ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Curtiss-Wright / VMETRO / Micro Memory MM-6490DR* at our website: *Click HERE*

MM–6490ADR

USERS GUIDE

9540 Vassar Ave.
Chatsworth, Ca. 91311-4141

Tel:(818) 998-0070
Fax:(818) 998-4459

September 1998

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Micro Memory inc.

The information in this document is subject to change without notice.

Micro Memory Inc. makes no warranty of any kind with regard to this material. However the information in this document has been carefully checked and is believed to be entirely correct and no responsibility is assumed for inaccuracies.

Any information regarding this product or any other Micro Memory products may be obtained from:

MICRO MEMORY INC.
9540 VASSAR AVE
CHATSWORTH, CA. 91311
TEL: (818) 998-0070
FAX: (818) 998-4459

CONVENTIONS

This manual uses the following notational conventions.

NOTE: A Note focuses attention on supplementary or unique information and should be read before any action is taken

- or / Used after a signal names (for example DS0* or DS0/) to indicate that the signal is active low and a signal without an asterisk or slash indicates that the signal is active high.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table of Contents

GENERAL INFORMATION 1
1.1 INTRODUCTION 1
1.2 GENERAL DESCRIPTION 1
1.2.1 Memory capacity options 2
1.2.2 Operational features 3
1.3 GENERAL SPECIFICATIONS 4
1.4 INTERFACE TIMING 5
1.4.1 Refresh Timing 5
1.4.2 VMEbus Interface Timing 6
1.5 MEMORY ARRAY ORGANIZATION 9

INSTALLATION 10
2.1 INTRODUCTION 10
2.2 UNPACKING INSTRUCTIONS 10
2.3 HARDWARE PREPARATION 10
2.3.1 Handling Procedure 10
2.4 OPERATING ENVIRONMENT 11
2.4.1 Airflow and Cooling 11
2.4.2 Power Requirements 11
2.5 JUMPER OPTIONS 11
2.5.1 Address Modifier Evaluation (Factory, A32 and A24 ENABLED) 13
2.5.2 Rescinded DTACK* (Factory, DISABLED) 13
2.5.3 VMEbus Fast! DTACK* (Factory, DISABLED) 13
2.5.4 System FAIL (SYSFAIL*) Refresh Timeout (Factory ENABLED) 13
2.6 CONFIGURATION JUMPER & SWITCH SETTINGS 14
2.6.1 VMEbus Module Selection 14
2.6.2 RACEway module Selection 14
2.6.3 Adjusted Addressing 15
2.6.4 Control Status Register (CSR) Selection 15
2.7 INSTALLATION IN THE SYSTEM 15

PROGRAMMING INFORMATION 16
3.1 Control Status Register 16
3.2 Parity Error Detection 16
3.3 RP - Ram Private 17
3.4 Write Wrong Parity 17
3.5 RACEway Programming 17
3.5.1 System Configuration and Initialization 17
3.5.2 Mapping Bulk Memory 19
3.5.3 Block Transfers using DMA 22
3.5.4 Measuring Transfer Rates 26
3.5.5 Hardware Description File for MM6490 31

MODULE SELECTION 34

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

U1 PLD Equations 37

Tables

Table 1 Memory Capacity 2
Table 2 VME P1 Connector Pin-out 7
Table 3 VME / RACEway P2 Connector Pin-out.....8
Table 4 Memory Organization.....9
Table 5 C.S.R. Bit Definitions.....16

Figures

Figure 1 Refresh Timing.....5
Figure 2 VME Timing 6
Figure 3 Jumper Locations.....12

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SECTION I

GENERAL INFORMATION

1.1 INTRODUCTION

This manual describes the elements of operation and installation procedures of the **MM-6490ADR** Dual Port Dynamic Random Access Memory Module.

1.2 GENERAL DESCRIPTION

The **MM-6490ADR** Dual Ported Memory is compatible with the **VMEbus, RACEway Interlink VITA 5-1994 (RACEway)** specifications. Inherent to the module are **VMEbus** options D64, D32, D16 & D8 (64, 32, 16 & 8 bit data path width), and A32, A24 (32, 24 bit address path width) and is a **RACEway Slave.** The module also generates and stores an even parity bit for each byte written on write cycles and checks parity for each byte read on read cycles. Then if a parity error is detected, the module sets a **Control Status Register** (CSR) bit and may be programmed to assert the bus error signal on the requesting port. (See Section III for CSR programming)

For Extremely High Performance operation, the **MM-6490ADR** Dual Ported Memory is equipped with a Dual Port **PAGE MODE** High-Speed ***BLOCK Transfer*** capability. This allows the use of High Performance Block Mode Controllers to transfer up to 2048 (MLBT)/256 (BLT) bytes on the **VMEbu**s, and 2048 bytes on the **RACEway**, for increased system performance.

While the **MM-6490ADR** Dual Ported Memory was designed as a 64 bit wide (data path) memory board, which complies with **VMEbus** Specifications (ANSI/VITA 1-1995) and can be addressed as 8 bit bytes, 16 bit words, 32 bit longwords or 64 bit octalwords. It also complies with the **RACEway** Specifications (VITA 5-1995) as a *Raw Memory Slave*.

The **MM-6490ADR** Dual Ported Memory can be configured from 64M to 512M bytes capacity by populating it with 8192K X 72 60ns CMOS DRAM Single Inline Packages (SIP) pairs (MM950 and MM960).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.2.1 Memory capacity options

The **MM-6490ADR** Dual Ported Memory modules are available in several options depending on the capacity required. Table 1 Memory Capacity, lists the optional part numbers for ordering purposes, the total memory capacity provided by each version.

Table 1 Memory Capacity

Model Number	Capacity
MM-6490ADR/64M	64.0M Bytes
MM-6490ADR/128	128.0M Bytes
MM-6490ADR/256M	256.0M Bytes
MM-6490ADR/512M	512.0M Bytes

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.2.2 Operational features

The **MM-6490ADR** Dual Ported Memory module contains its own address register and data buffers for each port for total compatibility with the **VMEbus** Specification (VITA 1-1995) and **RACEway** Specification (VITA 5-1995)

Module selection for the **MM-6490ADR** resides on any 1024K (100000 hex) bytes boundaries (A24/A32). The module may be mapped anywhere within the 4 G bytes of physical address space on the **VMEbus**.

There is no module selection for the **RACEway** port. The **MM-6490ADR** Dual Ported Memory module is addressed as a RAW MEMORY DEVICE using Shared Memory Buffers (SMB).

** Note: See **RACEway** 'MASTER' users manual, (not supplied) for more information.

The **MM-6490ADR** Dual Ported Memory array can be addressed as 8 bit bytes, 16 bit words, or as 32 bit longwords, using conventional Read/Write, Read/Modify/Write (RMW), or HIGH Performance BLOCK Transfers (MBLT) & (BLT).

The **MM-6490ADR** Dual Ported Memory responds to various Address Modifier Codes. The codes are decoded by U1 (a socketed PLD), Various combinations are available to the user (See Section II, jumpers).

The user may select to create his/her own Address Modifier code, in this event the Address Modifier evaluation PLD (U1) can be replaced to accommodate the systems' requirements (See Section II, jumpers).

A **Control Status Register** (CSR) is available to allow various dynamic options to to be utilized by the software programmer. Included in the CSR is a bit to make the **MM-6490ADR** Dual Ported Memory a Private Bus only board (ie: **RACEway** only). The CSR is an A16 Slave and has separate address selection jumpers, and may be addressed on any WORD boundary in the **VMEbus** I/O address space, using a combination of jumper & VMEbus *Low Limit* switch settings. (See Section II for selection & Section III for programming details)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.3 GENERAL SPECIFICATIONS

Characteristics	Specifications
Capacity	64M, 128M, 256M or 512Mbytes
Cycle Time	250ns. (non-Block mode) VME 100ns. (Block mode) VME
RACEway Transfer Rate	160 MBytes/sec during transfer
Access Time (Non-Block mode)	Read 195nsec VME Write 120nsec VME
Access Time (Block Mode)	Read 34nsec VME Write 34nsec VME
Address	32 bits A32/BLT/MBLT/UAT
Data In/Data Out	8/16/24/32/64 bits bidirectional with three-state output D08/D16/D32
Modes of Operation	Read, Write, Read/Modify/Write (RMW), Block Transfers (BLT), (MBLT) Un-aligned (UAT)
Address Modifiers	6 bits, jumper selectable, or user programmable PLD
Parity	EVEN, 1 bit for each byte Generated on each WRITE Checked on each read.
VMEbus	Memory selected on 1024K byte ($100000) boundary
Control Status Register Selection	CSR Selection based on VMEbus Start Address, in VMEbus 'A16'I/O Address space.
Inputs	TTL-compatible
Outputs	48ma. Three-state, TTL-compatible
Operating Temperature	0 to 60 degrees C
Storage Temperature	-40 to +85 degrees C
Relative Humidity	95% without condensation
Power Requirements: +5V (fully-populated)	Standby Operate 2.4A 2.8A

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.4 INTERFACE TIMING

The **MM-6490ADR** Dual Ported Memory modules are designed to accept 8192K x 72 DYNAMIC Random Access Memory module pairs with a 60nsec typical access delay Time.

1.4.1 Refresh Timing

The refresh cycle timing is illustrated by Figure 1.1. Refresh cycles are repeated approximately once every 15usec. A free running oscillator providing a refresh transparent to memory access cycles generates the cycle interval timing.

The **MM-6490ADR** contains a refresh request counter and can store up to 15 refresh requests. During memory cycles that take more than the 15us refresh interval, this counter will record how many refresh requests have been made. Upon completion of the cycle (The memory board is released i.e.: REQI* negated for **RACEway** cycles and AS* negated for **VME** cycles) the refresh circuit run up to the 15 stored refresh requests.

(See Section 2.5.4 regarding **VME** SYSFAIL* - Refresh Timeout)

- NOTE: Memory cycles must be completed within 220us or REFRESH cycles will be lost. Resulting in possible data corruption.

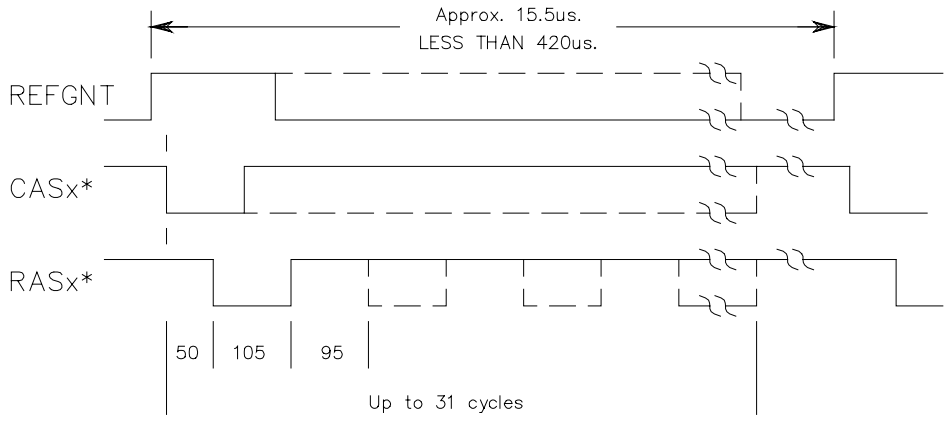


Figure 1 Refresh Timing

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.4.2 VMEbus Interface Timing

The **VMEbus** Interface Timing Diagram (Figure 2 VME Timing) illustrates the timing relationships on the Data Transfer bus, for Read and Write Cycles. Access delay time is measured from the leading edge of the Data strobes.

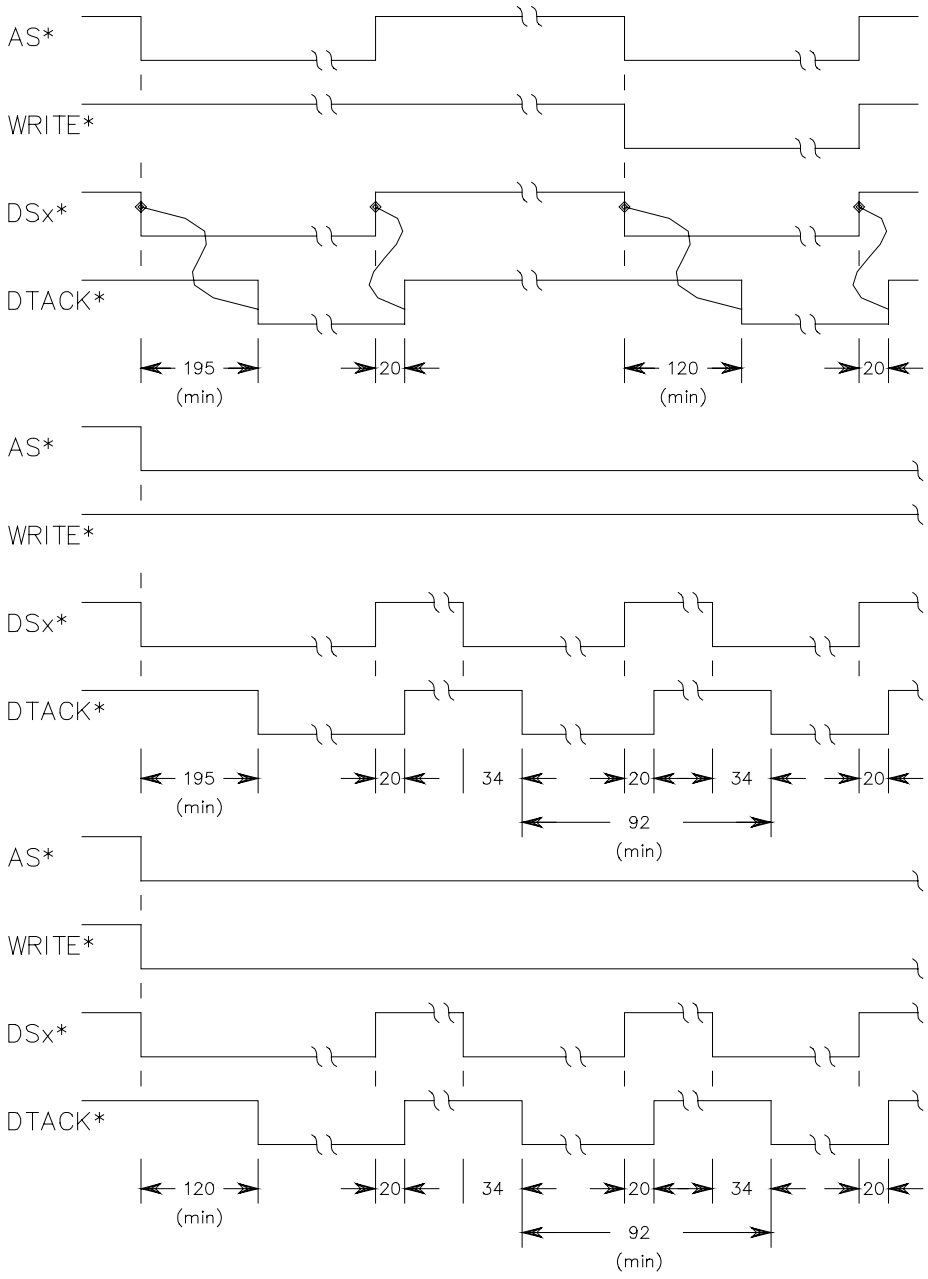


Figure 2 VME Timing

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 2 VME P1 Connector Pin-out

	A	B	C
1.	D00	BBSY*	D08
2.	D01	BCLR*	D09
3.	D02	ACFAIL*	D10
4.	D03	BG0IN*	D11
5.	D04	BG0OUT*	D12
6.	D05	BG1IN*	D13
7.	D06	BG1OUT*	D14
8.	D07	BG2IN*	D15
9.	GND	BG2OUT*	GND
10.	SYSCLK	BG3IN*	SYSFAIL*
11.	GND	BG3OUT*	BERR*
12.	DS1*	BR0*	SYSRESET*
13.	DS0*	BR1*	LWORD*
14.	WRITE*	BR2*	AM5
15.	GND	BR3*	A23
16.	DTACK*	AM0	A22
17.	GND	AM1	A21
18.	AS*	AM2	A20
19.	GND	AM3	A19
20.	IACK*	GND	A18
21.	IACKIN*	SERCLK	A17
22.	IACKOUT*	SERDAT	A16
23.	AM4	GND	A15
24.	A07	** IRQ7*	A14
25.	A06	** IRQ6*	A13
26.	A05 *	** IRQ5	A12
27.	A04	** IRQ4*	A11
28.	A03	** IRQ3*	A10
29.	A02	** IRQ2*	A09
30.	A01	** IRQ1*	A08
31.	** -12v	** +5vSTDBY	** +12v
32.	+5v	+5v	+5v

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3 VME / RACEway P2 Connector Pin-out

	A	B	C
1.	XCLKI	+5v	XRESETIO*
2.	GND	GND	Reserved
3.	XBIO09	RESERVED	XSYNCI*
4.	XBIO08	A24	GND
5.	GND	A25	XBIO07
6.	XBIO06	A26	GND
7.	GND	A27	XBIO11
8.	XBIO10	A28	GND
9.	XBIO04	A29	STROBEIO
10.	GND	A30	REPLYIO
11.	XBIO05	A31	GND
12.	XBIO03	GND	REQI
13.	GND	+5v	REQO
14.	RDCONIO	D16	GND
15.	Reserved	D17	XBIO02
16.	GND	D18	XBIO01
17.	XBIO00	D19	GND
18.	XBIO15	D20	XBIO12
19.	GND	D21	XBIO25
20.	XBIO24	D22	GND
21.	XBIO31	D23	XBIO29
22.	GND	GND	XBIO30
23.	XBIO28	D24	GND
24.	XBIO27	D25	XBIO26
25.	GND	D26	XBIO23
26.	XBIO22	D27	GND
27.	XBIO20	D28	XBIO19
28.	GND	D29	XBIO21
29.	XBIO18	D30	GND
30.	XBIO17	D31	XBIO16
31.	GND	GND	XBIO14
32.	XBIO13	+5v	GND

**** - No Connection on MM-6490ADR Dual Ported Memory**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.5 MEMORY ARRAY ORGANIZATION

The memory array is partitioned into eight (8) 8192K x 72 bit segments. The minimum memory configuration would be 8192K x 72 bits, including the eight (8) parity bit DRAMs in segment zero. This would translate to 64Megabytes as a minimum configuration for the **MM-6490ADR Dual Ported Memory**. Each partition or block is organized as 72 bits each (64 plus 8 parity), however the system will only see 64 bits

of data during memory references, as the parity is generated and checked transparent to the user. The 64 bits of data are divided into 8 bytes (8 bits each). While the memory reads and writes may be 8, 16, 32 or 64 bits wide, the memory is organized as 64 bit wide locations with logic controlling which byte the user is reading or writing.

If the contents of the first 8 locations in memory are $01 $23 $45 $67 $89 $AB $CD $EF, determining which bits constitute what byte may be somewhat confusing. Seeing the same data in the various formats used in most systems can clear this up (IE: Byte, Word, and Longword).

Table 4 Memory Organization

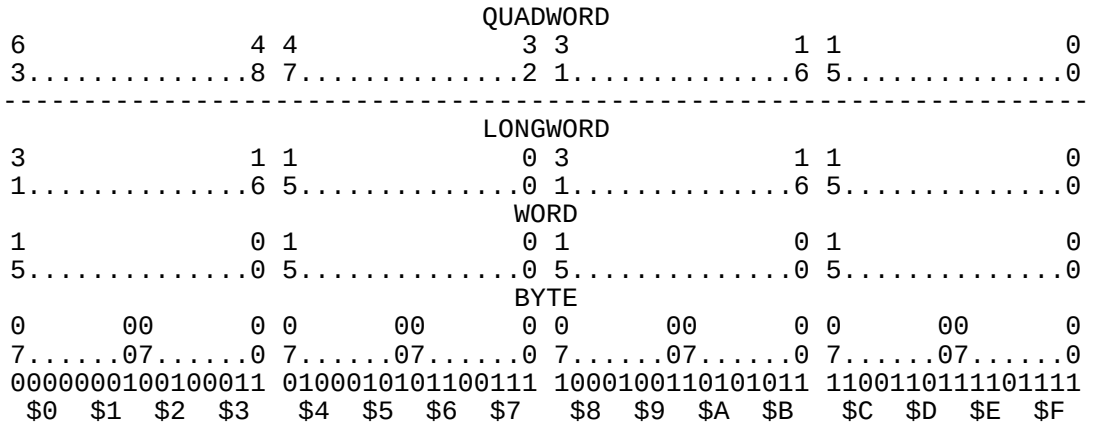
BYTE Format	
Location	Data
$00000000	$01
$00000001	$23
$00000002	$45
$00000003	$67
$00000004	$89
$00000005	$AB
$00000006	$CD
$00000007	$EF

WORD Format	
Location	Data
$00000000	$0123
$00000002	$4567
$00000004	$89AB
$00000006	$CDEF

LONGWORD Format	
Data	Location
$00000000	$01234567
	$89ABCDEF

LONGWORD Format	
Location	Data
$00000000	$01234567
$00000004	$89ABCDEF

BIT Organizations



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SECTION II

INSTALLATION

2.1 INTRODUCTION

This section details a step by step procedure to interface the **MM-6490ADR Dual Ported Memory** to both the VMEbus and RACEway Interlink.

2.2 UNPACKING INSTRUCTIONS

Unpack module from shipping carton. If carton is damaged upon receipt, request that the carrier's agent be present during unpacking and inspection of the equipment. Refer to packing list and verify that all items are present. Save packing material for storing or reshipping the equipment. For repairs or replacement of a board damaged during shipment, contact **Micro Memory Inc.** to obtain a Return Authorization number and further instructions.

2.3 HARDWARE PREPARATION

The memory module should be inspected and prepared for jumper placement prior to system installation. The following sections describe the proper jumper options necessary for system operation (see Section 2.5, for jumper options).

2.3.1 Handling Procedure

The **MM-6490ADR** Dual Ported Memory uses CMOS components that are susceptible to damage if exposed to static electrical charges. To avoid damage of these components during handling, testing or operation, the following procedure should be used.

- Device leads should contact conductive material to avoid building of any static charge, except during testing or operation.
- Soldering iron tips, metal fixtures, tools, and handling facilities used in preparing the module for operation should be grounded.
- Devices should never be removed or inserted while power is applied to the module because voltage transients may permanently damage the devices and/or module.
- The memory module should never be plugged in or out of the cardcage while power is applied.
- External signals should not be applied to device inputs while power is removed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Any memory module removed from the system should be transferred to either non-conductive foam or an anti-static plastic bag for storage or shipment.

2.4 OPERATING ENVIRONMENT

The **MM-6490ADR** Dual Ported Memory is an extremely High-Density memory product. Containing, 96 bits of high current (48ma.) data bus drivers, numerous high-speed components. And as a result, the environment into which the **MM-6490ADR**Dual Ported Memory is placed must be carefully considered.

2.4.1 Airflow and Cooling

Due to the extremely high component density of the **MM-6490ADR** Dual Ported Memory, an adequate airflow is required to maintain the operating Temperature within specifications of the memory module. Failure to adhere to these requirements may result in poor long-term reliability.

2.4.2 Power Requirements

The power requirements of **MM-6490ADR** Dual Ported Memory are such, that the module should never be operated while powered only by the P1 connector. Hence power should be applied to the appropriate pins (See Tables 3 & 4) of both the P1 and the P2 connector prior to operation of the **MM-6490ADR** Dual Ported Memory.

2.5 JUMPER OPTIONS

There are several jumper options attainable on the **MM-6490ADR** Dual Ported Memory module, depending on the user application. All of the available options are selected by installing or removing mini jumpers on wire wrap pins on 0.100-inch centers. (See Figure 3 Jumper Locations)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

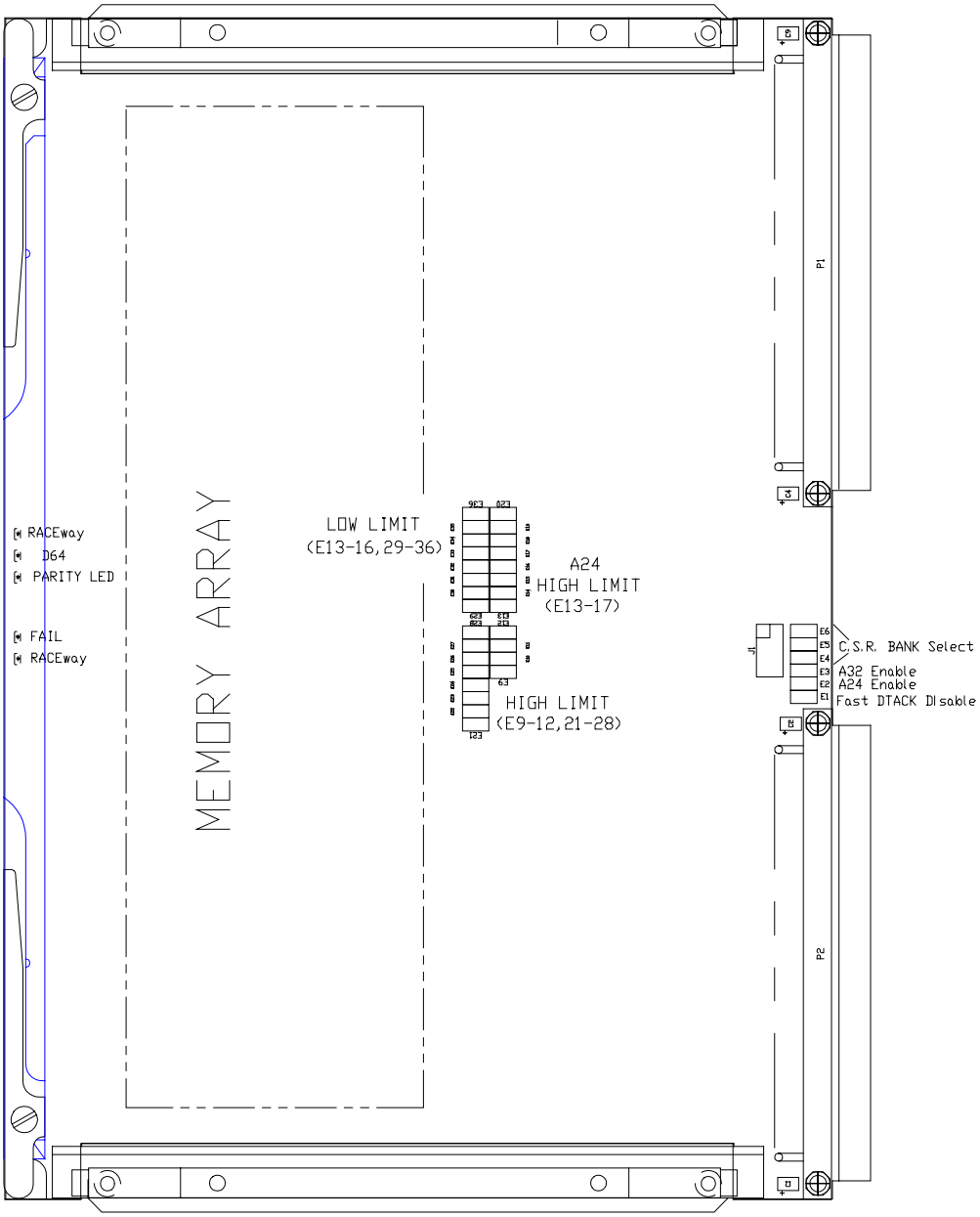


Figure 3 Jumper Locations

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.5.1 Address Modifier Evaluation (Factory, A32 and A24 ENABLED)

The **MM-6490ADR**Dual Ported Memory evaluates the Address Modifiers from the VMEbus to determine if they are to be responded to. As shipped from the factory, the **MM-6490ADR** Dual Ported Memory responds to the EXTENDED A32 (using all 32 address bits) Address Modifier codes and the STANDARD A24 (ignoring the upper 8 bits). Optional, the USER may remove E3 to disable the EXTENDED A32 selection and/or E2 STANDARD A24 selection. These Address Modifier codes are described in Appendix

B. The user may select to replace the Address Modifier selection PLD (U1), to allow for some other combination or the creation of a special (user defined) code or codes.

2.5.2 Rescinded DTACK* (Factory, DISABLED)

As shipped from the factory the **MM-6490ADR** Dual Port Memory is configured to use an 74S38 (an OPEN Collector device) to generate DTACK* (VMEbus). This conforms to the VMEbus specifications. However, on heavily loaded systems (ie: full 21 slot cardcage) this can cause DTACK* to take over 40ns to be negated. This can prevent most systems from attaining the maximum bus bandwidth of 40 Megabytes per second

(100ns bus cycles). On certain system configurations (ie: those without 'PARTICIPATING SLAVES') the designer MAY wish to use a 'RESCINDED DTACK*'. That is to say an 74FCT240 (with 3 state control) is used in place of the 74S38. Moving jumper E7 from pins 2 to 3, to pins 1 to 2 will enable the VMEbus 'RESCINDED DTACK*'.

- NOTE: This conforms to the <B>VME64<D> specification VITA11995.

2.5.3 VMEbus Fast! DTACK* (Factory, DISABLED)

Due to **VMEbus** specification, the **MM-6490** Dual Ported Memory is required to wait a minimum of 30ns before responding to DS0* or DS1* with the DTACK* signal. If jumper E1 is removed, the **MM-6490** Dual Ported Memory can respond in as little as 10ns from the earliest Data Strobe (DS0* or DS1*). Data Strobe skew (DS0* or DS1*) may cause unwanted results.

- **Caution!** This will *violate* **VMEbus** specifications, and should ONLY be attempted with a thorough understanding of the system and its configuration.

2.5.4 System FAIL (SYSFAIL*) Refresh Timeout (Factory ENABLED)

If memory cycles exceed the refresh counter (See Section 1.4.1 Refresh

Timing) capacity of 31 requests the **MM-6490ADR**light the FAIL L.E.D. and attempt to report a 'System FAILURE' by asserting SYSFAIL* on the **VME** bus. Removal of jumper E8 will prevent SYSFAIL* from being asserted.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.6 CONFIGURATION JUMPER & SWITCH SETTINGS

The **MM6490ADR** Dual Port Memory contains six (6) jumper blocks that must be configured prior to installation before proper operation can be obtained. (See Figure 2.1 for jumper locations)

The **VMEbus** port of the **MM6490ADR** Dual Port Memory must be addressed within it's four (4) Gigabyte address space. The **VMEbus** port is controlled by jumpers E13 through E40, also by jumpers E2 (A24 Address Modifier Enable) and E3 (A32 Address Modifier Enable (See Section 2.5.1).

While the **RACEway** port is controlled by software routing on the **RACEway** MASTER.

2.6.1 VMEbus Module Selection

Selection of the **VMEbus** lower limit and upper limit on 1024Kilobyte byte ($100000) boundaries and accomplished using jumpers E25 through E36 for the LOW LIMIT and jumpers E13 through E24 for the A32 space UPPER LIMIT. Jumpers E13 and E25 are the most significant address (A31) and jumpers E24 and E36 are the least significant address (A20) for both UPPER and LOWER limits. While jumper E37 is the most significant address (A23) and jumper E40 is the least significant address (A20) for A24 space UPPER LIMIT.

When the **MM-6490ADR** Dual Ported Memory is selected using an A24 address modifier, the **MM-6490ADR** will automatically force a 0 on internal addresses A24 thru A31. Therefore forcing the address request into the FIRST 16Megabyte block and will use jumpers E37 through E40 for the UPPER LIMIT selection (E37 is the most significant address (A23) while E40 is the least significant (A20)).

NOTE: The UPPER LIMIT is an ***inclusive*** address.

Therefore an 128MByte board addressed to start at $00000000 would have the LOW LIMIT set to $00000000 and the UPPER LIMIT set to $07FFFFFF ($000 to $07F on the jumpers). **NOT** $000 to $080.

A jumper **REMOVED** equates to a one (1) while **INSERTED** equates to a zero (0)

- **OPEN** or **OFF** equates to a one (1) while **CLOSED** or **ON** equates to a zero (0).

2.6.2 RACEway module Selection

RACEway module selection is accomplished via ROUTING information based on the **RACEway** MASTER. The **MM-6490** Dual Ported Memory module is considered a 'RAW MEMORY DEVICE' and is normally addressed using Shared Memory Buffers (SMB). (See Section 3.5 **RACEway** Programming, for details).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.6.3 Adjusted Addressing

In normal operation, the **MM-6490ADR**Dual Ported Memory would be selected on a 512 Megabyte boundary and simply set the *UPPER* and **LOWER** limits accordingly. However, in some applications the user may wish to more accurately control the starting and ending addresses. An example would be when the user wants to start the memory immediately following some on-board memory (IE: 1 Megabyte CPU on-board Memory). To keep both ports aligned (IE: first byte on **RACEway** port is first byte on **VMEbus** port), the addresses applied to the **MM-6490ADR** Dual Ported Memory must be adjusted. This is accomplished by on board circuitry that subtracts the LOW LIMIT address from the address applied from the bus. The **MM-6490ADR** Dual Ported Memory allows the user to select an address on any 1-Megabyte boundary, and still be fully aligned.

2.6.4 Control Status Register (CSR) Selection

The CSR address selection is a combination of jumpers E4, E5, E6 and the **VMEbus** LOW LIMIT selection. The CSR is located in **VMEbus** A16 (or I/O) space, and responds to ONLY A16 Address Modifiers ($29 & $2D. See Appendix B for more details).

Jumper E4 corresponds to the most significant address A15 while E6 corresponds to the least significant address A13. Addresses A12 down through A1 correspond to the **VMEbus** LOW LIMIT address positions for addresses A31 through A20. Where E25 (A31) corresponds to A12, E26 (A30) corresponds to A11 and so on.

2.7 INSTALLATION IN THE SYSTEM

After selecting the memory for the proper address locations by opening/closing the appropriate switches (see Section 2.6) and installing/removing the user defined option jumpers (see Section 2.5). And after following the guidelines set forth in Section 2.3, remove power, install the **MM-6490ADR** Dual Ported Memory into the system, reapply power to the system.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

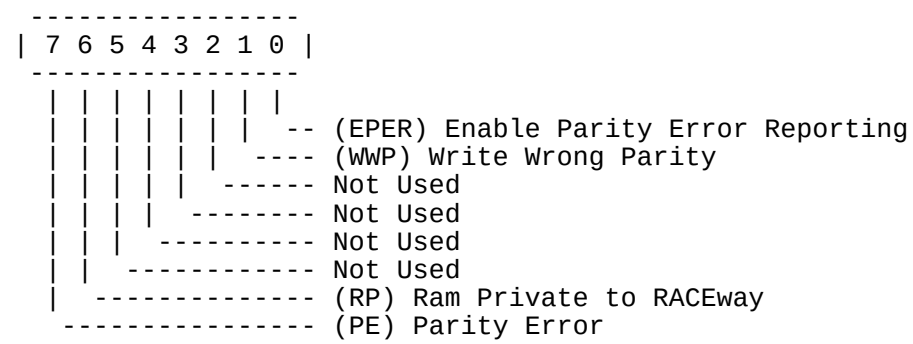
SECTION III

PROGRAMMING INFORMATION

3.1 Control Status Register

The Control Status Register (CSR) provides a **VMEbus** accessible resister to control and determine the status of the **MM-6490ADR Dual Ported Memory**. In the following sections, the CSR bits are defined and their use is explained.

Table 5 C.S.R. Bit Definitions



3.2 Parity Error Detection

The **MM-6490ADR Dual Ported Memory** generates and checks for errors in the memory by calculating the parity of each byte in memory. During write cycles, the **MM-6490ADR Dual Ported Memory** writes this bit into the array, and on read cycles it checks this bit against the bit in the array. If, during a read cycle, a difference is detected the CSR PE bit is set and if the CSR EPER bit is also set, the appropriate ERROR lead is asserted. In the case of a **VMEbus** reference, the BERR* lead is asserted instead of the DTACK* lead. The PE bit is cleared by reset or by writing a 0 into CSR bit7.

- To insure proper parity operation, the **MM-6490ADR Dual Ported Memory** must be *initialized*. That is to say a memory location **MUST** be written *before* it is read.

- **NOTE:** It is recommended that *ALL* memory be *Initialized* using a 32 bit write instruction (IE: The 680x0 “CLR.L” instruction or equivalent), to prevent unexpected Parity ERRORS.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.3 RP - Ram Private

The **MM6490DR Dual Ported Memory** may be reserved in ***TOTAL*** by setting the RP bit in the CSR to a 1. Any references made of the **MM6490DR Dual Ported Memory** from the **VMEbus** port are ignored while this bit is set.

3.4 Write Wrong Parity

When set, this bit will force the **MM6490DR** to generate the **WRONG** parity for each byte written. This allows testing of the parity bit memory device for increased system confidence. This mode may also be used to test BUS ERROR reporting circuits on other boards.

3.5 RACEway Programming

3.5.1 System Configuration and Initialization

The first step in programming a RACEway system is to create a configuration file which describes the hardware. This file is used by Mercury's configurator utility (configmc) to establish a configuration database for the MC/OS operating system. The database is built in memory allocated by the Mercury device driver and is consulted by MC/OS and runtime utilities to allocate resources and establish routes between processors and other devices such as bulk memory. A simple configuration file is shown below.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
# test6490.cfg
#
# Configuration file for one MCH6 and one MM6490 connected
# by an ILK4 RACEway interlink module.
#

board MCH6 -board_name MCH6
port VME -pVME 0x10000000 -pVMEws 32M -VMEil 3 -VMEiv 0xF8
port ILK
node A -memsize 16M -ptesize 0x80000
node B -memsize 16M -ptesize 0x80000

board MM6490 -board_name MM6490
port ILK
device A -device_name MEM10 -memsize 256M
#device B -device_name MEM11 -memsize 256M
#device C -device_name MEM12 -memsize 256M
#device D -device_name MEM13 -memsize 256M

board ILK4 -board_name ILK4          # Interlink module
port ILK -num 0
port ILK -num 1

connect ILK ILK4 0 MCH6         # Interlink port 0
connect ILK ILK4 1 MM6490       # Interlink port 1
```

This simple configuration file shows a single MCH6 motherboard with two compute nodes connected to the MM6490 memory board through an ILK4 interlink module. Bulk memory is configured in blocks of 256 MBytes, even if physical memory is actually smaller, so this configuration file may be used “as is” for memory boards with 64, 128 or 256 MBytes of physical memory. For a 512 MByte board, device B would be uncommented. For a 1 GByte board, devices C and D would also be uncommented. Configuration of a 2 GByte board would require addition of devices E, F, G, and H.

In order to configure the system, a *hardware description file* for the MM6490 must be placed in the directory /usr/mercury/include. This file, MM6490.desc, is not part of the standard MC/OS distribution and may be obtained from Micro Memory. The file specifies static hardware attributes of a board in terms of components and connections between components. A complete listing of the MM6490 hardware description file is provided in Section 3.5.5. Once the description file and the actual hardware are installed, the system is ready for configuration.

There are two typical host environments for a RACEway system, Solaris and VxWorks, and configuration is slightly different for each of them. Under Solaris, simply type

```
% configmc -v -cf test6490.cfg init
```

at the shell prompt to build the configuration database. The -v option provides a verbose output and can be omitted. Under VxWorks, executable versions of the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MC/OS utilities must first be loaded into the host processor before performing the configuration. From the VxWorks shell, type

```
-> ld </usr/mercury/xbin/vxworks/lib/mc_programs
-> configmc "-v -cf test6490.cfg init"
```

to load the utilities and build the configuration database. The utilities only need to be loaded once after the system is booted.

After the RACEway system has been successfully configured, a compute node must be initialized to run MM6490 test software. This is accomplished using the 'sysmc' utility under Solaris

```
% sysmc -bcs=0 init
```

or under VxWorks

```
-> sysmc "-bcs=0 init"
```

to initialize the first compute node. Once a node is initialized, it is ready to run programs under the MC/OS operating system.

3.5.2 Mapping Bulk Memory

Once the **MM-6490ADR** board is configured into the system, it can be mapped and accessed over RACEway using MC/OS system calls. This section describes how to read and write the **MM-6490ADR** one word at a time using memory pointers. The following section shows how to use the compute node's DMA controller to perform high-speed block transfers.

The first step is to attach the **MM-6490ADR** to the calling process as a raw memory device. Because the configuration file specified a **RACEway** connection between the **MCH6** and the **MM-6490ADR** through the **ILK4** Interlink module, the dev_attach() call will use a **RACEway** link from the processor to bulk memory.

```
rc = dev_attach("MEM10", 0, &mmi_devhandle);
```

Next, create a shared memory buffer on the MM6490 to make its memory accessible to MC/OS processes.

```
rc = smb_create("BUF1", SMB_RAW, BUFSIZE, MEM_DEVICE,
                  (ulong) mmi_devhandle, 0, &mmi_smbhandle);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Finally, create a mapping to the shared memory buffer and obtain a virtual address pointer to the bulk memory.

```
rc = smb_map(mmi_smbhandle, 0, 0, 0, 0, 0, 0, &mmi_maphandle,
             &mmi_mapinfo);

mmiptr = (ulong *)mmi_mapinfo.process.vaddr;
```

Once the mapping has been performed, the **MM-6490ADR** board may be accessed by dereferencing the pointer. To read from bulk memory

```
dataval = *mmiptr;
```

Or to write to bulk memory

```
*mmiptr = dataval;
```

Here we have shown unsigned long accesses, but the same method may be used read and write other types of data such as byte (unsigned char), float, or double precision.

Below is a simple test program which illustrates the techniques presented in this section.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/* mmitest1.c */

#include <stdio.h>
#include <stdlib.h>
#include <mcos.h>

#define BUFSIZE 0x1000000     /* 16 MByte */

typedef unsigned long ulong;

void main(void)
{

  DEV_handle        mmi_devhandle;
  SMB_handle        mmi_smbhandle;
  SMB_mapping_handle     mmi_maphandle;
  SMB_mapping_info  mmi_mapinfo;

  ulong   i, rc, *mmiptr, dataval, errors=0;


  /* Attach MM6490 as a RACEway device */

  rc = dev_attach("MEM10", 0, &mmi_devhandle);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Create shared memory buffer on MM6490 board */

  rc = smb_create("BUF1", SMB_RAW, BUFSIZE, MEM_DEVICE,
                  (ulong) mmi_devhandle, 0, &mmi_smbhandle);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Create mapping to shared memory buffer on MM6490 */

  rc = smb_map(mmi_smbhandle, 0, 0, 0, 0, 0, 0, &mmi_maphandle,
               &mmi_mapinfo);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Obtain a virtual address pointer to the buffer */

  mmiptr = (ulong *)mmi_mapinfo.process.vaddr;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  /* Fill the buffer with data */

  printf("Writing data to MM6490...\n");
  for (i=0; i<BUFSIZE/sizeof(ulong); i++)
    *mmiptr++ = i;
  printf("%ld words written.\n", i);


  /* Read back data and compare */

  printf("Comparing data...\n");
  mmiptr = (ulong *)mmi_mapinfo.process.vaddr;
  for (i=0; i<BUFSIZE/sizeof(ulong); i++) {
    dataval = *mmiptr++;
    if (dataval != i) {
    printf("Data miscompare: value=%ld, should be %ld\n",
dataval, i);
      errors++;
    }
  }

  /* Print results */

  printf("%ld words checked.\n", i);
  printf("Errors = %ld.\n", errors);

  /* Exit program */

  exit(0);
}
```

3.5.3 Block Transfers using DMA

In order to achieve maximum data transfer performance over RACEway, it is necessary to use the DMA controller found on each Mercury compute node. In this section we build on the simple memory mapping program described above to demonstrate how high-speed data transfers can be performed between a compute node and the **MM-6490ADR**.

The **MC/OS** operating system provides a high-speed data transfer facility known as DX. The DX facility transfers data between two *endpoints,* which are typically, shared memory buffers, using the compute node's DMA controller. In the previous section we showed how to create an SMB on the **MM-6490ADR** board. In order to use DX, we must also create an SMB on the Mercury compute node. For simplicity, we assume that the compute node is a **PowerPC**, but the same techniques apply to **i860** and **SHARC** compute nodes as well. First, create a local shared memory buffer:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
rc = smb_create("BUF2", SMB_ZERO, BUFSIZE, 0, 0, 0,
&ppc_smbhandle);
```

Then, perform the necessary mapping:

```
rc = smb_map(ppc_smbhandle, 0, 0, 0, 0, 0, 0, &ppc_maphandle,
             &ppc_mapinfo);
ppcptr = (ulong *)ppc_mapinfo.process.vaddr;
```

Once the SMBs are created and mapped, a DX transfer template is created which specifies the endpoints of the transfer:

```
rc = dx_create(mmi_smbhandle, ppc_smbhandle, 0, 0, 0, 0, 0,
               &rd_template);
```

Next, a transfer object is created which specifies the size of the transfer, buffer offsets, and an optional polling flag:

```
rc = dx_copy(rd_template, 0, 0, BUFSIZE, 0, 0, DX_POLL,
              &rd_transfer);
```

Finally, the transfer is initiated. The option DX_POLL causes the application program to wait until the transfer is completed before continuing.

```
rc = dx_start(rd_transfer, DX_POLL);
```

In some applications, the polling option is not used and the transfer occurs in the background while the CPU performs other tasks. Below is a simple test program which builds on the previous example and demonstrates high-speed data transfers using the DX facility.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/* mmitest2.c */

#include <stdio.h>
#include <stdlib.h>
#include <mcos.h>

#define BUFSIZE 0xC00000   /* 12 MByte */

typedef unsigned long ulong;

void main(void)
{

  DEV_handle          mmi_devhandle;
  SMB_handle          mmi_smbhandle, ppc_smbhandle;
  SMB_mapping_handle  mmi_maphandle, ppc_maphandle;
  SMB_mapping_info    mmi_mapinfo, ppc_mapinfo;
  DX_template         rd_template;
  DX_transfer         rd_transfer;

  ulong   i, rc, *mmiptr, *ppcptr, dataval, errors=0;


  /*****************************************/
  /* Set up shared memory buffer on MM6490 */
  /*****************************************/

  /* Attach MM6490 as a RACEway device */

  rc = dev_attach("MEM10", 0, &mmi_devhandle);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Create shared memory buffer on MM6490 board */

  rc = smb_create("BUF1", SMB_RAW, BUFSIZE, MEM_DEVICE, (ulong) mmi_devhandle,
                  0, &mmi_smbhandle);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Create mapping to shared memory buffer on MM6490 */

  rc = smb_map(mmi_smbhandle, 0, 0, 0, 0, 0, 0, &mmi_maphandle, &mmi_mapinfo);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Obtain a virtual address pointer to the buffer */

  mmiptr = (ulong *)mmi_mapinfo.process.vaddr;


  /************************************************/
  /* Set up shared memory buffer on PPC compute node */
  /************************************************/

  /* Create shared memory buffer on compute node */

  rc = smb_create("BUF2", SMB_ZERO, BUFSIZE, 0, 0, 0, &ppc_smbhandle);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Create mapping to shared memory buffer on compute node */

  rc = smb_map(ppc_smbhandle, 0, 0, 0, 0, 0, 0, &ppc_maphandle, &ppc_mapinfo);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Obtain a virtual address pointer to the buffer */

  ppcptr = (ulong *)ppc_mapinfo.process.vaddr;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  /*****************************************************/
  /* Set up DX facility for high-speed data transfers */
  /*****************************************************/

  /* Create template for PPC reading from MMI */

  rc = dx_create(mmi_smbhandle, ppc_smbhandle, 0, 0, 0, 0, 0, &rd_template);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Create transfer request for PPC reading from MMI */

  rc = dx_copy(rd_template, 0, 0, BUFSIZE, 0, 0, DX_POLL, &rd_transfer);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /*************************************/
  /* Initialize data and perform transfers */
  /*************************************/

  /* Fill the MMI buffer with data using single word access */

  printf("Writing data to MM6490...\n");
  for (i=0; i<BUFSIZE/sizeof(ulong); i++)
    *mmiptr++ = i;
  printf("%ld words written.\n", i);


  /* Transfer data from MMI buffer to PPC buffer using high-speed DX.  */
  /* DX_POLL option causes function to wait until transfer is complete */

  rc = dx_start(rd_transfer, DX_POLL);


  /* Check data in local buffer */

  printf("Comparing data...\n");
  ppcptr = (ulong *)ppc_mapinfo.process.vaddr;
  for (i=0; i<BUFSIZE/sizeof(ulong); i++) {
    dataval = *ppcptr++;
    if (dataval != i) {
      printf("Data miscompare: value = %ld, should be %ld\n", dataval, i);
      errors++;
    }
  }

  /* Print results */

  printf("%ld words checked.\n", i);
  printf("Errors = %ld.\n", errors);

  /* Exit program */

  exit(0);

}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.5.4 Measuring Transfer Rates

Often it is desirable to measure the rate at which data transfers occur. This can be accomplished using the tmr_timestamp() function of **MC/OS**. This function provides access to a high-resolution timer on the compute node and can be used to obtain accurate timing estimates. In this section we present an example of timing DMA transfers.

The setup for DX is essentially the same as in the previous section. Now we introduce the dx_modify(), tmr_timestamp(), and tmr_diff() functions. First, a loop is created to measure the transfer rate for several different size transfers. The function dx_modify() is used to change the transfer size each time through the loop. The tmr_timestamp() function is called just before the transfer is started and then immediately after the transfer completes. The tmr_diff() function returns the elapsed time which is then converted to a MByte/second rate. Results are printed in a table.

```
   for (xfersize = MINXFER; xfersize <= BUFSIZE; xfersize *= 2) {
       dx_modify(rd_transfer, 0, 0, xfersize, 0, DX_NBYTES);
       tmr_timestamp(&time1);
       rc = dx_start(rd_transfer, DX_POLL);
       tmr_timestamp(&time2);
       tmr_diff(time1, time2, 0, &elapsed);
       seconds = (float)elapsed.tv_sec +
                 (float)elapsed.tv_nsec/1000000000.0;
       megabytes = (float)xfersize/(float)MEGBYTE;
       printf("%8ld      %8.3f\n", xfersize,
   (megabytes/seconds));
     }
```

Below is a test program illustrating the use of timers. A test run of this program using a PowerPC compute node produced the following output:

```
% runmc mmitest3
Writing data to MM6490...
2097152 words written.
Timing reads using DX...
XferSize       MBytes/Sec
--------       ----------
      16         1.298
      32         2.569
      64         5.061
     128         9.376
     256        17.769
     512        32.815
    1024        53.894
    2048        78.063
    4096       100.625
    8192       117.517
   16384       127.219
   32768       133.650
   65536       136.576
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  131072       138.125
  262144       139.060
  524288       139.463
 1048576       139.685
 2097152       139.791
 4194304       139.814
 8388608       139.846

Comparing data...
2097152 words checked.
Errors = 0.
Timing writes using DX...
XferSize       MBytes/Sec
--------       ----------
      16         1.272
      32         2.649
      64         5.217
     128        10.072
     256        18.926
     512        33.628
    1024        53.364
    2048        80.774
    4096       102.044
    8192       118.156
   16384       128.347
   32768       134.235
   65536       136.971
  131072       138.778
  262144       139.498
  524288       139.908
 1048576       140.118
 2097152       140.148
 4194304       140.227
 8388608       140.256
```

The test program is presented below.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/* mmitest3.c */

#include <stdio.h>
#include <stdlib.h>
#include <mcos.h>

#define BUFSIZE 0x800000   /* 8 MByte */
#define MINXFER 16
#define MEGBYTE (1024*1024)

typedef unsigned long ulong;

void main(void)
{

  DEV_handle              mmi_devhandle;
  SMB_handle              mmi_smbhandle, ppc_smbhandle;
  SMB_mapping_handle      mmi_maphandle, ppc_maphandle;
  SMB_mapping_info        mmi_mapinfo, ppc_mapinfo;
  DX_template       rd_template, wr_template;
  DX_transfer       rd_transfer, wr_transfer;
  TMR_ts                  time1, time2;
  TMR_timespec            elapsed;

  ulong   i, rc, *mmiptr, *ppcptr, dataval, errors=0;
  ulong   xfersize;
  float   seconds, megabytes;


  /*****************************************/
  /* Set up shared memory buffer on MM6490 */
  /*****************************************/

  /* Attach MM6490 as a RACEway device */

  rc = dev_attach("MEM10", 0, &mmi_devhandle);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Create shared memory buffer on MM6490 board */

  rc = smb_create("BUF1", SMB_RAW, BUFSIZE, MEM_DEVICE,
                  (ulong) mmi_devhandle, 0, &mmi_smbhandle);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Create mapping to shared memory buffer on MM6490 */

  rc = smb_map(mmi_smbhandle, 0, 0, 0, 0, 0, 0, &mmi_maphandle, &mmi_mapinfo);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Obtain a virtual address pointer to the buffer */

  mmiptr = (ulong *)mmi_mapinfo.process.vaddr;


  /************************************************/
  /* Set up shared memory buffer on PPC compute node */
  /************************************************/

  /* Create shared memory buffer on compute node */

  rc = smb_create("BUF2", SMB_ZERO, BUFSIZE, 0, 0, 0, &ppc_smbhandle);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Create mapping to shared memory buffer on compute node */

  rc = smb_map(ppc_smbhandle, 0, 0, 0, 0, 0, 0, &ppc_maphandle, &ppc_mapinfo);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Obtain a virtual address pointer to the buffer */

  ppcptr = (ulong *)ppc_mapinfo.process.vaddr;


  /*************************************************/
  /* Set up DX facility for high-speed data transfers */
  /*************************************************/

  /* Create template for PPC reading from MMI */

  rc = dx_create(mmi_smbhandle, ppc_smbhandle, 0, 0, 0, 0, 0, &rd_template);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Create template for PPC writing to MMI */

  rc = dx_create(ppc_smbhandle, mmi_smbhandle, 0, 0, 0, 0, 0, &wr_template);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Create transfer request for PPC reading from MMI */

  rc = dx_copy(rd_template, 0, 0, BUFSIZE, 0, 0, DX_POLL, &rd_transfer);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /* Create transfer request for PPC writing to MMI */

  rc = dx_copy(wr_template, 0, 0, BUFSIZE, 0, 0, DX_POLL, &wr_transfer);
  if (rc != 0) err_print(rc, ERR_GET_ALL);


  /*************************************/
  /* Initialize data and perform transfers */
  /*************************************/

  /* Fill the MMI buffer with data using single word access */

  printf("Writing data to MM6490...\n");
  for (i=0; i<BUFSIZE/sizeof(ulong); i++)
    *mmiptr++ = i;
  printf("%ld words written.\n", i);


  /*****************************************************************/
  /* This section times the reads from MMI to PPC.  The size of the   */
  /* transfer is modified each time by calling dx_modify() with the   */
  /* DX_NBYTES flag.  The function tmr_timestamp() is used to read    */
  /* the compute node's high-resolution counter just before starting  */
  /* the transfer and just after it complets.  The rate for each size */
  /* transfer is then printed to form a table of transfer size versus */
  /* transfer rate.                                                   */
  /*****************************************************************/

  printf("Timing reads using DX...\n");
  printf("XferSize       MBytes/Sec\n");
  printf("--------       ----------\n");
  for (xfersize = MINXFER; xfersize <= BUFSIZE; xfersize *= 2) {
    dx_modify(rd_transfer, 0, 0, xfersize, 0, DX_NBYTES);
    tmr_timestamp(&time1);
    rc = dx_start(rd_transfer, DX_POLL);
    tmr_timestamp(&time2);
    tmr_diff(time1, time2, 0, &elapsed);
    seconds = (float)elapsed.tv_sec + (float)elapsed.tv_nsec/1000000000.0;
    megabytes = (float)xfersize/(float)MEGBYTE;
    printf("%8ld      %8.3f\n", xfersize, (megabytes/seconds));
  }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  /* After the final read, check data in local buffer */

  printf("Comparing data...\n");
  ppcptr = (ulong *)ppc_mapinfo.process.vaddr;
  for (i=0; i<BUFSIZE/sizeof(ulong); i++) {
    dataval = *ppcptr++;
    if (dataval != i) {
      printf("Data miscompare: value = %ld, should be %ld\n", dataval, i);
      errors++;
    }
  }


  /* Print results of data check */

  printf("%ld words checked.\n", i);
  printf("Errors = %ld.\n", errors);


  /* Now time the writes */

  printf("Timing writes using DX...\n");
  printf("XferSize       MBytes/Sec\n");
  printf("--------       ----------\n");
  for (xfersize = MINXFER; xfersize <= BUFSIZE; xfersize *= 2) {
    dx_modify(wr_transfer, 0, 0, xfersize, 0, DX_NBYTES);
    tmr_timestamp(&time1);
    rc = dx_start(wr_transfer, DX_POLL);
    tmr_timestamp(&time2);
    tmr_diff(time1, time2, 0, &elapsed);
    seconds = (float)elapsed.tv_sec + (float)elapsed.tv_nsec/1000000000.0;
    megabytes = (float)xfersize/(float)MEGBYTE;
    printf("%8ld      %8.3f\n", xfersize, (megabytes/seconds));
  }

  /* Exit program */

  exit(0);

}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.5.5 Hardware Description File for MM6490

```
# /usr/mercury/include/MM6490.desc
#
# Description file for MM6490, Micro Memory Inc's RACEway memory card.
# This describes a MM6490 that has ficticious xbars, one connected to
# the other.  This will allow us to address up to 256MB at each endpoint
# of the xbar up to a total of 2GB.

MM6490    # Name
0x1FFFFF  # Nodes Memory Alignment
None      # Default CE Type

8 # Slots
#Slot 0
  DEV  A
       END                          # Flags
       {XBar 1 F}                   # Device 0 Connection List
       {Slot DEV 0 {None},          # Device 0 Route List
        Slot DEV 1 {ILK 1 E},
        Slot DEV 2 {ILK 1 D},
        Slot DEV 3 {ILK 1 C},
        Slot DEV 4 {ILK 3 AFF},
        Slot DEV 5 {ILK 3 AFE},
        Slot DEV 6 {ILK 3 AFD},
        Slot DEV 7 {ILK 3 AFC},
        Port ILK 0 {ILK 2 AA}}

  DEV  B
       END                          # Flags
       {XBar 1 E}                   # Device 1 Connection List
       {Slot DEV 0 {ILK 1 F},       # Device 0 Route List
        Slot DEV 1 {None},
        Slot DEV 2 {ILK 1 D},
        Slot DEV 3 {ILK 1 C},
        Slot DEV 4 {ILK 3 AFF},
        Slot DEV 5 {ILK 3 AFE},
        Slot DEV 6 {ILK 3 AFD},
        Slot DEV 7 {ILK 3 AFC},
        Port ILK 0 {ILK 2 AA}}

  DEV  C
       END                          # Flags
       {XBar 1 D}                   # Device 2 Connection List
       {Slot DEV 0 {ILK 1 F},       # Device 0 Route List
        Slot DEV 1 {ILK 1 E},
        Slot DEV 2 {None},
        Slot DEV 3 {ILK 1 C},
        Slot DEV 4 {ILK 3 AFF},
        Slot DEV 5 {ILK 3 AFE},
        Slot DEV 6 {ILK 3 AFD},
        Slot DEV 7 {ILK 3 AFC},
        Port ILK 0 {ILK 2 AA}}

  DEV  D
       END                          # Flags
       {XBar 1 C}                   # Device 3 Connection List
       {Slot DEV 0 {ILK 1 F},       # Device 0 Route List
        Slot DEV 1 {ILK 1 E},
        Slot DEV 2 {ILK 1 D},
        Slot DEV 3 {None},
        Slot DEV 4 {ILK 3 AFF},
        Slot DEV 5 {ILK 3 AFE},
        Slot DEV 6 {ILK 3 AFD},
        Slot DEV 7 {ILK 3 AFC},
        Port ILK 0 {ILK 2 AA}}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  DEV  E
       END                          # Flags
       {XBar 2 F}                   # Device 4 Connection List
       {Slot DEV 0 {ILK 3 AFF},     # Device 0 Route List
        Slot DEV 1 {ILK 3 AFE},
        Slot DEV 2 {ILK 3 AFD},
        Slot DEV 3 {ILK 3 AFC},
        Slot DEV 4 {None},
        Slot DEV 5 {ILK 1 E},
        Slot DEV 6 {ILK 1 D},
        Slot DEV 7 {ILK 1 C},
        Port ILK 0 {ILK 2 AA}}

  DEV  F
       END                          # Flags
       {XBar 2 E}                   # Device 5 Connection List
       {Slot DEV 0 {ILK 3 AFF},     # Device 0 Route List
        Slot DEV 1 {ILK 3 AFE},
        Slot DEV 2 {ILK 3 AFD},
        Slot DEV 3 {ILK 3 AFC},
        Slot DEV 4 {ILK 1 F},
        Slot DEV 5 {None},
        Slot DEV 6 {ILK 1 D},
        Slot DEV 7 {ILK 1 C},
        Port ILK 0 {ILK 2 AA}}

  DEV  G
       END                          # Flags
       {XBar 2 D}                   # Device 6 Connection List
       {Slot DEV 0 {ILK 3 AFF},     # Device 0 Route List
        Slot DEV 1 {ILK 3 AFE},
        Slot DEV 2 {ILK 3 AFD},
        Slot DEV 3 {ILK 3 AFC},
        Slot DEV 4 {ILK 1 F},
        Slot DEV 5 {ILK 1 E},
        Slot DEV 6 {None},
        Slot DEV 7 {ILK 1 C},
        Port ILK 0 {ILK 2 AA}}

  DEV  H
       END                          # Flags
       {XBar 2 C}                   # Device 7 Connection List
       {Slot DEV 0 {ILK 3 AFF},     # Device 0 Route List
        Slot DEV 1 {ILK 3 AFE},
        Slot DEV 2 {ILK 3 AFD},
        Slot DEV 3 {ILK 3 AFC},
        Slot DEV 4 {ILK 1 F},
        Slot DEV 5 {ILK 1 E},
        Slot DEV 6 {ILK 1 D},
        Slot DEV 7 {None},
        Port ILK 0 {ILK 2 AA}}

1 # Ports
  ILK                           # Interlink Port 0
       {XBar 0 A}               # Connection List
       {Slot DEV 0 {ILK 2 HH},  # Route List
        Slot DEV 1 {ILK 2 HG},
        Slot DEV 2 {ILK 2 HF},
        Slot DEV 3 {ILK 2 HE},
        Slot DEV 4 {ILK 2 HD},
        Slot DEV 5 {ILK 2 HC},
        Slot DEV 6 {ILK 2 HB},
        Slot DEV 7 {ILK 2 HA}}

3 # XBars
  6PORT END                             # XBar Type
    {Port ILK 0; None; None;            # XBar 0 Connection List
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        None; XBar 2 A; XBar 1 A}

   6PORT END                        # XBar Type
     {XBar 0 F; None; Slot DEV 3;  # XBar 1 Connection List
        Slot DEV 2; Slot DEV 1; Slot DEV 0}
   6PORT END                              # XBar Type
     {XBar 0 E; None; Slot DEV 7;  # XBar 2 Connection List
        Slot DEV 6; Slot DEV 5; Slot DEV 4}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDIX A

MODULE SELECTION

VME Jumper Table (A31 - A28)

LOW LIMIT E24 thru E27					HIGH LIMIT E13 thru E16				
	1= Removed				0=Installed				
Jumper POSITION	**24**	**25**	**26**	**27**	**Jumper POSITION**	**13**	**14**	**15**	**16**
$0yz00000	0	0	0	0	$0yzFFFFF	0	0	0	0
$1yz00000	0	0	0	1	$1yzFFFFF	0	0	0	1
$2yz00000	0	0	1	0	$2yzFFFFF	0	0	1	0
$3yz00000	0	0	1	1	$3yzFFFFF	0	0	1	1
$4yz00000	0	1	0	0	$4yzFFFFF	0	1	0	0
$5yz00000	0	1	0	1	$5yzFFFFF	0	1	0	1
$6yz00000	0	1	1	0	$6yzFFFFF	0	1	1	0
$7yz00000	0	1	1	1	$7yzFFFFF	0	1	1	1
$8yz00000	1	0	0	0	$8yzFFFFF	1	0	0	0
$9yz00000	1	0	0	1	$9yzFFFFF	1	0	0	1
$Ayz00000	1	0	1	0	$AyzFFFFF	1	0	1	0
$Byz00000	1	0	1	1	$ByzFFFFF	1	0	1	1
$Cyz00000	1	1	0	0	$CyzFFFFF	1	1	0	0
$Dyz00000	1	1	0	1	$DyzFFFFF	1	1	0	1
$Eyz00000	1	1	1	0	$EyzFFFFF	1	1	1	0
$Fyz00000	1	1	1	1	$FyzFFFFF	1	1	1	1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

VME Jumper Table (A27 - A24)

LOW LIMIT E28 thru E31					HIGH LIMIT E17 thru E20				
Jumper POSITION	**28**	**29**	**30**	**31**	**Jumper POSITION**	**17**	**18**	**19**	**20**
$x0z00000	0	0	0	0	$x0zFFFFF	0	0	0	0
$x1z00000	0	0	0	1	$x1zFFFFF	0	0	0	1
$x2z00000	0	0	1	0	$x2zFFFFF	0	0	1	0
$x3z00000	0	0	1	1	$x3zFFFFF	0	0	1	1
$x4z00000	0	1	0	0	$x4zFFFFF	0	1	0	0
$x5z00000	0	1	0	1	$x5zFFFFF	0	1	0	1
$x6z00000	0	1	1	0	$x6zFFFFF	0	1	1	0
$x7z00000	0	1	1	1	$x7zFFFFF	0	1	1	1
$x8z00000	1	0	0	0	$x8zFFFFF	1	0	0	0
$x9z00000	1	0	0	1	$x9zFFFFF	1	0	0	1
$xAz00000	1	0	1	0	$xAzFFFFF	1	0	1	0
$xBz00000	1	0	1	1	$xBzFFFFF	1	0	1	1
$xCz00000	1	1	0	0	$xCzFFFFF	1	1	0	0
$xDz00000	1	1	0	1	$xDzFFFFF	1	1	0	1
$xEz00000	1	1	1	0	$xEzFFFFF	1	1	1	0
$xFz00000	1	1	1	1	$xFzFFFFF	1	1	1	1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

VME Jumper Table (A23 - A20)

LOW LIMIT E32 thru E35					HIGH LIMIT E21 thru E24				
Jumper POSITION	**32**	**33**	**34**	**35**	**Jumper POSITION**	**21**	**22**	**23**	**24**
$xy000000	0	0	0	0	$xy0FFFFF	0	0	0	0
$xy100000	0	0	0	1	$xy1FFFFF	0	0	0	1
$xy200000	0	0	1	0	$xy2FFFFF	0	0	1	0
$xy300000	0	0	1	1	$xy3FFFFF	0	0	1	1
$xy400000	0	1	0	0	$xy4FFFFF	0	1	0	0
$xy500000	0	1	0	1	$xy5FFFFF	0	1	0	1
$xy600000	0	1	1	0	$xy6FFFFF	0	1	1	0
$xy700000	0	1	1	1	$xy7FFFFF	0	1	1	1
$xy800000	1	0	0	0	$xy8FFFFF	1	0	0	0
$xy900000	1	0	0	1	$xy9FFFFF	1	0	0	1
$xyA00000	1	0	1	0	$xyAFFFFF	1	0	1	0
$XyB00000	1	0	1	1	$xyBFFFFF	1	0	1	1
$XyC00000	1	1	0	0	$xyCFFFFF	1	1	0	0
$XyD00000	1	1	0	1	$xyDFFFFF	1	1	0	1
$XyE00000	1	1	1	0	$xyEFFFFF	1	1	1	0
$XyF00000	1	1	1	1	$xyFFFFFF	1	1	1	1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDIX B

U1 PLD EQUATIONS

```
module          u1;
TITLE          'U1 - (PN46xxx) - MM-6490 VMEbus AM Control';
" REVISION      NC
" AUTHOR        Michael Hasenfratz - Created: May 17, 1995 [04:17pm]
" COMPANY       Micro Memory Inc. Chatsworth, Ca.
" DATE          May 30, 1995 [01:53pm]

"   This device controls the VMEbus Address Modifier/Selection
u1              device 'p22v10';

am0             pin  1  ; " Address Modifier 0
am1             pin  2  ; " Address Modifier 1
am2             pin  3  ; " Address Modifier 2
am3             pin  4  ; " Address Modifier 3
am4             pin  5  ; " Address Modifier 4
am5             pin  6  ; " Address Modifier 5
!iack           pin  7  ; " Interrupt ACKnowlage
bsysrst         pin  8  ; " System RESET
as              pin  9  ; " Address Strobe
dis             pin 10  ; " VMEbus DISable
wr              pin 11  ; " VME WRite
selcsr          pin 13  ; " Select CSR
d64             pin 15  ; " VME D64 (MBLT) AM
shtblk          pin 16  ; " SHORT BLOCK AM
blk             pin 17  ; " BLOCK MODE REQUESTED
!wrcsr          pin 18  ; " WRITE to CSR
!rdcsr          pin 19  ; " READ from CSR
!amcsr          pin 20  ; " VME I/O (A16) AM
!am24           pin 22  ; " VME A24 AM
!am32           pin 23  ; " VME A32 AM
```

EQUATIONS

```
am32    = !am5 & !am4 &  am3 & !am2 & !am1 &  am0 & !bsysrst & !iack & !dis
        # !am5 & !am4 &  am3 & !am2 &  am1 & !am0 & !bsysrst & !iack & !dis
        # !am5 & !am4 &  am3 & !am2 &  am1 &  am0 & !bsysrst & !iack & !dis
        # !am5 & !am4 &  am3 &  am2 & !am1 &  am0 & !bsysrst & !iack & !dis
        # !am5 & !am4 &  am3 &  am2 &  am1 & !am0 & !bsysrst & !iack & !dis
        # !am5 & !am4 &  am3 &  am2 &  am1 &  am0 & !bsysrst & !iack & !dis
        # !am5 & !am4 &  am3 & !am2 & !am1 & !am0 & !bsysrst & !iack & !dis
        # !am5 & !am4 &  am3 &  am2 & !am1 & !am0 & !bsysrst & !iack & !dis

am24    =  am5 &  am4 &  am3 & !am2 & !am1 &  am0 & !bsysrst & !iack & !dis
        #  am5 &  am4 &  am3 & !am2 &  am1 & !am0 & !bsysrst & !iack & !dis
        #  am5 &  am4 &  am3 & !am2 &  am1 &  am0 & !bsysrst & !iack & !dis
        #  am5 &  am4 &  am3 &  am2 & !am1 &  am0 & !bsysrst & !iack & !dis
        #  am5 &  am4 &  am3 &  am2 &  am1 & !am0 & !bsysrst & !iack & !dis
        #  am5 &  am4 &  am3 &  am2 &  am1 &  am0 & !bsysrst & !iack & !dis
        #  am5 &  am4 &  am3 & !am2 & !am1 & !am0 & !bsysrst & !iack & !dis
        #  am5 &  am4 &  am3 &  am2 & !am1 & !am0 & !bsysrst & !iack & !dis
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
amcsr   =  am5 & !am4 &  am3 & !am2 & !am1 &  am0 & !bsysrst & !iack
        #  am5 & !am4 &  am3 &  am2 & !am1 &  am0 & !bsysrst & !iack

blk     = !am5 & !am4 &  am3 & !am2 &  am1 &  am0 & !bsysrst & !iack & !dis
        # !am5 & !am4 &  am3 &  am2 &  am1 &  am0 & !bsysrst & !iack & !dis
        # !am5 & !am4 &  am3 & !am2 & !am1 & !am0 & !bsysrst & !iack & !dis
        # !am5 & !am4 &  am3 &  am2 & !am1 & !am0 & !bsysrst & !iack & !dis
        #  am5 &  am4 &  am3 & !am2 &  am1 &  am0 & !bsysrst & !iack & !dis
        #  am5 &  am4 &  am3 &  am2 &  am1 &  am0 & !bsysrst & !iack & !dis
        #  am5 &  am4 &  am3 & !am2 & !am1 & !am0 & !bsysrst & !iack & !dis
        #  am5 &  am4 &  am3 &  am2 & !am1 & !am0 & !bsysrst & !iack & !dis

d64     = !am5 & !am4 &  am3 & !am2 & !am1 & !am0 & !bsysrst & !iack & !dis
        # !am5 & !am4 &  am3 &  am2 & !am1 & !am0 & !bsysrst & !iack & !dis
        #  am5 &  am4 &  am3 & !am2 & !am1 & !am0 & !bsysrst & !iack & !dis
        #  am5 &  am4 &  am3 &  am2 & !am1 & !am0 & !bsysrst & !iack & !dis

shtblk  =  am5 &  am4 &  am3 & !am2 &  am1 &  am0 & !bsysrst & !iack & !dis
        #  am5 &  am4 &  am3 &  am2 &  am1 &  am0 & !bsysrst & !iack & !dis
        # !am5 & !am4 &  am3 & !am2 &  am1 &  am0 & !bsysrst & !iack & !dis
        # !am5 & !am4 &  am3 &  am2 &  am1 &  am0 & !bsysrst & !iack & !dis

rdcsr   =  am5 & !am4 &  am3 & !am2 & !am1 &  am0 & !bsysrst & !iack & !wr &  selcsr
        #  am5 & !am4 &  am3 &  am2 & !am1 &  am0 & !bsysrst & !iack & !wr &  selcsr

wrcsr   =  am5 & !am4 &  am3 & !am2 & !am1 &  am0 & !bsysrst & !iack &  wr &  selcsr
        #  am5 & !am4 &  am3 &  am2 & !am1 &  am0 & !bsysrst & !iack &  wr &  selcsr
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP