module in slot 2 arbitrates for the M-bus.

P2 Slot 2 asserts its MBRQ signal to confirm that it won the M-bus. It also drives MCMD with READ and MDAL with the memory address.

P3 Modules monitoring the M-bus transaction start decoding the address.

P4 Modules monitoring the M-bus transaction continue servicing the request.

P5 Modules monitoring the M-bus transaction continue servicing the request.

P6 The cache of the module in slot 7 contains the referenced line, which is unmodified, so it asserts MSHARED, but leaves its MBRQ signal deasserted.

P7 The memory module in slot 4 supplies the first longword.

P8 Slot 4 supplies the second longword.

P19 Slot 4 supplies the third longword.

P10 Slot 4 supplies the fourth longword, ending the transaction.

P11 There are no pending transactions, so the M-bus remains idle.

4.6.3. Memory Read to Shared Dirty Line

Figure 4-5 shows a memory read for a shared, dirty cache line.

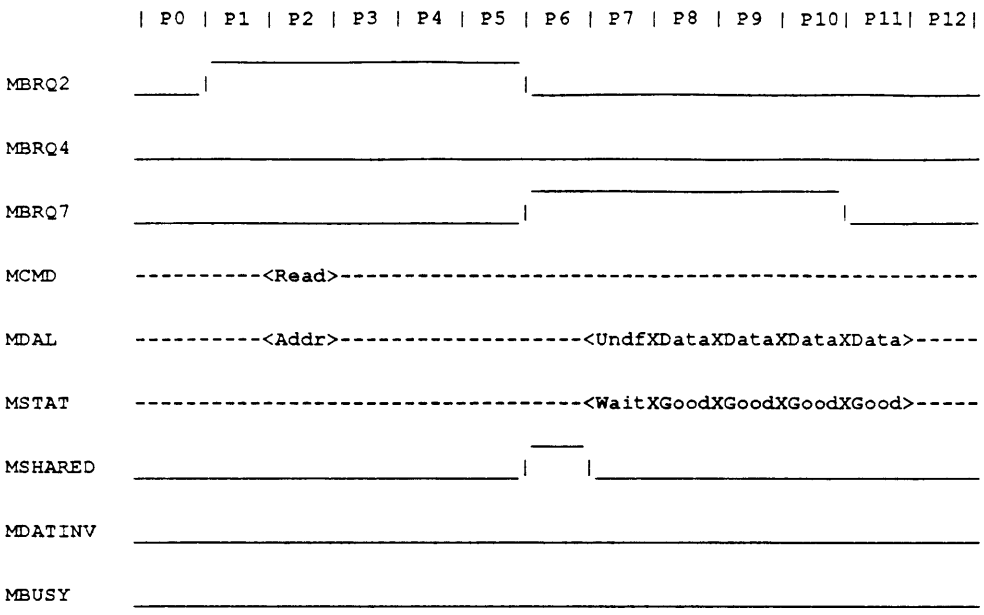


Figure 4-5: Memory Read Transaction to Shared Dirty Line

P0 The M-bus is idle.

P1 The module in slot 2 arbitrates for the M-bus.

P2 Slot 2 asserts its MBRQ signal to confirm that it won the M-bus. It also drives MCMD with READ and MDAL with the memory address.

P3 Modules monitoring the M-bus transaction start decoding the address.

P4 Modules monitoring the M-bus transaction continue servicing the request.

P5 Modules monitoring the M-bus transaction continue servicing the request.

P6 The cache of the module in slot 7 contains the referenced line, which has been modified, so it asserts both MSHARED and its MBRQ signal.

P7 Slot 7 continues to assert its MBRQ and indicates wait status. The selected memory module in slot 4 aborts its read operation.

P8 Slot 7 supplies the first longword.

P9 Slot 7 supplies the second longword.

P10 Slot 7 supplies the third longword.

P11 Slot 7 supplies the fourth longword, ending the transaction.

P12 There are no pending transactions, so the M-bus remains idle.

4.6.4. Memory Read with Uncorrectable ECC Error

Figure 4-6 shows a no-wait-state memory read for an unshared cache line with an uncorrectable ECC error in the second quadword.

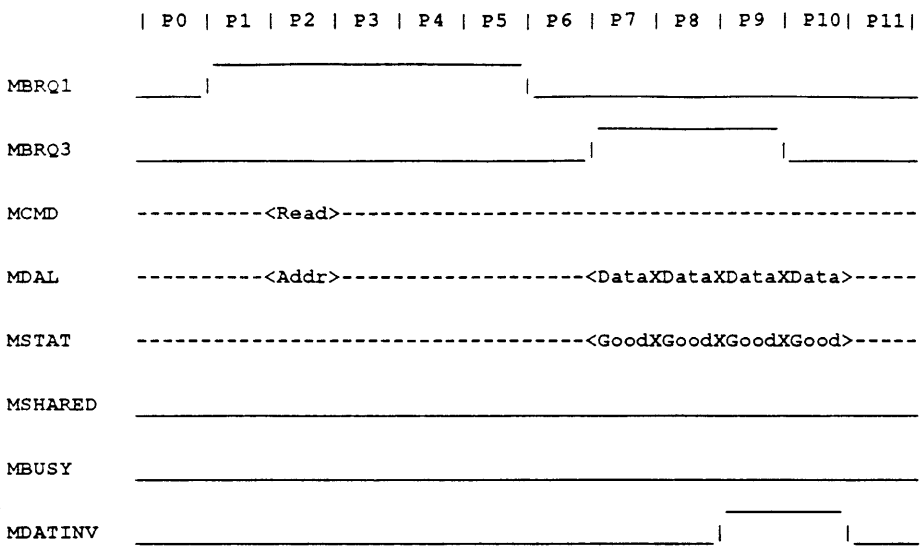


Figure 4-6: Memory Read Transaction with Uncorrectable ECC Error

P0 The M-bus is idle.

P1 The module in slot 1 arbitrates for the M-bus.

P2 Slot 1 wins the M-bus arbitration and continues to assert its MBRQ signal to confirm this, at the same time that it drives MCMD with READ and MDAL with the memory address.

P3 Modules monitoring the M-bus transaction start decoding the address.

P4 Modules monitoring the M-bus transaction continue servicing the request.

P5 Modules monitoring the M-bus transaction continue servicing the request.

P6 No caches contain the referenced line, so MSHARED remains deasserted.

P7 The memory module in slot 3 contains the referenced line. It asserts its MBRQ, indicates good data, and supplies the first longword.

P8 Slot 3 supplies the second longword.

P9 Slot 3 supplies the third longword, and it asserts MDATINV to indicate that the data is known to have an error.

P10 Slot 3 supplies the fourth longword, and it asserts MDATINV to indicate that the data is known to have an error.

P11 This is the earliest cycle that a new transaction may start.

4.6.5. Memory Read to Non-Existent Memory

Figure 4-7 shows a memory read to a non-existent memory-space address.

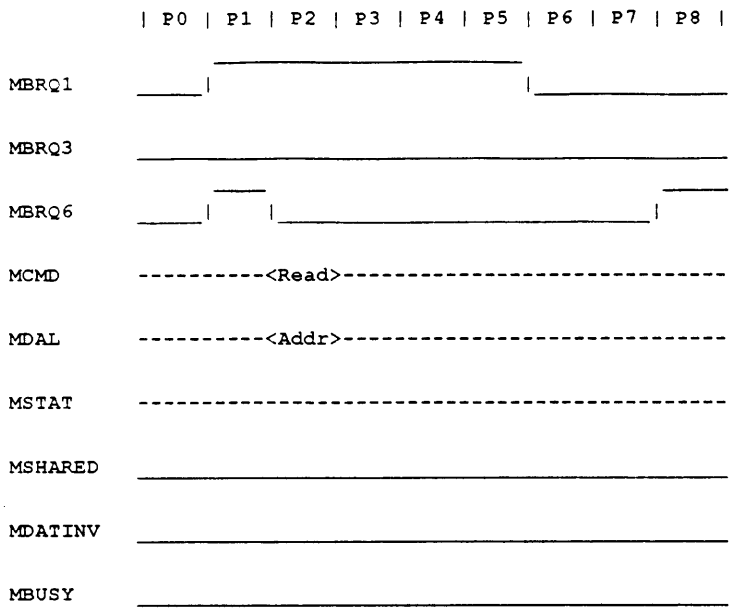


Figure 4-7: Memory-Read Transaction with No Slave Response

P0 The M-bus is idle.

P1 The modules in slots 1 and 6 arbitrate for the M-bus.

P2 Slot 1 has higher priority, wins the M-bus arbitration, and continues to assert its MBRQ signal to confirm this, at the same time that it drives MCMD with READ and MDAL with the memory address. Slot 6 deasserts its MBRQ, since it lost the M-bus arbitration.

P3 Modules monitoring the M-bus transaction start decoding the address.

P4 Modules monitoring the M-bus transaction continue servicing the request.

P5 Modules monitoring the M-bus transaction continue servicing the request.

P6 No caches contain the referenced line, so MSHARED remains deasserted.

P7 No memory module contains the referenced line, so all the MBRQ signals remain deasserted. This is the last cycle of the transaction.

P8 Slot 6 rearbitrates for its pending transaction. This is the first possible cycle for a new transaction.

4.6.6. Victim Write

Figure 4-8 shows a victim write to flush out a dirty cache line.

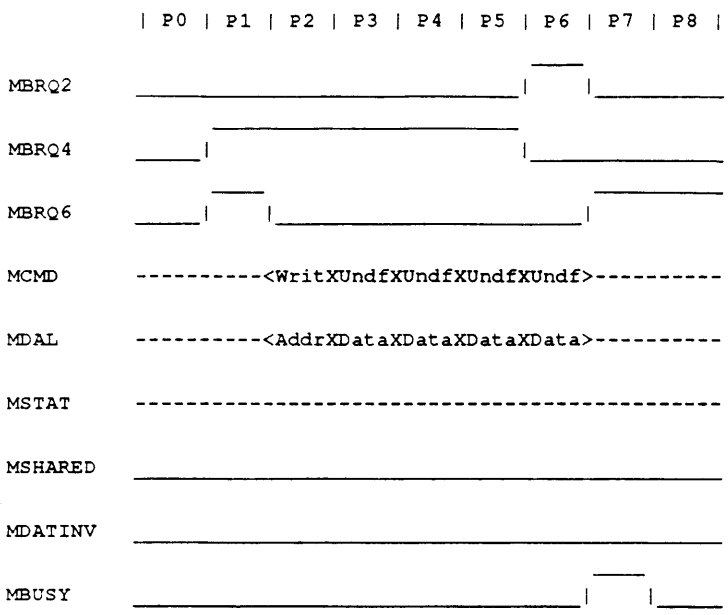


Figure 4-8: Victim Write Transaction

P0 The M-bus is idle.

P1 The modules in slots 4 and 6 arbitrate for the M-bus.

P2 Slot 4 wins the arbitration and asserts its MBRQ signal to confirm that it won the M-bus. It also drives MCMD with WRITE and MDAL with the memory address.

P3 Slot 4 drives MDAL with the first longword.

P4 Slot 4 drives MDAL with the second longword.

P5 Slot 4 drives MDAL with the third longword.

P6 Slot 4 drives MDAL with the fourth longword. The memory module in slot 2 asserts MBRQ to indicate that it is the slave.

P7 The memory module in slot 2 asserts MBUSY while it complete the write. The module in slot 6 rearbitrates for the M-bus.

P8 The memory module has completed the memory write, so it deasserts MBUSY. Since MBUSY was asserted in P7, the module in slot 6 continues to rearbitrate for the M-bus.

4.6.7. Victim Write with Internal Parity Error

Figure 4-9 shows a memory victim write to flush out a dirty cache line that has a parity error in the second longword.

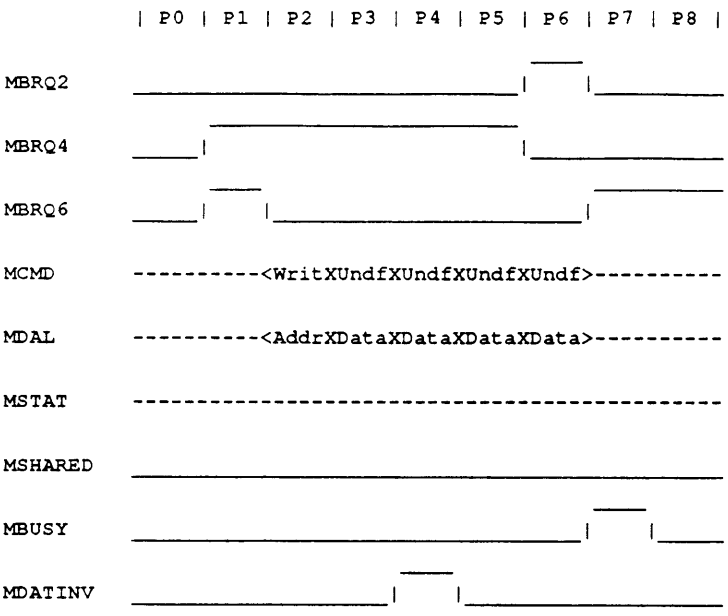


Figure 4-9: Victim Write Transaction with Internal Parity Error

P0 The M-bus is idle.

P1 The modules in slots 4 and 6 arbitrate for the M-bus.

P2 Slot 4 wins the arbitration and asserts its MBRQ signal to confirm that it won the M-bus. It also drives MCMD with WRITE and MDAL with the memory address.

P3 Slot 4 drives MDAL with the first longword.

P4 Slot 4 drives MDAL with the second longword and asserts MDATINV to indicate that an internal parity error was detected on the data.

P5 Slot 4 drives MDAL with the third longword.

P6 Slot 4 drives MDAL with the fourth longword. The memory module in slot 2 asserts MBRQ to indicate that it is the slave.

P7 The memory module in slot 2 asserts MBUSY while it complete the write. The module in slot 6 rearbitrates for the M-bus.

P8 The memory module has completed the memory write, so it deasserts MBUSY. Since MBUSY was asserted in P7, the module in slot 6 continues to rearbitrate for the M-bus.

4.6.8. Write-Through to Unshared Line

Figure 4-10 shows a memory write-through for a cache line that was shared but has become unshared. The memory address references a memory module in slot 2.

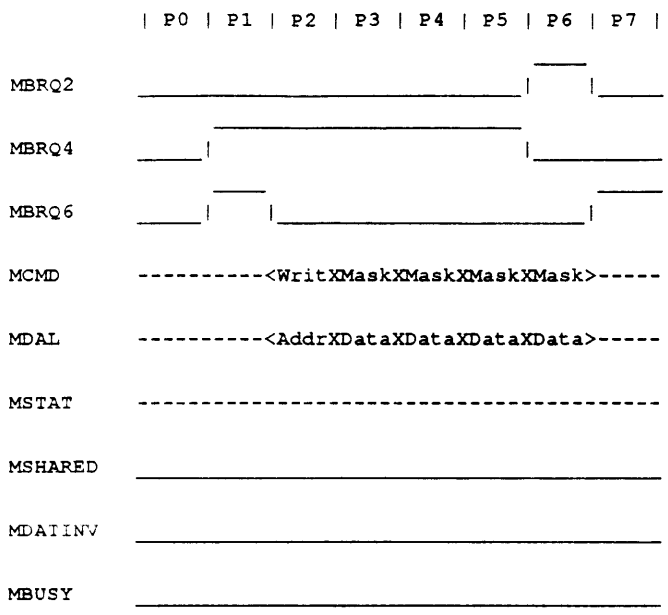


Figure 4-10: Write-Through Transaction to Unshared Line

P0 The M-bus is idle.

P1 The modules in slots 4 and 6 arbitrate for the M-bus.

P2 Slot 4 wins the arbitration and asserts its MBRQ signal to confirm that it won the M-bus. It also drives MCMD with WRITE and MDAL with the memory address.

P3 Slot 4 drives MCMD and MDAL with a byte mask and the first longword.

P4 Slot 4 drives MCMD and MDAL with a byte mask and the second longword.

P5 Slot 4 drives MCMD and MDAL with a byte mask and the third longword.

P6 Slot 4 drives MCMD and MDAL with a byte mask and the fourth longword. The referenced line was not in any other cache, so MSHARED remains deasserted. The memory module in slot 2 asserts MBRQ to indicate that it is the slave.

P7 The memory module in slot 2 has completed the write transaction, so it does not assert MBUSY. The module in slot 6 rearbitrates for the M-bus.

4.6.9. Write-Through to Shared Line

Figure 4-11 shows a memory write-through for a shared cache line.

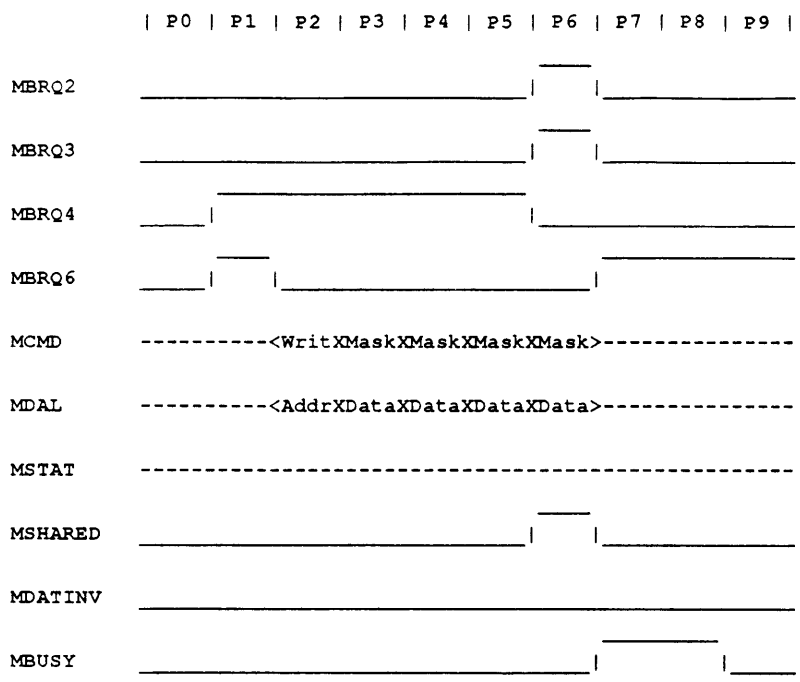


Figure 4-11: Write-Through Transaction to Shared Line

P0 The M-bus is idle.

P1 The modules in slots 4 and 6 arbitrate for the M-bus.

P2 Slot 4 wins the arbitration and asserts its MBRQ signal to confirm that it won the M-bus. It also drives MCMD with WRITE and MDAL with the memory address.

P3 Slot 4 drives MCMD and MDAL with a byte mask and the first longword.

P4 Slot 4 drives MCMD and MDAL with a byte mask and the second longword.

P5 Slot 4 drives MCMD and MDAL with a byte mask and the third longword.

P6 Slot 4 drives MCMD and MDAL with a byte mask and the fourth longword. The referenced line is in another cache in slot 3, which asserts both MSHARED and MBRQ. The memory module in slot 2 asserts MBRQ to indicate that it is the slave.

P7 The memory module in slot 2 has completed the write transaction so it does not assert MBUSY. The cache is not finished with the write-through, so it asserts MBUSY. The module in slot 6 rearbitrates for the M-bus.

P8 The cache is not finished with the write-through, so it continues to assert MBUSY. The module in slot 6 continues to rearbitrate for the M-bus, since MBUSY was asserted in P7.

P9 The cache has completed the write through so it deasserts MBUSY. The module in slot 6 continues rearbitrates for the M-bus since MBUSY was asserted in P8.

4.6.10. Victim Write with Address Parity Error

Figure 4-12 shows a memory victim write with a MDAL parity error during P2.

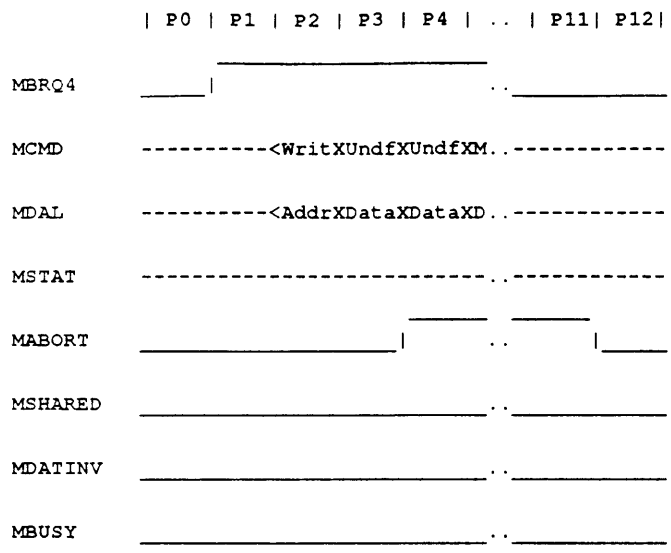


Figure 4-12: Victim Write Transaction with Address Parity Error

P0 The M-bus is idle.

P1 The module in slot 4 arbitrates for the M-bus.

P2 Slot 4 asserts its MBRQ signal to confirm that it won the M-bus. It also drives MCMD with WRITE and MDAL with the memory address.

P3 A slot detected a parity error on the value of MDAL/MDPAR on the M-bus during cycle P2.

P4 Slot 4 drives MDAL with the second longword. The slot that detected the parity error asserts MABORT.

P5-P10 M-bus interfaces return to idle state.

P11 The module that detected the error continues to assert MABORT. All M-bus interfaces should initiate a machine check of their internal logic.

P12 MABORT is deasserted after eight cycles.

4.6.11. I/O Read

Figure 4-13 shows an I/O read.

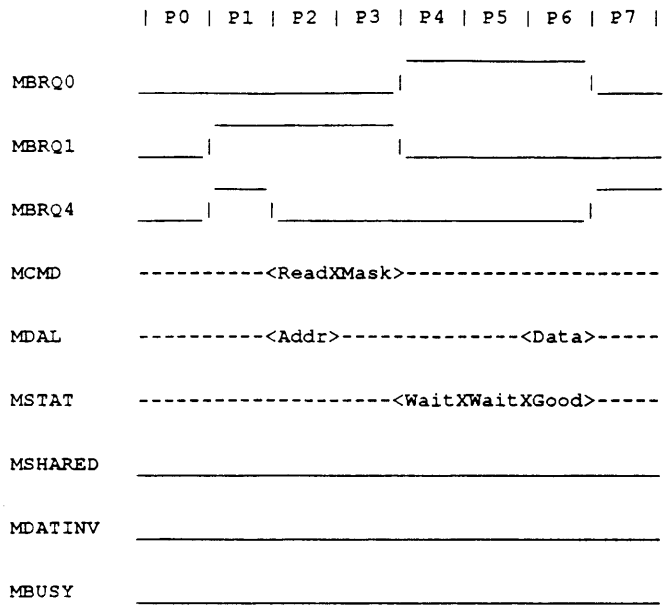


Figure 4-13: I/O-Read Transaction

P0 The M-bus is idle.

P1 The modules in slots 1 and 4 arbitrate for the M-bus.

P2 Slot 1 wins the arbitration and asserts its MBRQ signal to confirm that it won the M-bus. It also drives MCMD with READ and MDAL with the I/O address.

P3 Modules decode the I/O address. Slot 1 specifies the byte mask on MCMD.

P4 Slot 0 asserts its MBRQ to indicate that it is processing the I/O read, but it specifies WAIT on MSTAT, since data is not yet available.

P5 Slot 0 asserts its MBRQ to indicate that it is processing the I/O read, but it specifies WAIT on MSTAT, since data is not yet available.

P6 Slot 0 drives read data onto MDAL and specifies GOOD on MSTAT to complete the transaction.

P7 Slot 4 rearbitrates for its pending transaction. This is the first possible cycle for a new transaction.

4.6.12. I/O Read with No Slave Response

Figure 4-14 shows an I/O read.

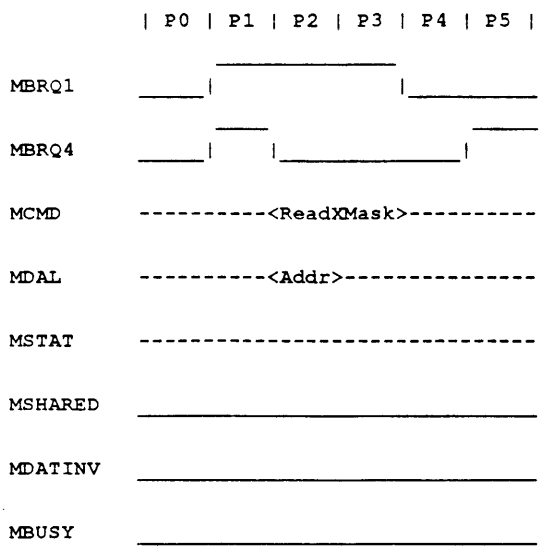


Figure 4-14: I/O-Read Transaction with No Slave Response

P0 The M-bus is idle.

P1 The modules in slots 1 and 4 arbitrate for the M-bus.

P2 Slot 1 wins the arbitration and asserts its MBRQ signal to confirm it won the M-bus. It also drives MCMD with READ and MDAL with the I/O address.

P3 Modules decode the I/O address. Slot 1 specifies the byte mask on MCMD.

P4 The address referenced non-existent I/O, so no slaves responded. The M-bus master's M-bus interface should indicate an error to its internal logic. This is the last cycle of the transaction.

P5 Slot 4 rearbitrates for its pending transaction. This is the first possible cycle for a new transaction.

4.6.13. I/O Write

Figure 4-15 shows an I/O write.

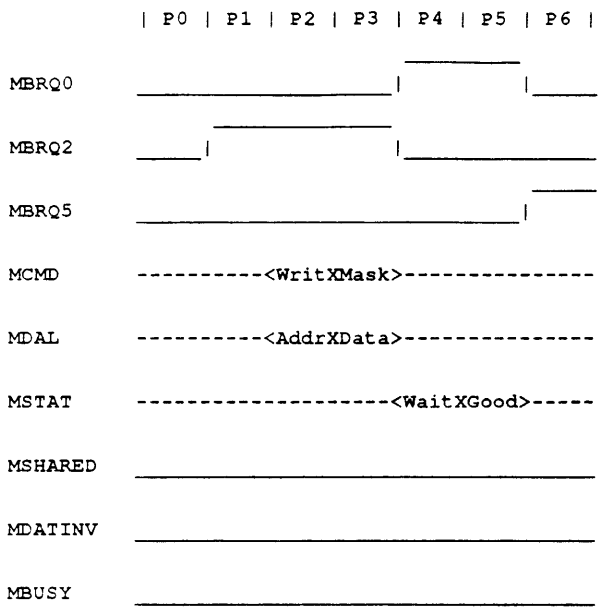


Figure 4-15: I/O-Write Transaction

P0 The M-bus is idle.

P1 The module in slot 2 arbitrates for the M-bus.

P2 Slot 2 asserts its MBRQ signal to confirm it won the M-bus. It also drives MCMD with WRITE and MDAL with the I/O address.

P3 Modules decode the I/O address. Slot 2 specifies the byte mask on MCMD and supplies data on MDAL.

P4 Slot 0 asserts its MBRQ to indicate it is processing the I/O write, but it specifies WAIT on MSTAT, since the write has not been completed.

P5 Slot 0 continues to assert its MBRQ and specifies GOOD on MSTAT to complete the transaction.

P6 Slot 5 rearbitrates for a pending transaction. This is the first possible cycle for a new transaction.

4.6.14. I/O Write with No Slave Response

Figure 4-16 shows an I/O write with no slave response.

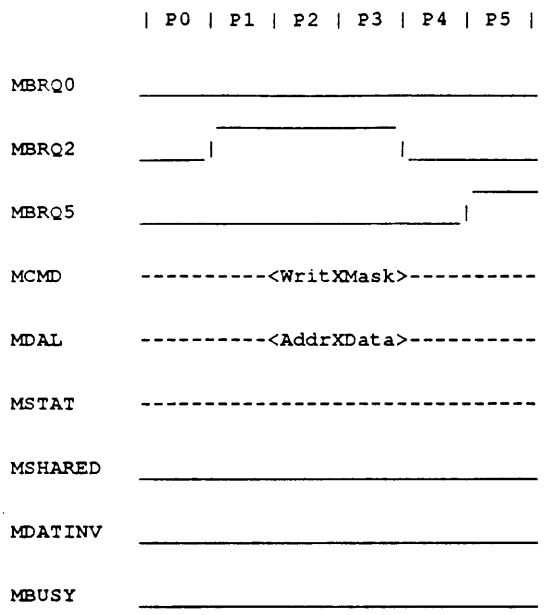


Figure 4-16: I/O Write Transaction with No Slave Response

P0 The M-bus is idle.

P1 The module in slot 2 arbitrates for the M-bus.

P2 Slot 2 asserts its MBRQ signal to confirm it won the M-bus. It also drives MCMD with WRITE and MDAL with the I/O address.

P3 Modules decode the I/O address. Slot 2 specifies the byte mask on MCMD and supplies data on MDAL.

P4 No module was referenced by the I/O address, so all the MBRQ signals remain deasserted. This is the last cycle of the transaction.

P5 Slot 5 rearbitrates for a pending transaction. This is the first possible cycle for a new transaction.

4.6.15. Interrupt Acknowledge

Figure 4-17 shows an interrupt acknowledge.

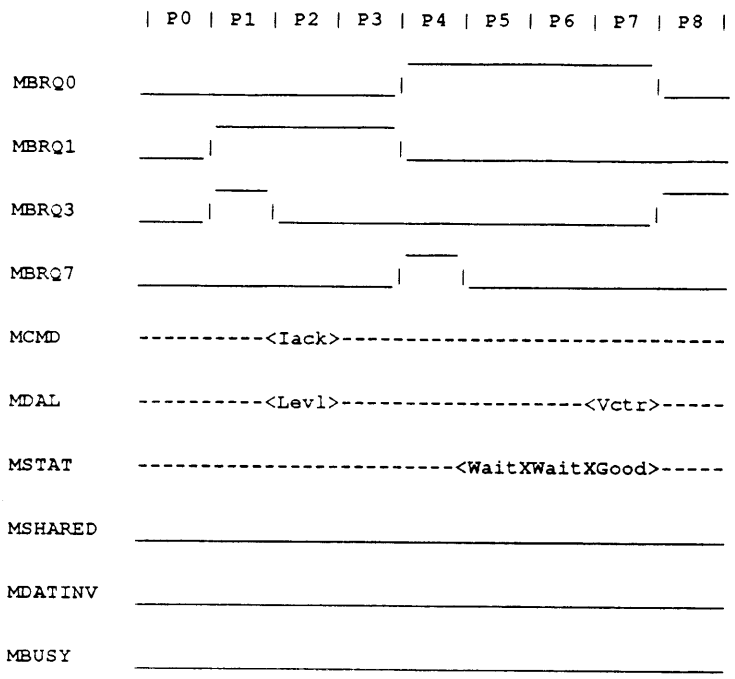


Figure 4-17: Interrupt Acknowledge Transaction

P0 The M-bus is idle.

P1 The modules in slots 1 and 3 arbitrate for the M-bus.

P2 Slot 1 wins the arbitration and asserts its MBRQ signal to confirm it won the M-bus. It also drives MCMD with INTACK and MDAL with the interrupt level.

P3 Modules check to see if they are asserting MIRQ<Level>.

P4 Slots 0 and 7 assert their MBRQ to indicate they have a pending interrupt at this level.

P5 Slot 0 asserts its MBRQ to indicate it was the highest-priority interrupter and specifies WAIT on MSTAT, while it generates an internal interrupt-acknowledge cycle.

P6 Slot 0 continues to assert MBRQ and specify WAIT on MSTAT, while its internal interrupt-acknowledge cycle proceeds.

P7 Slot 0 drives the vector onto MDAL and specifies GOOD on MSTAT to complete the transaction.

P8 Slot 3 rearbitrates for its pending transactions. This is the first possible cycle for a new transaction.

4.6.16. Interrupt Acknowledge with No Slave Response

Figure 4-18 shows an interrupt acknowledge with no slave response.

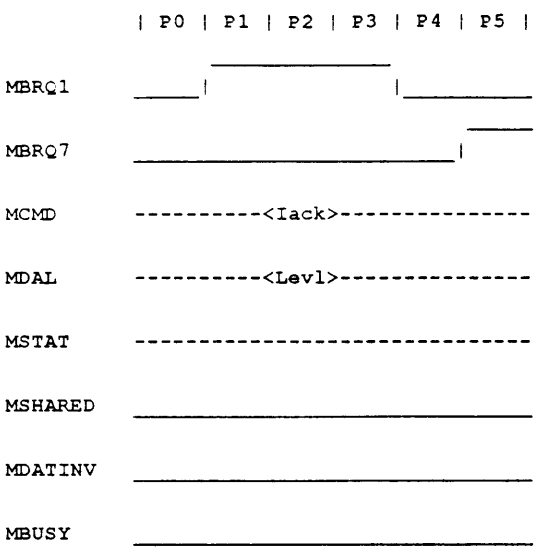


Figure 4-18: Interrupt Acknowledge Transaction with No Slave Response

P0 The M-bus is idle.

P1 The module in slot 1 arbitrates for the M-bus.

P2 Slot 1 wins the arbitration and asserts its MBRQ signal to confirm it won the M-bus. It also drives MCMD with INTACK and MDAL with the interrupt level.

P3 Modules check to see if they are asserting MIRQ<Level>.

P4 No modules believe they are asserting MIRQ<Level>. The M-bus master's M-bus interface should indicate a passive release to its internal logic. This is the last cycle of the transaction.

P5 Slot 7 rearbitrates for its pending transaction. This is the first possible cycle for a new transaction.

4.7. M-Bus Interface Registers

Every M-bus module must implement one register within its slot-assigned, 32-Mbyte region of I/O space. Table 4-27 lists the register name; the address offset from the base of the 32-Mbyte, slot-specific region, and the function of the register.

Table 4-27: Required Interface Registers for the M-Bus

Register	Offset	Function
MODTYPE	01FFFFFC	Module type

4.7.1. M-Bus MODTYPE Interface Register

Figure 4-19 shows the format of the read-only MODTYPE register, which indicates the class of the module, class-specific information, the interface-chip type, and the interface-chip revision. The class-specific information is specific to the interface chip. For example, a memory module might indicate the DRAM size and number of banks.

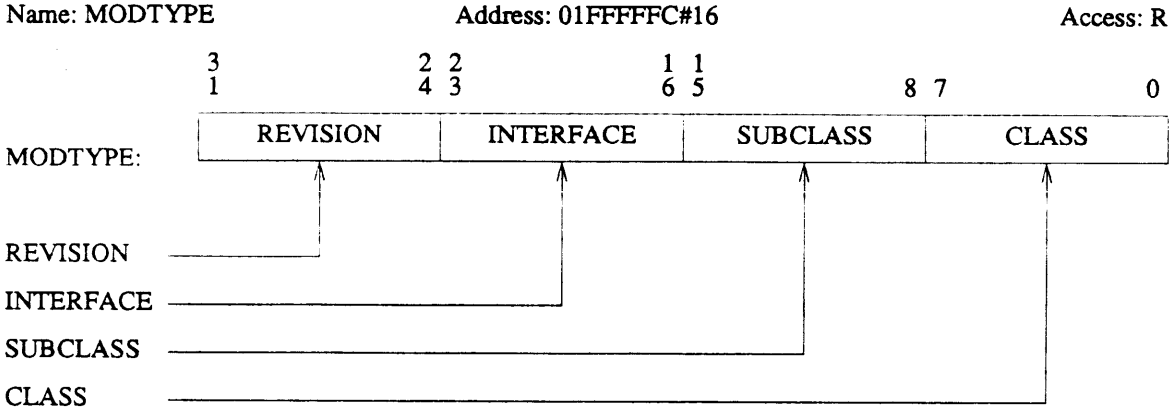


Figure 4-19: MODTYPE Register Format

To read the MODTYPE register for slot 0, issue a longword I/O read to address 91FFFFFC (VAX address 31FFFFFC); to read the MODTYPE register for slot 5, issue a longword I/O read to address 9BFFFFFC (VAX address 3BFFFFFC).

Table 4-28 lists the module classes currently defined.

Table 4-28: Defined Module Classes for the M-Bus

CLASS	Module class
01#16	Bus-adapter class
02#16	Graphics class
04#16	I/O class
08#16	CPU class
10#16	Memory class
20#16	Reserved
40#16	Reserved
80#16	Reserved

Table 4-29 lists the interface-chip types currently defined.

Table 4-29: Defined Interface Chip Types for the M-Bus

INTERFACE	Interface chip
0	Reserved
1	Firefox Bus Interface Chip (FBIC)
2	Firefox Memory Data Path and Control (FMDC)
3	PixelStamp
4	Rigel processor
5	uPrism processor
6..255	Reserved

4.8. Initialization

M-bus initialization is coordinated by the MRESET, MPOK, and MDCOK signals. When the MRESET signal is asserted, the state of the entire Firefox workstation is (re)initialized. When the MPOK signal is asserted, the AC input to the power supplies is within specification. Processor modules may use the MPOK signal as a power-fail interrupt to initiate power-fail processing. When the MDCOK signal is asserted, the DC output of the power supplies is within specification. Modules with nonvolatile storage may use the MDCOK signal to freeze the state of their storage device.

In the following figures, MRESET is shown with its backplane active-low assertion state.

4.8.1. Powerup

When a workstation powers-up, the power supplies assert MDCOK when their DC output is within specification. Then the power supplies assert MPOK when their AC input is within specification. The MRESET signal, which the Firefox Workstation I/O Module generates in most configurations, is held asserted for approximately 70 milliseconds after DC power is available to allow the M-bus clock generator and module internal logic to stabilize. Figure 4-20 illustrates this sequence.

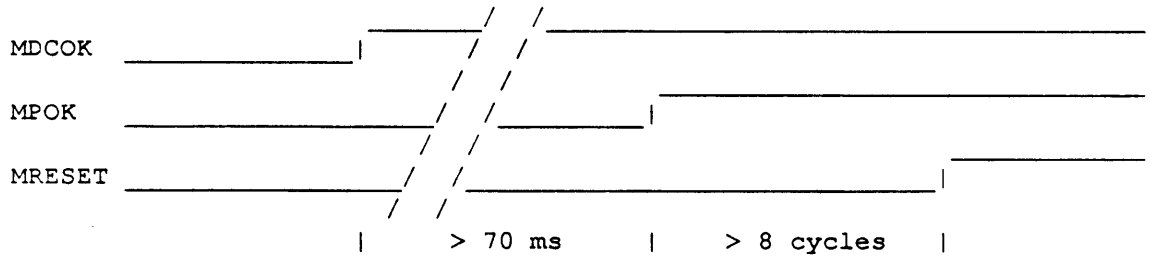


Figure 4-20: Powerup Sequence

4.8.2. Powerdown

When AC input to the power supplies falls out of specification, they deassert MPOK. If the power supplies can no longer maintain their DC outputs within specification, they deassert MDCOK. The power supplies continue to supply DC power within specification for at least 4 milliseconds after AC power is lost. In response to the loss of MDCOK, MRESET is asserted. Figure 4-21 illustrates this sequence.

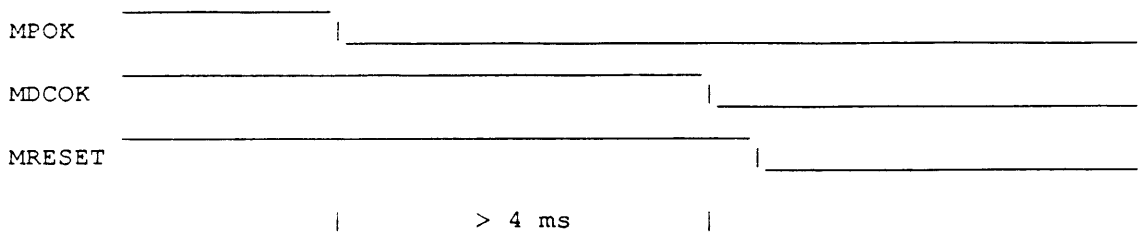


Figure 4-21: Power-down Sequence

4.8.3. Workstation Reset

When the MRESET signal is asserted, all modules initialize their internal logic. In addition to powerup and powerdown events, modules may implement logic to assert MRESET either from a switch or by writing to a control register. Figure 4-22 illustrates this sequence.

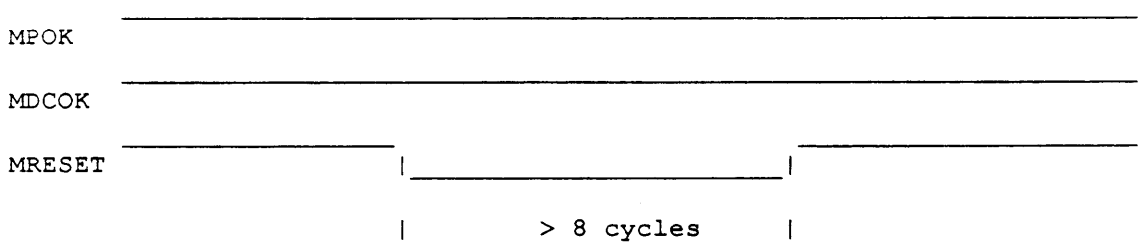


Figure 4-22: Workstation-Reset Sequence

4.9. Electrical

All M-bus modules must use the same type, number, and physical placement of M-bus transceivers/drivers.

4.9.1. M-Bus Transceivers/Drivers and Input Loads

Table 4-30 lists the mandatory M-bus transceiver/driver components for M-bus modules. All transceiver/driver components must be in SOIC packages. When termination is specified, a series resistor is required immediately after the M-bus transceiver/driver output. Series resistors must be discrete, surface-mounted resistors of at least 0.125-watt rating. When pull-up is specified, a pull-up resistor to +5.0 volts is present on the backplane. Backplane resistors must be discrete and of 0.250-watt rating, at least.

Table 4-30: M-Bus Module Transceiver/Driver Components

Component	Termination	Pull-up	Signal(s)
74F244	20 ohm	4.7K ohm	MBRQ
74F245	20 ohm	4.7K ohm	MDAL<31:24>
74F245	20 ohm	4.7K ohm	MDAL<23:16>
74F245	20 ohm	4.7K ohm	MDAL<15:08>
74F245	20 ohm	4.7K ohm	MDAL<07:00>
74F245	20 ohm	4.7K ohm	MDPAR
74F245	20 ohm	4.7K ohm	MCMD<3:0>,MCPAR
74F245	20 ohm	4.7K ohm	MSTAT<1:0>, MSPAR
74AS760		143/768 ohm	MSHARED, MDATINV, MBUSY, MABORT
74AS760		1.5K ohm	MIRQ<3:0>

Table 4-31 lists the mandatory M-bus-driver components for the M-bus backplane. When termination is specified, a series resistor is required immediately after the M-bus transceiver/driver output. Series resistors must be discrete of 0.125-watt rating, at least. The MRESET and MCLKI signals may be driven by an M-bus module in some configurations, in which case the indicated driver must be used on that module.

Table 4-31: M-Bus Backplane Driver Components

Component	Termination	Pull-up	Signal(s)
74F244	10 ohm		MCLKA, MCLKB
74AS760		143/768 ohm	MRESET, MCLKI
74XXXX		180/390 ohm	MPOK, MDCOK, MHALT, MRUN

Table 4-32 lists the allowed per-module loading for each M-bus signal. Loading is in addition to the output driver, if appropriate. For example, the MABORT signal has one 74F244 output driver and two CMOS input loads for each M-bus module.

Table 4-32: Allowed Input Loading Per Module for the M-Bus

Loading	Signal(s)
1(2) CMOS INPUTS	MBRQ, MBUSY
1 74F245 TRANSCEIVER	MDAL, MDPAR
1 74F245 TRANSCEIVER	MCMD, MCPAR
1 74F245 TRANSCEIVER	MSTAT, MSPAR
1(2) CMOS INPUTS	MSHARED, MDATINV, MABORT
1(2) CMOS INPUTS	MRESET
1(2) CMOS INPUTS	MPOK
1(2) CMOS INPUTS	MDCOK
NONE	MRUN
1(2) CMOS INPUTS	MIRQ
1(2) CMOS INPUTS	MHALT
2 CMOS INPUTS	MCLKA, MCLKB
1(2) CMOS INPUTS	MCLKI

Signals with a loading of 1(2) indicate that dual processor modules are allowed two loads on those signals, whereas all other modules may only have one load on those signals. The M-bus clocks must have two loads on all modules to minimize clock skew between backplane slots with dual processor modules and backplane slots with other module types.

4.9.2. M-Bus Driver/Receiver DC Characteristics

Table 4-33 lists the DC characteristics for the various driver/receiver classes. All input and output voltages are in volts. All input, output, and leakage currents are in milliamperes. The F245 transceiver class is associated with the MDAL, MDPAR, MCMD, MCPAR, MSTAT, and MSPAR signals. The F244 driver class is associated with the MBRQ, MCLKA, and MCLKB signals. The AS760 driver class is associated with the MSHARED, MDATINV, MBUSY, MABORT, MIRQ, MRESET, MCLKI, MPOK, MDCOK, MHALT, and MRUN signals. The CMOS receiver class is associated with the MBRM, MCLKA, MCLKB, MSHARED, MDATINV, MBUSY, MABORT, MIRQ, MRESET, MCLKI, MPOK, MDCOK, and MHALT signals.

Table 4-33: M-Bus Driver/Receiver DC Characteristics

Class	Voh	Ioh	Vol	Iol	Vih	Iih	Vil	Iil	Iz
F245	2.0	-15.0	0.55	64.0	2.0	0.07	0.8	-1.0	0.05
F244	2.0	-15.0	0.55	64.0	-	-	-	-	0.05
AS760	-	0.1	0.55	64.0	-	-	-	-	-
CMOS	-	-	-	-	2.0	0.01	0.8	-0.01	-

4.9.3. M-Bus Signal Capacitance

Table 4-34 lists the allowed module capacitance for the various signal classes. All capacitance values are in picofarads. Capacitance values in parentheses are for dual processor modules.

Table 4-34: M-Bus Module Signal Capacitance

Signals	Cin	Cout	Cio
MCLKA,MCLKB	40	-	-
MBRQ	-	13	-
MBRM	17(35)	-	-
MDAL,MCMD,MSTAT,MXPAR	-	-	13
MSHARED,MDATINV,MBUSY,MABORT,MRESET,MCLKI,MHALT,MIRQ	-	-	27(45)
MPOK,MDCOK	20(40)		

4.9.4. M-Bus Timing

The M-bus AC timing is determined by four components:

- Bus interface output propagation delay
- M-bus driver/receiver propagation delay
- Bus interface input setup/hold time requirements
- M-bus clock distribution skew

Figure 4-23 shows the wave forms generated by the M-bus clock generator. The clock generator is based on a divide-by-six circuit of the master oscillator. The clock generator must be free-running and self-initializing from an arbitrary power-up state within a few oscillator cycles.

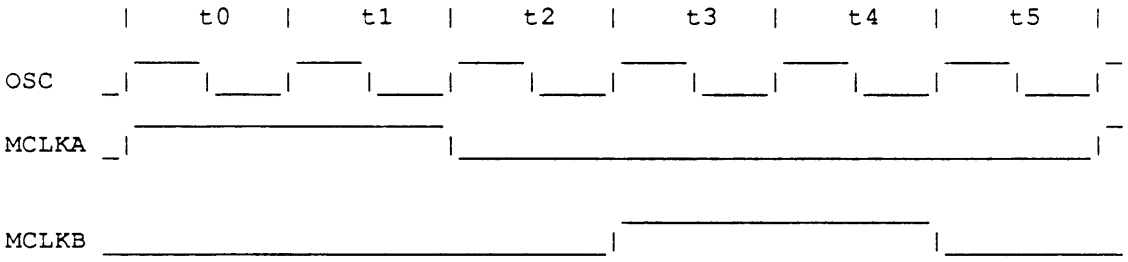


Figure 4-23: M-Bus MCLKA/MCLKB Waveforms

All M-bus output signals and bus driver control signals must be outputs of flip-flops clocked on the rising edge of MCLKA. All M-bus input signals must be latched on the falling edge of MCLKB. Table 4-35 shows the timing budget for each component of the module-to-module communication path. All times are in nanoseconds. Refer to the *Firefox M-bus Data Line Signal Integrity Study* and the *M-bus Interface Logic Clock Distribution System Signal Integrity Study* for a detailed discussion of the SPICE simulations conducted to derive the F245-driver timing and the clock-distribution-skew timing.

Table 4-35: M-Bus Module-To-Module Timing Budget

SComponent	Symbol	Minimum Time	Maximum Time
Bus interface output delay	To	0.0	TBD
F245 M-bus driver delay	Td	2.5	18.0
F245 M-bus receiver delay	Tr	2.5	7.0
Bus interface input setup time	Ts	-	TBD
Bus interface input hold time	Th	-	TBD
M-bus clock distribution skew	Tc	-	9.2

The minimum M-bus cycle time is determined by: Tcycle = (To.max + Td.max + Tr.max + Ts + Tc) * 6/5

The available input hold time is determined by: Th = (Tcycle/6) - Tc - To.min

Note that the available-input-hold-time equation does not include a term for the backplane driver/receiver propagation delay. This is necessary for dual processor modules which do not have a backplane driver/receiver in the path between the two bus interfaces.

4.9.5. Module AC Characteristics

Table 4-36 lists the AC characteristics that M-bus modules must meet for input and output responses. All timing is measured with respect to the threshold voltage of M-bus signals, Vt equal to 1.4 volts, at the backplane connector.

Table 4-36: Module AC Characteristics

Class	To	Ts	Th
Outputs to MCLKA rising	TBD	-	-
Inputs to MCLKB falling	-	TBD	TBD

4.9.6. DC Power

Each M-bus slot has sixteen +5 volt pins, three +12 volt pins, and thirty-six ground pins. Each connector pin is rated for a minimum of 1.0 amperes, resulting in a maximum of 16.0 amperes at +5 volts and 3.0 amperes at +12 volts.

4.9.7. AC Power

The M-bus has a +5 volt or ground pin for each three signal pins. The power pins for the two rows of the backplane connectors are staggered. This provides an AC ground within 100 mils of each signal pin.

M-bus modules must implement sufficient power supply decoupling to limit ripple imposed back on the backplane +5 volt plane to 25 millivolts.

4.9.7.1. Operation of M-Bus with Extended Modules

The M-bus worst-case timing margins do not allow operation of the M-bus with modules connected to the backplane via an extender module at the nominal M-bus clock period.

With typical timing margins, it should be possible to operate the M-bus at nominal M-bus clock period with one module on an extender.

SPICE simulations indicate that the M-bus backplane switching time increases by 5.0 ns with a 12.0-inch extender. Calculations indicate that clock skew is increased by 2.5 ns. This implies that the M-bus cycle time must be increased by 15.0 ns to run a worst-case system with one module on an extender.

The extender module must be a four-layer PCB with +5-volt and ground planes. Signal traces must be routed on alternate sides to minimize crosstalk. The etch length must not exceed 12.0 inches. There must only be one additional connector in the signal path.

If the M-bus cycle time is increased, memory modules may not receive adequate DRAM refresh.

Under no circumstances is extension of more than one M-bus module supported.

4.9.8. Backplane Signal Assignments

Table 4-37 shows the backplane connector-pin assignment for each of the M-bus signals. Each of the two connector blocks has a standard power and ground pattern that gives one AC ground for each three signals.

Table 4-37: M-Bus Backplane Signal Assignments

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
A01	GND	A02		B01	GND	B02	+5VBAT
A03		A04		B03	MCLKA	B04	GND
A05		A06	+5V	B05	GND	B06	+5V
A07		A08		B07	MCLKB	B08	GND
A09	GND	A10		B09	GND	B10	+12V
A11		A12		B11	+12V	B12	+12V
A13		A14	GND	B13	MBRQ	B14	GND
A15		A16		B15	MBRM0	B16	MBRM1
A17	+5V	A18		B17	+5V	B18	MBRM2
A19		A20		B19	MBRM3	B20	MBRM4
A21		A22	GND	B21	MBRM5	B22	GND
A23		A24		B23	MBRM6	B24	MSPAR
A25	GND	A26		B25	GND	B26	MSTAT0
A27		A28		B27	MSTAT1	B28	MIRQ0
A29		A30	+5V	B29	MIRQ1	B30	+5V
A31		A32		B31	MIRQ2	B32	MIRQ3
A33	GND	A34		B33	GND	B34	MCPAR
A35		A36		B35	MCMD0	B36	MCMD1
A37		A38	GND	B37	MCMD2	B38	GND
A39		A40		B39	MCMD3	B40	MRSVB40
A41	+5V	A42		B41	+5V	B42	MSLOT
A43		A44		B43	MDPAR	B44	MID0
A45		A46	GND	B45	MID1	B46	GND
A47		A48		B47	MID2	B48	MCLKI
A49	GND	A50		B49	GND	B50	GND
A51		A52		B51	MDAL0	B52	MDAL1
A53		A54	+5V	B53	MDAL2	B54	+5V
A55		A56		B55	MDAL3	B56	MDAL4
A57	GND	A58		B57	GND	B58	MDAL5
A59		A60		B59	MDAL6	B60	MDAL7
A61		A62	GND	B61	MRESET	B62	GND
A63		A64		B63	MDAL8	B64	MDAL9
A65	+5V	A66		B65	+5V	B66	MDAL10
A67		A68		B67	MDAL11	B68	MDAL12
A69		A70	GND	B69	MDAL13	B70	GND
A71		A72		B71	MDAL14	B72	MDAL15
A73	GND	A74		B73	GND	B74	MDAL16
A75		A76		B75	MDAL17	B76	MDAL18
A77		A78	+5V	B77	MDAL19	B78	+5V
A79	-12V	A80	-12V	B79	MDAL20	B80	MDAL21
A81	GND	A82	MRUN	B81	GND	B82	MDAL22
A83	MRSVA83	A84	MRSVA84	B83	MDAL23	B84	MDAL24
A85	MDATINV	A86	GND	B85	MDAL25	B86	GND
A87	MBUSY	A88	MSHARED	B87	MDAL26	B88	MDAL27
A89	+5V	A90	MABORT	B89	+5V	B90	MDAL28
A91	MHALT	A92	MDCOK	B91	MDAL29	B92	MDAL30
A93	MPOK	A94	GND	B93	MDAL31	B94	GND

The signals on the B block labeled MBRM<0:6> are the M-bus-request signals from the other slots. Table 4-38 lists the connections for the MBRQ signal from each of the slots to the other 6 slots.

Table 4-38: M-Bus MBRQ Connections per Slot

	Slot0	Slot1	Slot2	Slot3	Slot4	Slot5	Slot6	Slot7
MBRQ0	B13	B15	B15	B15	B15	B15	B15	B15
MBRQ1	B15	B13	B16	B16	B16	B16	B16	B16
MBRQ2	B16	B16	B13	B18	B18	B18	B18	B18
MBRQ3	B18	B18	B18	B13	B19	B19	B19	B19
MBRQ4	B19	B19	B19	B19	B13	B20	B20	B20
MBRQ5	B20	B20	B20	B20	B20	B13	B21	B21
MBRQ6	B21	B21	B21	B21	B21	B21	B13	B23
MBRQ7	B23	B23	B23	B23	B23	B23	B23	B13

The MRSRVD signals are bussed across all slots.

MCLKA and MCLKB are radially distributed to each slot from the M-bus clock subsystem. The M-bus clock-generator components are located on the M-bus backplane.

The unspecified signals on the A connector block are reserved and must not be connected to any internal logic of M-bus modules. However, M-bus modules must still connect the assigned power pins, as they are part of the DC power-distribution network.

The maximum stub length for the MCMD, MSTAT, MDAL, and MXPAR signals is 1.5 inches, where stub length is the amount of etch between the top of the edge finger and the IC pin. Stub length of all other M-bus signals must not exceed 3.0 inches.

4.10. Mechanical

M-bus modules use the new *L-series-quad* module format with two L-series one-piece connectors resulting in two paddles of 94 edge-fingers on 100-mil centers.