MVME6000UM/D1

MVME6000
VMEbus Interface
User's Manual

MVME6000UM/D1

July 1989

MVME6000
VMEbus INTERFACE
USER'S MANUAL

First Edition

TABLE OF CONTENTS

CHAPTER 4 LOCAL CONTROL AND STATUS REGISTER SET

CHAPTER 5 GLOBAL CONTROL AND STATUS REGISTER SET

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 GENERAL

The MVME6000 is the interface between MC68020 and MC68030 microprocessors and the VMEbus. With external buffers, decoders and oscillators, it provides all of the functions that a CPU board requires to connect to the VMEbus, allowing a significant savings in board space. Using all of the chip's modules, a board can operate as the VMEbus system controller and provide a full master/slave interface. In addition to the VMEbus defined function, the chip includes functions to support some multiprocessor architectures.

1.2 SUMMARY OF MAJOR FEATURES

- **VMEbus Master interface:**
  - MC68020/MC68030 local CPU interface
  - 16-bit or 32-bit local data width
  - 16-bit or 32-bit VMEbus data width
  - 16-bit, 24-bit, or 32-bit VMEbus address width
  - Software programmed AM code
  - Software configured VMEbus access watch-dog timer
  - Software enabled 4-byte cache fill mode
  - Software enabled write-posting mode

- **VMEbus Slave interface:**
  - MC68020/MC68030 local CPU interface
  - 16-bit or 32-bit local data width
  - 16-bit or 32-bit VMEbus data width
  - 16-bit, 24-bit or 32-bit VMEbus address width
  - Software programmed AM code
  - Software enabled write-posting mode

- **VMEbus Requester:**
  - Software configured RWD/ROR/RNEVER release modes
  - Software configured FAIR/RODWB/ROSELECT request modes
  - Software configured BR0* - BR3* request level

- **VMEbus Interrupter:**
  - Software configured IRQ1* - IRQ7* interrupt request level
  - 8-bit Software programmed status/ID register

- **VMEbus interrupt handler:**
  - Can handle all 7 VMEbus interrupts
  - Software programmed individual masks
  - MC68020/MC68030 local bus interface

INTRODUCTION

- **VMEbus System Controller:**
  - Software configured PRI/RRS/SGL Arbiter
  - Software configured Arbitration Timer
  - IACK Daisy-Chain Driver
  - SYSCLK Driver
  - Software configured Bus Timer
  - SYSRESET* Driver

- **Global Register Set:**
  - Four location monitors
  - Global control of locally detected failures
  - Global control of local reset
  - Two global attention interrupt bits
  - An 8-bit Board ID register
  - Five 8-bit dual-ported general purpose communication registers
  - A 4-bit chip ID number

- **Utility Interrupt Handler:**
  - Individually masked interrupts:
    - After completing a VMEbus IACK cycle
    - When SYSFAIL* is asserted
    - When BERR* is received during a Master write-posted cycle
    - When Location Monitor 0 or 1 detect a VMEbus cycle
    - When either of the two attention control bits is set
  - Unique interrupt vectors for all the above
  - MC68020/MC68030 local bus interface

- A software configured local bus timer

- With the exception of ACFAIL*, A08-A31 and D08-D31, directly connects to all VMEbus signal lines.

1.3 TERMINOLOGY

The terms *asserted* and *negated* are used extensively in this document to avoid confusion when dealing with a mixture of active-low and active-high signals. The term asserted is used to indicate that a signal is in its active, or true state, regardless of its voltage level. The term negated is used to indicate that a signal is in its inactive, or false, state.

To help define their use, some signal names have an asterisk (*) suffix. When an asterisk follows the name of a level significant signal it denotes that the signal is valid when it is at a low voltage level. When an asterisk follows the name of a signal that is edge significant it means that the actions initiated by that signal occur when it makes a transition from a high to a low level voltage.

The terms *control bit* and *status bit* are used extensively in this document. *Control bit* is used to describe a bit in a register which can be set and cleared under software control, affecting the operation of the MVME6000. *Status bit* is used to describe a bit in a register which can be read by software to determine operational or exception conditions. The term *true* is used to indicate that a bit is in the state that enables the function it controls. The term *false* is used to indicate that the bit is in the state that disables the function it controls. In all tables, the terms *0* and *1* are used to describe the actual value that should be written to the bit, or the value that it will yield when read.

All numbers given in this document are either decimal or hexadecimal. Hexadecimal numbers are are prefixed by '$'. For example, '12' is the decimal number twelve, and '$12' is the decimal number eighteen.

1.4 RELATED DOCUMENTS

The following publications are applicable to the MVME6000 and may provide additional helpful information. If not shipped with this product, they may be purchased from the Motorola Literature Distribution Center, 616 West 24th Street, Tempe, Arizona 85282; telephone (602) 994-6561.

DOCUMENT TITLE	MOTOROLA PART NUMBER
MC68020 32-Bit Microprocessor User's Manual	MC68020UM/AD
MC68030 Enhanced 32-Bit Microprocessor User's Manual	MC68030UM/AD

The following publication is available from the source indicated:

Versatile Backplane Bus: VMEbus, part number IEEE 1014-87: The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017

INTRODUCTION

1

CHAPTER 2
FUNCTIONAL BLOCKS

2.1 GENERAL INFORMATION

The MVME6000 is composed of nine functional blocks. These are master, slave, requester, arbiter, interrupter, interrupt handler, global and local timers, the local control and status register set, and the global control and status register set. Figure 2-1 shows the block diagram of the chip.

2.2 MASTER

The master provides the signals necessary to interface a microprocessor to the VMEbus. With the exception of ACFAIL*, data lines D08-D31, and address lines A08-A31, the chip directly connects to all signals that a VMEbus master can drive and monitor. On the local side, the master is designed to easily interface to a 68020 or a 68030 microprocessor. The chip also provides signals to control the external address and data buffers/latches and to automatically enable data swapping buffers when required.

Using input signal lines and register bits, the chip can be configured to interface to either a 16-bit or a 32-bit local data bus. On the VME side, the chip can be configured to provide the following capabilities:

Addressing capabilities: A16, A24, A32
Data transfer capabilities: D08, D16, D32, UAT

The master partially supports the dynamic bus sizing protocols of the 68020/68030 microprocessor. When the microprocessor initiates a quad-byte write cycle to a VMEbus slave that only has the D16 capability, the chip executes a double-byte write cycle on the VMEbus. It will then acknowledge the microprocessor as required for a double-byte slave. The microprocessor can then execute additional cycles as required.

Using the Address-Modifier-Out control register (refer to Chapter 3), the address modifier (AM) code that the master places on the VMEbus can be programmed under software control. Alternately, the AM code reflects the state of the three function code lines driven by the 68020 or 68030 microprocessor.

Included is a software controlled access watch-dog timer. It starts when the chip is requested to do a data transfer or an interrupt acknowledge cycle. The timer stops once the chip has started the data transfer of the relevant VMEbus cycle or if the cycle has been write-posted and the chip has acknowledged the processor. If neither of these conditions happen, the timer will time-out, driving the processor's bus error signal, and setting the appropriate status bit in the Local Control and Status Register (LCSR). Using control bits, the timer can be disabled or it can be enabled to drive the processor's bus error signal after 64 usecs, 1 msec, or 32 msecs.

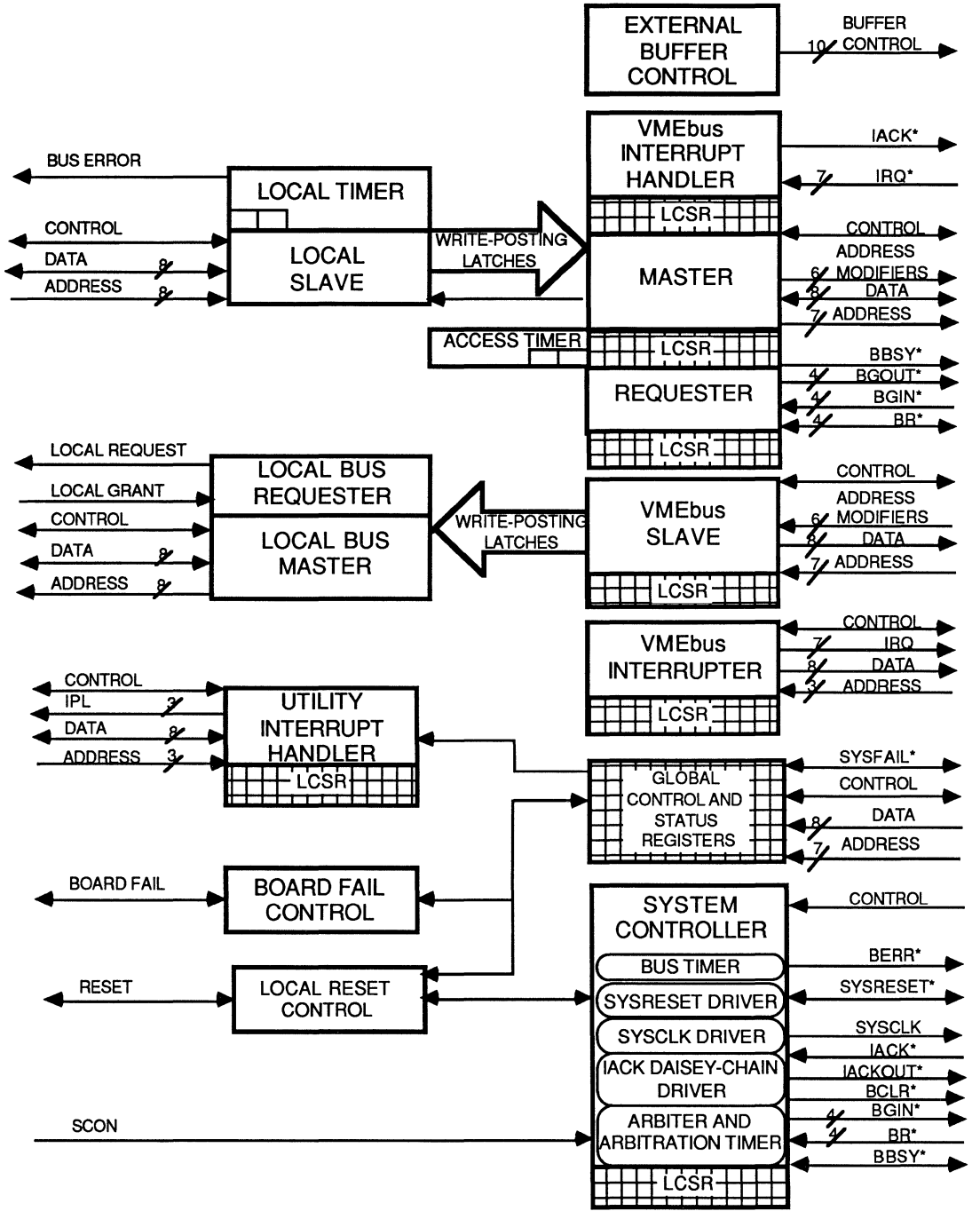


Figure 2-1. Block Diagram of the MVME6000

In addition, software can configure the master to operate in the following modes:

(a) Write Posting - i.e., control the latching of outgoing data and addressing information, provide an early acknowledge to the local microprocessor, and complete the VMEbus write cycle on its own.

(b) Cache Fill - i.e., execute a 4-byte VMEbus read cycle regardless of the amount of data that the local microprocessor requests during a read cycle that accesses a D32 VMEbus Slave.

2.3 SLAVE

The slave provides all the signals necessary to allow a VMEbus Master access to on-board resources. With the exception of data lines D08-D31 and address lines A08-A31 the MVME6000 directly connects to all signals that a VMEbus slave can drive and monitor.

Adhering to the IEEE 1014-87 VMEbus Specification, the slave module can withstand address only cycles, as well as address pipelining. Using input signal lines and control register bits, it can be configured to provide the following VMEbus capabilities:

Addressing capabilities: A16, A24, A32
Data transfer capabilities: D08, D16, D32, UAT

Using the Address-Modifier-In control register (refer to Chapter 3), the slave can be configured under software control to respond to any of the defined AM codes. In addition, the slave can be configured under software control to perform Write-Posting operations. When in this mode the chip will control the latching of incoming data and addressing information, request control of the local bus and provide an early DTACK* during a VMEbus write cycle that accesses an on-board resource. The chip will then complete the local access after it has been granted the local bus.

2.4 REQUESTER

The requester provides all the signals necessary to allow the local processor to request and be granted use of the VMEbus. The MVME6000 directly connects to all signals that a VMEbus requester can drive and monitor.

Requiring no external jumpers, the MVME6000 provides the means for software to program the MVME6000 to request the bus on any one of the four bus request levels, automatically establishing the bus grant daisy-chains for the three inactive levels.

The requester will request the bus if any of the following conditions occur:

(a) The local CPU initiates either a data transfer cycle or an interrupt acknowledge cycle to the VMEbus.

(b) The MVME6000 is requested to acquire control of the VMEbus as defined by either an input signal, or a control bit.

Software controls when the requester releases the bus as follows:

(a) Release When Done (RWD) - i.e., after acquiring control of the bus, release it immediately after completing the bus cycle.

(b) Release-On-Request (ROR) - i.e., after acquiring control of the bus, maintain control and monitor all the bus request levels. The requester will then relinquish control of the bus only if a bus request is pending on any of the request lines.

(c) Release Never (RNEVER) - i.e., after acquiring control of the bus, maintain control until the RNEVER bit is cleared.

To minimize the timing overhead of the arbitration process, the requester in the MVME6000 releases the bus early. If it is about to release the bus and it is executing a VMEbus cycle, the requester releases BBSY* before its associated master completes the cycle. This allows the arbiter to arbitrate any pending requests, and grant the bus to the next requester, at the same time that the active master completes its cycle.

2.5 INTERRUPTER

The interrupter provides all the signals necessary to allow on-board logic to request interrupt service from a VMEbus Interrupt Handler. The MVME6000 directly connects to all signals that a VMEbus interrupter can drive and monitor.

The MVME6000 requires no external jumpers and permits the software to program the MVME6000 to request an interrupt on any of the seven interrupt request lines. The chip also controls the propagation of the acknowledge on the IACK daisy-chain.

The Interrupter in the MVME6000 operates in the Release-On-Acknowledge (ROAK) mode. An 8-bit control register provides software with the means to provide a dynamically programmed status/ID information. Upon reset, this register is initialized to a status/ID of $0F (the uninitialized vector in the 68K based environment).

2.6 INTERRUPT HANDLER

The interrupt handler provides all the signals necessary to allow on-board logic to identify and handle VMEbus as well as internally generated utility interrupt requests. The MVME6000 directly connects to all signals that a VMEbus handler can drive and monitor. On the local bus, the interrupt handler is designed to comply with the interrupt handling signaling protocol of the 68K family of microprocessors.

Requiring no external jumpers, the MVME6000 provides the means for software to program the interrupt handler to handle any combination of the seven VMEbus request lines. In addition, the chip provides individual interrupt mask bits which allow software to mask any of the VMEbus or the internally generated utility interrupts.

The MVME6000 can be programmed to request an interrupt from the local processor when any of the following conditions occur:

a. A VMEbus interrupt request line is asserted.

b. The MVME6000 has completed a VMEbus interrupt-acknowledge cycle.

c. The SYSFAIL* line is asserted.

d. BERR* is asserted during a VMEbus write-posted operation.

e. A VMEbus access was detected to the short I/O address that location monitor 0 or 1 were assigned to monitor.

f. Another VMEbus master set one of the two global signaling bits in the global register set.

A unique vector is provided by the MVME6000 for all the utility interrupts (cases b through f above). This vector is comprised of a 3-bit pre-assigned source identification code and a 5-bit software programmable base.

2.7 SYSTEM CONTROLLER

With the exception of the optional SERCLK Driver and the Power Monitor, the MVME6000 includes all the functions that a VMEbus System Controller can provide. The System Controller is enabled/disabled with the aid of an external jumper. Figure 2-1 includes a block diagram of the modules that form the system controller (the dotted lines denote signals which are internal to the MVME6000). All of the System Controller's VMEbus signals can be directly connected to the VMEbus.

2.7.1 Arbiter

The arbitration algorithm used by the MVME6000's arbiter is selected by software. All three arbitration modes defined in the VMEbus Specification are supported: Priority (PRI), Round-Robin-Select (RRS), as well as Single (SGL). When operating in the PRI mode, the arbiter asserts the BCLR* line whenever it detects a request for the bus whose level is higher than the one being serviced.

The MVME6000 includes an arbitration timer, preventing a bus lock-up when no requester assumes control of the bus after the arbiter has issued a grant. Using a control bit, this timer can be enable or disabled. When enabled, it assumes control of the bus by driving the BBSY* signal after 256 usecs, releasing it after satisfying the requirements of the VMEbus specification, and then re-arbitrating any pending bus requests.

2.7.2 IACK Daisy-Chain Driver

Complying with the latest revision of the VMEbus Specification, the System Controller in the MVME6000 includes an IACK Daisy-Chain Driver, ensuring that the timing requirements of the IACK daisy-chain are satisfied.

2.7.3 SYSCLK Driver

The MVME6000 includes a SYSCLK Driver that provides the required 16MHz system clock signal, and that can be directly connected to the VMEbus SYSCLK signal line. The chip generates the system clock signal using an external 16MHz oscillator.

2.7.4 Bus Timer

The Bus Timer in the MVME6000 is enabled/disabled by software to terminate a VMEbus cycle by asserting BERR* if any of the VMEbus data strobes is maintained in its asserted state for longer than the programmed time-out period, which can be set to 64, 128, or 256 usecs.

In addition to the VMEbus timer, the MVME6000 contains a local bus timer. This timer asserts the local PBERR* when the local data strobe is maintained in its asserted state for longer than the programmed time-out period. This timer too can be enabled or disabled under software control. The time out period can be programmed for 64, 128, or 256 usecs.

2.7.5 Reset Driver

The MVME6000 includes both a global and a local reset driver. When the MVME6000 operates as the VMEbus System Controller, the reset driver provides a global system reset by asserting the VMEbus signal SYSRESET*. This is controlled by either an input signal, or with the aid of a control bit, allowing both hardware and software to initiate a global reset sequence. SYSRESET* will remain asserted for at least 200 mseconds, as required by the VMEbus Specification.

Similarly, the MVME6000 provides an input signal and a control bit to initiate a local reset operation. By setting a control bit, executive software can maintain a board in a reset state, disabling a faulty board from participating in normal system operation. The local reset driver is enabled even when the MVME6000 is not the System Controller.

CHAPTER 3

SIGNAL LINE DESCRIPTION

3.1 GENERAL

In addition to the power and ground pins, the MVME6000 provides 120 active signals as shown in Figure 3-1. They are divided into three groups: VMEbus signals, buffer control signals, and onboard signals.

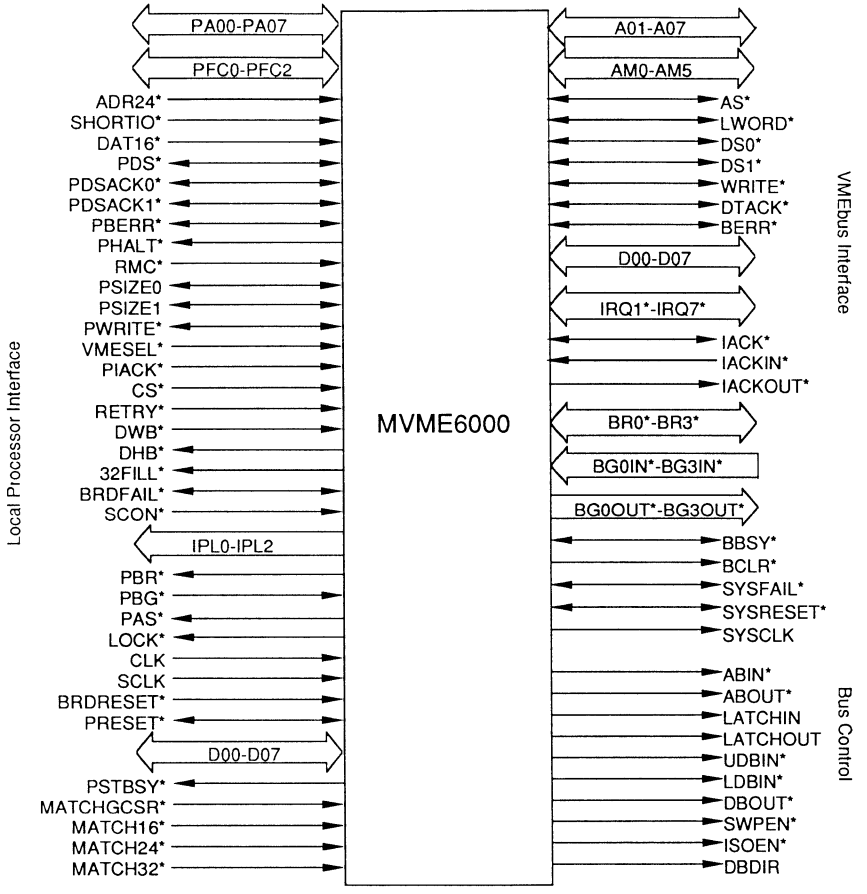


Figure 3-1. Signals of the MVME6000

SIGNAL LINES

3.2 VMEbus SIGNALS

With the exclusion of A08-A31, D08-D31 and ACFAIL*, the MVME6000 directly drives and monitors all VMEbus signals. A detailed description of the VMEbus signal lines can be found in the VMEbus Specification, and is not repeated here.

3

3.3 LOCAL INTERFACE SIGNALS

This chapter describes the signals that the MVME6000 uses to interface to on-board logic.

3.3.1 32FILL* - 4-Byte Cache Fill

Type: Totem-Pole Output

32FILL* is asserted by the MVME6000's VMEbus master whenever it is executing a VMEbus BYTE(0-3) read cycle.

The MVME6000 allows software to configure the master to attempt to read 4 bytes of data from a VMEbus slave, regardless of the number of bytes requested by the local processor. This function better supports cache architectures. This is done by setting the CFILL control bit to one as described in paragraph 4.5. As shown in Table 3-3, the chip will execute a 4-byte read cycle regardless of the requested size only if its master is configured to do a 32-bit transfer as defined by the MASD16 control bit (refer to paragraph 4.5) and the DAT16* signal line (refer to paragraph 3.3.6), and if the CFILL control bit is set to 1.

3.3.2 ADR24* - Standard Address Space

Type: Input

ADR24* is driven by on-board logic (e.g., an address decoder) to configure the address modifier code that the MVME6000's VMEbus master drives during a VMEbus cycle. The line is asserted to prohibit, and negated to enable the master to drive an extended (A32) address modifier code. However, the AM code that the master actually drives during the VMEbus cycle is determined in one of two ways:

1. Dynamically determined from the levels of the ADR24* and the SHORTIO* signal lines (refer to paragraph 3.3.35) in conjunction with the MASA24 and MASA16 control bits (refer to paragraph 4.5 and Table 4-6).

2. From the contents of the Master Address Modifier register as described in paragraph 4.9.

3.3.3 BRDFAIL* - Board Failed

Type: Input; Open-Collector Output

BRDFAIL* is driven by the MVME6000 whenever the BRDFAIL bit in the System Controller Configuration register is set to 1 (refer to paragraph 4.3).

BRDFAIL* can be asserted by other devices on the board to signal a failure. The global status bit BRDFAIL reflects the state of the BRDFAIL* signal line regardless of who drives it.

3.3.4 BRDRESET* - Board Reset

Type: Input

BRDRESET* is asserted by on-board logic, and causes the MVME6000 to assert PRESET* (refer to paragraph 3.3.28) and, if the chip is the system controller, assert the VMEbus SYSRESET* signal. When asserting SYSRESET* the MVME6000 guarantees that it remains asserted for a minimum of 200 milliseconds. Figure 3-3 is an example of connecting the BRDRESET* and PRESET* signals to on-board logic.

3.3.5 CLK - Clock

Type: Input

The CLK input is intended to connect to an external TTL clock source. It is used to generate the timing signal for the internal state machines that control and synchronize the operation of the MVME6000.

3.3.6 CS* - Chip Select

Type: Input

As described in Chapters 4 and 5, the MVME6000 includes a Local Control and Status Register set (LCSR) and a Global Control and Status Register set (GCSR). Both are accessible to the local processor. In conjunction with the PDS* signal line (refer to paragraph 3.3.23), the CS* input signal line, is used to access the GCSR and the LCSR. The specific register is then selected using address lines PA00-PA07 at the offsets defined in Chapters 4 and 5.

3.3.7 DAT16* - Data 16-Bit

Type: Input

DAT16* is driven by on-board logic (e.g., an address decoder) to allow dynamic configuration of the chip's VMEbus master to include the D32 data transfer capability. The line is asserted to prohibit, and negated to enable the master to execute a BYTE(1-2), BYTE(0-2), BYTE(1-3), and a BYTE(0-3) VMEbus cycle. However, the master will actually execute such a cycle only if it has been enabled to do so as defined by the MASD16 and the MASUAT control bits (refer to paragraph 4.5, and Tables 3-3 and 3-4).

3.3.8 DHB* - Device Has Bus

Type: Totem-Pole Output

DHB* is driven by the MVME6000 in response to the DWB* input. When DWB* is asserted and the MVME6000 has VMEbus mastership it asserts DHB*. DHB* remains asserted until DWB* is negated.

3.3.9 DWB* - Device Wants Bus

Type: Input

DWB* can be used by on-board logic to request the MVME6000 to acquire and maintain VMEbus mastership. Asserting DWB* causes the chip to request the bus (if it does not already own it). VMEbus mastership is then maintained DWB* is negated. After negating DWB* the MVME6000 might either release or maintain VMEbus mastership in accordance with the configuration of the release modes as described in paragraph 4.4.

3.3.10 HALT* - Halt

Type: Input; Open-Collector Output

NOTE

HALT* is designed to be directly connected to the HALT* pin on the MPU (68020/68030).

3

In conjunction with PBERR* (refer to paragraph 3.3.19), HALT* is asserted by the MVME6000 to indicate that the local processor should retry the current cycle. A retry sequence is initiated in response to either of the following two events:

1. On-board logic asserts RETRY* (refer to paragraph 3.3.32) while the chip's VMEbus master is executing a VMEbus cycle.

2. A dual-port lockup condition is detected. The MVME6000 determines a lockup condition when a VMEbus master is attempting to access local resources through the MVME6000's VMEbus slave at the same time that the local processor is attempting to access VMEbus resources using the chip's VMEbus master.

NOTE

If the 020 bit is set and the RMC* signal is asserted during either of the two cases, the MVME6000 initiates a bus error, not a retry.

3.3.11 IPL0*-IPL2* - Interrupt Priority Level 0-2

Type: Totem-Pole Output

The MVME6000 drives IPL0*-IPL2* to interrupt the local processor. The encoding of the three IPL lines, as well as the interrupt source for each level are shown in Table 3-1. Further information about the utility interrupts can be found in paragraphs 4.11 and 4.12. IPL0*-IPL2* normally are not connected directly to the local MPU if devices other than the MVME6000 can interrupt the local MPU.

Table 3-1. Encoding of IPL0*-IPL2*

IPL2*	IPL1*	IPL0*	Interrupt Level	VMEbus Interrupt	Utility Interrupt
H	H	H	----	----	----
H	H	L	1	IRQ1*	Signal low
H	L	H	2	IRQ2*	Location monitor 0
H	L	L	3	IRQ3*	VMEbus IACK
L	H	H	4	IRQ4*	Location monitor 1
L	H	L	5	IRQ5*	Signal high
L	L	H	6	IRQ6*	SYSFAIL*
L	L	L	7	IRQ7*	Write-post BERR*

3.3.12 LOCK* - Lock

Type: Totem-Pole Output

LOCK* is used to signal on-board logic that the MVME6000's slave is in the process of responding to a VMEbus cycle. The MVME6000 asserts the LOCK* signal when it has been granted control of the local bus, as described in paragraphs 3.3.21 and 3.3.22. It then maintains LOCK* in its asserted state, negating it after the VMEbus AS* is negated.

The intended use of the LOCK* line is to prevent the local processor from accessing local resources when the MVME6000 temporarily releases control of the local bus between the read and write portions of a read-modify-write cycle being performed by a VMEbus master on local resources.

CAUTION

IF LOCK* IS USED IN THE MANNER DESCRIBED ABOVE, IT IS THE RESPONSIBILITY OF ON-BOARD LOGIC TO GUARANTEE THE INDIVISIBILITY OF THE ACCESS TO MEMORY, I.E., THAT LOCAL MEMORY IS NOT CHANGED BETWEEN THE READ AND WRITE PORTIONS OF AN INDIVISIBLE CYCLE.

3.3.13 MATCH16* - Match A16 Access

Type: Input

MATCH16* is used by the MVME6000's VMEbus slave to determine whether the address carried on address lines A01-A15 matches the address of an on-board resource. The line is asserted by an external address decoder when a match is detected. However, the chip's VMEbus slave will respond only if it is configured to respond to cycles in the short address range (refer to paragraph 4.8), and if the cycle is indeed selecting byte locations in the short address range as determined from the AM code carried on AM0-AM5. Figure 3-4 shows how the MATCH16* signal might be used.

3.3.14 MATCH24* - Match A24 Access

Type: Input

The MATCH24* input is used by the MVME6000's VMEbus slave to determine whether the address carried on address lines A01-A23 matches the address of on-board resources. The line is asserted by an external address decoder when a match is detected. However, the slave will respond only if it is configured to respond to cycles in the standard address range (refer to paragraph 4.8), and if the cycle is indeed selecting byte location in the standard address range as determined from the AM code carried on AM0-AM5. Figure 3-4 shows how the MATCH24* input signal might be used.

3.3.15 MATCH32* - Match A32 Access

Type: Input

The MATCH32* input is used by the MVME6000's VMEbus slave to determine whether the address carried on address lines A01-A31 matches the address of on-board resources. The line is asserted by an external address decoder when a match is detected. However, the slave will respond only if it has been configured to respond to cycles in the extended address range (refer to paragraph 4.8), and if the cycle is indeed selecting byte location in the extended address range as determined from the AM code carried on AM0-AM5. Figure 3-4 shows how the MATCH32* signal might be used.

3.3.16 MATCHGCSR* - Match GCSR Access

Type: Input

The MATCHGCSR* input is used by the MVME6000 to determine whether the short address carried on address lines A08-A15 matches the base address of the chip's Global Control and Status Register (GCSR) set. The line is asserted by an external address decoder when a match is detected. Note that the external decoder should not assert MATCH16* or MATCH24* or MATCH32* at the same time it asserts MATCHGCSR*. Figure 3-4 shows how the MATCHGCSR* signal might be used.

3.3.17 PA00-PA01 - Processor Address Lines 0-1

Type: Input; Three-State Output

NOTE

PA00-PA01 is designed to be directly connected to the PA00-PA01 pin on the MPU (68020/68030).

PA00-PA01 are received by the chip's VMEbus master when it executes VMEbus cycles. In conjunction with other signal lines and control bits, the master uses PA00 and PA01 to determine the size of the data transfer, and drive the appropriate levels on the VMEbus signals DS0*, DS1*, A01, and LWORD*, as shown in Tables 3-3 and 3-4.

PA00-PA01 are driven by the chip's VMEbus slave when it responds to VMEbus cycles. The slave drives address lines PA00 and PA01 to establish the byte locations accessed, as determined from DS0*, DS1*, A01, and LWORD* (refer to Tables 3-5 and 3-6).

PA00-PA01 are received by the chip's local slave during a local access to the chip's LCSR and GCSR. When the local processor is accessing the chip's registers, address lines PA00-PA01, in conjunction with PA02-PA07, are used to select the specific register that is being accessed.

3.3.18 PA01-PA03 - Processor Address Lines 1-3

Type: Input; Three-State Output

NOTE

PA01-PA03 is designed to be directly connected to the PA01-PA03 pin on the MPU (68020/68030).

PA01-PA03 are used by the interrupt handler to determine to which level the local processor is responding during an interrupt acknowledge cycle. If the interrupt acknowledge cycle becomes a VMEbus interrupt acknowledge cycle, then PA01-PA03 are also used to create the value driven onto A01-A03 by the MVME6000 (refer to Table 3-2).

Table 3-2. Using PA01-PA03 to Determine the Interrupt Level

PA3	PA2	PA1	INTERRUPT BEING ACKNOWLEDGED	A03	A02	A01
L	L	L	None	---	---	---
L	L	H	Level 1	L	L	H
L	H	L	Level 2	L	H	L
L	H	H	Level 3	L	H	H
H	L	L	Level 4	H	L	L
H	L	H	Level 5	H	L	H
H	H	L	Level 6	H	H	L
H	H	H	Level 7	H	H	H

3.3.19 PA02-PA07 - Processor Address Lines 2-7

Type: Input; Three-State Output

NOTE

PA02-PA07 is designed to be directly connected to the PA02-PA07 pin on the MPU (68020/68030).

PA02-PA07 are received by the chip's VMEbus master, and driven onto VMEbus address lines A02-A07, when it executes VMEbus cycles.

PA02-PA07 are driven by the chip's VMEbus slave to carry the levels received on VMEbus address lines A02-A07 when it responds to VMEbus cycles.

PA02-PA07 are received by the chip's local slave when the local CPU accesses the LCSR and the GCSR. When the local processor is accessing the chip's registers, address lines PA02-PA07, in conjunction with PA00-PA01, select the register being accessed.

3.3.20 PAS* - Processor Address Strobe

Type: Three-State Output

PAS* is driven by the chip's VMEbus slave when it responds to a VMEbus cycle. It is asserted to indicate the start of an access to local resources, and is negated when the access is completed.

3.3.21 PBERR* - Processor Bus Error

Type: Input; Open-Collector Output

NOTE

PBERR* is designed to be directly connected to the PBERR* pin on the MPU (68020/68030).

PBERR* is driven by on-board logic and received by the MVME6000's VMEbus slave when it is responding to a VMEbus cycle. On-board logic asserts PBERR* to indicate that the data transfer request has not completed successfully (e.g., parity error), and is negated to indicate a successful transfer. When the MVME6000's VMEbus slave is responding to a VMEbus cycle, the assertion of PBERR* causes the MVME6000 to assert the VMEbus' BERR* signal.

PBERR*, in conjunction with HALT* (refer to paragraph 3.3.8), is asserted by the MVME6000 to indicate that the local processor should retry the current VMEbus bound cycle. A retry sequence is initiated in response to the following two events:

1. On-board logic asserts RETRY* (refer to paragraph 3.3.32) while the chip's VMEbus master is executing a VMEbus cycle.

2. A dual-port lockup condition is detected. The MVME6000 determines a lockup condition when a VMEbus master is attempting to access local resources through

the MVME6000's VMEbus slave at the same time that the local processor is attempting to access VMEbus resources using the chip's VMEbus master.

NOTE

If the 020 bit is set and the RMC* signal is asserted during either of the two cases, the MVME6000 initiates a bus error, not a retry.

3.3.22 PBG* - Processor Bus Grant

Type: Input

NOTE

PBG* should not be connected directly to the MPU (68020/68030).

PBG* is monitored by the MVME6000 to detect when it is granted control of the local bus. PBG* is asserted by on-board logic to give the MVME6000 mastership of the local bus.

3.3.23 PBR* - Processor Bus Request

Type: Open-Collector Output

NOTE

PBR* should not be connected directly to the MPU (68020/68030).

PBR* is asserted by the MVME6000 to request mastership of the local bus. The chip requests mastership of the local bus when it determines that its VMEbus slave was selected to access an on-board resource. The MVME6000 holds PBR* asserted until its input signal PBG* is asserted, and the chip has asserted PAS* and PDS*.

NOTE

If the local processor requests that the MVME6000's master execute a VMEbus cycle after the chip asserts PBR*, or if the local bus is requested by the chip's VMEbus slave after the chip's VMEbus master was selected to execute a VMEbus cycle, the chip will perform a retry as described in paragraph 3.3.31.

3.3.24 PD00-PD07 - Local Data Lines 0-7

Type: Input; Three-State Output

The MVME6000's VMEbus slave monitors PD00-PD07, driving the binary value they carry onto the VMEbus D00-D07 data lines when it responds to a VMEbus read cycle. When the MVME6000's VMEbus slave responds to a VMEbus write cycle, it drives PD00-PD07 with the binary value carried on the VMEbus D00-D07 data lines.

The chip's VMEbus master monitors PD00-PD07, driving the binary value they carry onto the VMEbus D00-D07 data lines when it executes a VMEbus write cycle. When the MVME6000's VMEbus master executes a VMEbus read cycle, it drives PD00-PD07 with the binary value carried on the VMEbus D00-D07 data lines.

In addition, the local processor uses PD00-PD07 to access the MVME6000's GCSR and LCSR. When the local processor reads the registers, the MVME6000 drives PD00-PD07 with the binary value stored in the accessed register. When the processor writes to the registers, the chip monitors PD00- PD07, and stores the binary value they carry into the accessed register.

3.3.25 PDS* - Processor Data Strobe

Type: Input; Three-State Output

NOTE

PDS* is designed to be directly connected to the PDS* pin on the MPU (68020/68030).

PDS* is driven by the MVME6000's VMEbus slave to indicate the beginning of an access to local resources. The slave asserts PDS* when it responds to a VMEbus cycle, but not until after it has been granted mastership of the local bus (refer to paragraphs 3.3.22 and 3.3.23). The slave negates PDS* after the local resource acknowledges the data access as described in paragraph 3.3.26.

PDS* is received by the MVME6000's VMEbus master to determine the beginning and end of the VMEbus bound read or write cycle. It begins (or attempts to begin) a VMEbus read/write when both PDS* and VMESEL are asserted (refer to paragraph 3.3.38). On-board logic then negates PDS* to terminate the cycle.

PDS* is received by the MVME6000's interrupt handler to determine the start and end of an interrupt acknowledge cycle. The VMEbus interrupt-handler begins (or attempts to begin) a VMEbus IACK cycle when it detects both PDS* and PIACK* asserted (refer to paragraph 3.3.28). Local logic then negates PDS* to terminate the interrupt acknowledge cycle.

PDS* is received by the MVME6000 to determine the start and end of an access to the LCSR and GCSR. The MVME6000 allows access to the registers after it detects both CS* (refer to paragraph 3.3.6) and PDS* asserted. The local processor then negates PDS* to terminate the access to the MVME6000's control and status registers.

CAUTION

DURING VMEBUS MASTER READ/WRITE/IACK CYCLES, PDS* MUST REMAIN ASSERTED UNTIL THE MVME6000 ASSERTS PDSACK0*, PDSACK1*, OR PBERR*. ANY VIOLATION OF THIS MAY CAUSE ERRATIC OPERATION.

The intended use of PDS*, PIACK*, and VMESEL* is shown in Figure 3-2.

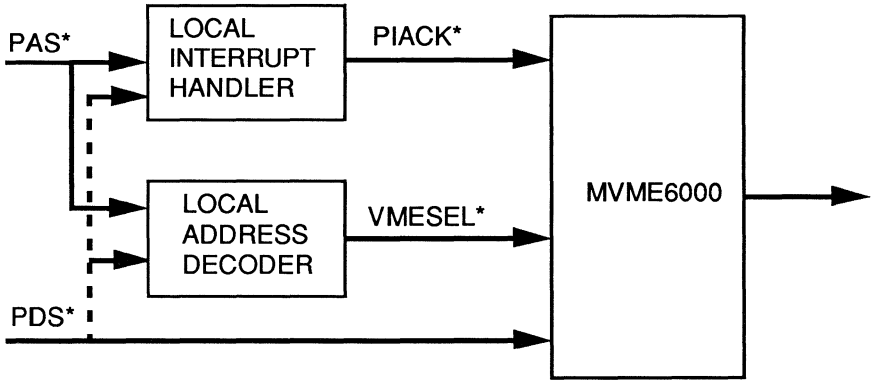


Figure 3-2. Connecting PDS*, VMESEL*, and PIACK* to the MVME6000

3.3.26 PDSACK0*-PDSACK1* - Data Size Acknowledge 0-1

Type: Input; Three-State Output

NOTE

PDSACK0*-PDSACK1* is designed to be directly connected to the PDSACK0*-PDSACK1* pin on the MPU (68020/68030).

The MVME6000's VMEbus slave monitors PDSACK0* and PDSACK1* to determine when the local device has retrieved the requested data during a read cycle, or when it has stored the data during write cycles. Since the VMEbus does not support dynamic bus sizing, local logic needs only assert one of the two signal lines.

The MVME6000's VMEbus master drives PDSACK0*-PDSACK1* to signal the local processor that the VMEbus cycle has been executed. Asserting either PDSACK0* or PDSACK1* during a VMEbus read cycle signals that the VMEbus slave has responded to the cycle by asserting DTACK*, and that the requested data is now valid. When the chip asserts PDSACK0* or PDSACK1* during a VMEbus write cycle, it signals the local processor that its request has been acknowledged, and that it can terminate the cycle. However, when the master has been configured to execute write-posting operations, the chip asserts the PDSACK lines as soon as it has latched (or controlled the latching of) the required cycle information, and before the VMEbus slave actually responded to it. The chip drives PDSACK0*-PDSACK1* as shown in Tables 3-3 and 3-4.

The MVME6000 asserts PDSACK1* to indicate to the local processor that its requested access to the LCSR or GCSR has been successfully completed.

3.3.27 PFC0-PFC2 - Processor Function Code 0-2

Type: Input; Three-State Output

NOTE

PFC0-PFC1 is designed to be directly connected to the PFC0-PFC1 pin on the MPU (68020/68030).

PFC0-PFC2 are driven by the chip's VMEbus slave to the binary value carried on the VMEbus AM0-AM2 lines in response to a VMEbus cycle. If the slave's Address Modifier is cleared to zeros the MVME6000 does not drive PFC0-PFC2.

PFC0-PFC2 are monitored by the chip's VMEbus master to dynamically determine the level it is to drive on the VMEbus AM0-AM2 address modifier lines. However, the levels carried on PFC0- PFC2 are actually driven onto AM0-AM2 only if the Masters Address Modifier register is not selected, as described in paragraph 4.9.

3.3.28 PIACK* - Processor Interrupt Acknowledge

Type: Input

The MVME6000 monitors PIACK to determine when the local MPU is acknowledging an interrupt request. When both PIACK* and PDS* are asserted, the VMEbus interrupt handler checks for an internal interrupt request. If an internal MVME6000 interrupt request is pending on the acknowledge level, the MVME6000 responds immediately. Otherwise, it waits and performs a VMEbus IACK* cycle that it uses to respond to the local interrupt acknowledge cycle. Figure 3-2 shows how the on-board logic might use PIACK*.

3.3.29 PRESET* - Processor Reset

Type: Input; Open-Collector Output

The MVME6000 asserts PRESET* when either the local input signal BRDRESET* (refer to paragraph 3.3.27), or the VMEbus SYSRESET* signal are asserted. When asserting PRESET*, the MVME6000 holds it asserted for a minimum of 200 milliseconds.

PRESET* might also be asserted by external logic. When the MVME6000 detects that PRESET* is asserted, it initializes some of the control register bits to their default value. The bits that are affected by PRESET*, as well as the value they are set to, are defined in Chapters 4 and 5. Figure 3-3 shows an example of connecting the BRDRESET* and PRESET* signals to onboard logic.

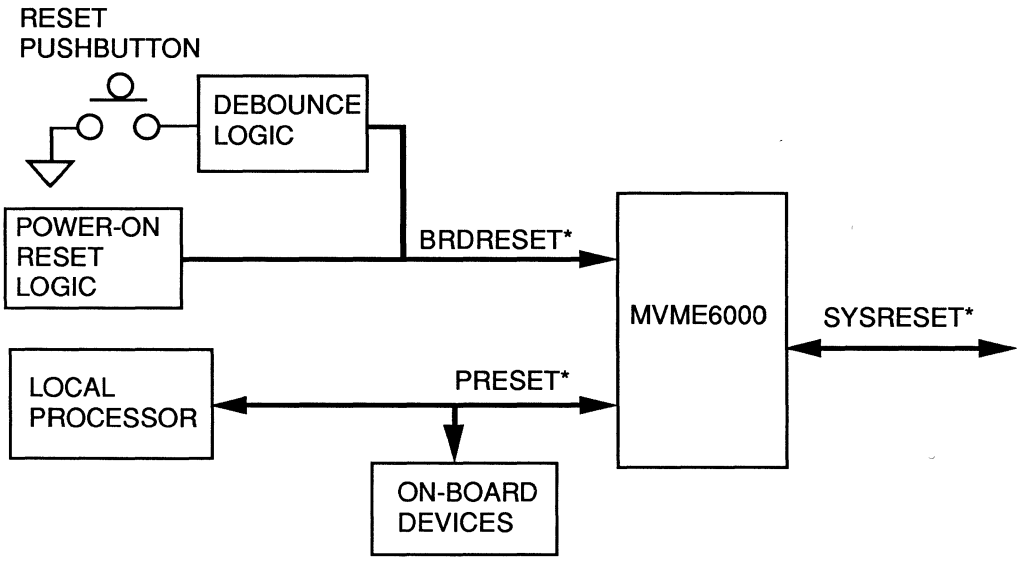


Figure 3-3. Connecting BRDRESET* and PRESET* to On-Board Logic

3.3.30 PSIZ0, PSIZ1 - Processor Size Request 0-1

Type: Input; Three-State Output

NOTE

PSIZ0-PSIZ1 is designed to be directly connected to the PSIZ0-PSIZ1 pin on the MPU (68020/68030).

PSIZ0-PSIZ1 are received by the chip's VMEbus master when it executes a VMEbus cycle. In conjunction with other signal lines and control bits, the master uses PSIZ0 and PSIZ1 to determine the size of the data transfer, and drive the appropriate levels on the VMEbus signals DS0*, DS1*, A01, and LWORD*, as shown in Tables 3-3 and 3-4.

PSIZ0-PSIZ1 are driven by the chip's VMEbus slave when it responds to a VMEbus cycle. The levels driven on PSIZ0 and PSIZ1 establish the byte locations accessed during the cycle, as determined from DS0*, DS1*, A01, and LWORD*, as shown in Table 3-5 and Table 3-6.

3.3.31 PWRITE* - Processor Write

Type: Input; Three-State Output

NOTE

PWRITE* is designed to be directly connected to the PWRITE* pin on the MPU (68020/68030).

PWRITE* is driven by the MVME6000's VMEbus slave in response to a VMEbus cycle. The level that the slave drives PWRITE* to, is directly derived from the VMEbus signal WRITE*. The slave asserts PWRITE* to execute a write cycle to local resources, and negates it to execute a read cycle to local resources. Refer to Tables 3-5 and 3-6.

PWRITE* is received by the MVME6000's VMEbus master when it executes a VMEbus cycle. The master uses the incoming level on PWRITE* to determine whether it is to execute a VMEbus read or a VMEbus write cycle. The master executes a write cycle when PWRITE* is asserted, and a read cycle when it is negated.

PWRITE* is received by the MVME6000 when its LCSR or GCSR are accessed. The local processor asserts PWRITE* when it wishes to store data in the registers, and negates PWRITE* when it wishes to read data from the registers.

3.3.32 RETRY* - Retry

Type: Input

RETRY* allows on-board logic to request that the local processor retry the cycle. The MVME6000 initiates a retry sequence when it detects RETRY* asserted, when its master is executing a VMEbus cycle, but only if it has not yet asserted the DS1*/DS0*. It initiates the retry sequence by asserting both PBERR* and HALT*, adhering to the timing requirements of a 68020 type processor. However, if RMC* is asserted and the 020 bit is set when the chip detects RETRY* asserted, it will only assert PBERR*.

3.3.33 RMC* - Read-Modify-Write Cycle

Type: Input

NOTE

RMC* is designed to be directly connected to the RMC* pin on the MPU (68020/68030).

RMC* is asserted by on-board logic in conjunction with VMESEL* and PDS* to request that the MVME6000's VMEbus master execute a VMEbus RMW cycle. When the chip detects RMC* low and PSIZ0 high, it determines that the local processor is requesting a single-address indivisible cycle. In response, the chip's master will maintain the VMEbus AS* asserted throughout the cycle, toggling the data strobes, and terminating the cycle after RMC* is negated. If the MPU executes a multiple-address indivisible cycle (PSIZ0 low during the first cycle after RMC* is asserted), the MVME6000 ignores RMC* and performs normal read and write cycles as if RMC* were not asserted.

NOTE

The PA01 - PA31, PFC0-2, ADR24, SHORTIO, and DAT16 pins should remain valid during both the read and the write portions of read-modify-write cycles.

3.3.34 SCLK - System Clock

Type: Input

SCLK is a 16MHz input clock that is used to generate the VMEbus SYSCLK signal.

3.3.35 SCON* - System Controller On

Type: Input

SCON* is driven by on-board logic to configure the MVME6000 as the VMEbus system controller. When the MVME6000 detects SCON* asserted, it enables the VMEbus arbiter and arbitration timer, the bus timer, the IACK daisy-chain driver, the SYSCLK driver, and the SYSRESET* driver. These functions are disabled when SCON* is negated.

NOTE

When SCON* is high, the MVME6000's SYSRESET* driver will not drive SYSRESET* low when BRDRESET* is driven low. However, the chip will still drive SYSRESET* low whenever the SRESET control bit in the System Controller Register in the LCSR is set, regardless of the state of SCON*.

3.3.36 SHORTIO* - Short I/O Cycle

Type: Input

SHORTIO* is driven by on-board logic (an address decoder) to configure the address modifier code that the MVME6000's VMEbus master drives during a VMEbus cycle. An A16 address modifier code is driven whenever the line is asserted and the Master Address Modifier register is not selected (paragraph 4.9). When the line is negated, the MVME6000 automatically determines the address modifier code in one of two ways:

1. Dynamically from the levels of the ADR24* signal line (paragraph 3.3.2) and the MASA24 and MASA16 control bits (paragraph 4.5 and Table 4-6).
2. From the Master Address Modifier register, if enabled as described in paragraph 4.9.

3.3.37 VMESEL* - Select VMEbus Cycle

Type: Input

The MVME6000 uses VMESEL* in conjunction with PDS* to determine that its VMEbus master is requested to perform a VMEbus cycle, and that mastership of the VMEbus should be obtained. The MVME6000 responds to this request once it detects both VMESEL* and PDS* asserted. The on- board logic that drives VMESEL* should maintain it asserted until after PDS* is negated. Figure 3-2 shows how on-board logic might use VMESEL*.

3.4 BUFFER CONTROL SIGNALS

3.4.1 ABIN* - Address Bus In

Type: Totem-Pole Output

ABIN* is driven by the chip's VMEbus slave when it responds to a VMEbus cycle. The MVME6000 asserts ABIN* to indicate that, having been granted mastership of the local bus, it is enabling the VMEbus address bus to be driven onto the local bus.

The intended use of ABIN* is to enable the output of the external buffers that receive address lines A08-A31 of the VMEbus. See Figure 3-4 for an example of connecting the MVME6000 to the 74FCT543 registered transceivers to receive the VMEbus address bus .

3.4.2 ABOUT* - Address Bus Out

Type: Totem-Pole Output

ABOUT* is driven by the chip's VMEbus master when it executes a VMEbus cycle. It also is driven by the chip's VMEbus interrupt handler when it executes VMEbus interrupt acknowledge cycles.

The MVME6000 asserts ABOUT* to indicate that, having been granted mastership of the VMEbus, the board can now drive the VMEbus address bus.

ABOUT* is used to enable the output of the external buffers that are used to drive address lines A08-A31 of the VMEbus. The MVME6000 controls the assertion of ABOUT* to provide the required 35 nanoseconds set-up time of the address bus to AS* (Timing Parameter 4 in the VMEbus Specification). An example of connecting the MVME6000 to the 74FCT543 registered transceivers to drive the VMEbus address bus is given in Figure 3-4.

3.4.3 LATCHIN - Latch In

Type: Totem-Pole Output

LATCHIN is driven by the chip's VMEbus slave when it responds to VMEbus cycles. It is asserted by the MVME6000 whenever it is responding to a VMEbus data transfer cycle. LATCHIN is used to connect to the latch enable input of external latches (such as the 74FCT543) that are used to receive the VMEbus A08-A31 address lines, and D08-D31 data lines. An example of connecting the MVME6000 to the 74FCT543 registered transceivers to receive the VMEbus address and data buses is given in Figure 3-4 and Figure 3-5.

NOTE

Latching the VMEbus data bus into the local registers of the 74FCT543s is only meaningful when the chip is executing a cycle that receives data from the VMEbus and drives it onto the local data bus. The output of the latches is enabled in accordance with the type of cycle the chip executes as described in paragraphs 3.4.6 and 3.4.

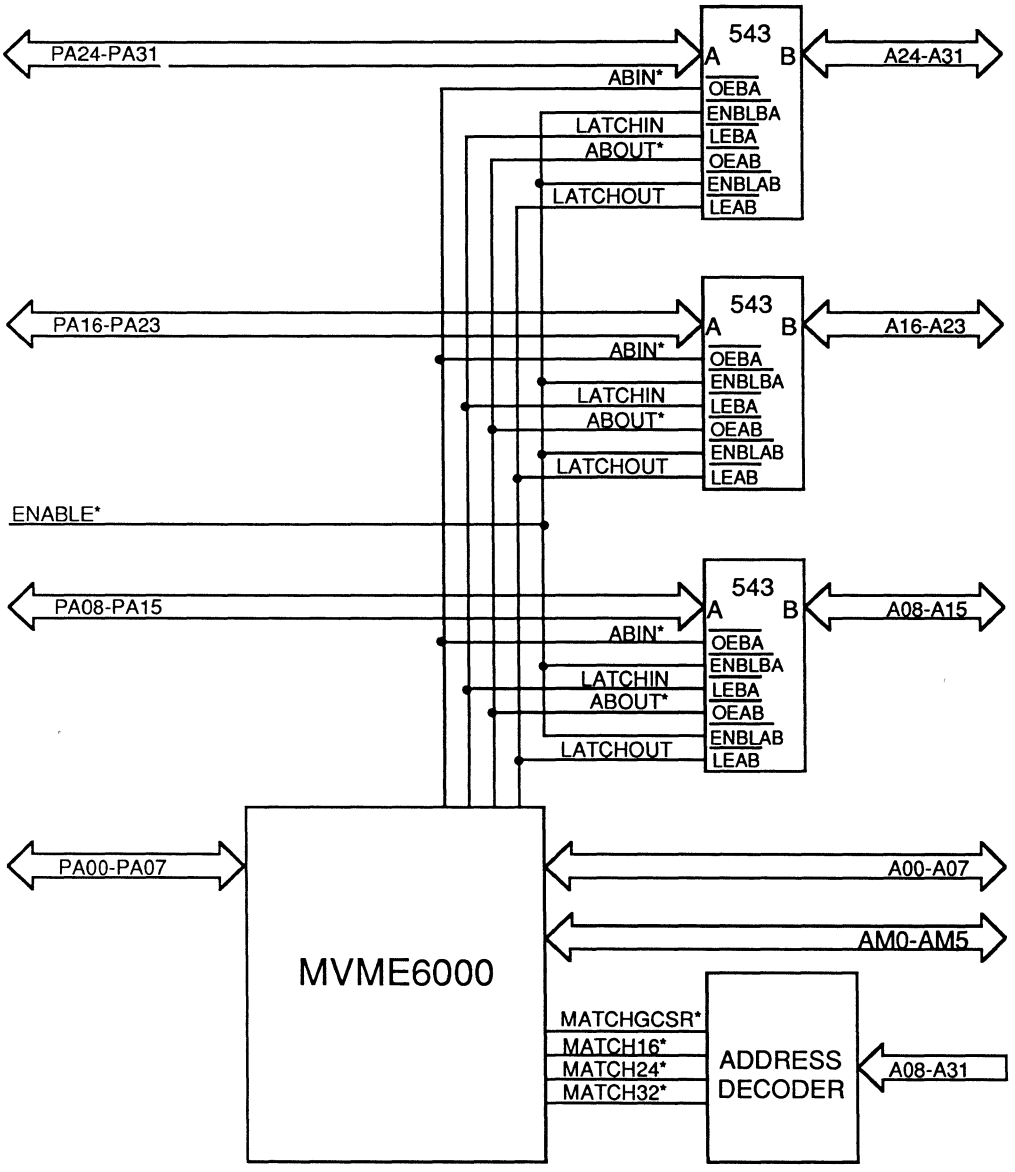


Figure 3-4. Connecting the MVME6000 to External Address Buffers

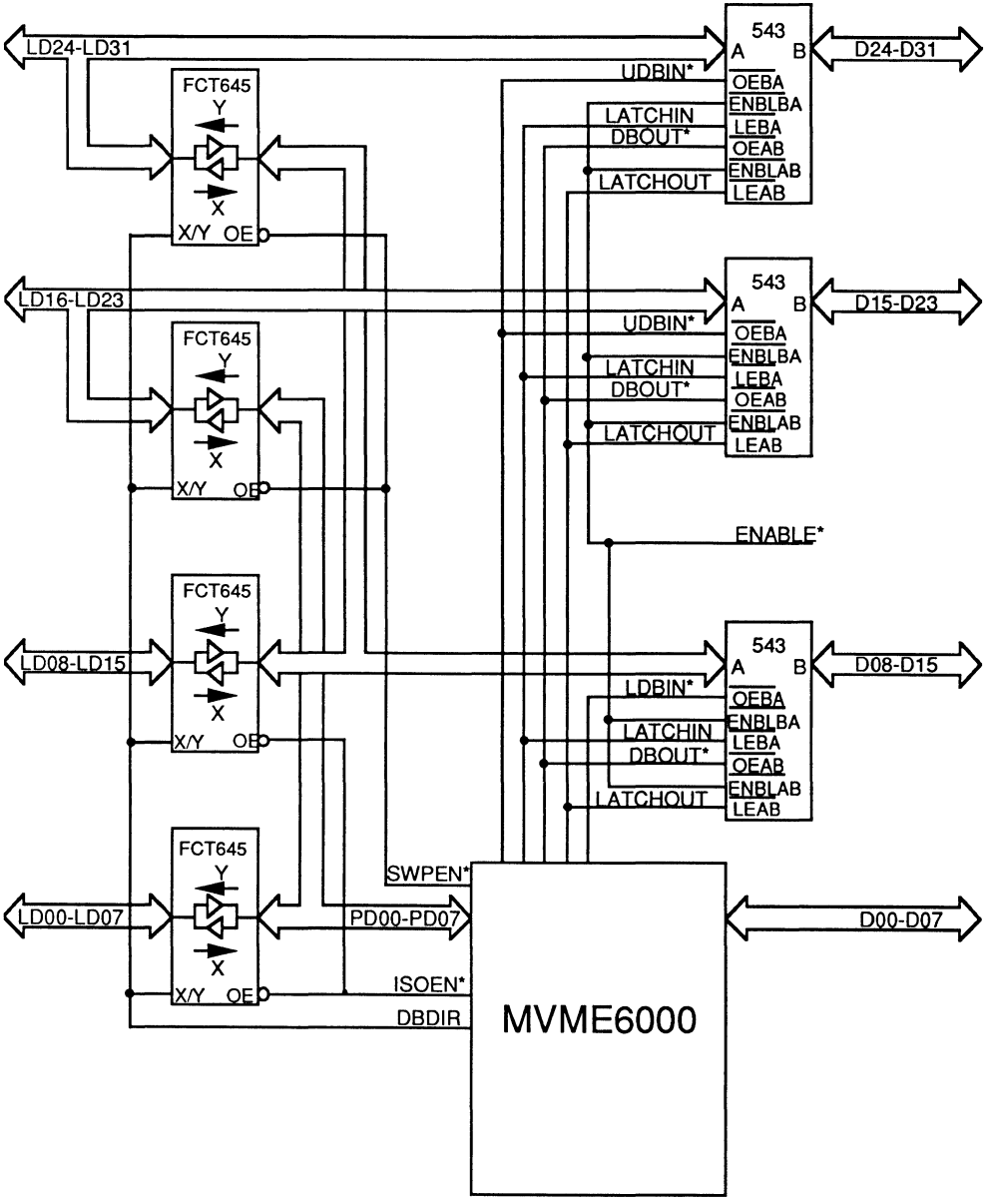


Figure 3-5. Connecting the MVME6000 to External Data Buffers

3.4.4 LATCHOUT - Latch Out

Type: Totem-Pole Output

LATCHOUT is driven by the chip's VMEbus master when it executes VMEbus cycles.

LATCHOUT is driven by the chip's VMEbus interrupt handler when it executes VMEbus interrupt acknowledge cycles.

LATCHOUT is driven by the chip's VMEbus slave when it responds to VMEbus cycles.

LATCHOUT is asserted by the MVME6000 whenever it is initiating or responding to a VMEbus cycle. The intended use of LATCHOUT is to connect to the latch enable input of external latches (such as the 74FCT543) that are used to drive the VMEbus A08-A31 address lines, and D08-D31 data lines. An example of connecting the MVME6000 to the 74FCT543 registered transceivers to drive the VMEbus address and data buses is given in Figures 3-4 and 3-5.

NOTE

Latching the local data bus into the VMEbus registers of the 74FCT543s is only meaningful when the chip is executing a cycle that drives data from on-board logic onto the VMEbus data lines. The output of the latches is enabled in accordance with the type of cycle that the chip executes as described in paragraph 3.4.5.

3.4.5 DBOUT* - Data Bus Out

Type: Totem-Pole Output

DBOUT* is asserted by the chip's VMEbus master when it executes a VMEbus write cycle (see Tables 3-3 and 3-4). DBOUT* is asserted by the chip's VMEbus slave whenever it responds to a VMEbus read cycle (see Tables 3-5 and 3-6).

DBOUT* is used to enable the buffers that are used to drive the local data lines PD08-PD31 onto the VMEbus data lines D08-D31. See Figure 3-4 for an example of connecting the MVME6000 to the 74FCT543 registered transceivers to drive the VMEbus data bus.

3.4.6 LDBIN* - Lower Data Bus In

Type: Totem-Pole Output

LDBIN* is driven by the chip's VMEbus master when it executes a VMEbus read cycle. The level to which the chip drives the LDBIN* signal is determined by the type of the VMEbus read cycle its VMEbus master is executing, as shown in Tables 3-3 and 3-4.

LDBIN* is driven by the chip's VMEbus slave when it responds to a VMEbus write cycle. The level to which the chip drives the LDBIN* signal is determined by the type of write cycle its VMEbus slave is responding to, as shown in Tables 3-5 and 3-6.

The intended use of LDBIN* is to enable the buffers that are used to receive the VMEbus data lines D08-D15 and drive local data lines PD08-PD15. An example of connecting the MVME6000 to the 74FCT543 registered transceivers to receive the VMEbus data bus is given in Figure 3-5.

3.4.7 UDBIN* - Upper Data Bus In

Type: Totem-Pole Output

UDBIN* is driven by the chip's VMEbus master when it executes a VMEbus read cycle. The level to which the chip drives the UDBIN* signal is determined by the type of the VMEbus read cycle its VMEbus master is executing as shown in Tables 3-3 and 3-4.

UDBIN* is driven by the chip's VMEbus slave when it responds to a VMEbus write cycle. The level to which the chip drives the UDBIN* signal is determined by the type of write cycle to which its VMEbus slave is responding. Refer to Tables 3-5 and 3-6.

The intended use of UDBIN* is to enable the buffers that are used to receive the VMEbus data lines D16-D31 to drive the local data lines PD16-PD31. An example of connecting the MVME6000 to the 74FCT543 registered transceivers to receive the VMEbus data bus is shown in Figure 3-5.

3.4.8 DBDIR - Data Bus Direction

Type: Totem-Pole Output

DBDIR is driven by the chip's VMEbus master, the chip's VMEbus slave, and the chip's local slave to indicate the direction of the data.

DBDIR is driven low by the chip's VMEbus master when it executes a VMEbus read cycle, and high when it executes a VMEbus write cycle (see Tables 3-3 and 3-4).

DBDIR is driven high when the chip's VMEbus slave is responding to a VMEbus read cycle, and low when it is responding to a VMEbus write cycle (see Tables 3-5 and 3-6).

DBDIR is driven low by the chip's local slave when it responds to a local read cycle, and high when it responds to a local write cycle.

The intended use of the DBDIR output signal is to control the direction of the various data buffers that are used to interface the local data bus to the VMEbus data bus. Figure 3-5 is an example of how DBDIR is used.

3.4.9 ISOEN* - Isolation Enable*

Type: Totem-Pole Output

ISOEN* is driven by the chip's VMEbus master when it executes VMEbus read or write cycle. The level to which the master drives ISOEN* is determined by the type of cycle it executes, as well as by the capabilities it was configured to support, as shown in Tables 3-3 and 3-4.

ISOEN* is driven by the chip's VMEbus slave when it is responding to VMEbus read and write cycles. The level to which the slave drives ISOEN* is determined by the type of cycle it is responding to, and by the capabilities it includes. See Tables 3-5 and 3-6.

The chip's local slave negates ISOEN* when it responds to an access to the LCSR and the GCSR.

ISOEN* is used to provide the necessary control required to interface D16 and D08(EO) VMEbus slaves to a 68020/68030 microprocessor. This class of processors access D16 devices over data lines LD16-LD31, while the VMEbus uses data lines D00-D15 to access them. Figure 3-5 shows how ISOEN* is used to support this requirement. In the example, if ISOEN* is asserted during cycles that drive data onto the VMEbus, the isolation buffers are enabled to route the local data lines LD00-LD15 onto the VMEbus data lines D00-D15. Similarly, if ISOEN* is asserted during cycles that receive data from the VMEbus, the VMEbus data lines D00-D15 will be routed onto the local data lines LD00-LD15. When ISOEN* is negated, the isolation buffers are turned off.

3.4.10 SWPEN* - Swap Enable

Type: Totem-Pole Output

SWPEN* is driven by the chip's VMEbus master when it executes a VMEbus cycle. The level to which the master drives SWPEN* is determined by the type of cycle it executes and by the capabilities it was configured to support, as shown in Tables 3-3 and 3-4.

SWPEN* is driven by the chip's VMEbus slave when it is responding to a VMEbus cycle. The level to which the slave drives SWPEN* is determined by the type of cycle it is responding to and by the capabilities it was configured to support, as shown in Tables 3-5 and 3-6.

SWPEN* is asserted by the chip's local slave when it responds to a local access to the LCSR and GCSR.

The intended use of the SWPEN* signal is to provide the necessary control that will allow to interface D16 and D08(EO) VMEbus slaves to a 68020/68030 microprocessor. This class of processors access D16 devices over data lines LD16-LD31, while the VMEbus uses data lines D00- D15 to access them. Figure 3-5 shows how SWPEN* is used to support this requirement. In the example, if SWPEN* is asserted during cycles that drive data onto the VMEbus, the swap buffers are enabled to route the local data lines LD16-LD31 onto the VMEbus data lines D00-D15. Similarly, if SWPEN* is asserted during cycles that receive data from the VMEbus, the VMEbus data lines D00-D15 will be routed onto the local data lines LD16-LD31.

Table 3-3. Signaling During VMEbus Read Cycles

INPUT SIGNALS AND CONTROL BITS							OUTPUT SIGNALS							
PSIZ1	PSIZ0	PA1	PA0	CFILL	D32EN	MASUAT	PDSACK0*	SWPEN*	ISOEN*	UDBIN*	DS1*	DS0*	A01	LWORD*
L	L	L	L	X	F	X	H	L	H	H	L	L	L	H
L	L	L	L	0	T	X	L	H	L	L	L	L	L	L
L	L	L	H	0	X	0	H	L	H	H	H	L	L	H
L	L	L	H	X	F	X	H	L	H	H	H	L	L	H
L	L	L	H	0	T	1	L	H	L	L	H	L	L	L
L	L	H	L	0	X	X	H	L	H	H	L	L	H	H
L	L	H	H	0	X	X	H	L	H	H	H	L	H	H
L	H	L	L	0	X	X	H	L	H	H	L	H	L	H
L	H	L	H	0	X	X	H	L	H	H	H	L	L	H
L	H	H	L	0	X	X	H	L	H	H	L	H	H	H
L	H	H	H	0	X	X	H	L	H	H	H	L	H	H
H	L	L	L	0	X	X	H	L	H	H	L	L	L	H
H	L	L	H	X	F	X	H	L	H	H	H	L	L	H
H	L	L	H	0	X	0	H	L	H	H	H	L	L	H
H	L	L	H	0	T	1	L	H	L	L	L	L	H	L
H	L	H	L	0	X	X	H	L	H	H	L	L	H	H
H	L	H	H	0	X	X	H	L	H	H	H	L	H	H
H	H	L	L	0	X	0	H	L	H	H	L	L	L	H
H	H	L	L	X	F	X	H	L	H	H	L	L	L	H
H	H	L	L	0	T	1	L	H	L	L	L	H	L	L
H	H	L	H	0	X	0	H	L	H	H	H	L	L	H
H	H	L	H	X	F	X	H	L	H	H	H	L	L	H
H	H	L	H	0	T	1	L	H	L	L	H	L	L	L
H	H	H	L	0	X	X	H	L	H	H	L	L	H	H
H	H	H	H	0	X	X	H	L	H	H	H	L	H	H
X	X	X	X	1	T	X	L	H	L	L	L	L	L	L

Notes: 1. F = False, T = True, H = High, L = Low, X = Don't care
2. D32EN is True if DAT16* = H and MASD16 = 0.
3. D32EN is False if either DAT16* = L or MASD16 = 1
4. PDSACK1* = L, DBDIR = L, LDBIN = L, DBOUT* = H During all cycles above.

Table 3-4. Signaling During VMEbus Write Cycles

INPUT SIGNALS AND CONTROL BITS							OUTPUT SIGNALS							
PSIZ1	PSIZ0	PA1	PA0	CFILL	D32EN	MASUAT	PDSACK0*	SWPEN*	ISOEN*	UDBIN*	DS1*	DS0*	A01	LWORD*
L	L	L	L	X	F	X	H	L	H	H	L	L	L	H
L	L	L	L	X	T	X	L	H	L	H	L	L	L	L
L	L	L	H	X	X	0	H	L	H	H	H	L	L	H
L	L	L	H	X	F	X	H	L	H	H	H	L	L	H
L	L	L	H	X	T	1	L	H	L	H	H	L	L	L
L	L	H	L	X	X	X	H	L	H	H	L	L	H	H
L	L	H	H	X	X	X	H	L	H	H	H	L	H	H
L	H	L	L	X	X	X	H	L	H	H	L	H	L	H
L	H	L	H	X	X	X	H	L	H	H	H	L	L	H
L	H	H	L	X	X	X	H	L	H	H	L	H	H	H
L	H	H	H	X	X	X	H	L	H	H	H	L	H	H
H	L	L	L	X	X	X	H	L	H	H	L	L	L	H
H	L	L	H	X	F	X	H	L	H	H	H	L	L	H
H	L	L	H	X	X	0	H	L	H	H	H	L	L	H
H	L	L	H	X	T	1	L	H	L	H	L	L	H	L
H	L	H	L	X	X	X	H	L	H	H	L	L	H	H
H	L	H	H	X	X	X	H	L	H	H	H	L	H	H
H	H	L	L	X	X	0	H	L	H	H	L	L	L	H
H	H	L	L	X	F	X	H	L	H	H	L	L	L	H
H	H	L	L	X	T	1	L	H	L	H	L	H	L	L
H	H	L	H	X	X	0	H	L	H	H	H	L	L	H
H	H	L	H	X	F	X	H	L	H	H	H	L	L	H
H	H	L	H	X	T	1	L	H	L	H	H	L	L	L
H	H	H	L	X	X	X	H	L	H	H	L	L	H	H
H	H	H	H	X	X	X	H	L	H	H	H	L	H	H

Notes: 1. F = False, T = True, H = High, L = Low, X = Don't care
2. D32EN is True if DAT16* = H and MASD16 = 0.
3. D32EN is False if either DAT16* = L or MASD16 = 1
4. PDSACK1* = L, DBDIR = H, LDBIN = H, DBOUT* = L During all cycles above.

Table 3-5. Signaling When Responding to a VMEbus Read Cycle

INPUT SIGNALS AND CONTROL BITS					OUTPUT SIGNALS									
SLVD16	LWORD*	A01	DS0*	DS1*	PSIZ1	PSIZ0	PA1	PA0	SWPEN*	ISOEN*	LDBIN*	UDBIN*	DBOUT*	BYTE LOCATION ACCESSED
0	L	L	L	L	L	L	L	L	H	L	H	H	L	BYTE(0-3)
0	L	L	L	H	H	H	L	H	H	L	H	H	L	BYTE(1-3)
0	L	L	H	L	H	H	L	L	H	L	H	H	L	BYTE(0-2)
0	L	H	L	L	H	L	L	H	H	L	H	H	L	BYTE(1-2)
0	L	H	L	H	H	H	H	H	H	H	H	H	H	NONE
0	L	H	H	L	H	H	H	H	H	H	H	H	H	NONE
0	H	L	L	L	H	L	L	L	L	H	H	H	L	BYTE(0-1)
0	H	L	L	H	L	H	L	H	L	H	H	H	L	BYTE(1)
0	H	L	H	L	L	H	L	L	L	H	H	H	L	BYTE(0)
0	H	H	L	L	H	L	H	L	H	L	H	H	L	BYTE(2-3)
0	H	H	L	H	L	H	H	H	H	L	H	H	L	BYTE(3)
0	H	H	H	L	L	H	H	L	H	L	H	H	L	BYTE(2)
1	L	L	L	L	H	H	H	H	H	H	H	H	H	NONE
1	L	L	L	H	H	H	H	H	H	H	H	H	H	NONE
1	L	L	H	L	H	H	H	H	H	H	H	H	H	NONE
1	L	H	L	L	H	H	H	H	H	H	H	H	H	NONE
1	L	H	L	H	H	H	H	H	H	H	H	H	H	NONE
1	L	H	H	L	H	H	H	H	H	H	H	H	H	NONE
1	H	L	L	L	H	L	L	L	L	H	H	H	L	BYTE(0-1)
1	H	L	L	H	L	H	L	H	L	H	H	H	L	BYTE(1)
1	H	L	H	L	L	H	L	L	L	H	H	H	L	BYTE(0)
1	H	H	L	L	H	L	H	L	H	L	H	H	L	BYTE(2-3)
1	H	H	L	H	L	H	H	H	H	L	H	H	L	BYTE(3)
1	H	H	H	L	L	H	H	L	H	L	H	H	L	BYTE(2)

Notes: 1. H = High, L = Low
2. PWRITE* = H, DBDIR = H
During all cycles above.

Table 3-6. Signaling When Responding to a VMEbus Write Cycle

INPUT SIGNALS AND CONTROL BITS					OUTPUT SIGNALS									
SLVD16	LWORD*	A01	DS0*	DS1*	PSIZ1	PSIZ0	PA1	PA0	SWPEN*	ISOEN*	LDBIN*	UDBIN*	DBDIR	BYTE LOCATION ACCESSED
0	L	L	L	L	L	L	L	L	H	L	L	L	L	BYTE(0-3)
0	L	L	L	H	H	H	L	H	H	L	L	L	L	BYTE(1-3)
0	L	L	H	L	H	H	L	L	H	L	L	L	L	BYTE(0-2)
0	L	H	L	L	H	L	L	H	H	L	L	L	L	BYTE(1-2)
0	L	H	L	H	H	H	H	H	H	H	H	H	H	NONE
0	L	H	H	L	H	H	H	H	H	H	H	H	H	NONE
0	H	L	L	L	H	L	L	L	L	L	L	H	L	BYTE(0-1)
0	H	L	L	H	L	H	L	H	L	L	L	H	L	BYTE(1)
0	H	L	H	L	L	H	L	L	L	L	L	H	L	BYTE(0)
0	H	H	L	L	H	L	H	L	L	L	L	H	L	BYTE(2-3)
0	H	H	L	H	L	H	H	H	L	L	L	H	L	BYTE(3)
0	H	H	H	L	L	H	H	L	L	L	L	H	L	BYTE(2)
1	L	L	L	L	H	H	H	H	H	H	H	H	H	NONE
1	L	L	L	H	H	H	H	H	H	H	H	H	H	NONE
1	L	L	H	L	H	H	H	H	H	H	H	H	H	NONE
1	L	H	L	L	H	H	H	H	H	H	H	H	H	NONE
1	L	H	L	H	H	H	H	H	H	H	H	H	H	NONE
1	L	H	H	L	H	H	H	H	H	H	H	H	H	NONE
1	H	L	L	L	H	L	L	L	L	L	L	H	L	BYTE(0-1)
1	H	L	L	H	L	H	L	H	L	L	L	H	L	BYTE(1)
1	H	L	H	L	L	H	L	L	L	L	L	H	L	BYTE(0)
1	H	H	L	L	H	L	H	L	L	L	L	H	L	BYTE(2-3)
1	H	H	L	H	L	H	H	H	L	L	L	H	L	BYTE(3)
1	H	H	H	L	L	H	H	L	L	L	L	H	L	BYTE(2)

Notes: 1. H = High, L = Low
2. PWRITE* = L, DBOUT* = H
During all cycles above.

CHAPTER 4

LOCAL CONTROL AND STATUS REGISTER SET

4.1 GENERAL

The MVME6000 has two sets of registers: (1) the Local Control and Status Registers (LCSR) and (2) the Global Control and Status Registers (GCSR). The LCSR controls and configures the operation of the chip and provides status information that allows software to monitor its interaction with the environment. It is only accessible to the local processor. The GCSR (refer to Chapter 5) allows other masters to monitor the status of the board, affect its operation and communicate with the local CPU. This chapter describes the functions that the LCSR controls and monitors.

4.2 LCSR

Only the local CPU can access the LCSR. It does so by asserting the CS* and PDS* signals together. The value on PA00-PA07 selects which register is accessed within the LCSR as shown in Table 4-1.

Table 4-1. The Local Control and Status Register Set

OFFSET FROM BASE ADDRESS	DESCRIPTION
$01	System Controller Configuration Register
$03	Requester Configuration Register
$05	Master Configuration Register
$07	Slave Configuration Register
$09	Timer Configuration Register
$0B	Slave Address Modifier Register
$0D	Master Address Modifier Register
$0F	Interrupt Handler Mask Register
$11	Utility Interrupt Mask Register
$13	Utility Interrupt Vector Register
$15	Interrupt Request Register
$17	Status/ID Register
$19	Bus Error Status Register
$1B	GCSR Base Address Register

THE LCSR

PA07-00	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT0
$01	- - -	- - -	- - -	- - -	ROBIN S0	BRDFAIL S1; P1	SRESET S0; P0	SCON
$03	DWB S0; P0	DHB S0; P0	RONR S0; P0	RWD S0; P0	RNEVER S0; P0	- - -	RQLEV1 S1; P1	RQLEV0 S1; P1
$05	DDSACK S0	020 S0	MASWP S0	CFILL S0	MASUAT S0	MASA16 S0	MASA24 S0	MASD16 S0
$07	SLVEN S0	- - -	SLVWP S0	- - -	- - -	- - -	- - -	SLVD16 S0
$09	- - -	ARBTO S1	VBTO1 S1	VBTO0 S0	ACTO1 S1; P1	ACTO0 S1; P1	LBTO1 S1; P1	LBTO0 S1; P1
$0B	SUPER S0	USER S0	EXTEND S0	STND S0	SHORT S0	BLOCK S0	PROGRM S0	DATA S0
$0D	AMSEL S0	- - -	AM5 S0	AM4 S0	AM3 S0	AM2 S0	AM1 S0	AM0 S0
$0F	IEN7 S0; P0	IEN6 S0; P0	IEN5 S0; P0	IEN4 S0; P0	IEN3 S0; P0	IEN2 S0; P0	IEN1 S0; P0	- - -
$11	WPERREN S0; P0	SFIEN S0; P0	SIGHEN S0; P0	LMIEN S0; P0	IACKEN S0; P0	LMOEN S0; P0	SIGLEN S0; P0	- - -
$13	UVB7 S0; P0	UVB6 S0; P0	UVB5 S0; P0	UVB4 S0; P0	UVB3 S0; P0	UID2	UID1	UID0
$15	- - -	- - -	- - -	- - -	- - -	IL2 S0	IL1 S0	IL0 S0
$17	D07 S0	D06 S0	D05 S0	D04 S0	D03 S1	D02 S1	D01 S1	D00 S1
$19	- - -	- - -	- - -	- - -	RMCERR S0; P0	VBERR S0; P0	ACTO S0; P0	LBTO S0; P0
$1B	- - -	- - -	- - -	- - -	GCSRA7 S1	GCSRA6 S1	GCSRA5 S1	GCSRA4 S1

S0 indicates that SYSRESET* clears bit to 0.
S1 indicates that SYSRESET* sets bit to 1.
P0 indicates that PRESET* clears bit to 0.
P1 indicates that PRESET* sets bit to 1.
Absence of S0 or S1 indicates that the bit is not affected by SYSRESET*.
Absence of P0 or P1 indicates that the bit is not affected by PRESET*.

Figure 4-1. MVME6000 LCSR

4.3 SYSTEM CONTROLLER CONFIGURATION REGISTER

Table 4-2. System Controller Configuration Register

PA07-00	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
$01	---	---	---	---	ROBIN S=0	BRDFAIL S=1, P=1	SRESET S=0, P=0	SCON

Bit 0 - SCON - System Controller On
The SCON status bit is a reflection of the SCON* input signal line. When the SCON* pin is low, the MVME6000 is the System Controller and SCON =1. When the SCON* pin is high, the MVME6000 is not the System Controller and SCON=0.

Bit 1 - SRESET - System Reset
This bit allows software to initiate a global reset sequence. Setting the SRESET bit will assert the SYSRESET* signal on the VMEbus. Note that SYSRESET* assertion will cause the MVME6000 to also assert the PRESET* signal. After the minimum time elapses, SRESET is cleared and the SYSRESET* and PRESET* signals are no longer asserted by the MVME6000.

CAUTION

IN MOST CASES, PRESET* IS CONNECTED TO THE LOCAL PROCESSOR. IN SUCH CASES, SETTING PRESET* WILL RESET THE LOCAL PROCESSOR.

Bit 2 - BRDFAIL - Board Fail
Setting BRDFAIL to 1 causes the MVME6000 to attempt to assert SYSFAIL* on the VMEbus. The GCSR bit Inhibit SYSFAIL (ISF), described in Chapter 5, enables the chip to cause SYSFAIL* to be asserted as a result of the state of BRDFAIL. In addition, when the bit is set, the chip asserts the BRDFAIL* signal (which can be used, for example, to turn on a fail LED). Clearing BRDFAIL to 0 will negate the chip's contribution to the assertion of SYSFAIL*, as well as its contribution to the assertion of the BRDFAIL* signal.

The BRDFAIL bit is set to one by SYSRESET* or PRESET*, signaling to the system that the board is not yet ready to participate in normal system operation. The bit should then be cleared by software after the board is ready to participate, typically at the successful completion of power-up self-test procedures.

Bit 3 - ROBIN - Round Robin Select
The ROBIN bit configures the VMEbus arbiter's arbitration mode. ROBIN=1 forces the Round Robin mode. ROBIN=0 forces the priority mode.

4.4 VMEbus REQUESTER

This register configures the MVME6000's VMEbus Requester. It allows software to select the active request level, to configure the conditions under which control of the bus will be relinquished, and the conditions under which the chip can request the bus.

Table 4-3. Requester Configuration Register

PA07-00	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
$03	DWB S0, P0	DHB S0, P0	RONR S0, P0	RWD S0, P0	RNEVER S0, P0	- - -	RQLEV1 S1, P1	RQLEV0 S1,P1

BIT 0, BIT 1 - RQLEV1, RQLEV0 - Request Level 1 and 0

These control bits configure the VMEbus Requester as shown in Table 4-4 below.

Table 4-4. Selecting The Bus Request Level

RQLEV1	RQLEV0	LEVEL
0	0	0
0	1	1
1	0	2
1	1	3

NOTE

Writes to REQLEV1,0 do not change the actual requester level until the MVME6000 has VMEbus mastership and then releases it. This means that there are times when the value written into REQLEV1,0 does not match the current requester level (the request level is lagging). During such times, reads to REQLEV1,0 reflect the actual requester level, not the value written into REQLEV1,0.

BIT 3 - RNEVER - Release Never

Setting this bit to 1 prevents the requester from releasing the VMEbus. However, unlike the DWB control bit described below, setting the RNEVER bit does not cause the requester to request the VMEbus. Clearing the RNEVER bit allows the requester to relinquish the VMEbus in accordance with the other control bits of the Requester Configuration register.

Bit 4 - RWD - Release When Done

The RWD bit allows software to configure the requester's release mode. When the bit is set to 1, and if RNEVER and DWB are both cleared to 0, the requester will release the bus after the local processor completes a VMEbus cycle.

When this bit is cleared to 0, and if RNEVER and DWB are both cleared to 0, the requester will operate in the Release-On-Request (ROR) mode. After acquiring control of the VMEbus, it will maintain control until it detects another request pending on the VME. It will then relinquish the bus.

Bit 5 - RONR - Request On No Request
The RONR bit controls the manner in which the MVME6000 requests the VMEbus. When the bit is set and the MVME6000 is bus master and gives it up, the MVME6000 will not request the VMEbus again until it detects the bus request (BR*) signal, on its level, negated. When the MVME6000 detects BR* negated, it waits another 2 CLK periods before asserting its contribution to BR*.

NOTE

In order for the MVME6000 to detect the negation of BR*, BR* must remain negated for at least 1.5 CLK periods.

Bit 6 - DHB - Device Has Bus
The DHB status bit is 1 when the MVME6000 is VMEbus master and 0 when it is not.

Bit 7 - DWB - Device Wants Bus
Setting the DWB control bit to 1 causes the MVME6000 to request the bus. Once bus mastership has been obtained, it will not be relinquished until after the DWB and the RNEVER bits are both cleared to 0.

4.5 MASTER CONFIGURATION REGISTER

Table 4-5. Master Configuration Register

PA07-00	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
$05	DDSACK S0	020 S0	MASWP S0	CFILL S0	MASUAT S0	MASA16 S0	MASA24 S0	MASD16 S0

Bit 0 - MASD16 - Master D16
The MASD16 bit, in conjunction with the DAT16* input signal line, allows both software and hardware to dynamically configure the data transfer capabilities of the master. A summary of how the MVME6000 uses the MASD16 bit in conjunction with other signal and control bits is shown in Chapter 3, Tables 3-3 and 3-4.

If both the MASD16 bit is cleared and the DAT16* signal is negated, the MVME6000's master will have the D32, D16, and D08 data transfer capability. This allows it to execute a single-byte, double-byte, or quad-byte cycle on the VMEbus, as requested by the local processor.

When either the control bit is set or the signal line is asserted, the master will only have the D16 and D08 capability, and the VMEbus transfers will be limited to single-byte and double-byte aligned transfers.

While providing flexibility in mixing 16-bit and 32-bit slaves in the system, the above feature does not constitute full support of dynamic bus sizing. To do so, the responding slave must dynamically inform the master during the cycle what its data transfer capability is, and this is not possible with the VMEbus. Rather, the DAT16* input signal line is driven by on-board decode logic which was pre-configured to identify areas in the global memory map as containing D16 and D08(EO) VMEbus slaves. The MVME6000 asserts the two data acknowledge lines PDSACK0* and PDSACK1*, and enables data and swap buffers as required by the data transfer capabilities it is configured to have. This allows the local processor to correctly identify the placement of valid data during read cycles, and to determine when more cycles are required to complete its intended data transfer.

Bit 1 - MASA24 - Master A24

The MASA24 bit operates in conjunction with the input signal ADR24* to dynamically configure the addressing capability of the MVME6000's master. If either the MASA24 bit is set or the ADR24* signal is asserted, the master will drive one of the standard (24-bit) address modifier codes during VMEbus cycles (unless the master is configured to use the Master's Address Modifier register as described in paragraph 4.9). The specific Standard AM code is determined from the levels that the local processor drives on the three function code lines during the cycle, as shown in Table 4-6.

Bit 2 - MASA16 - Master A16

The MASA16 bit operates in conjunction with the input signal SHORTIO* to dynamically configure the addressing capability of the MVME6000's master. If either the MASA16 bit is set or the SHORTIO* signal is asserted, a Short (16-bit) AM code will be used, regardless of the state of the MAS24 bit or ADR24* signal line (unless the master is configured to use the contents of the Master Address Modifier register as described in paragraph 4.9). The specific Short AM code is determined from the levels that the local processor drives on the three function code lines during the cycle, as shown in Table 4-6.

CAUTION

NO ATTEMPT IS MADE TO DETERMINE WHETHER THE COMPUTED A M CODE IS RESERVED OR NOT. IT IS THE RESPONSIBILITY OF THE BOARD DESIGNER TO PREVENT FUNCTION CODES WHICH MIGHT YIELD A RESERVED ADDRESS MODIFIER CODE FROM REACHING THE MVME6000.

Bit 3 - MASUAT - Master Unaligned Transfers

The MASUAT bit allows software to configure the master to provide the UAT data transfer capability. Setting the MASUAT bit to one will configure the master to execute unaligned VMEbus cycles when necessary.

If the bit is cleared, the MC68030 will be acknowledged so as to break the unaligned transfer into multiple aligned cycles.

NOTE

While making it optional for the master to provide the UAT data transfer capability, the IEEE 1014-87 VMEbus standard requires that all D32 slaves support it.

TABLE 4-6. Determining the Master's AM Code

MAA16	SHORTIO*	MASA24	ADR24*	PFC1	PFC2	PFC0	VMEbus Address Modifier						
							AM5	AM4	AM3	AM2	AM1	AM0	Code
0	H	0	H	L	L	H	L	L	H	L	L	H	$09
0	H	0	H	L	H	L	L	L	H	L	H	L	$0A
0	H	0	H	H	L	H	L	L	H	H	L	H	$0D
0	H	0	H	H	H	L	L	L	H	H	H	L	$0E
1	X	X	X	L	L	H	H	L	H	L	L	H	$29
1	X	X	X	L	H	L	H	L	H	L	H	L	$2A
X	L	X	X	L	L	H	H	L	H	L	L	H	$29
X	L	X	X	L	H	L	H	L	H	L	H	L	$2A
1	X	X	X	H	L	H	H	L	H	H	L	H	$2D
1	X	X	X	H	H	L	H	L	H	H	H	L	$2E
X	L	X	X	H	L	H	H	L	H	H	L	H	$2D
X	L	X	X	H	H	L	H	L	H	H	H	L	$2E
0	H	1	X	L	L	H	H	H	H	L	L	H	$39
0	H	1	X	L	H	L	H	H	H	L	H	L	$3A
0	H	1	X	H	L	H	H	H	H	H	L	H	$3D
0	H	1	X	H	H	L	H	H	H	H	H	L	$3E
0	H	X	L	L	L	H	H	H	H	L	L	H	$39
0	H	X	L	L	H	L	H	H	H	L	H	L	$3A
0	H	X	L	H	L	H	H	H	H	H	L	H	$3D
0	H	X	L	H	H	L	H	H	H	H	H	L	$3E

H = High, L = Low, X = Don't Care

Note that AM2, 1, 0 track FC2, 1, 0

Bit 4 - CFILL - Cache Fill

The CFILL bit allows software to force the VMEbus read cycle to access 4 bytes, but only if the accessed memory can support it. Tables 3-3 and 3-4 in Chapter 3 show how the MVME6000 uses the CFILL in conjunction with other control bits and signals to determine the VMEbus cycle.

Bit 5 - MASWP - Master Write-Posting

Setting the MASWP bit speeds up local processor writes to the VMEbus. It should be used with caution. When MASWP (Master Write-Processing) is set, local processor write cycles to the VMEbus are acknowledged by the MVME6000 before they have finished on the VMEbus. The MVME6000 finishes the write cycles on its own, allowing the local processor to continue with new cycles. If the SLVEN bit is cleared (slave disabled), the MVME6000 acknowledges VME writes before it has obtained VMEbus mastership. If the SLVEN bit is set, it waits until it has obtained VMEbus mastership.

CAUTION

THE MPU IS NOT NOTIFIED VIA BERR* IF AN ERROR OCCURS WHILE THE MVME6000 IS FINISHING A WRITE-POSTED CYCLE. THE MVME6000 CAN BE PROGRAMMED TO INTERRUPT THE MPU IN THIS EVENT. THIS INTERRUPT NOTIFICATION COULD BE WELL AFTER THE OCCURRENCE OF THE ERROR. IT IS THE USER'S RESPONSIBILITY TO DETERMINE WHETHER THIS FEATURE IS USEFUL AND DURING WHAT PHASE OF THE SYSTEM'S INITIALIZATION SEQUENCE IT MIGHT BE ENABLED.

Bit 6 - 020 - Local CPU is a 68020

When this bit is cleared the MVME6000 asserts PBERR* and PHALT* to break RMC bus lockups. When it is set, it asserts only PBERR* to break RMC bus lockups. The reason for this is as follows: When the local MPU attempts to execute a VMEbus cycle at the same time that another VMEbus master is attempting to access local resources, a lockup situation occurs. The local MPU will not relinquish the local bus until the VMEbus bound cycle finishes. But the VMEbus bound cycle cannot finish until the MPU releases the local bus so that the other VMEbus master can finish its cycle. Normally, this lockup is broken by the MVME6000 asserting the PBERR* and PHALT* together, signaling to the local MPU that it should abort the current cycle, relinquish local bus mastership, and retry the aborted cycle later. However, if the local processor is an MC68020 and it is asserting the RMC* pin during its VMEbus bound cycle, it will not give up the local bus before it attempts to retry the aborted cycle. To get it to give up the local bus the MVME6000 asserts PBERR* without asserting PHALT*, causing the MC68020 to take a BERR* exception. If the local processor is an MC68030 that is trying the RMC cycle, the retry is sufficient to cause it to give up local bus mastership only if it is in the first cycle of the RMC sequence. If it is not in the first cycle of the sequence, the MC68030 does not give up the local bus.

Bit 7 - DDTACK

When this bit is set, extra delay is added between the assertion of DTACK* and PDSACK1*/PDSACK0* for VMEbus master cycles. This is needed when the MVME6000 is used with a 33MHz MC68020/MC68030 and the PDSACKX* signals are connected directly (as they normally are) to the 020/030's DSACKX* pins.

4.6 SLAVE CONFIGURATION REGISTER

This register configures the slave to operate in the environment that the local bus provides.

Table 4-7. Slave Configuration Register

PA07-00	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
$07	SLVEN S0	---	SLVWP S0	---	---	---	---	SLVD16 S0

CAUTION

THE BITS IN THE SLAVE CONFIGURATION REGISTER MUST BE CHANGED ONLY WHEN THE MVME6000 HAS CONTROL OF THE VMEbus. THE RECOMMENDED PROCEDURE FOR CHANGING THE SLAVE'S CONFIGURATION IS AS FOLLOWS:

(A) IN THE REQUESTER CONFIGURATION REGISTER, SET THE DWB BIT TO 1.
(B) READ THE DHB STATUS BIT UNTIL IT IS 1.
(C) CHANGE THE SLAVE CONFIGURATION REGISTER.
(D) CLEAR DWB TO 0 (IF THERE IS NO NEED FOR THE BUS).

Bit 0 - SLVD16 - Slave D16

Setting SLVD16 to one configures the MVME6000's slave to provide only the D08(EO) and D16 data transfer capabilities. It is typically set when the local bus is only 16-bits wide. Clearing the SLVD16 bit to 0 configures the MVME6000's slave to provide the D08(EO), D16, and D32/UAT data transfer capabilities. The effect of the SLVD16 control bit on the operation of the slave is shown in Tables 3-5 and 3-6 in Chapter 3.

Bit 5 - SLVWP - Slave Write-Posting

Setting the SLVWP bit speeds up VMEbus writes to other onboard resources. When SLVWP is set VMEbus write cycles to onboard resources are acknowledged by the MVME6000 before the data has actually been written into the device. This allows the VMEbus master to end its cycle quickly, causing the MVME6000 to complete the write on its own.

CAUTION

THE EXECUTION OF WRITE POSTING OPERATIONS TO MEMORY ASSUMES THAT WRITE CYCLES WILL COMPLETE SUCCESSFULLY, I.E., THE WRITE ACCESS WILL NOT RESULT IN A BERR* ACKNOWLEDGE. IT IS UP TO THE USER TO DETERMINE WHETHER THIS MODE OF OPERATION IS DESIRED.

Bit 7 - SLVEN - Slave Enable
Setting the SLVEN control bit to 1 enables the MVME6000's slave.

4.7 TIMER CONFIGURATION REGISTER

This register is used to configure the time-out periods of the four timers contained in the MVME6000. Two of the timers, the Arbitration Timer and the Bus Timer, are only active when the MVME6000 is configured to operate as the VMEbus System Controller. The other two, the Access Timer and the Local Bus Timer, can be enabled regardless of the state of the SCON* input signal.

Table 4-8. Timer Configuration Register

PA07-00	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
$09	- - -	ARBTO S1	VBTO1 S1	VBTO0 S0	ACTO1 S1, P1	ACTO0 S1, P1	LBTO1 S1, P1	LBTO0 S1, P1

Bits 0,1 - LBTO0, LBTO1 - Local Bus Time-Out
The Local Bus Timer allows the local processor to recover from a lockup condition when a local resource does not respond to the cycle. When a local time-out occurs, the MVME6000 asserts the PBERR* output signal, and the LBTO bit in the Bus Error Status register is set. The Local Bus Timer starts timing when DS* is activated and stops when one of the following occurs: (1) DS* is negated, (2) VMESEL* is asserted, or (3) PIACK* is asserted. The time-out period of the local bus timer is shown in Table 4-9.

CAUTION

TAKE CARE NOT TO CHANGE THE STATE OF VMESEL* OR PIACK* AFTER 3/4 OF THE LOCAL TIME-OUT HAS EXPIRED (IF DS* IS STILL ASSERTED).

Table 4-9. Programming the Local Bus Timer

LBTO1	LBTO0	Time-Out Period
0	0	1024 • T
0	1	2048 • T
1	0	4096 • T
1	1	Timer disabled

T= Period of the CLK pin.

Bits 2,3 - ACTO0, ACTO1 - Access Time-Out
The VMEbus Access Timer allows the local processor to recover from a cycle it can't complete because the MVME6000 can't use the VMEbus. The timer starts when the DS* signal is asserted along with either VMESEL* or PIACK*. The timer stops counting when VMEbus mastership is attained and all conditions are met to allow the MVME6000 to assert DS1*/DS0*. If a time-out does occur, the MVME6000 asserts PBERR* to the local processor. The time-out period of the Access Timer is encoded as shown in Table 4-10.

Table 4-10. Programming the VMEbus Access Timer

ACTO1	ACTO0	Time-Out Period
0	0	1024 • T
0	1	16,384 • T
1	0	524,290 • T
1	1	Timer disabled

T= Period of the CLK pin.

Bits 4,5 - VBTO0, VBTO1 - VMEbus Time-Out

The VMEbus Bus Timer allows the local processor to recover from cycles that cannot complete because no VMEbus resource responds to them. The timer starts counting when DS1* or DS0* is asserted on the VMEbus. It stops counting when both DS1* and DS0* are negated. If a VMEbus time-out occurs, the MVME6000 asserts BERR*. The time-out period of the VMEbus Bus Timer is encoded as shown in Table 4-11.

Table 4-11. Programming the VMEbus Bus Timer

VBTO1	VBTO0	Time-Out Period
0	0	1024 • T
0	1	2048 • T
1	0	4096 • T
1	1	Timer disabled

T= Period of the CLK pin.

Bit 6, ARBTO - Arbitration Timer

The Arbitration Timer is used to prevent system lockup when no requester claims the VMEbus after the arbiter has issued a bus grant. Setting ARBTO to 1 enables the VMEbus arbitration timer. When enabled, the timer asserts BBSY* if it (BBSY*) has not been asserted within 4096 • T after the MC68030's arbiter issues a bus grant. The timer then maintains BBSY* for at least 90ns (as required by the VMEbus specification) then negates it. This causes the arbiter to arbitrate any pending requests for the bus.

4.8 SLAVE ADDRESS MODIFIER REGISTER

This register allows software to configure which Address Modifier codes VMEbus masters use to access onboard resources. The 8 bits of the register are organized into 3 groups. At least one bit in each group must be set or the Address Modifier used by the master is ignored. Clearing all the register's bits to 0 disables the MVME6000 from decoding the AM code, i.e., it responds to all AM codes. When this is the case, the chip maintains the function code signals PFC2-PFC0 in a high impedance state.

Table 4-12. Slave Address Modifier Register

PA07-00	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
$0B	SUPER S0	USER S0	EXTED S0	STND S0	SHORT S0	BLOCK S0	PRGRM S0	DATA S0

Bit 0, Bit 1, Bit 2 - DATA, PROGRAM, BLOCK

These three bits form the first group which configures the slave's AM code. Setting any of the bits to one enables the slave to respond to cycles as described in the example below. BLOCK should never be set.

Bit 3, Bit 4, Bit 5 - SHORT, STND, EXTED

These three bits form the second group. Setting any of the bits to one enables the slave to respond to cycles as described in the example below.

Bit 6, Bit 7 - USER, SUPER

These bits form the third group. Setting any of the bits to one enables the slave to respond to cycles as described in the example below.

EXAMPLE: If the SUPER, STND and DATA control bits are set to 1, the only AM code accepted is $3D: Standard Supervisor Data access. When more than one bit is set in a group, the accepted AM codes will include all permutations of the bits that are set. For example, if the SUPER, USER, EXTEND, PROGRAM and DATA bits are set, the accepted AM codes are $09, $0A, $0D and $0E. These are Extended User Data access, Extended User Program access, Extended Supervisor Data access and Extended Supervisor Program access.

CAUTION

ALTHOUGH ALL BITS IN THE SLAVE ADDRESS MODIFIER REGISTER MAY BE CHANGED DYNAMICALLY, THEY MUST ONLY BE CHANGED WHEN THE MVME6000 HAS CONTROL OF THE VMEbus. A POSSIBLE PROCEDURE FOR CHANGING THE SLAVE ADDRESS MODIFIER REGISTER IS GIVEN BELOW.

IN THE REQUESTER CONFIGURATION REGISTER:

(A) SET THE DWB CONTROL BIT TO 1.
(B) READ THE DHB STATUS BIT UNTIL IT IS 1.
(C) CHANGE THE SLAVE ADDRESS MODIFIER REGISTER.
(D) CLEAR DWB TO 0 (IF THERE IS NO NEED FOR THE BUS).

4.9 MASTER ADDRESS MODIFIER REGISTER

This register allows software to program the Address Modifier code that is driven by the MVME6000's master during a VMEbus cycle.

Table 4-13. Master Address Modifier Register

PA07-00	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
$0D	AMSEL S0	- - -	AM5 S0	AM4 S0	AM3 S0	AM2 S0	AM1 S0	AM0 S0

Bits 0-5 - AM0-AM5 - Address Modifier Code.

These six bits, in conjunction with AMSEL, allow software to dynamically select the address space that the master accesses during the VMEbus cycle. Setting any of these six bits to one will cause the master to drive the corresponding address modifier line to high (if AMSEL is set to 1). Clearing any of the bits to 0 will cause the master to drive the corresponding line to low (if AMSEL is set to 1).

Bit 7 - AMSEL - Address Modifier Select

Software uses the AMSEL control bit to define the source of the AM code driven by the master during a VMEbus cycle. Setting the bit to 1 causes the master to drive the contents of the lower 6 bits onto the address modifier lines. The value stored in this register is not checked for reserved or illegal AM codes. Clearing the AMSEL bit causes the master to dynamically determine the AM code.

NOTE

When the chip is selected to execute a VMEbus cycle, and at the same time the processor drives all three function code lines to 1, the contents of bits 0-5 of the register will be driven onto AM0-AM5, regardless of the state AMSEL.

4.10 INTERRUPT HANDLER MASK REGISTER

This register is used to enable the local processor to respond to specific VMEbus interrupt requests. When the Interrupt Handler detects an interrupt on one of the enabled interrupt lines, it responds by requesting the local processor to initiate an interrupt acknowledge cycle. Setting any of bits 1 through 7 unmasks an interrupt request from the VMEbus IRQ signal at the corresponding level. Clearing the bit to 0, masks the corresponding interrupt level. Only one VMEbus master is allowed to handle each level of VMEbus IRQ. The software should set these bits accordingly.

Table 4-14. Interrupt Handler Mask Register

PA07-00	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
$0F	IEN7 S0, P0	IEN6 S0, P0	IEN5 S0, P0	IEN4 S0, P0	IEN3 S0, P0	IEN2 S0, P0	IEN1 S0, P0	- - -

4.11 UTILITY INTERRUPT MASK REGISTER

This register enables the chip's Interrupt Handler to respond to specific utility interrupt requests. If the Interrupt Handler detects an interrupt request from an enabled function, it responds by requesting the local processor to initiate an interrupt acknowledge cycle.

Table 4-15. Utility Interrupt Mask Register

PA07-00	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
$11	WPPERRN S0, P0	SFIEN S0, P0	SIGHEN S0, P0	LM1EN S0, P0	IACKEN S0, P0	LM0EN S0, P0	SIGLEN S0, P0	- - -

Bit 1 - SIGLEN - Signal Low Interrupt Enable

As described in Chapter 5, the GCSR provides two global attention interrupt bits: SIGLP and SIGHP, which allow other VMEbus masters to interrupt the local processor on a low priority (level 1) and on a high priority (level 5). Setting the SIGLEN control bit to 1 unmasks the SIGLP interrupt.

Bit 2 - LM0EN - Location Monitor 0 Interrupt Enable

As described in Chapter 5, the GCSR provides 4 location monitors. Location monitors 0 and 1 cause a local interrupt when the VMEbus address they are configured to monitor is accessed. The LM0EN control bit allows software to mask the interrupt request when an access is detected to the address monitored by location monitor 0. Setting LM0EN to 1 unmasks the interrupt. Local interrupt levels are shown in Table 4-16.

Bit 3 - IACKEN - IACK Interrupt Enable

The MVME6000 allows software to program the Interrupt Handler to generate a local interrupt after it concludes a VMEbus IACK cycle. Setting IACKEN to 1 enables the IACK interrupt. Local interrupt levels are shown in Table 4-16.

Bit 4 - LM1EN - Location Monitor 1 Interrupt Enable

As described in Chapter 5, the GCSR provides 4 location monitors. Two of them, location monitor 0 and 1, cause a local interrupt when the VMEbus address they are configured to monitor is accessed. The LM1EN control bit allows software to mask the interrupt request when an access is detected to the address that is monitored by location monitor 1. Setting LM1EN to 1 unmasks the interrupt.

Table 4-16. Utility Interrupts And Their Assigned Levels

UTILITY INTERRUPT	ASSIGNED PRIORITY
SIGLP	Level 1
LM0	Level 2
IACK	Level 3
LM1	Level 4
SIGHP	Level 5
SYSFAIL	Level 6
WPBERR	Level 7

Bit 5 - SIGHEN - Signal High Interrupt Enable
As described in Chapter 5, the GCSR provides a global high priority attention interrupt bit - SIGHP - which allows other VMEbus masters to interrupt the local processor. Setting SIGHEN to 1 unmasks the SIGHP interrupt. Local interrupt levels are shown in Table 4-16.

Bit 6 - SFIEN - SYSFAIL Interrupt Enable
Setting SFIEN to 1 enables a low level on the VMEbus SYSFAIL* line to cause an interrupt to the local processor. Local interrupt levels are shown in Table 4-16.

Bit 7 - WPERREN - Write Posting Bus Error Enable
The MVME6000 allows software to configure the VMEbus master to operate in the write-posting mode; i.e., acknowledge the local processor's VMEbus bound write cycle before the chip has executed the cycle on the VMEbus. If the chip encounters a VMEbus error as it attempts to complete the write-posed cycle, it signals the local processor via level 7 interrupt if the WPERREN bit is set.

4.12 UTILITY INTERRUPT VECTOR REGISTER

This register provides the local CPU with a unique vector for each of the utility interrupts.

Table 4-17. Utility Interrupt Vector Register

PA07-00	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
$13	UVB7 S0, P0	UVB6 S0, P0	UVB5 S0, P0	UVB4 S0, P0	UVB3 S0, P0	UID2	UID1	UID0

NOTE

The assigned level of each utility interrupt is the same as its assigned ID. This was implemented by reflecting the state of address lines A01-A03 (that the local CPU drives when it acknowledges an interrupt) into bits 0-2 of the Utility Vector register. So when accessing this register in the course of a normal CPU read cycle, bits 0-2 yield the register's offset address.

CAUTION

THE CONTENTS OF THE UTILITY INTERRUPT REGISTER MUST NOT BE ALTERED WHILE ONE OF THE UTILITY INTERRUPTS IS ACTIVE.

Table 4-18. Encoding The Utility Interrupt ID

Utility Interrupt Source	Bit 2	Bit 1	Bit 0
SIGLP	0	0	1
LM0	0	1	0
IACK	0	1	1
LM1	1	0	0
SIGHP	1	0	1
SYSFAIL	1	1	0
WPBERR	1	1	1

Bits 0-2, UID0, UID1, UID2 - Utility Interrupt ID 0-2

The lower three bits of the Utility Interrupt Vector register are encoded by the MVME6000 to identify the function that caused the utility interrupt request as shown in Table 4-18.

Bits 3-7 - UVB3 through UVB7 - Utility Vector Base bits

The upper five bits of the register are programmable by software to supply a base for the vector provided in the course of acknowledging one of the utility interrupts. These bits are cleared by any reset.

4.13 INTERRUPT REQUEST REGISTER

This register is used to configure the interrupt request line that the Interrupter asserts to request an interrupt on the VMEbus.

Table 4-19. Interrupt Request Register

PA07-00	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
$15	---	---	---	---	---	IL2 S0	IL1 S0	IL0 S0

Bit 0, bit 1, bit 2 - IL0, IL1, IL2

The three interrupt level select bits are encoded as shown in Table 4-20. Writing a non zero value to these 3 bits will cause the interrupter to assert the corresponding VMEbus IRQ line. Since the interrupter operates in the Release-On-Acknowledge (ROAK) mode, the Interrupt Request register is cleared, negating the IRQ* line when the chip responds to a VMEbus interrupt acknowledge cycle.

Table 4-20. Configuring The Interrupt Request Level

INTERRUPT REQUEST Line driven	IL2	IL1	IL0
None	0	0	0
IRQ1*	0	0	1
IRQ2*	0	1	0
IRQ3*	0	1	1
IRQ4*	1	0	0
IRQ5*	1	0	1
IRQ6*	1	1	0
IRQ7*	1	1	1

CAUTION

ONCE THE BITS OF THE INTERRUPT REQUEST REGISTER ARE SET TO DRIVE ONE OF THE IRQ* LINES, THEY MUST NOT BE CHANGED. THE THREE BITS MAY BE CHANGED ONLY WHEN THEY ARE ALL CLEARED, SIGNIFYING THAT THE PREVIOUS INTERRUPT REQUEST HAS BEEN SERVICED.

4.14 VMEbus STATUS/ID REGISTER

This register allows software to dynamically program the status/ID that the interrupter provides during an interrupt acknowledge cycle.

Table 4-21. VMEbus Status/ID Register

PA07-00	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
$17	D07 S0	D06 S0	D05 S0	D04 S0	D03 S1	D02 S1	D01 S1	D00 S1

4.15 BUS ERROR STATUS REGISTER

This register allows the local processor to determine the cause of a bus error condition flagged by the MVME6000. Reading the register causes all of its bits to be cleared to 0. The Bus Error Status register is designed to only indicate the cause of the latest bus error condition; i.e., when there is cause to set any of the bits, all other bits are cleared.

Table 4-22. Bus Error Status Register

PA07-00	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
$19	- - -	- - -	- - -	- - -	RMCERR S0, P0	VBERR S0, P0	ACTO S0, P0	LBTO S0, P0

Bit 0 - LBTO - Local Bus Time Out

When set, this status bit indicates that the local timer has timed out.

Bit 1 - ACTO - Access Time Out

When set, this status bit indicates that the VMEbus access timer has timed out.

Bit 2 - VBERR - VMEbus Bus Error

When set, this status bit indicates that the VMEbus BERR* signal was asserted in the course of a non write-posted cycle that was initiated by the MVME6000. It should be noted that this bit will not be set if the VMEbus Timer timed out in response to a VMEbus cycle that was initiated by another VMEbus master.

Bit 3 - RMCERR - RMC Lock Error

When set, this bit indicates that an access lock-up between the local bus and the VMEbus during an RMC cycle was broken by the MVME6000 asserting PBERR*. Refer to the description of the 020 bit in the Master Configuration Register.

4.16 GCSR BASE ADDRESS CONFIGURATION REGISTER

This register allows software to set the base address of the Global Control and Status Register set (GCSR) in the VMEbus Short I/O map.

The value contained in bits 0-3 of this register will configure bits 4-7 of the GCSR's base address. Address lines A08-A15 must be decoded by an external decoder and fed into the MATCHMPR* signal pin. Bits 1-3 of the VMEbus address select the specific registers in the GCSR.

Table 4-23. GCSR Base Address Configuration Register

PA07-00	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
$1B	- - -	- - -	- - -	- - -	GCCSRA7 S1	GCSRA6 S1	GCSRA5 S1	GCSRA4 S1

Table 4-24. MVME6000 GCSR As Viewed From The VMEbus

	GCSRA7-4
$0	$XX00-$XX0F
$1	$XX10-$XX1F
$2	$XX20-$XX2F
$3	$XX30-$XX3F
$4	$XX40-$XX4F
$5	$XX50-$XX5F
$6	$XX60-$XX6F
$7	$XX70-$XX7F
$8	$XX80-$XX8F
$9	$XX90-$XX9F
$A	$XXA0-$XXAF
$B	$XXB0-$XXBF
$C	$XXC0-$XXCF
$D	$XXD0-$XXDF
$E	$XXE0-$XXEF
$F	DOES NOT RESPOND

CHAPTER 5

GLOBAL CONTROL AND STATUS REGISTER SET

5.1 GENERAL

The Global Control and Status Register set (GCSR) allows the board to operate in an environment that includes more than one microprocessor based board. It provides facilities that let other CPU boards interrupt the local processor, communicate with it, determine its operational status, and disable it. The GCSR also contains four location monitors which allow a single CPU to broadcast a signal to other CPU boards. All registers of the GCSR are accessible to both the local processor and to other VMEbus masters. Table 5-1 lists the various registers in the GCSR. Figure 5-1 details the organization of the GCSR and how its bits are affected by SYSRESET* and PRESET*.

5.2 ADDRESSING THE GCSR

The GCSR is located in the Short I/O map, and is accessed in the Supervisor space, AM code $2D. The chip allows access to the GCSR to both the local processor over the local bus, and to other VMEbus masters over the VMEbus.

The chip allows access to the GCSR over the local bus when its input signal CS* is asserted, and the address carried on PA00-PA07 selects one of the registers of the GCSR. The local offset address of the registers of the GCSR is shown in Table 5-1.

The chip allows other VMEbus masters to access the GCSR over the VMEbus when it detects its input signal MATCHGCSR* asserted. The specific register that is then accessed will be determined by the address carried on VMEbus address lines A01-A15. The addressing convention for the GCSR allows to group the system's boards into separate groups with up to 15 boards in each group.

Address lines A08-A15 form a group address. They must be decoded by an external decoder, which will select the GCSR by asserting the MATCHGCSR* input signal. All the boards in the group must have the same group address.

Address lines A04-A07 form the board address within the group. They are compared against bits 0-3 of the GCSR Base Address Register in the LCSR (see section 4.16). The board address of each of the boards in the group must be set to a unique value. Board address $F (i.e., address lines A04- A07 are all ones) is reserved as a Group Broadcast address and should not be used.

Address lines A01-A03 select the specific address register in the GCSR.

THE GCSR

Table 5-1. The Registers of the GCSR

GLOBAL OFFSET	LOCAL OFFSET	REGISTER
$01	$21	Global Register 0
$03	$23	Global Register 1
$05	$25	Board Identification Register
$07	$27	General Purpose Register 0
$09	$29	General Purpose Register 1
$0B	$2B	General Purpose Register 2
$0D	$2D	General Purpose Register 3
$0F	$2F	General Purpose Register 4

REGISTER OFFSET	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
$01	LM3 S1	LM2 S1	LM1 S1	LM0 S1	CHIPID3 0	CHIPID2 0	CHIPID1 0	CHIPID0 1
$03	R&H S0	SCON	ISF S0	BRDFAIL			SIGHP S0	SIGLP S0
$05	BRDID7 S0	BRDID6 S0	BRDID5 S0	BRDID4 S0	BRDID3 S0	BRDID2 S0	BRDID1 S0	BRDID0 S0
$07	GENERAL PURPOSE CONTROL AND STATUS REGISTER 0 S0							
$09	GENERAL PURPOSE CONTROL AND STATUS REGISTER 1 S0							
$0B	GENERAL PURPOSE CONTROL AND STATUS REGISTER 2 S0							
$0D	GENERAL PURPOSE CONTROL AND STATUS REGISTER 3 S0							
$0F	GENERAL PURPOSE CONTROL AND STATUS REGISTER 4 S0							

Figure 5-1. The Organization of the GCSR

5.3 GLOBAL REGISTER 0

Global Register 0 includes four location monitors, and a hardwired chip identification number. The register is accessed at an offset of $01 from the base address of the GCSR. In addition the local processor can access the register at an offset of $21 from the base address of the LCSR.

Table 5-2. Global Register 0

BIT	NAME	STATE UPON SYSRESET*	STATE UPON PRESET*	LOCAL ACCESS	GLOBAL ACCESS
0	CHIPID0	No Change	No Change	Read Only	Read Only
1	CHIPID1	No Change	No Change	Read Only	Read Only
2	CHIPID2	No Change	No Change	Read Only	Read Only
3	CHIPID3	No Change	No Change	Read Only	Read Only
4	LM0	1	No Change	Read & Set	Read & Set
5	LM1	1	No Change	Read & Set	Read & Set
6	LM2	1	No Change	Read & Set	Read & Set
7	LM3	1	No Change	Read & Set	Read & Set

Note: Read & Set means that the bit can only be set to 1 by software.

Bit 0-3 - CHIPID - MVME6000 Identification Number

Bits 0-3 of the register provide a unique identification number for the MVME6000. The MVME6000 presents a hardwired ID of $0001. Other VMEbus interface chips which implement a different software architecture will be assigned other codes.

Bit 4 - LM0 - Location Monitor 0

Location monitor 0 is configured to monitor double-byte accesses to the short I/O address $XXF0, and single-byte accesses to the short I/O address $XXF1, where $XX denotes the group address as described in section 5.2. Table 5-3 shows the various location monitors and their assigned addresses.

When cleared LM0 indicates that an access to address $XXF0 was detected. At such a time utility interrupt level 2 will be requested, if the interrupt is enabled as described in paragraph 4.11.

The bit is set when its interrupt is acknowledged. In addition, the bit can be set under software control.

Bit 5 - LM1 - Location Monitor 1

Location monitor 1 is configured to monitor double-byte accesses to the short I/O address $XXF2, and single-byte accesses to the short I/O address $XXF3, where $XX denotes the group address as described in paragraph 5.2. Table 5-3 shows the various location monitors and their assigned addresses.

When cleared, LM1 indicates that an access to address $XXF1 was detected. At such a time utility interrupt level 4 will be requested, if the interrupt is enabled as described in paragraph 4.11.

The bit is set when its interrupt is acknowledged. In addition, the bit can be set under software control.

Table 5-3. Location Monitors and their Assigned Addresses

LOCATION MONITOR	A15 - A08	A07	A06	A05	A04	A03	A02	A01
LM0	Group Address	H	H	H	H	L	L	L
LM1	Group Address	H	H	H	H	L	L	H
LM2	Group Address	H	H	H	H	L	H	L
LM3	Group Address	H	H	H	H	L	H	H
Reserved	Group Address	H	H	H	H	H	L	L
Reserved	Group Address	H	H	H	H	H	L	H
Reserved	Group Address	H	H	H	H	H	H	L
Reserved	Group Address	H	H	H	H	H	H	H

Bit 6 - LM2 - Location Monitor 2

Location monitor 2 is configured to monitor double-byte accesses to the short I/O address $XXF4, and single-byte accesses to the short I/O address $XXF5, where $XX denotes the group address as described in section 5.2. Table 5-3 shows the various location monitors and their assigned addresses.

When cleared, LM2 indicates that an access to address $XXF2 was detected. The bit is set, under software control.

Bit 7 - LM3 - Location Monitor 3

Location monitor 3 is configured to monitor double-byte accesses to the short I/O address $XXF6, and single-byte accesses to the short I/O address $XXF7, where $XX denotes the group address as described in section 5.2. Table 5-3 shows the various location monitors and their assigned addresses.

When cleared, LM3 indicates that an access to address $XXF3 was detected. The bit is set under software control.

5.4 GLOBAL REGISTER 1

Global register 1 allows other processors in the system to determine what is the operational status of the board, to reset it, and to interrupt it. The register is accessed at an offset of $03 from the base address of the GCSR. In addition it can be accessed by the local processor at an offset of $23 from the base address of the LCSR.

Table 5-4. Global Register 1

BIT	NAME	STATE UPON SYSRESET*	STATE UPON PRESET*	LOCAL ACCESS	GLOBAL ACCESS
0	SIGLP	0	No Change	Read & Clear	Read & Set
1	SIGHP	0	No Change	Read & Clear	Read & Set
2	---	---	---	---	--- ---
3	---	---	---	---	--- ---
4	BRDFAIL	H	H	Read Only	Read Only
5	ISF	0	No Change	Read & Write	Read & Write
6	SCON	No Change	No Change	Read Only	Read Only
7	R&H	0	No Change	Read & Write	Read & Write

Note: Read & Clear = Read and Clear, i.e., can only be cleared to 0 by software.
Read & Set = Read and Set, i.e., can only be set to 1 by software.

Bit 0 - SIGLP - Signal Low Priority
The SIGLP control bit allows other VMEbus masters to alert the local processor by generating a level 1 local. Setting the bit to 1 causes the MVME6000 to request an interrupt. However, the local processor will be interrupted only if the SIGLP interrupt is enabled as described in paragraph 4.11. After using the bit to request an interrupt, the local processor can clear the interrupt and enable further interrupts by writing a 1 into it.

Bit 1 - SIGHP - Signal High Priority
The SIGHP control bit allows other VMEbus masters to alert the local processor by generating a level 4 local interrupt. Setting the bit to 1 causes the MVME6000 to request an interrupt. However, the local processor will be interrupted only if the SIGHP interrupt is enabled as described in paragraph 4.11. After using the bit to request an interrupt, the local processor can clear the interrupt and enable further interrupts by writing a 1 into it.

Bit 4 - BRDFAIL - Board Fail
The BRDFAIL bit is a reflection of the BRDFAIL* input/output signal line. The status bit is set to 1 whenever the signal line is asserted by the MVME6000, or by logic external to the MVME6000. The bit is cleared only when the BRDFAIL signal line is negated.

Bit 5 - ISF - Inhibit SYSFAIL*
The ISF control bit allows other VMEbus master to cause the MVME6000 to release its contribution to the VMEbus SYSFAIL* line. This is provided so that software can determine how many boards have failed. It should be noted that the ISF bit has no effect on the BRDFAIL status bit. Setting the bit to 1 inhibits the MVME6000 from asserting the VMEbus SYSFAIL* line.

Bit 6 - SCON - System Controller On
The SCON status bit is a reflection of the SCON* input signal line. When the SCON* input signal line is grounded, enabling the the MVME6000 to operate as the VMEbus System Controller, the SCON status bit is set to one.

Bit 7 - R&H - Reset and Hold
The R&H bit allows other VMEbus masters to reset the local processor. The local PRESET* line is maintained in its asserted state for as long as the R&H bit is set to 1. This allows a board to be disabled in case it disrupts the system.

CAUTION

IF THE LOCAL PROCESSOR SETS ITS OWN R&H BIT, IT WILL CAUSE ITSELF TO BE MAINTAINED IN A RESET STATE UNTIL SOME OTHER MASTER CLEARS THE BIT TO 0.

THE MVME6000 DOES NOT GUARANTEE A MINIMUM ASSERTION TIME ON THE PRESET* LINE. IT IS LEFT UP TO SOFTWARE TO ENSURE THAT THE R&H BIT HAS BEEN SET FOR THE MINIMUM REQUIRED ASSERTION TIME OF PRESET* BEFORE THE BIT IS CLEARED.

5.5 BOARD IDENTIFICATION REGISTER

The Board Identification register is provided to allow software to establish a way to uniquely identify the boards in the system. The register is accessed at an offset of $05 from the base address of the GCSR. In addition it can be accessed by the local processor at an offset of $25 from the base address of the LCSR.

All the bits of the register are cleared to 0 upon the assertion of SYSRESET* (PRESET* has no effect on it). The register allows both read and write accesses to the local processor. VMEbus masters can only read this register.

5.6 GENERAL PURPOSE CSR 0

General Purpose CSR 0 is accessed at an offset of $07 from the base address of the GCSR. In addition, it can be accessed by the local processor at an offset of $27 from the base address of the LCSR. All the bits of this register are set to 1 upon the assertion of SYSRESET* (PRESET* has no effect on it). The local processor, as well as other VMEbus masters, can both read and write to it.

5.7 GENERAL PURPOSE CSR 1

General Purpose CSR 1 is accessed at an offset of $09 from the base address of the GCSR. In addition, it can be accessed by the local processor at an offset of $29 from the base address of the LCSR. The register is cleared to 0 upon the assertion of SYSRESET* (PRESET* has no effect on it). The local processor, as well as other VMEbus masters, can both read and write to it.

5.8 GENERAL PURPOSE CSR 2

General Purpose CSR 2 is accessed at an offset of $0B from the base address of the GCSR. In addition, it can be accessed by the local processor at an offset of $2B from the base address of the LCSR. The register is cleared to 0 upon the assertion of SYSRESET* (PRESET* has no effect on it). The local processor, as well as other VMEbus masters, can both read and write to it.

5.9 GENERAL PURPOSE CSR 3

General Purpose CSR 3 is accessed at an offset of $0D from the base address of the GCSR. In addition, it can be accessed by the local processor at an offset of $2D from the base address of the LCSR. The register is cleared to 0 upon the assertion of SYSRESET* (PRESET* has no effect on it). The local processor, as well as other VMEbus masters, can both read and write to it.

5.10 GENERAL PURPOSE CSR 4

General Purpose CSR 4 is accessed at an offset of $0F from the base address of the GCSR. In addition, it can be accessed by the local processor at an offset of $2F from the base address of the LCSR. The register is cleared to 0 upon the assertion of SYSRESET* (PRESET* has no effect on it). The local processor, as well as other VMEbus masters, can both read and write to it.

APPENDIX A

ELECTRICAL CHARACTERISTICS

Table A-1. Maximum Ratings

RATING	SYMBOL	VALUE	UNIT
Supply Voltage	Vcc	-0.5 to +7.0	V
Input Voltage	Vin	-0.5 to +5.5	V
Storage Temperature Range	Tstg	-65 to 150	°C

Table A-2. Operating Range

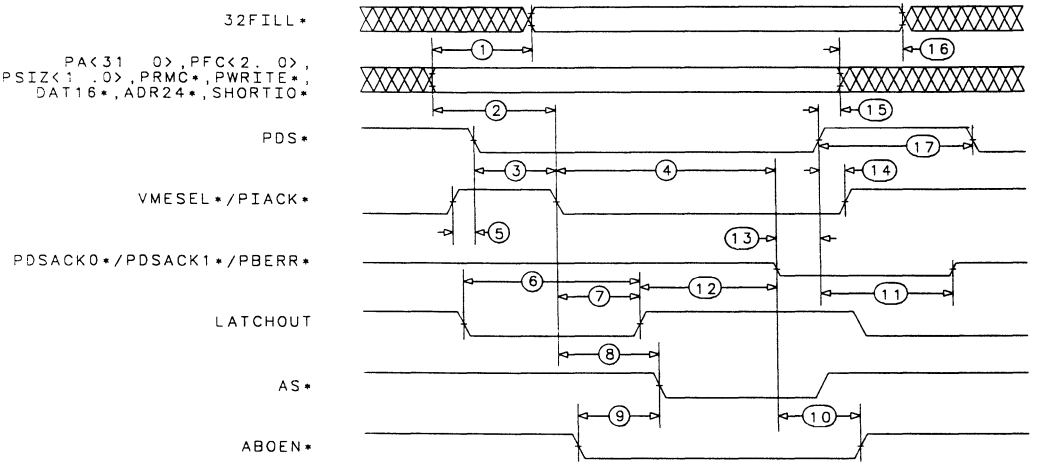
CHARACTERISTIC	SYMBOL	MIN	MAX	UNIT
Temperature	Ta	0	70	°C
Voltage	Vcc	4.75	5.25	Vdc

Table A-3. DC Electrical Characteristics (Over Operating Temperature and Voltage Range)

CHARACTERISTIC	SYMBOL	MIN	MAX	UNIT
Input High Voltage	Vih	2.0		V
Input Low Voltage	Vil		0.8	V
High Level Input Current	Iih		20	uA
Low Level Input Current	Iil		0.6	mA
Hi-Z (Off-state) Leakage Current	Itsi		10	uA
Output High Voltage	Voh			V
Ioh=400uA 32FILL*, ABIN*, ABOUT*, DBIR, DBOUT, IPL0*–IPL2*, ISOEN*, LATCHIN, LATCHOUT, LDBIN*, LOCK*, PA00–PA07, PAS*, PD0-PD7, PDS* PFC0–PFC2, DHB*,PSIZ0-PSIZ1, PWRITE*, SWPEN*, UDBIN*		2.4		
Output Low Voltage	Vol			V
Io l=48mA BRDFAIL*		0.6		
Io l=8mA 32FILL*, ABIN*, ABOUT*, DBDIR, DBOUT*, IPL0*–IPL2*, ISOEN*, LATCHIN, LATCHOUT, PBERR*, LDBIN*, LOCK*, PA00-PA 7, PAS*, PBR*, PD0–PD7, PDS*, PDSACK0*, PDSACK*1*, PFC0–PFC2, DHB*, PHALT*, PRESET*, PSIZ0–PSIZ1, PWRITE*, SWPEN*, UDBIN*		0.5		
Power Dissapation (Ta=0 °C)	Pd		4.5	W
Capacitance (Applies to non-VMEbus signals only)	Cin		20	pF

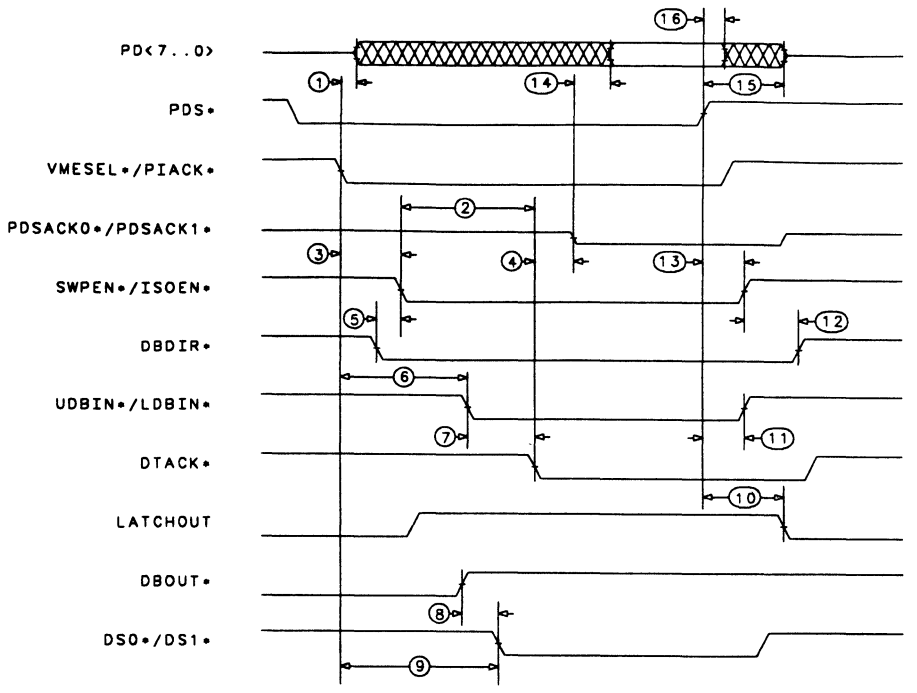
APPENDIX A

Table A-4. VMEbus Master: Times Common to Read, Write, and IACK Cycles



NUM.	CHARACTERISTIC	MIN	TYP	MAX
1	PFC, PSIZ, PRMC, PWRITE, DAT16, SHORTIO valid to 32FILL* valid			85ns
2	PA, PFC, PSIZ, PRMC, PWRITE, DAT16, ADR24, SHORTIO valid to VMESEL*PIACK* low	30ns		
3	PDS* low to VMESEL*/PIACK* low	0ns		
4	VMESEL*/PIACK* low to PDSACK0*/PDSACK1*/PBERR* low	60ns		
5	VMESEL*/PIACK* high to PDS* low	10ns		
6	LATCHOUT pulse width low	40ns		
7	VMESEL*/PIACK* low to LATCHOUT low	60ns		
8	VMESEL*/PIACK* low to AS* low	60ns		
9	ABOEN* low to AS* low	50ns		
10	PDSACK0*/PDSACK1*/PBERR* low to ABOEN* high	0ns		
11	PDS* high to PDSACK0*/PDSACK1*/PBERR* high Z	5ns		60ns
12	LATCHOUT high to PDSACK0*/PDSACK1*/PBERR* low	5ns		
13	PDSACK0*/PDSACK1*/PBERR* low to PDS* high	20ns		
14	PDS* high to VMESEL*/PIACK* high	0ns		
15	PDS* high to PA, PFC, PSIZ, PRMC, PWRITE, DAT16, ADR24, SHORTIO invalid	0ns		
16	PFC, PSIZ, PRMC/PWRITE, DAT16, SHORTIO invalid to 32FILL* i nvalid	5ns		
17	PDS* pulse width high	20ns		

Table A-5. VMEbus Master: Times Specific To Read and IACK Cycles

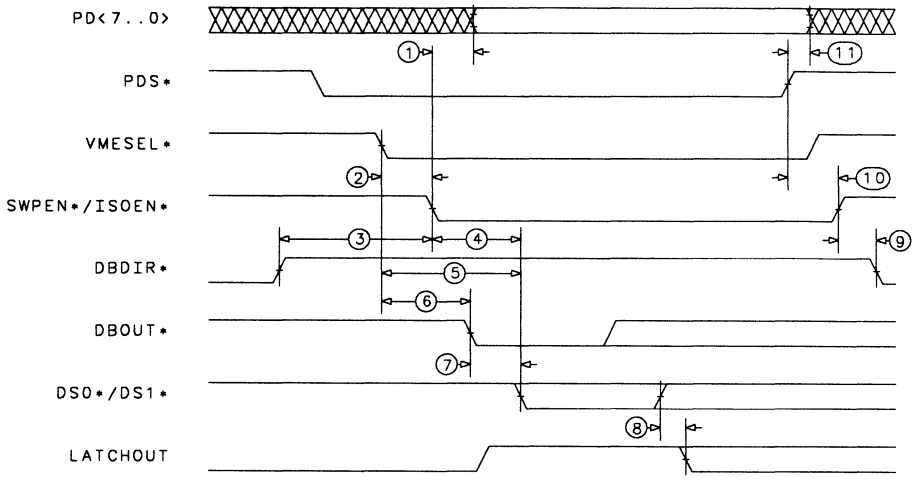


NUM.	CHARACTERISTIC	MIN	TYP	MAX
1	VMESEL*/PIACK* low to PD<7..0> driven	0ns		
2	SWPEN*/ISOEN* low to DTACK* low	30ns		
3	VMESEL*/PIACK* low to SWPEN*/ISOEN* low	0ns		
4	DTACK* low to PDSACK0*/PDSACK1* low (DDSACK=0)	13ns	40ns	
4	DTACK* low to PDSACK0*/PDSACK1* low (DDSACK=1)	25ns	75ns	
5	DBDIR* low to SWPEN*/ISOEN* low	3ns		
6	VMESEL*/PIACK* low to UDBIN*/LDBIN* low	60ns		
7	UDBIN*/LDBIN* low to DTACK* low	0ns		
8	DBOUT* high to DS0*/DS1* low	20ns		
9	VMESEL*/PIACK* low to DS0*/DS1* low (Current Bus Master)		1.5T+40ns	
10	PDS* high to LATCHOUT low	0ns		
11	PDS* high to UDBIN*/LDBIN* high	0ns		45ns
12	SWPEN*/ISOEN* high to DBDIR* high	5ns		
13	PDS* high to SWPEN*/ISOEN* high	0ns		45ns
14	PDSACK0*/PDSACK1* low to PD<7..0> valid (DDSACK=0)			20ns
14	PDSACK0*/PDSACK1* low to PD<7..0> valid (DDSACK=1)			10ns
15	PDS* high to PD<7..0> high Z			60ns
16	PDS* high to PD<7..0> invalid	0ns		

Notes:

1 – During IACK* cycles, VMESEL* should not be asserted.
2 – During READ/WRITE cycles, PIACK* should not be asserted.
3 – T is the CLK period.

Table A-6. VMEbus Master: Times Specific To Write Cycles

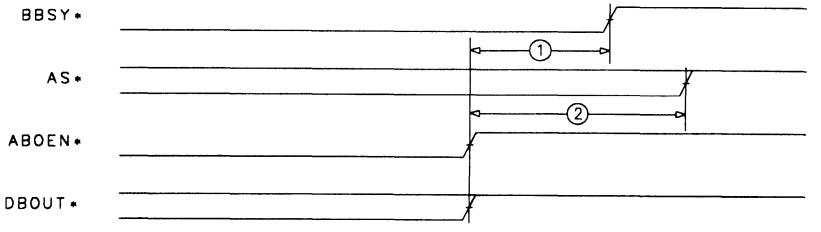


NUM.	CHARACTERISTIC	MIN	TYP	MAX
1	SWPEN*/ISOEN* low to PD<7..0> valid			11ns
2	VMESEL* low to SWPEN*/ISOEN* low	0ns		45ns
3	DBDIR* high to SWPEN*/ISOEN* low	5ns		
4	SWPEN*/ISOEN* low to DS0*/DS1* low	60ns		
5	VMESEL* low to DS0*/DS1* low (current bus master)		2T+40ns	
6	VMESEL* low to DBOUT* low	0ns		
7	DBOUT* low to DS0*/DS1* low	50ns		
8	DS0*/DS1* high to LATCHOUT low	0ns		
9	SWPEN*/ISOEN* high to DBDIR* low	5ns		
10	PDS* high tp SWPEN*/ISOEN* high	0ns		45ns
11	PDS* high to PD<7..0> invalid	0ns		
12	LATCHOUT high to D08–D31, A08–A31 frozen			10ns

Notes:

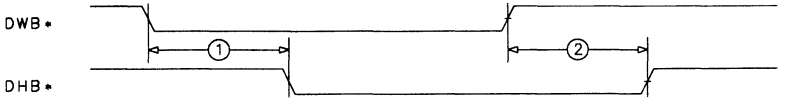
1 – T is the CLK period.

Table A-7. VMEbus Master: Releasing VMEbus Mastership



NUM.	CHARACTERISTIC	MIN	TYP	MAX
1	ABOEN*/DBOUT* high to BBSY* high (AS* already high)	15ns		
2	ABOEN*/DBOUT* high to AS* high (BBSY* already high)	15ns		

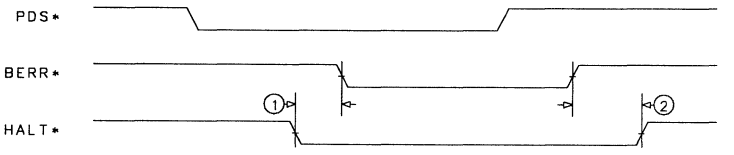
Table A-8. VMEbus Master: DWB*/DHB* Timing



NUM.	CHARACTERISTIC	MIN	TYP	MAX
1	DWB* low to DHB* low	5ns		
2	DWB* high to DHB* high			50ns

APPENDIX A

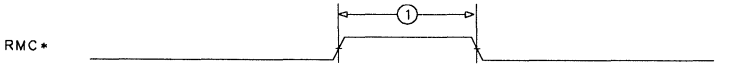
Table A-9. VMEbus Master: Local Bus Retries



NUM.	CHARACTERISTIC	MIN	TYP	MAX
1	HALT* low to BERR* low	60ns		
2	BERR* high to HALT* high	5ns		

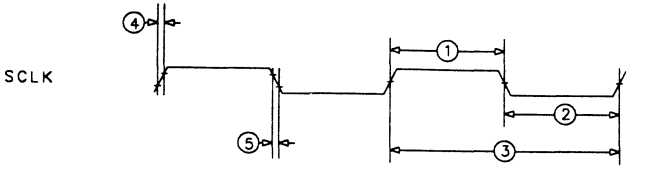
A

Table A-10. VMEbus Master: RMC* Timing



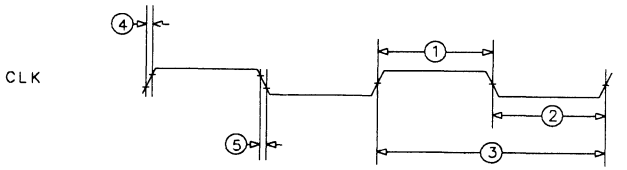
NUM.	CHARACTERISTIC	MIN	TYP	MAX
1	RMC* pulse width high	20ns		

Table A-11. SCLK Waveforms



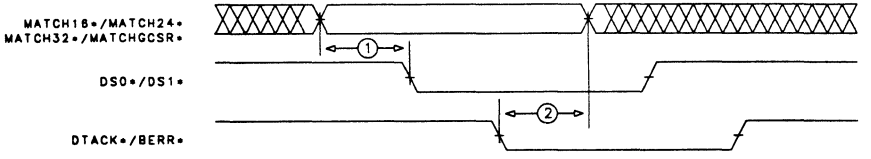
NUM.	CHARACTERISTIC	MIN	TYP	MAX
1	SCLK pulse width high	28ns		35ns
2	SCLK pulse width low	28ns		35ns
3	SCLK cycle time	61.5ns	62.5ns	63.5ns
4	SCLK rise time			5ns
5	SCLK fall time			5ns

Table A-12. CLK Waveforms



NUM.	CHARACTERISTIC	MIN	TYP	MAX
1	CLK pulse width high	40ns		
2	CLK pulse width low	40ns		
3	CLK cycle time	100ns		
4	CLK rise time			5ns
5	CLK fall time			5ns

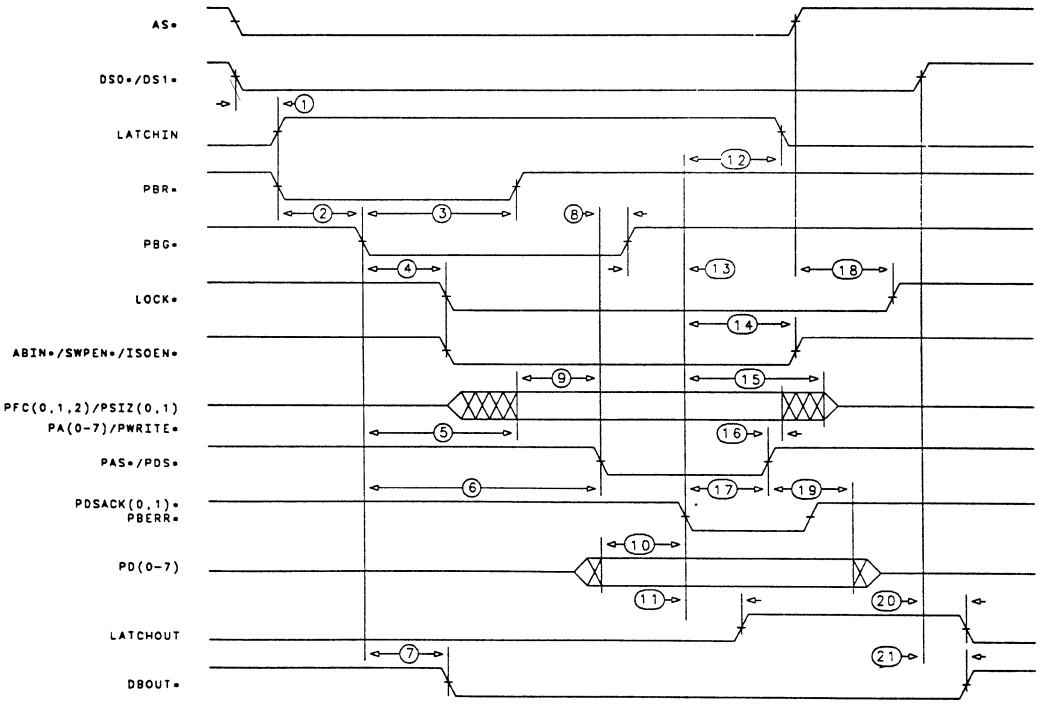
Table A-13. VMEbus Slave: MATCH16*, MATCH24*, MATCH32, MATCHGCSR*, Timing



NUM.	CHARACTERISTIC	MIN	TYP	MAX	UNIT
1	MATCH (16, 24, 32, GCSR)* valid to DS0*/DS1* low	0			ns
2	DTACK*/BERR* low to MATCH (16, 24, 32, GCSR)* invalid	0			ns

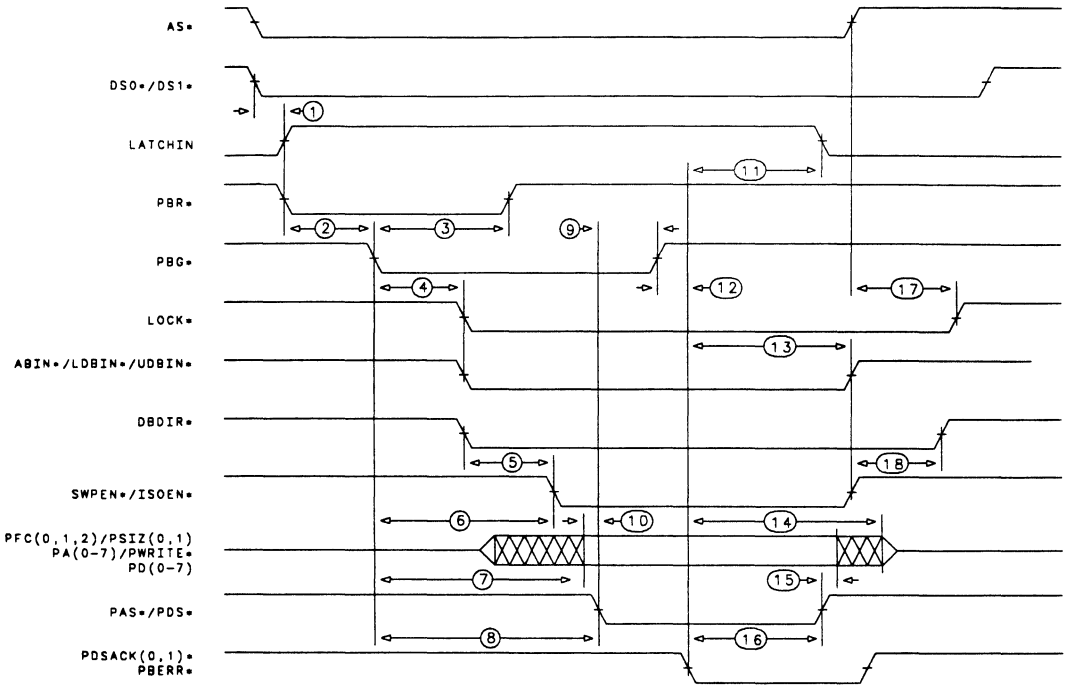
APPENDIX A

Table A-14. VMEbus To Local Bus: Read Cycle



NUM.	CHARACTERISTIC	MIN	TYP	MAX	UNIT
1	DS0*/DS1* low to LATCHIN high / PBR* low	15		95	ns
2	PBR* low to PBG* low	0			ns
3	PBG* low to PBR* high Z	20		100	ns
4	PBG* low to LOCK*/ABIN*/SWPEN*/ISOEN* low	5		35	ns
5	PBG* low to PFC/PSIZ/PA/PWRITE* valid	10		70	ns
6	PBG* low to PAS*/PDS* low	15		90	ns
7	PBG* low to DBOUT* low	10		55	ns
8	PAS*/PDS* low to PBG* high	0			ns
9	PFC/PSIZ/PA/PWRITE* valid to PAS*/PDS* low	5			ns
10	PD valid to PDSACK* low	0			ns
11	PDSACK*/PBERR* low to LATCHOUT high	10		50	ns
12	PDSACK*/PBERR* low to LATCHIN low	15		95	ns
13	PBG* high to PDSACK*/PBERR* low	0			ns
14	PDSACK*/PBERR* low to ABIN*/SWPEN*/ISOEN* high	10		65	ns
15	PDSACK*/PBERR* low to PFC/PSIZ/PA/PWRITE* high Z	20		110	ns
16	PAS*/PDS* high to PFC/PSIZ/PA/PWRITE* invalid	5			ns
17	PDSACK*/PBERR* low to PAS*/PDS* high	10		65	ns
18	AS* high to LOCK* high	5		30	ns
19	PAS*/PDS* high to PD invalid	0			ns
20	DS0*/DS1* high to LATCHOUT low	10		50	ns
21	DS0*/DS1* high to DBOUT* high	5		40	ns

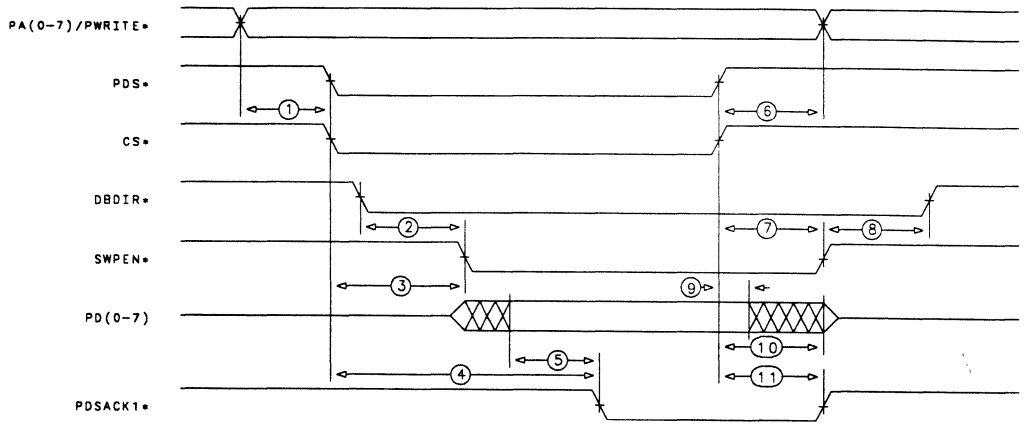
Table A-15. VMEbus To Local Bus: Write Cycle



NUM.	CHARACTERISTIC	MIN	TYP	MAX	UNIT
1	DS0*/DS1* low to LATCHIN high / PBR* low	15		95	ns
2	PBR* low to PBG* low	0			ns
3	PBG* low to PBR* high Z	20		100	ns
4	PBG* low to LOCK*/ABIN*/LDBIN*/UDBIN* low	5		35	ns
5	DBDIR* low to SWPEN*/ISOEN* low	5			ns
6	PBG* low to SWPEN*/ISOEN* low	10		55	ns
7	PBG* low to PFC/PSIZ/PA/PD/PWRITE* valid	10		70	ns
8	PBG* low to PAS*/PDS* low	15		90	ns
9	PAS*/PDS* low to PBG* high	0			ns
10	PFC/PSIZ/PA/PD/PWRITE* valid to PAS*/PDS* low	5			ns
11	PDSACK*/PBERR* low to LATCHIN low	15		95	ns
12	PBG* high to PDSACK*/PBERR* low	0			ns
13	PDSACK*/PBERR* low to ABIN*/LDBIN*/UDBIN*/ SWPEN*/ISOEN* high	10		65	ns
14	PDSACK*/PBERR* low to PFC/PSIZ/PA/PD/PWRITE* high Z	20		110	ns
15	PAS*/PDS* high to PFC/PSIZ/PA/PD/PWRITE* invalid	5			ns
16	PDSACK*/PBERR* low to PAS*/PDS* high	10		65	ns
17	AS* high to LOCK* high	5		30	ns
18	SWPEN*/ISOEN* high to DBDIR* high	5			ns

APPENDIX A

Table A-16. Local Bus To LCSR: Read Cycle

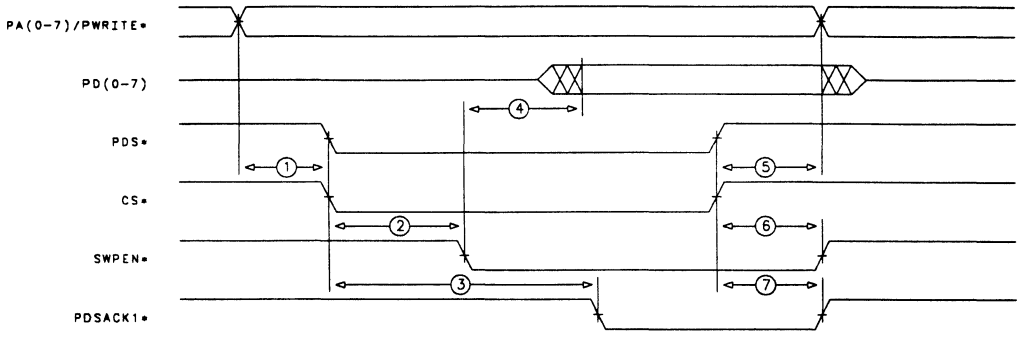


NUM.	CHARACTERISTIC	MIN	TYP	MAX	UNIT
1	PA/PWRITE* valid to PDS* and CS* low (Note 1)	10			ns
2	DBDIR* low to SWPEN* low	5			ns
3	PDS* and CS* low to SWPEN* low	10		65	ns
4	PDS* and CS* low to PDSACK1* low (Note 2)			3T+75	ns
5	PD valid to PDSACK1* low (Note 2)	T-10			ns
6	PDS* or CS* high to PA/PWRITE* invalid	0			ns
7	PDS* or CS* high to SWPEN* high	5		40	ns
8	SWPEN* high to DBIR* high	5			ns
9	PDS* or CS* high to PD invalid	5			ns
10	PDS* or CS* high to PD high Z			75	ns
11	PDS* or CS* high to PDSACK1* high	5		40	ns

Notes:

1. The LCSR is selected when PDS* and CS* are both low.
2. T is the CLK period.

Table A-17. Local Bus To LCSR: Write Cycle



NUM.	CHARACTERISTIC	MIN	TYP	MAX	UNIT
1	PA/PWRITE* valid to PDS* and CS* low (Note 1)	10			ns
2	PDS* and CS* low to SWPEN* low	5		45	ns
3	PDS* and CS* low to PDSACK1* low (Note 2)			3T+75	ns
4	SWPEN* low to PD valid			11	ns
5	PDS* or CS* high to PA/PD/PWRITE* invalid	0			ns
6	PDS* or CS* high to SWPEN* high	5		40	ns
7	PDS* or CS* high to PDSACK1* high	5		40	ns

Notes:

1. The LCSR is selected when PDS* and CS* are both low.
2. T is the CLK period.

Table A-18. Buffer Timing Requirements (See Figures 3-4 and 3-5)

The propagation delays for the X543 Devices must be no greater than the following:

NUM.	CHARACTERISTIC	MIN	TYP	MAX
1	A valid to B valid (LEBA* low)			10ns
2	B valid to A valid (LEAB* low)			10ns
3	LEBA* low to A valid			15ns
4	LEAB* low to B valid			15ns
5	OEBA* low to B enabled			12ns
6	OEAB* low to A enabled			12ns
7	OEAB* high to B disabled			10ns
8	OEBA* high to A disabled			10ns

The X543 devices used must have minimum setup, hold, pulse width requirements no greater than the following:

NUM.	CHARACTERISTIC	MIN	TYP	MAX
7	A valid to LEAB* high (SETUP)	5ns		
8	B valid to LEBA* high (SETUP)	5ns		
9	LEAB* high to A invalid (HOLD)	5ns		
10	LEBA* high to B invalid (HOLD)	5ns		
11	LEAB* pulse width low	5ns		
12	LEBA* pulse width low	5ns		

The propagation delays for the X645 devices must be no greater than the following:

NUM.	CHARACTERISTIC	MIN	TYP	MAX
1	A valid to B valid			10ns
2	B valid to A valid			10ns
3	OE* low to A or B enabled			11ns
4	X/Y* valid to A or B enabled			11ns
5	OE* high to A or B disabled			12ns
6	X/Y* valid to A or B disabled			12ns

APPENDIX B
PINOUT OF THE MVME6000

The signal lines of the MVME6000 are divided into two groups: VMEbus signals, and local signals. Table B-1 summarizes the VMEbus signal lines and Table B-2 summarizes the local signal lines. Figure B-1 shows the pin-assignments of the MVME6000 and Figure B-2 shows the package dimensions .

Table B-1. Summary of VMEbus Signal Lines

SIGNAL LINE	DIRECTION	TYPE
A01-A07	I/O	TS
AM0-AM5	I/O	TS
AS*	I/O	TS
BBSY*	I/O	OC
BCLR*	O	TS
BERR*	I/O	OC
BG0IN*-BG3IN*	I	
BG0OUT*-BG3OUT*	O	TP
BR0*-BR3*	I/O	OC
D00-D07	I/O	TS
DS0*, DS1*	I/O	TS
DTACK*	I/O	OC
IACK*	I/O	TS
IACKIN*	I	
IACKOUT*	O	TP
IRQ1*-IRQ7*	I/O	OC
LWORD*	I/O	TS
SYSCLK	O	TS
SYSFAIL*	I/O	OC
SYSRESET*	I/O	OC
WRITE*	I/O	TS

Notes:

1. I = Input only
2. O = Output only
3. I/O = Input and Output
4. TS = Three State
5. OC = Open Collector
6. TP = Totem Pole

APPENDIX B

Table B-2. Summary of Local Signal Lines

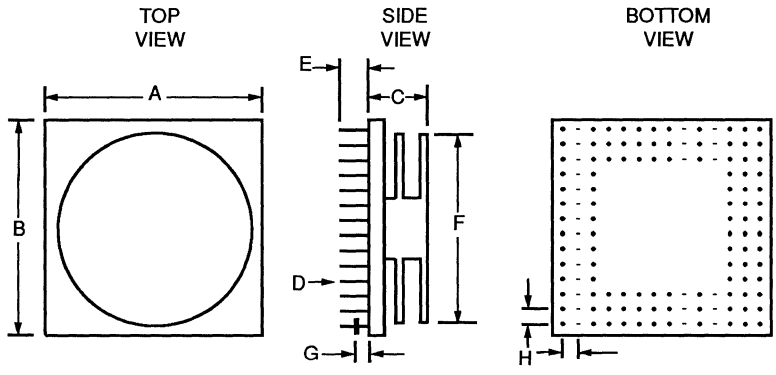
SIGNAL LINE	DIRECTION	TYPE
32FILL*	O	TP
ABIN*	O	TP
ABOUT*	O	TP
ADR24*	I	
BRDFAIL*	I/O	OC
BRDRESET*	I	
CLK	I	
CS*	I	
DAT16*	I	
DBDIR	O	TP
DBOUT*	O	TP
DWB*	I	
IPL0*-IPL2*	O	TP
ISOEN*	O	TP
LATCHIN	O	TP
LATCHOUT	O	TP
LDBIN*	O	TP
LOCK*	O	TP
MATCH16*	I	
MATCH24*	I	
MATCH32*	I	
MATCHGCSR*	I	
PA00-PA07	I/O	TS
PAS*	O	TS
PBERR*	I/O	OC
PBG*	I	
PBR*	O	OC
PD0-PD7	I/O	TS
PDS*	I/O	TS
PDSACK0*, PDSACK1*	I/O	OC
PFC0-PFC2	I/O	TS
PHALT*	O	OC
PIACK*	I	
PRESET*	I/O	OC
PSTBSY*	O	TP
PSIZ0-PSIZ1	I/O	TS
PWRITE*	I/O	TS
RMC*	I	
RETRY*	I	
SCLK	I	
SCON*	I	
SHORTIO*	I	
SWPEN*	O	TP
UDBIN*	O	TP
VMESEL*	I	

14	13	12	11	10	9	8	7	6	5	4	3	2	1
SCON*	BCLR*	PD2	PD5	CLK	LATCH-IN	BRD-RESET*	PRE-SET*	BG2IN*	DS1*	IRQ1*	IRQ3*	IRQ7*	D01
SYS-CLK	GND	PD1	PD4	PD6	LATCH-OUT	SYS-RESET*	BG1IN*	BG3IN*	AS*	IRQ4*	IRQ5*	D00	D02
PA3	PA1	SCLK	PD0	PD3	PD7	GND	BG0IN*	DS0*	IRQ2*	IRQ6*	GND	D04	D05
PA5	PA2	PA0									D03	D07	VCC
VCC	PA6	PA4									D06	MATCH-GCSR*	PBE-RR*
A02	A01	PA7									BRD-FAIL*	BBSY*	DTACK*
A05	A03	A04									GND	IACKIN*	IACK-OUT*
A06	A07	GND									WRITE*	LWORD*	DB-OUT*
SYS-FAIL*	CS*	MA-TCH16*									DWB*	RMC*	IACK*
PWRITE*	MA-TCH24*	PBG*									BR1*	ADR24*	VCC
VCC	MA-TCH32*	ABIN*									BG1-OUT*	BR3*	BR0*
LOCK*	ABOUT*	GND	AM1	AM5	BERR*	PDSA-CK1*	GND	LDBIN*	SWPEN*	PBR*	BG3-OUT*	BG0-OUT*	BR2*
ISOEN*	PSIZ1	AM2	AM3	PFC1	PDS*	PDS-ACK0*	IPL2*	DAT16*	UDBIN*	DBDIR*	RETRY*	GND	BG2-OUT*
PSIZ0	AM0	AM4	PFC0	PFC2	PAS*	IPL0*	IPL1*	PIACK*	SHOR-TIO*	DHB*	32FILL*	VME-SEL*	PHALT*
14	13	12	11	10	9	8	7	6	5	4	3	2	1

PINS VIEWED FROM BOTTOM

Figure B-1. Pin Assignments of the MVME6000

APPENDIX B



PACKAGE DIMENSIONS, 132 PIN GRID ARRAY				
	INCHES		MILLIMETERS	
SYMBOL	MIN.	MAX.	MIN.	MAX.
A	1.380	1.420	35.052	36.068
B	1.380	1.420	35.052	36.068
C	0.320	0.380	8.128	9.652
D	0.016	0.021	0.406	0.533
E	0.165	0.195	4.191	4.953
F	1.150	1.300	29.210	33.020
G	0.045	0.055	1.143	1.397
H	.100 NOMINAL		2.54 NOMINAL	

Figure B-2. MVME6000 Package Dimensions

MOTOROLA INC.

READER COMMENTS

Motorola welcomes your comments on its documentation. Please use this form.

Manual Title _____

Part Number _____ Date _____

Your Name _____

Your Title _____ Telephone Number _____

Company _____

Address _____

_____

_____

General Information:

Do you read this manual in order to:

☐ Install the product ☐ Use the product ☐ Repair the product

☐ Reference information ☐ Other _____

In general, how do you locate information within this manual?

☐ Index ☐ Table of Contents ☐ Page Headings ☐ Other _____

Completeness: ☐ Excellent ☐ Very Good ☐ Good ☐ Fair ☐ Poor

What topic would you like more information on:

_____

_____

Presentation: ☐ Excellent ☐ Very Good ☐ Good ☐ Fair ☐ Poor

What features of the manual are most useful (tables, figures, appendixes, index, etc.)?

_____

_____

Is the information easy to understand? ☐ Yes ☐ No If you checked no, please explain:

_____

_____

Is the information easy to find? ☐ Yes ☐ No If you checked no, please explain:

_____

_____

Technical Accuracy: ☐ Excellent ☐ Very Good ☐ Good ☐ Fair ☐ Poor

If you have found technical or typographical errors, please list them here.

Page Number	Description of Error

MOTOROLA INC.

Microcomputer Division
2900 South Diablo Way
Tempe, Arizona 85282
P.O. Box 2953
Phoenix, Arizona 85062

10835 PRINTED IN USA 7-89 IRONWOOD LITHO 7523-B 1,500

68NW9209I51A