Sun StorEdge™ A7000 MEMORY CHANNEL™ IV System Diagnostic Reference Manual

THE NETWORK IS THE COMPUTER™

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900 USA
650 960-1300 Fax 650 969-9131

Part No. 805-4899-10
April 1999, Revision A

Send comments about this document to: docfeedback@sun.com

Please
Recycle

Contents

Figures

Tables

Preface

Sun StorEdge A7000 MEMORY CHANNEL IV System Diagnostic Reference Manual is specific to the Extended Diagnostic provided for testing the MEMORY CHANNEL™ IV module and the Transition Module Interface (TMI). This manual contains the following information:

- Special requirements for program execution
- Program initialization procedures
- Test descriptions
- Descriptions of messages produced by this program

How This Book Is Organized

Chapter 1 “Diagnostic Overview” describes the Extended Diagnostic used for testing the MEMORY CHANNEL IV module and Transition Module Interface (TMI). It identifies special requirements for test execution and describes the testing modes supported by this program.

Chapter 2 “Program Initialization” describes the initialization procedure used to run this diagnostic in interactive mode. It provides descriptions of the initialization options, program initialization procedures, and diagnostic execution examples.

Chapter 3 “Program Tests” describes the individual diagnostic tests.

Chapter 4 “Program Messages” describes the messages produced by the diagnostic during interactive mode execution.

Typographic Conventions

TABLE P-1 Typographic Conventions

Typeface or Symbol	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output.	Use the `setconfig` command to change the testing configuration. `Do you need help? Enter Y or N (CR)`
`AaBbCc123`	What you type, when contrasted with on-screen computer output.	`MCA: continue on error; [cr,d,q]?`**`d`**
AaBbCc123	Book titles, new words or terms, words to be emphasized. Variable expressions; replaced with a real name or value.	Read Chapter 2 in the *Sun StorEdge A7000 ROM Monitor Reference Manual.* `slot` *`n`*`:Loading extended image mc4diag`
|	In system output displays, a vertical line indicates that a choice will be made between the values.	In the following example, either `unexpected` or `missing` will be displayed: *bit* `0x`*`bbbbbbbb`* `is unexpected|missing`

TABLE P-1 Typographic Conventions

Typeface or Symbol	Meaning	Examples
`[ ]`	In system output examples, brackets indicate optional values. If several values are placed inside brackets, any or none of them can be displayed. Brackets are also used in system prompts to enclose the response choices.	In the following example, displaying the slot number is optional: `[Slot` *n*`:]`
. . .	The vertical ellipsis indicates continuation of system output.	In the following example, additional information would be displayed in place of the vertical ellipsis: `0x`*addr* . . .
`< >`	In examples of command input, an item surrounded by a greater than and less than sign must be replaced with an action. The greater than and less than signs are omitted when performing the action.	`<CR>` means press the Return key.

Related Documentation

TABLE P-2 Related Documentation

Type	Title
User interface	*Sun StorEdge A7000 ROM Monitor Reference Manual*
Diagnostic reference	*Sun StorEdge A7000 Diagnostics Reference Manual*

Sun Documentation on the Web

The `docs.sun.com`[sm] web site enables you to access Sun technical documentation on the Web. You can browse the `docs.sun.com` archive or search for a specific book title or subject at:

```
http://docs.sun.com
```

Sun Welcomes Your Comments

We are interested in improving our documentation and welcome your comments and suggestions. You can email your comments to us at:

```
docfeedback@sun.com
```

Please include the part number of your document in the subject line of your email.

CHAPTER 1

Diagnostic Overview

Introduction

The MEMORY CHANNEL IV System Diagnostic is an extended diagnostic used to verify the functionality of the MEMORY CHANNEL IV module and to ensure that the module will work properly in the actual target environment. This diagnostic is also used to verify the functionality of the Transition Module Interface (TMI). The diagnostic can run tests in one of two major modes: Local Loopback or Node-to-node testing.

In the Local Loopback mode, you can test the MEMORY CHANNEL IV module with or without a TMI. The diagnostic initialization defaults to No TMI Testing. When Local Loopback and No TMI testing are selected, only the MEMORY CHANNEL IV module is tested and no transactions take place with other modules on the MEMORY CHANNEL Bus (the MEMORY CHANNEL IV module operates in Local Loopback). When both Local Loopback and TMI testing are selected, the TMI can be placed in either Internal Loopback or External Loopback. In both cases, the MEMORY CHANNEL IV module operates in normal mode (loopback is disabled), which permits transactions to take place with the TMI modules on the MEMORY CHANNEL Bus. In Internal Loopback, the TMI does not transmit commands or data to another TMI, but sends data back across the MEMORY CHANNEL Bus. In External Loopback, the TMI sends commands and data to a remote TMI, receives the same data from the remote TMI, and sends it back across the originating MEMORY CHANNEL Bus.

In Node-to-node testing, one MEMORY CHANNEL IV module must be the host and all others will be remote nodes. In this mode, at least two nodes must be used. They may be connected using TMI boards.

Each MEMORY CHANNEL IV module consists of two boards: the ECC Memory board (EDRAM) and the MEMORY CHANNEL IV Adapter board (MCA). The EDRAM contains 128MB, 256MB, or 512MB of memory and interfaces to the Local Bus, VMEbus, and MCA board. The MCA board is mounted as a Mezzanine board on the EDRAM and provides the interface to the MEMORY CHANNEL Bus. This diagnostic tests the following functional areas of the EDRAM board:

- **ECC Memory.** This area contains a 128MB, 256MB, or 512MB memory array of DRAMs with error correcting capabilities.
- **Local Bus Slave Interface.** This area provides any Local Bus Master with a path to access the on-board memory.
- **VMEbus Slave Interface.** This area provides any VMEbus Master with a path to access the on-board memory through VMEbus extended (A32) address space. It also supports D32:BLT and D64:BLT block transfers.
- **Mezzanine Interface.** This area transmits and receives addresses and data from the MCA.
- **Expansion DRAM Command and Status Registers.** The Command and Status Registers (CSRs) provide software control of all memory, Local Bus interface, and VMEbus interface functions.

The diagnostic also tests the following functional areas of the MCA board:

- **Read Sense Controller (RSC).** This area monitors inputs from the MCbus and clocks those falling within the enabled receive windows into the receive FIFOs.
- **Write Sense Controller (WSC).** This area monitors memory write transaction addresses from the Local Bus and VMEbus ports of the Expansion DRAM. If the write in progress is within an enabled transmit window, the data and translated address together with the appropriate flags are clocked into the transmit FIFOs.
- **MEMORY CHANNEL Engine (MCE).** This area initiates transfers from the Expansion DRAM to the transmit FIFOs. It transmits DMA command block transfer data and tags into the transmit FIFOs.
- **Mezzanine Interface.** This area transmits and receives addresses between the MCA and the mezzanine port of Expansion DRAM. The interface internally arbitrates between the Read Sense Controller (RSC) and MEMORY CHANNEL Engine (MCE) for the use of the mezzanine port.
- **MCbus Interface.** This area transmits and receives data and addresses to and from a remote MCbus node.
- **MEMORY CHANNEL Command and Status Registers.** The Command and Status Registers (CSRs) allow software control of all MEMORY CHANNEL and REFLECTIVE MEMORY functions on the MCA. This includes transmit and receive map RAM, MC start address, transfer byte count, and various control, status, and diagnostic functions.

The Transition Module Interface (TMI) board converts the parallel MEMORY CHANNEL Bus to a high-speed serial data stream. The TMI provides a bus extender/bridge between MEMORY CHANNEL buses. This diagnostic tests the following functional areas of the TMI board:

- **MEMORY CHANNEL IV Bus Write Interface.** This area handles all write traffic from the serial hose to the MEMORY CHANNEL IV Bus.
- **MEMORY CHANNEL IV Bus Read Interface.** This area handles all read traffic from the MEMORY CHANNEL IV Bus to the serial hose.
- **Down Hose Interface.** This area reads frames from the serial hose and passes them to the down data FIFOs.
- **Up Hose Interface.** This area reads MEMORY CHANNEL block data from the up data FIFOs, optionally routes the data through the window RAMs, and writes it to the serialization logic.
- **Serialization Logic.** This area reads words from the data stream, encodes the data to embed the clock in the serial bit stream, computes a Cyclic Redundancy Code (CRC), and serially shifts the data out to the up hose connector.
- **Control Switch.** This area provides a route between the up data path and the down data path for the movement of control and programming data.

This is a disk based diagnostic normally executed during the power up testing sequence. If configured, the program is automatically called from the ROM Monitor after the Built-in Self Tests (BISTs) are executed. You can also run this diagnostic using individual ROM Monitor commands. Refer to the *Sun StorEdge A7000 Diagnostics Reference Manual* for related commands and procedures.

Special Requirements

This diagnostic has the following special requirements:

- When running the diagnostic on a MEMORY CHANNEL IV module with no cables attached, you must jumper the module as Master to enable the bus arbitration. If cables are attached and multiple modules are configured, only one module must be jumpered as Master to enable the bus arbitration.
- Multiple active nodes on the MEMORY CHANNEL Bus must be assigned unique node IDs in incrementing order.
- To prevent bus saturation, do not run the MEMORY CHANNEL IV System diagnostic while other nodes on the same MEMORY CHANNEL Bus are active.
- Refer to the Installation manuals provided with your system for MEMORY CHANNEL IV module jumpering information.
- Test 40 requires one of the following controller and disk drive configurations. No scratch disk is required.
  - V/SCSI and a WREN VII SCSI disk drive.
  - V/SCSI-2 and a WREN VII SCSI disk drive.
  - V/SCSI-2 and a ST15150 Barracuda disk drive.
- Test 41 requires a V/SCSI-2 and an ST15150 Barracuda disk drive. No scratch disk is required.

Testing Considerations

- When configuring the MEMORY CHANNEL IV System diagnostic for automatic testing, you can preselect the Local Bus address and VMEBus address using the `setconfig scsr 2` and `setconfig scsr 3` commands, respectively. Refer to the *Sun StorEdge A7000 Diagnostics Reference Manual* for configuration information.
- Multiple copies of the MEMORY CHANNEL IV System Diagnostic are required for node-to-node testing. Load one copy into the host and one copy into each remote node to be tested.
- Use Control-C to terminate an operation in interactive mode. Any environment variable changes made under the control of this diagnostic will be preserved when the Control-C sequence is initiated.
- Use the Escape key to abort an Extended Diagnostic and return control to the interactive mode options menu. Any environment variable changes made under the control of this diagnostic will be preserved when the escape sequence is initiated.

Testing Modes

The MEMORY CHANNEL IV System Diagnostic provides the following modes for testing single and multiple nodes:

- Local Loopback (LL)
- Host in Remote (HR)
- Node in Remote (NR)

In addition, the diagnostic provides the following modes for testing the Transition Module Interface (TMI):

- TMI Local Loopback
- TMI External Loopback

You specify the testing mode during program initialization when running the diagnostic in interactive mode. Refer to Chapter 2 for initialization information. For automatic testing, the default mode (Local Loopback) is used.

Local Loopback (LL) Mode

The Local Loopback (LL) testing mode uses the local loopback feature provided by the hardware to test a single MEMORY CHANNEL IV module. The module output is wrapped around to the input without transferring data to other nodes on the MEMORY CHANNEL Bus. Under Local Loopback mode, TMI testing may be selected. In this case, the diagnostic runs additional tests that operate the MEMORY CHANNEL IV module in normal mode (loopback disabled). This permits transactions onto the MEMORY CHANNEL Bus for testing the TMI modules. Local Loopback and No TMI Testing is the default testing mode.

Host in Remote (HR) Mode

Use the Host in Remote (HR) mode to test remote nodes on the same MEMORY CHANNEL Bus. One node is assigned as host and the others are assigned as nodes. The host controls and synchronizes the actions of each node on the MEMORY CHANNEL Bus. Only one host may be configured. In this mode, the host transmits and receives data. When TMI testing is enabled, node-to-node transactions are passed through the TMI modules.

Note – If the host or any node has 128 Megabytes of RAM, the maximum logical node number is 0xD.

If the host or any node has 256 or 512 Megabytes of RAM, the maximum logical node number is 0xF.

Node in Remote (NR) Mode

Use the Node in Remote (NR) mode to remotely test up to nine nodes on the same MEMORY CHANNEL Bus. This testing mode requires two or more nodes. One node is assigned as host and the other eight are assigned as nodes. The host transmits commands to the nodes for execution. The nodes execute the command, respond to the host, and wait for the next command. In this mode, the node transmits and receives data.

By using one or more TMI's, you can expand the total number of nodes to 16, one host and up to 15 remote nodes.

Note – If the host or any node has 128 Megabytes of RAM, the maximum logical node number is 0xD.

If the host or any node has 256 or 512 Megabytes of RAM, the maximum logical node number is 0xF.

TMI Local Loopback Mode

The TMI Local Loopback testing mode uses the TMI local loopback feature of the hardware to test a single Transition Module Interface (TMI) board. The board output is wrapped around to the input without transferring data across the serial hose to another TMI.

TMI External Loopback Mode

The TMI External Loopback testing mode transfers data from a local TMI through the FTM and back to the local TMI. This mode requires an external optical jumper cable connecting the local FTM's transmit port to its receive port. The local TMI sends data to the FTM and the external optical cable forces the data back into the FTM, through the local TMI's receive buffers, and back to the MEMORY CHANNEL IV bus. This mode is different from the TMI Local Loopback mode because it includes the FTM in the data path.

CHAPTER 2

Program Initialization

Introduction

When running in interactive mode, the MEMORY CHANNEL IV System Diagnostic provides initialization options that allow you to:

- Specify the bus configuration for testing
- Specify the EDRAM board slot number
- Specify the testing mode
- Select the node to be tested
- Select the memory test area
- Enable milestone messages
- Loop on individual subtests
- Select the SCSI disk drive used for testing
- Specify TMI testing
- Select the TMI testing mode
- Specify the TMI Bus and node ID

To initialize the diagnostic, select the `Initialize Diag.` option from the Interactive Mode Options menu. The interactive mode options are described in the *Sun StorEdge A7000 Diagnostics Reference Manual*.

Initialization Procedure

Once `Initialize Diag.` is selected, the program displays a series of initialization prompts. TABLE 2-1 lists the standard responses for all initialization prompts.

TABLE 2-1 Standard Prompt Responses

Response	Description
`cr`	Press the Return key to select the default response
`?`	Display help information
`^`	Return to the interactive mode options menu

1. The program first displays the following prompt:

```
Do you need help? Enter Y or N (CR)
```

   Enter `Y` to display a help message. Enter `N` or press the Return key to bypass displaying the help information. The default response is `N`.

2. The program then displays:

```
MCA PRESENT = Y   ;[cr,?,^,Y,N]?
```

   Enter one of the following or a standard response:

Response	Description
`Y`	Specifies that a MEMORY CHANNEL IV Adapter (MCA) board is configured.
`N`	Specifies that no MCA board is configured. Tests 1 through 3 can be run without an MCA.

3. The program displays:

```
Normal or VME ONLY testing = Normal   ;[cr,?,^,N,V]?
```

Enter one of the following or a standard response:

Response	Description
N	Specifies normal testing using both the Local Bus and the VME Bus.
Y	Specifies VME ONLY testing using only the VME Bus. Under this selection, Local Bus testing is bypassed.

4. The program displays:

```
Slot Number = n ;[cr,?,^,(2-6)]?
```

Enter the slot number of the EDRAM board or enter one of the standard responses. The variable *n* represents the default slot number. Valid slot numbers are 2 through 6.

5. The program displays:

```
Select test mode=LL   ;[cr,?,^,LL,HR,NR]?
```

Enter the response associated with the desired testing mode or one of the standard responses. Valid testing modes are:

Prompt Response	Testing Mode
LL	Local Loopback (default mode)
HR	Host in Remote
NR	Node in Remote

If you selected HR or NR, go to Step 11.

6. The program displays:

```
Select beginning mbyte test area = 0 ;[cr,?,^,(0-511)]?
```

Enter the beginning megabyte area within the memory address range to be used for testing or enter one of the standard responses. Valid beginning test areas are 0 through 511.

7. The program displays:

```
WARNING If a TMI is present in the system, the last mbyte is used
by the TMI
Select ending mbyte test area = 2 ;[cr,?,^,(1-512)]?
```

Enter the ending megabyte area within the memory address range to be used for testing or enter one of the standard responses. Valid ending test areas are 1 through 512.

8. The program displays:

```
Select SCSI base address = none ;[cr,?,^,none,########]?
```

Enter none if no SCSI disk drive is configured on the VME Bus. If a SCSI disk drive is configured on the VME Bus, enter the hexadecimal base address of the controller (for example, 0xfff46800) or enter one of the standard responses. Tests 40 and 41 are bypassed if no address is specified.

If you entered a SCSI base address, the program displays:

```
Which scsi disk do you wish to use = 1 ;[cr,?,^,(0-7)]?
```

Enter the number associated with the selected disk drive or one of the standard responses.

9. The program displays:

```
TMI PRESENT = N ;[cr,?,^,N,Y]?
```

Enter one of the following or a standard response:

Response	Description
Y	Enables TMI testing.
N	Inhibits TMI testing.

If you entered N, go to Step 14.

10. If you entered Y in Step 9, the program displays:

```
Select Local or External TMI testing = Local ;[cr,?,^,L,X]?
```

Enter one of the following or a standard response:

Response	Description
L	Specifies testing the TMI in Local Loopback mode.
X	Specifies testing the TMI in External Loopback mode.

The program displays:

```
Enter Local TMI Bus and Node Id = 0x0105 ;[cr,?,^,####)]? 0x
```

Enter the physical bus and node ID for the local TMI or one of the standard responses. In this example, the bus ID is 01 and the node ID is 05.

At this time, go to Step 4.

11. If you entered HR or NR in Step 5, the program displays:

```
Logical Node ID = 0x0   [cr,?,^,(0-F)]?
```

Enter the logical ID number for this node or enter one of the standard responses. Valid numbers are 0x0 through 0xF. Assign a unique number for the host and each node. If you are configuring NR mode, go to Step 14.

12. If you entered HR in Step 5, the program displays:

```
Enter all destination node(s) to test=2;[cr,?,^,(1,2,...F)]?
```

Enter the logical node ID number for each remote node to be tested or enter one of the standard responses. Valid numbers are 0x0 through 0xF. Select any one of sixteen logical node ID numbers as long as each is unique. There is no relationship between the logical node ID and the physical node ID jumpered on the board. If you are initializing the diagnostic copy on the host, do not include the host ID number.

13. The program displays:

```
ATB PRESENT = N   ;[cr,?,^,Y,N]?
```

Enter one of the following or a standard response:

Response	Description
Y	Enables running Test 102 on the Arbitration Termination Board (ATB).
N	Disables running Test 102 on the ATB.

14. The program displays:

```
Do you wish to enable milestones? = No ;[cr,?,^,Y,N]?
```

Enter Y to enable the display of expanded milestone messages. Enter N to disable expanded milestone messages or enter one of the standard responses.

15. The program displays:

```
Do you wish to loop on subtest=No  ;[cr,?,^,Y,N]?
```

Enter Y to enable looping on a specific subtest. Enter N to disable looping on a subtest or enter one of the standard responses.

If you entered Y to enable looping on a subtest, the program displays:

```
Enter the subtest(s) you wish to loop = a ;[cr,?,^,(b c ...)]?
```

Enter the characters associated with the desired subtests or enter one of the standard responses. Valid subtests range from a through z.

At this time program initialization is complete and the program returns to the Interactive Mode Options menu.

Note – To run tests in interactive mode, use the a, e, or t option from the Interactive Mode Options Menu to select the desired tests. Then use the r option to start test execution. The interactive mode options are described in the *Sun StorEdge A7000 Diagnostics Reference Manual.*

Execution Examples

The following examples use the MEMORY CHANNEL IV System Diagnostic to test a two node system and a four node system.

Note – The diagnostic can be initialized and started on the host and nodes in any order.

In all examples, values displayed for addresses and program revisions are for example only. The actual values may be different depending on the hardware and software configurations.

If no response is shown for a prompt, press the Return key to select the default response.

TMI testing is disabled.

Two Node System

The following example uses the MEMORY CHANNEL IV System Diagnostic to test a two node system.

Initialize and Start the Node

The following sequence initializes and starts diagnostic execution on the node. For this example, the EDRAM board is in slot 3.

Start the program in interactive mode. The diagnostic displays its start message and the Interactive Mode Options menu as follows:

```
ROM >> go 0 0x3f600000
Extended Memory Channel IV Diagnostic
Diagnostic Revision: 1.13
MCA   Board CSR Addr: 0xfff40000 - 0xfff407ff
EDRAM Board CSR Addr: 0xfffb0000 - 0xfffbffff
Copyright 1998 Sun Microsystems, Inc.

Valid Options                    Current State
-------------                    -------------
a  -  Automatic testing.         (disabled)
c  -  Select reset CSR.          (0xfff40000)
d  -  Loop on diagnostic.        (1)
e  -  Extensive testing.         (enabled)
f  -  Loop on failing test.      (disabled)
h  -  Help.
i  -  Initialize Diag.
q  -  Quit.
r  -  Run.
s  -  Stop on error.             (enabled)
t  -  Select test range.         (disabled)
Enter your selection: i          (Selects the Initialize Diag. option.)
```

Select the `Initialize Diag.` option to begin the initialization process. The program displays the following initialization prompts:

```
Do you need help? Enter Y or N (CR)
MCA PRESENT = Y ;[cr,?,^,Y,N]?
Normal or VME ONLY testing = Normal  ;[cr,?,^,N,V]?
Slot Number = 3 ;[cr,?,^,(2-6)?
Select test mode=LL   ;[cr,?,^,LL,HR,NR]? nr
Logical Node ID = 0x0 ;[cr,?,^,(0-F)]? 2
TMI PRESENT = N ;[cr,?,^,Y,N]?
Do you wish to enable milestones? = No ;[cr,?,^,Y,N]?
Do you wish to loop on subtest=No ;[cr,?,^,Y,N]?
```

Notice that the testing mode selected is Node in Remote (NR).

Once initialization is complete, the program redisplays the Interactive Mode Options menu.

```
Valid Options                    Current State
-------------                    -------------
a  -  Automatic testing.         (disabled)
c  -  Select reset CSR.          (0xfff40000)
d  -  Loop on diagnostic.        (1)
e  -  Extensive testing.         (enabled)
f  -  Loop on failing test.      (disabled)
h  -  Help.
i  -  Initialize Diag.
q  -  Quit.
r  -  Run.
s  -  Stop on error.             (enabled)
t  -  Select test range.         (disabled)
Enter your selection: i          (Selects the Initialize Diag. option.)
```

Select the `Run.` option to start running Test 90 on this node. At this time, begin initializing the host.

Initialize and Start the Host

The following sequence initializes and starts diagnostic execution on the host.

Start the program in interactive mode. The diagnostic displays its start message and the Interactive Mode Options menu as follows:

```
ROM >> go 0 0x3f600000
Extended Memory Channel IV Diagnostic
Diagnostic Revision: 1.13
MCA   Board CSR Addr: 0xfff40000 - 0xfff407ff
EDRAM Board CSR Addr: 0xfffb0000 - 0xfffbffff
Copyright 1998 Sun Microsystems, Inc.

Valid Options                    Current State
-------------                    -------------
a  -  Automatic testing.         (disabled)
c  -  Select reset CSR.          (0xfff40000)
d  -  Loop on diagnostic.        (1)
e  -  Extensive testing.         (enabled)
f  -  Loop on failing test.      (disabled)
h  -  Help.
i  -  Initialize Diag.
q  -  Quit.
r  -  Run.
s  -  Stop on error.             (enabled)
t  -  Select test range.         (disabled)
Enter your selection: i          (Selects the Initialize Diag. option)
```

Select the `Initialize Diag.` option to start program initialization for the host.

The program displays the following initialization prompts:

```
Do you need help? Enter Y or N (CR)
MCA PRESENT = Y ;[cr,?,^,Y,N]?
Normal or VME ONLY testing = Normal  ;[cr,?,^,N,V]?
Slot Number = 3 ;[cr,?,^,(2-6)?
Select test mode=LL   ;[cr,?,^,LL,HR,NR]? hr
Logical Node ID = 0x0 ;[cr,?,^,(0-F)]?
Enter all destination node(s) to test=2 ;[cr,?,^,(0,1,2,...F)]?
ATB PRESENT = N ;[cr,?,^,Y,N]?
TMI PRESENT = N ;[cr,?,^,Y,N]?
Do you wish to enable milestones? = No ;[cr,?,^,Y,N]?
Do you wish to loop on subtest=No ;[cr,?,^,Y,N]?
```

Notice that the testing mode selected is Host in Remote (HR).

Once initialization is complete, the program redisplays the Interactive Mode Options menu:

```
Valid Options                Current State
-------------                -------------
a  -  Automatic testing.     (disabled)
c  -  Select reset CSR.      (0xfff40000)
d  -  Loop on diagnostic.    (1)
e  -  Extensive testing.     (enabled)
f  -  Loop on failing test.  (disabled)
h  -  Help.
i  -  Initialize Diag.
q  -  Quit.
r  -  Run.
s  -  Stop on error.         (enabled)
t  -  Select test range.     (disabled)
Enter your selection: r      (Selects the Run. option)
```

Select the `Run.` option to start running Tests 91 through 100 on the host.

Four Node System

The following example uses the MEMORY CHANNEL IV System Diagnostic to test a four node system. TMIs are required for this configuration. The host and one node are in a local system and the other two nodes are in a remote system. The TMI in the local system uses BUS ID 1 and the TMI in the remote system uses BUS ID 2.

Initialize and Start the First Node

The following sequence initializes and starts diagnostic execution on the first node.

Start the program in interactive mode. The diagnostic displays its start message and the Interactive Mode Options menu as follows:

```
ROM >> go 0 0x3f600000
Extended Memory Channel IV Diagnostic
Diagnostic Revision: 1.13
MCA   Board CSR Addr: 0xfff40000 - 0xfff407ff
EDRAM Board CSR Addr: 0xfffb0000 - 0xfffbffff
Copyright 1998 Sun Microsystems, Inc.

Valid Options                    Current State
-------------                    -------------
a  -  Automatic testing.         (disabled)
c  -  Select reset CSR.          (0xfff40000)
d  -  Loop on diagnostic.        (1)
e  -  Extensive testing.         (enabled)
f  -  Loop on failing test.      (disabled)
h  -  Help.
i  -  Initialize Diag.
q  -  Quit.
r  -  Run.
s  -  Stop on error.             (enabled)
t  -  Select test range.         (disabled)
Enter your selection: i          (Selects the Initialize Diag. option.)
```

Select the `Initialize Diag.` option to begin the initialization process. The program displays the following initialization prompts:

```
Do you need help? Enter Y or N (CR)
MCA PRESENT = Y ;[cr,?,^,Y,N]?
Normal or VME ONLY testing = Normal  ;[cr,?,^,N,V]?
Slot Number = 3 ;[cr,?,^,(2-6)?
Select test mode=LL   ;[cr,?,^,LL,HR,NR]? nr
Logical Node ID = 0x0 ;[cr,?,^,(0-F)]? 1
TMI PRESENT = N ;[cr,?,^,Y,N]? y
Enter Local TMI Bus and Node Id = 0x0105 ;[cr,?,^,####]? 0x0101
Do you wish to enable milestones? = No ;[cr,?,^,Y,N]?
Do you wish to loop on subtest=No ;[cr,?,^,Y,N]?
```

Notice that the node is initialized for `nr` mode and Node ID 1. The TMI BUS ID is 1. Once initialization is complete, the program redisplays the Interactive Mode Options menu.

```
Valid Options                    Current State
-------------                    -------------
a  -  Automatic testing.         (disabled)
c  -  Select reset CSR.          (0xfff40000)
d  -  Loop on diagnostic.        (1)
e  -  Extensive testing.         (enabled)
f  -  Loop on failing test.      (disabled)
h  -  Help.
i  -  Initialize Diag.
q  -  Quit.
r  -  Run.
s  -  Stop on error.             (enabled)
t  -  Select test range.         (disabled)
Enter your selection: i          (Selects the Initialize Diag. option.)
```

Select the `Run.` option to start running Test 90 on this node.

Initialize and Start the Second Node

The following sequence initializes and starts diagnostic execution on the second node.

Start the program in interactive mode. The diagnostic displays its start message and the Interactive Mode Options menu as follows:

```
ROM >> go 0 0x3f600000
Extended Memory Channel IV Diagnostic
Diagnostic Revision: 1.13
MCA   Board CSR Addr: 0xfff40000 - 0xfff407ff
EDRAM Board CSR Addr: 0xfffb0000 - 0xfffbffff
Copyright 1998 Sun Microsystems, Inc.

Valid Options                    Current State
-------------                    -------------
a  -  Automatic testing.         (disabled)
c  -  Select reset CSR.          (0xfff40000)
d  -  Loop on diagnostic.        (1)
e  -  Extensive testing.         (enabled)
f  -  Loop on failing test.      (disabled)
h  -  Help.
i  -  Initialize Diag.
q  -  Quit.
r  -  Run.
s  -  Stop on error.             (enabled)
t  -  Select test range.         (disabled)
Enter your selection: i          (Selects the Initialize Diag. option.)
```

Select the `Initialize Diag.` option to begin the initialization process. The program displays the following initialization prompts:

```
Do you need help? Enter Y or N (CR)
MCA PRESENT = Y ;[cr,?,^,Y,N]?
Normal or VME ONLY testing = Normal  ;[cr,?,^,N,V]?
Slot Number = 3 ;[cr,?,^,(2-6)?
Select test mode=LL   ;[cr,?,^,LL,HR,NR]? nr
Logical Node ID = 0x0 ;[cr,?,^,(0-F)]? 2
TMI PRESENT = N ;[cr,?,^,Y,N]? y
Enter Local TMI Bus and Node Id = 0x0105 ;[cr,?,^,####]? 0x0201
Do you wish to enable milestones? = No ;[cr,?,^,Y,N]?
Do you wish to loop on subtest=No ;[cr,?,^,Y,N]?
```

Notice that the node is initialized in `nr` mode and the Node ID is 2. The TMI BUS ID is 2. This node is in the remote system. Once initialization is complete, the program redisplays the Interactive Mode Options menu.

```
Valid Options                    Current State
-------------                    -------------
a  -  Automatic testing.         (disabled)
c  -  Select reset CSR.          (0xfff40000)
d  -  Loop on diagnostic.        (1)
e  -  Extensive testing.         (enabled)
f  -  Loop on failing test.      (disabled)
h  -  Help.
i  -  Initialize Diag.
q  -  Quit.
r  -  Run.
s  -  Stop on error.             (enabled)
t  -  Select test range.         (disabled)
Enter your selection: i          (Selects the Initialize Diag. option.)
```

Select the `Run.` option to start running Test 90 on this node.

Initialize and Start the Third Node

The following sequence initializes and starts diagnostic execution on the third node.

Start the program in interactive mode. The diagnostic displays its start message and the Interactive Mode Options menu as follows:

```
ROM >> go 0 0x3f600000
Extended Memory Channel IV Diagnostic
Diagnostic Revision: 1.13
MCA   Board CSR Addr: 0xfff40000 - 0xfff407ff
EDRAM Board CSR Addr: 0xfffb0000 - 0xfffbffff
Copyright 1998 Sun Microsystems, Inc.

Valid Options                    Current State
-------------                    -------------
a  -  Automatic testing.         (disabled)
c  -  Select reset CSR.          (0xfff40000)
d  -  Loop on diagnostic.        (1)
e  -  Extensive testing.         (enabled)
f  -  Loop on failing test.      (disabled)
h  -  Help.
i  -  Initialize Diag.
q  -  Quit.
r  -  Run.
s  -  Stop on error.             (enabled)
t  -  Select test range.         (disabled)
Enter your selection: i          (Selects the Initialize Diag. option.)
```

Select the `Initialize Diag.` option to begin the initialization process. The program displays the following initialization prompts:

```
Do you need help? Enter Y or N (CR)
MCA PRESENT = Y ;[cr,?,^,Y,N]?
Normal or VME ONLY testing = Normal  ;[cr,?,^,N,V]?
Slot Number = 3 ;[cr,?,^,(2-6)?
Select test mode=LL   ;[cr,?,^,LL,HR,NR]? nr
Logical Node ID = 0x0 ;[cr,?,^,(0-F)]? 3
TMI PRESENT = N ;[cr,?,^,Y,N]? y
Enter Local TMI Bus and Node Id = 0x0105 ;[cr,?,^,####]? 0x0201
Do you wish to enable milestones? = No ;[cr,?,^,Y,N]?
Do you wish to loop on subtest=No ;[cr,?,^,Y,N]?
```

Notice that the node is initialized in `nr` mode and the Node ID is 3. The TMI BUS ID is 2. This node is in the remote system. Once initialization is complete, the program redisplays the Interactive Mode Options menu.

```
Valid Options                    Current State
-------------                    -------------
a  -  Automatic testing.         (disabled)
c  -  Select reset CSR.          (0xfff40000)
d  -  Loop on diagnostic.        (1)
e  -  Extensive testing.         (enabled)
f  -  Loop on failing test.      (disabled)
h  -  Help.
i  -  Initialize Diag.
q  -  Quit.
r  -  Run.
s  -  Stop on error.             (enabled)
t  -  Select test range.         (disabled)
Enter your selection: i          (Selects the Initialize Diag. option.)
```

Select the `Run.` option to start running Test 90 on this node.

Initialize and Start the Host

The following sequence initializes and starts diagnostic execution on the host.

Start the program in interactive mode. The diagnostic displays its start message and the Interactive Mode Options menu as follows:

```
ROM >> go 0 0x3f600000
Extended Memory Channel IV Diagnostic
Diagnostic Revision: 1.13
MCA   Board CSR Addr: 0xfff40000 - 0xfff407ff
EDRAM Board CSR Addr: 0xfffb0000 - 0xfffbffff
Copyright 1998 Sun Microsystems, Inc.

Valid Options                    Current State
-------------                    -------------
a  -  Automatic testing.         (disabled)
c  -  Select reset CSR.          (0xfff40000)
d  -  Loop on diagnostic.        (1)
e  -  Extensive testing.         (enabled)
f  -  Loop on failing test.      (disabled)
h  -  Help.
i  -  Initialize Diag.
q  -  Quit.
r  -  Run.
s  -  Stop on error.             (enabled)
t  -  Select test range.         (disabled)
Enter your selection: i          (Selects the Initialize Diag. option)
```

Select the `Initialize Diag.` option to start program initialization for the host.

The program displays the following initialization prompts:

```
Do you need help? Enter Y or N (CR)
MCA PRESENT = Y ;[cr,?,^,Y,N]?
Normal or VME ONLY testing = Normal  ;[cr,?,^,N,V]?
Slot Number = 3 ;[cr,?,^,(2-6)?
Select test mode=LL   ;[cr,?,^,LL,HR,NR]? hr
Logical Node ID = 0x0 ;[cr,?,^,(0-F)]?
Enter all destination node(s) to test=2
;[cr,?,^,(0,1,2,...F)]? 1,2,3
ATB PRESENT = N ;[cr,?,^,Y,N]?
TMI PRESENT = N ;[cr,?,^,Y,N]? y
Enter Local TMI Bus and Node Id = 0x0105 ;[cr,?,^,####]? 0x0101
Do you wish to enable milestones? = No ;[cr,?,^,Y,N]?
Do you wish to loop on subtest=No ;[cr,?,^,Y,N]?
```

Notice that the testing mode selected is Host in Remote (hr) and the destination nodes are 1, 2, and 3. The TMI BUS ID is 1. The host is in the local system.

Once initialization is complete, the program redisplays the Interactive Mode Options menu:

```
Valid Options                Current State
-------------                -------------
a  -  Automatic testing.     (disabled)
c  -  Select reset CSR.      (0xfff40000)
d  -  Loop on diagnostic.    (1)
e  -  Extensive testing.     (enabled)
f  -  Loop on failing test.  (disabled)
h  -  Help.
i  -  Initialize Diag.
q  -  Quit.
r  -  Run.
s  -  Stop on error.         (enabled)
t  -  Select test range.     (disabled)
Enter your selection: r      (Selects the Run. option)
```

Select the Run. option to start running Tests 91 through 100 on the host.

CHAPTER 3

Program Tests

Introduction

The MEMORY CHANNEL IV System Diagnostic contains 56 tests used to verify the functionality of single and multiple MEMORY CHANNEL IV modules. Each MEMORY CHANNEL IV module consists of two boards: an ECC Memory board (EDRAM) of 128MB, 256MB or 512MB that plugs directly into the Local Bus and a MEMORY CHANNEL IV Adapter (MCA) board that provides the connection to the MEMORY CHANNEL Bus (MCbus). The MCA board is mounted as a Mezzanine board on the EDRAM board.

Tests 1 through 43 are used for local testing, Test 90 is used for remote testing in Node in Remote (NR) mode, and Tests 91 through 102 are used for remote testing in Host in Remote (HR) mode. The testing modes are described in Chapter 1.

This diagnostic also contains tests for the Transition Module Interface (TMI) board. Tests 44, 46 through 47 and 53 through 79 are used to test the TMI in Local Loopback mode. The TMI testing modes are described in Chapter 1.

Test Summary

The MEMORY CHANNEL IV System Diagnostic tests are listed in TABLE 3-1.

TABLE 3-1 MEMORY CHANNEL IV System Diagnostic Tests

Test Number	Test Name
1	EDRAM Register Map
2	EDRAM Memory March on Private Memory
3	EDRAM Data Address on Private Memory
4	MCA Register Map
5	WSC - Transmit FIFO & MCbus Output Transfer Control (VMEbus)
6	WSC - Single Data Transfer Verification (VMEbus)
7	WSC - Stuck and Tied Address Lines
8	WSC - Data Pattern - All Ones Data
9	WSC - Data Pattern - All Zeros Data
10	WSC - Data Pattern - All 5's Data
11	WSC - Data Pattern - All A's Data
12	WSC - Data Pattern - Address in Data
13	WSC - Data Pattern - Alternating Data
14	WSC - Data Pattern - Random Data
15	WSC - TX/RX Windows Open/Close Verification (VMEBUS)
16	WSC - Longword Transfer using I/O buffer size equal to TA size
17	WSC - Word Transfer using I/O buffer size equal to TA size
18	WSC - Byte Transfer using I/O buffer size equal to TA size
19	WSC - Burst Contention LWord Address Xfrs & MCbus Busy (VMEBUS)
20	WSC - Burst Contention Mix Address Xfrs & MCbus Busy (VMEBUS)
21	WSC - Reflecting From Multiple CPUs each using Address in Data
22	DMA - Stuck and Tied Data Lines Transfer (VMEbus)
23	DMA - Stuck and Tied Address Lines Single Transfer (VMEbus)
24	DMA - Variable Transfer Count (VMEBUS)
25	DMA - All 5's Data (VMEBUS)

TABLE 3-1 MEMORY CHANNEL IV System Diagnostic Tests *(Continued)*

Test Number	Test Name
26	DMA - All A's Data (VMEBUS)
27	DMA - Alternating Data (VMEBUS)
28	DMA - Random Data & 64b ECC Sequenced (VMEBUS)
29	DMA - DMA Throttled by TX & RX FIFOs (VMEbus)
30	DMA - Source & Destination Address Cross 4K Boundary (VMEBUS)
31	DMA - Command Queuing (VMEBUS)
32	DMA - Command Queuing Address Cross 4K Boundary (VMEBUS)
33	DMA - Receiver Closes Odd Page Window Addresses (VMEbus)
34	DMA - Access EDRAM Memory while DMA Transfer is in Progress
35	DMA - Access Window RAMs while DMA Transfer is in Prog. (VMEbus)
36	WSC/DMA - Alternating WSC & DMA Subblock Transfers (VMEBUS)
37	WSC/DMA - Byte Transfer Concurrent with DMA Transfer
38	WSC/DMA - Longword Transfer Concurrent with DMA Transfer
39	WSC/DMA - WSC Transfer while Command Queuing (VMEBUS)
40	EDRAM VME32 Block Transfer Mode (SCSI & MC4)
41	EDRAM VME D64 Block Transfer Mode (SCSI & MC4)
42	DMA - VME Interrupts
43	DMA - Force ECC & Framing Errors (VMEBUS)
44	TMI Local Loopback... Control/Status Register
46	TMI Local Loopback... Programming to Operational Mode
47	TMI Local Loopback... Data Transfer
52	TMI/WSC - Single Data Transfer Verification (VMEbus)
53	TMI/WSC - Data Pattern - All Ones Data
54	TMI/WSC - Data Pattern - All Zeros Data
55	TMI/WSC - Data Pattern - All 5's Data
56	TMI/WSC - Data Pattern - All A's Data
57	TMI/WSC - Data Pattern - Address in Data
58	TMI/WSC - Data Pattern - Alternating Data
59	TMI/WSC - Data Pattern - Random Data
60	TMI/WSC - TX/RX Windows Open/Close Verification (VMEBUS)

TABLE 3-1 MEMORY CHANNEL IV System Diagnostic Tests *(Continued)*

Test Number	Test Name
61	TMI/WSC - Longword Transfer using I/O buffer size equal to TA size
62	TMI/WSC - Word Transfer using I/O buffer size equal to TA size
63	TMI/WSC - Byte Transfer using I/O buffer size equal to TA size
64	TMI/WSC - Reflecting from Multiple CPUs each using Address in Data
65	TMI/DMA - Stuck and Tied Data Lines Transfer (VMEbus)
66	TMI/DMA - Stuck and Tied Address Lines Single Transfer (VMEbus)
67	TMI/DMA - Variable Transfer Count (VMEBUS)
68	TMI/DMA - All 5's Data (VMEBUS)
69	TMI/DMA - All A's Data (VMEBUS)
70	TMI/DMA - Alternating Data (VMEBUS)
71	TMI/DMA - Random Data & 64b ECC Sequenced (VMEBUS)
72	TMI/DMA - DMA Throttled by TX & RX FIFOs (VMEbus)
73	TMI/DMA - Source & Destination Address Cross 4K Boundary (VMEBUS)
74	TMI/DMA - Receiver Closes Odd Page Window Addresses (VMEbus)
75	TMI/DMA - Access EDRAM Memory while DMA Transfer is in Progress
76	TMI/DMA - Access Window RAMs while DMA Transfer is in Prog. (VMEbus)
77	TMI/WSC/DMA - Alternating WSC & DMA Subblock Transfers (VMEBUS)
78	TMI/WSC/DMA - Byte Transfer Concurrent with DMA Transfer
79	TMI/WSC/DMA - Longword Transfer Concurrent with DMA Transfer
90	NR Node Test
91	HR WSC - Data Pattern - Address in Data
92	HR WSC - Data Pattern - Alternating Data
93	HR WSC - VL Transfers (Bus Valid Inhibit & Local Semaphore)
94	HR DMA - Stuck and Tied Data Lines Transfer
95	HR DMA - Data Pattern - Address in Data
96	HR DMA - Variable Transfer Count
97	HR DMA - VL Transfers (Bus Valid Inhibit & Local Semaphore)
98	HR WSC/DMA - Alternating WSC & DMA Subblock Transfers (VMEbus)
99	HR WSC/DMA - Byte Transfer Concurrent with DMA Transfer

TABLE 3-1 MEMORY CHANNEL IV System Diagnostic Tests *(Continued)*

Test Number	Test Name
100	HR WSC/DMA - Longword Transfer Concurrent with DMA Transfer
101	HR DMA - Data Pattern - 0x7E7E7E7E
102	HR DMA - ATB Test

Test Descriptions

The following sections describe each of the tests in the MEMORY CHANNEL IV System Diagnostic. The tests for MEMORY CHANNEL IV modules are grouped into the following categories:

Test Number	Test Type
1-3	EDRAM board functionality
4	MCA board Control/Status Registers (CSRs)
5-21	REFLECTIVE MEMORY System (RMS) functionality
22-35	MEMORY CHANNEL DMA transfer functionality
36-39, 42-43	WSC and MEMORY CHANNEL DMA transfer functionality
40-41	SCSI Block Transfer modes
90	Node in Remote mode
91-93	WSC transfer functionality in Host in Remote mode
94-97, 101	MEMORY CHANNEL DMA transfer functionality in Host in Remote mode
98-100	WSC and MEMORY CHANNEL DMA transfer functionality in Host in Remote mode
102	ATB board

The tests for Transition Module Interface (TMI) boards are grouped into the following categories:

Test Number	Test Type
44, 46-47	TMI board functionality in Local Loopback mode
52-64	REFLECTIVE MEMORY System (RMS) functionality
65-76	MEMORY CHANNEL DMA transfer functionality
77-79	WSC and MEMORY CHANNEL DMA transfer functionality

Note – Only Tests 1 through 3 and Subtests a through i in Test 4 are run during the power up testing sequence or in automatic mode. All tests can be run in interactive mode during extensive testing.

When writing an address-in-address data pattern, the test writes the address of a memory location as data into the same location.

Test 1 EDRAM Register Map

This test verifies the ability to read and write the Control and Status Registers (CSRs) on the Expansion DRAM (EDRAM) board. The test executes the following subtests:

Subtest	Subtest Name
a	Board ID Register
b	Memory Size Register
c	Local Bus Base Address Register
d	VMEbus Base Address Register

Subtest a - Board ID Register

This subtest reads the contents of the Board ID Register at address offsets 0x000 through 0x7fc and verifies that the four ATE Pattern parameters are programmed correctly. This ensures that the PROM contains data and is not blank. The subtest then reads all PROM locations and calculates a checksum value. The calculated checksum is compared to the checksum located in the last four PROM locations (0x7f0 through 0x7fc). This subtest is run only when the EDRAM is configured on the Local Bus.

Subtest b - Memory Size Register

This subtest reads the Memory Size Register and verifies that the memory size is 0x080 (128MB), 0x100 (256MB), or 0x300 (512MB).

Subtest c - Local Bus Base Address Register

This subtest writes, reads, and verifies the contents of the Local Bus Base Address Register. This register contains the starting address of the on-board memory in the Local Bus address space.

Note – This subtest is bypassed during power up testing and in automatic mode. The subtest is also bypassed if you selected the VME ONLY bus configuration during program initialization.

Subtest d - VMEbus Base Address Register

This subtest writes, reads, and verifies the contents of the VMEbus Base Address Register. This register contains the starting address of the on-board memory in the VMEbus address space.

Test 2 EDRAM Memory March on Private Memory

This test verifies global memory on the EDRAM board. The test writes data patterns through a range of memory. The data is then read and verified one location at a time from the ending address to the starting address. As each location is verified, the data pattern is complemented and rewritten. The memory is then read from the starting address to the ending address checking for the complemented pattern. Memory access is checked from both the VMEbus and the Local Bus. The test checks the status and verifies that no errors occurred. This test contains the following subtests:

Subtest	Subtest Name
a	All Fives Data Complemented (VMEbus)
b	All Zeros Data Complemented (VMEbus)
c	All Fives Data Complemented (Local Bus)
d	All Zeros Data Complemented (Local Bus)

Note – Subtests c and d are bypassed if you selected the VME ONLY bus configuration during program initialization.

Test 3 EDRAM Data Address on Private Memory

This test verifies global memory on the EDRAM board. The test writes an address-in-address data pattern into each memory location in the test range. The data is written from the starting to the ending address. The test reads and verifies each location from the starting to the ending address to ensure that patterns at high addresses do not corrupt lower memory locations. After data transfer testing is completed, the test checks the status and verifies that no errors occurred. Memory access is checked from either the VME Bus or the Local Bus. The testing procedure is performed for each of the following subtests:

Subtest	Subtest Name
a	Data Address (VMEbus)
b	Data Address (Local Bus)

Note – Subtest b is bypassed if you selected the VME ONLY bus configuration during program initialization.

Test 4 MCA Register Map

This test verifies the ability to read and write the Control and Status Registers (CSRs) on the MEMORY CHANNEL IV Adapter (MCA) board. The test executes the following subtests:

Subtest	Subtest Name
a	Board ID Register
b	VIC - VME Interrupt Control Register
c	MC.CTL - Control Register
d	TWRA - TX Window RAM Address Register
e	TX Window RAM Data Register TWRD & RAM
f	RWSRA - RX Window Segment RAM Address Register
g	RWPRA - RX Window Page RAM Address Register
h	RX Window RAM Data Register RWRD & Page RAM
i	RX Window RAM Data Register RWRD & Segment RAM

Subtest	Subtest Name
j	ONLINE / OFFLINE Register
k	Command FIFO and DMA w/o data transfer
l	Read SENSE & display Node ID & Subblock size

Note – Only Subtests a through i are run during the power up testing sequence or in automatic mode.

Subtest a - Board ID Register

This subtest reads the contents of the Board ID Register at address offsets 0x000 through 0x07c and verifies that the four ATE Pattern parameters are programmed correctly. This ensures that the PROM contains data and is not blank. The subtest then reads all PROM locations and calculates a checksum value. The calculated checksum is compared to the checksum located in the last four PROM locations (0x070 through 0x07c).

Subtest b - VIC - VME Interrupt Control Register

This subtest verifies the ability to set and reset each interrupt control bit in the VMEbus Interrupt Control Register. The following sequence of data patterns is used for testing: all ones, all fives, all A's, all zeros.

Subtest c - MC.CTL - Control Register

This subtest verifies the ability to set and reset each control bit in the MEMORY CHANNEL Control Register CSR. After the bits are checked, the CSR is loaded with all ones and a SOFT.CLEAR function is performed to reset the board and clear the register. The following sequence of data patterns is used for testing: all ones, all fives, all A's, all zeros.

Subtest d - TWRA - TX Window RAM Address Register

This subtest verifies the ability to set and reset the Window Address, and P and U bits in the TX Window RAM Address Register (TWRA). The following sequence of data patterns is used for testing: all ones, all fives, all A's, all zeros.

Subtest e - TX Window RAM Data Register TWRD & RAM

This subtest verifies the ability to set and reset the Transmit Translation Address, and the V, L, R, and Window Open and Close bits in the Transmit Window RAM Data Register (TWRD) and the RAM. The subtest exercises the entire address range of the TX RAM. All locations are first written, and then read back and verified. The following sequence of data patterns is used for testing: RAM address, all ones, all fives, all A's, all zeros.

Subtest f - RWSRA - RX Window Segment RAM Address Register

This subtest verifies the ability to set and reset the Window Segment RAM Address bits in the RX Window Segment RAM Address Register (RWSRA). This register contains a pointer to the Receive Windows Segment RAM. The following sequence of data patterns is used for testing: all ones, all fives, all A's, all zeros.

Subtest g - RWPRA - RX Window Page RAM Address Register

This subtest verifies the ability to set and reset the Window Page RAM Address bits in the RX Window Page RAM Address Register (RWPRA). This register contains a pointer to the Receive Windows Page RAM. The following sequence of data patterns is used for testing: all ones, all fives, all A's, all zeros.

Subtest h - RX Window RAM Data Register RWRD & Page RAM

This subtest verifies the ability to set and reset bits in the Receive Window RAM Data Register (RWRD) and Page RAM. The subtest exercises the entire address range of Page RAM while keeping the Segment RAM address at 0. Segment RAM location 0 is accessed for the Segment Open and Closed bit. A Segment Open or Segment Closed bit error indicates either a faulty Segment RAM or RWRD data bit. All locations are first written, and then read back for accuracy. The following sequence of data patterns is used for testing: all ones, all fives, all A's, all zeros.

Subtest i - RX Window RAM Data Register RWRD & Segment RAM

This subtest verifies the ability to set and reset bits in the Receive Window RAM Data Register (RWRD) and Segment RAM. The subtest exercises the entire address range of the Segment RAM while keeping the Page RAM address at 0. Page RAM location 0 is accessed for the Page Open and Closed bit and Translation Address field. A Page Open or Page Closed bit, Mailbox bit, or Translation Address bit error could be caused by faulty data in the Receive Window RAM Data Register or Page RAM. All locations are first written, and then read back for accuracy.

The data patterns used for testing yield an alternating bit pattern between adjacent addresses as follows:

- All even Segment RAM addresses have data = 0x0
- All odd Segment RAM addresses have data = 0x20000000 (REC_SEG_OPEN)
- All even Segment RAM addresses have data = 0x20000000 (REC_SEG_OPEN)
- All odd Segment RAM addresses have data = 0x0

Subtest j - ONLINE/OFFLINE Register

This subtest verifies the ability to place the MCA board Online and Offline. The subtest verifies that the SENSE CSR provides Online status and that no errors are detected in the Interrupt Status (INT.STATUS) CSR. This subtest performs the following operations:

- Resets the MCA board.
- Enables Local Loopback mode.
- Closes all RX and TX Windows.
- Turns the MCA board Online.
- Checks for errors.
- Turns the MCA board Offline.
- Checks for errors.

Subtest k - Command FIFO and DMA w/o data transfer

This subtest verifies the ability to load commands into the Command FIFO CSRs and verifies that the DMA engine (MCE) unloads FIFO and arms the Source Address Counter, Byte Counter, and Destination Address Counter. A byte count of 0 is used to prevent a data transfer request from the EDRAM mezzanine port. The subtest verifies the DMACI field and DMA.CMPLT status in the Interrupt Status CSR and the DMA.DONE and DMA.Q.DONE flags. The Source Address Counter is verified with the following sequence of address patterns: all ones, all fives, all A's, all zeros. The subtest also checks for a command sequence error. This subtest performs the following operations:

- Resets the MCA board.
- Enables Local Loopback mode.
- Closes all RX and TX Windows.
- Turns the MCA board online.
- Issues a series of commands and verifies the status and source address after each command is executed.
- Forces a command sequence error and checks for the correct error status.
- Issues another command and verifies that it is not executed because TX is halted.

Subtest l - Read SENSE & display Node ID & Subblock size

This subtest reads the Sense CSR and displays the Node ID and Subblock size in the following format:

```
NID = 0xn, SBZ = bb
```

Variable	Description
n	Hexadecimal Node ID value from 0 through 9.
bb	Decimal subblock size: 8, 16, or 32.

Test 5 WSC - Transmit FIFO & MCbus Output Transfer Control (VMEbus)

This test verifies the ability of the MEMORY CHANNEL Adapter to sense memory writes (snoop) and conditionally accept them based on the Transmit Window RAM Open bit. The test also verifies that snooping is inhibited if the MEMORY CHANNEL Adapter is stalled or offline. The first part of the test verifies that the snoop control logic allows data into the transmitter FIFO. When a stalled condition occurs, one additional transfer is written to EDRAM and held in the write post buffer instead of memory. The second part of the test verifies that the MCbus control logic can unload the FIFO and request and transfer data onto the MCbus. The data is only transmitted and is not looped back to the receiver; however, the EDRAM output buffer memory is checked for the correct data.

Note – The MCbus Multiple Grant Request bit (MCMULT) in the MC Control CSR is exercised by the MCbus Interface when the TX FIFO is half full.

The test uses an address-in-address data pattern.

The test writes one transfer to the first location in the memory test area with the TX Window closed. It then opens the TX Window and writes 448 or 449 additional transfers across the VMEbus in ascending order. The number of transfers is determined by the board type. These writes are performed while the TX FIFO is disabled to prohibit FIFO output. One additional write is issued to fill the TX FIFO and cause the Stall signal to be sent from the MEMORY CHANNEL Adapter to the EDRAM. The STALL LED will light for a short period of time. One more memory write is issued to hold the data in the EDRAM write post buffer. The TX FIFO is then enabled to permit data to flow from the FIFO to the MCbus interface. The data is only transmitted and is not looped back to the receiver; however, the EDRAM output buffer memory is checked for the correct data.

This test performs the following operations:

- Resets the MCA board.
- Enables Local Loopback mode.
- Disables the TX FIFO output.
- Turns the MCA board online.

The test verifies that setting the TX Window Close bit inhibits data into the TX FIFO with the following operations:

- Closes all TX and RX Windows.
- Writes to the first location in the memory test area.
- Reads and verifies the Interrupt Status and Sense CSRs.

The test verifies the TX Window Open bit and the ability to fill the TX FIFO with the following operations:

- Opens the TX Window for the first window in the memory test area.
- Writes to the next 448 or 449 locations in the test area.
- Reads and verifies the Interrupt Status and Sense CSRs.
- Writes to the 449th or 450th location in the test area.
- Reads and verifies the Interrupt Status and Sense CSRs.
- Writes one transfer into the next location in the test area to cause the EDRAM to queue the transfer in its write post buffer.
- Reads and verifies the Interrupt Status and Sense CSRs.

The test verifies, with the following operations, that turning the MCA offline disables the Snoop data:

- Turns the MCA board offline.
- Writes to the next 448 or 449 locations in the test area.
- Reads and verifies the Interrupt Status and Sense CSRs.

Finally, the test uses the following operations to verify the ability of the MCbus control logic to perform an output data transfer.

- Turns the MCA board online.
- Enables Local Loopback mode.
- Enables the TX FIFO output.
- Verifies that the output transfer was successful and the Interrupt Status and Sense CSRs contain the correct status.

Test 6 WSC - Single Data Transfer Verification (VMEbus)

This test verifies the ability of the MEMORY CHANNEL Adapter to transmit and receive a single data transfer. The test verifies the basic transfer control hardware in Local Loopback mode. This test contains the following subtests:

Subtest	Subtest Name
a	Long Transfer (VMEbus)
b	Byte Transfer (VMEbus)

Subtest a uses a longword data transfer to verify basic transfer control. Subtest b uses a byte transfer to verify the MEMORY CHANNEL Adapter receiver can perform a basic read-modify-write operation to EDRAM. Both subtests use a binary progression data pattern. Each subtest performs the following operations:

- Resets the MCA board.
- Reads and verifies the Interrupt Status CSR.

- Enables Local Loopback mode.
- Opens the RX and TX Windows for the first location in the memory test area.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Transfers the data and waits for reflection to complete.
- Checks the Interrupt Status and Sense CSRs for the expected status.
- Verifies the contents of the input and output data buffers.

Test 7 WSC - Stuck and Tied Address Lines

This test verifies the address line path from the EDRAM Local Bus and VMEbus ports through the MCA transmitter, MCbus interface and receiver, and back to the EDRAM receiver path used for writing data into the input buffer. Four different longword addresses are used to verify that no address lines are stuck or tied. The test also verifies the ability to access the first and last memory locations and to set and reset the V, L, and R bits and the bit fields within the Receive Source ID Error (RSIDE) and Receive Translated Address Error (RTAE) CSRs. This test contains the following subtests:

Subtest	Subtest Name
a	Ones & Zeros Address (LBUS)
b	Zeros & Ones Address (LBUS)
c	5's & A's Address (LBUS)
d	A's & 5's Address (LBUS)
e	Ones & Zeros Address (VMEbus)
f	Zeros & Ones Address (VMEbus)
g	5's & A's Address (VMEbus)
h	A's & 5's Address (VMEbus)

Each subtest uses two address patterns so the input and output buffers have unique addresses. The transmitter's output buffer address is translated by the Window RAM into the one's complement address which becomes the input buffer address.

The following address assignments are used for the subtests:

Subtest	ED_phy_adr (outbuf 28:3)	[MCbus Adr] TX_trans_adr (39:12) (11:3)	C	RX_trans_adr (28:12) (11:3)	ED_phy_adr (inbuf 28:3)
a and e	0x1ffffff8	0x00000000 ff8	8	0x00000 ff8	0x00000ff8
b and f	0x00000000	0xffffffff 000	f	0x1ffff 000	0x1ffff000
c and g	0x15555550	0xaaaaaaaa 550	a	0x0aaaa 550	0x0aaaa550
d and h	0x0aaaaaa8	0x55555555 aa8	d	0x15555 aa8	0x15555aa8

ED_phy_adr	Specifies the EDRAM physical address seen by the MEMORY CHANNEL Adapter (bits 28:3 from the input or output buffer).
TX_trans_adr	Specifies the translated address from the TX Window RAM (bits 39:12, 11:3).
RX_trans_adr	Specifies the translated address from the RX Window RAM (bits 28:12, 11:3).
C	Specifies control bit fields 3-0 of the TX Window RAM Data Register (Open, V, L, R bits).

Once started, the test performs the following operations:

- Resets the MCA board.
- Verifies that the MCA reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate TX and RX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that no unexpected status is present.

Each subtest performs the following operations:

- Writes data to the subtest-specific longword addresses.
- Waits for reflection to complete.
- Checks the Interrupt Status and Sense CSRs for the expected status.
- Checks the RSIDE CSR and verifies that the node ID and subblock ID are zero.
- Checks the RTAE CSR and verifies that the EDRAM physical address for the input buffer is correct.
- Verifies the contents of the input and output data buffers.

Tests 8-14 WSC - Data Pattern Tests

These tests verify the ability to transmit and receive various data patterns. Each test contains two subtests. The tests and subtests use different bus access types and data patterns. The following subtests are available:

Test	Subtest	Subtest Name
8	a	All Ones Data (LBUS)
	b	All Ones Data (VMEbus)
9	a	All Zeros Data (LBUS)
	b	All Zeros Data (VMEbus)
10	a	All 5's Data (LBUS)
	b	All 5's Data (VMEbus)
11	a	All A's Data (LBUS)
	b	All A's Data (VMEbus)
12	a	Address in Data (LBUS)
	b	Address in Data (VMEbus)
13	a	Alternating Data (LBUS)
	b	Alternating Data (VMEbus)
14	a	Random Data (LBUS)
	b	Random Data (VMEbus)

Note – Subtest a is bypassed if you selected the VME ONLY bus configuration during program initialization.

For Test 12, the address used as data is the input buffer address.

For Test 13, alternating longwords of 0x00000000 and 0xfefefefe are used as data.

Each test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens TX and RX Windows for the test area.
- Turns the MCA board online.
- Initializes the input and output data buffers.

Each subtest performs the following operations:

- Writes the data pattern to the output buffer.
- Waits for the data transfer to complete.
- Checks the Interrupt Status and Sense CSRs for the expected status.
- Verifies the contents of the input and output data buffers.

Test 15 WSC - TX/RX Windows Open/Close Verification (VMEBUS)

This test verifies that the Transmit and Receive Windows open and close properly. Windows 1 through 16 of the Translation Address are used during testing. This test contains the following subtests:

Subtest	Subtest Name
a	TX Open, RX Seg Wind 0 Open, Page Winds Open (VMEbus)
b	TX Close, RX Seg Wind 0 Open, Page Winds Open (VMEbus)
c	TX Open, RX Seg Wind 0 Open, Page Winds Close (VMEbus)
d	TX Open, RX Seg Wind 0 Close, Page Winds Open (VMEbus)

Each subtest writes one longword into the first location of each window for a total of 16 writes. Once started, the test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Turns the MCA board online.
- Verifies that the expected status is present.
- Initializes the input and output data buffers.

Subtest a

This subtest performs the following operations:

- Opens the first 16 TX Windows.
- Opens the first 16 RX Windows and RX Segment Window 0.
- Verifies that 16 data longwords were transferred to the input buffer.

Subtest b

This subtest performs the following operations:

- Closes the first 16 TX Windows.
- Verifies that no data was transferred to the input buffer.

Subtest c

This subtest performs the following operations:

- Opens the first 16 TX Windows.
- Closes the RX Page Windows.
- Verifies that no data was transferred to the input buffer.

Subtest d

This subtest performs the following operations:

- Closes RX Segment Window 0.
- Opens the RX Page Windows.
- Verifies that no data was transferred to the input buffer.

Test 16 WSC - Longword Transfer using I/O buffer size equal to TA size

This test verifies that a very large data buffer can be reflected using longword addressing across the Local Bus and VMEbus. The I/O buffer size is determined by the test area (TA) size. The test contains the following subtests:

Subtest	Subtest Name
a	WSC - Longword Transfer (LBUS)
b	WSC - Longword Transfer (VMEbus)

Note – Subtest a is bypassed if you selected the VME ONLY bus configuration during program initialization.

Each subtest uses a byte binary progression data pattern and performs the following operations:

- Resets the MCA board.
- Verifies that the MCA board reset correctly.

- Enables Local Loopback mode.
- Opens the appropriate TX and RX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Waits for the data transfer to complete.
- Checks the Interrupt Status and Sense CSRs for the expected status.
- Verifies the contents of the input and output data buffers.

Test 17 WSC - Word Transfer using I/O buffer size equal to TA size

This test verifies that a very large data buffer can be reflected using word addressing across the Local Bus and VMEbus. The I/O buffer size is determined by the test area (TA) size. The test contains the following subtests:

Subtest	Subtest Name
a	WSC - Word Transfer (LBUS)
b	WSC - Word Transfer (VMEbus)

Note – Subtest a is bypassed if you selected the VME ONLY bus configuration during program initialization.

Each subtest uses a byte binary progression data pattern and performs the following operations:

- Resets the MCA board.
- Verifies that the MCA board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate TX and RX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Waits for the data transfer to complete.
- Checks the Interrupt Status and Sense CSRs for the expected status.
- Verifies the contents of the input and output data buffers.

Test 18 WSC - Byte Transfer using I/O buffer size equal to TA size

This test verifies that a very large data buffer can be reflected using byte addressing across the Local Bus and VMEbus. The I/O buffer size is determined by the test area (TA) size. The test contains the following subtests:

Subtest	Subtest Name
a	WSC - Byte Transfer (LBUS)
b	WSC - Byte Transfer (VMEbus)

Note – Subtest a is bypassed if you selected the VME ONLY bus configuration during program initialization.

Each subtest uses a byte binary progression data pattern and performs the following operations:

- Resets the MCA board.
- Verifies that the MCA board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate TX and RX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Waits for the data transfer to complete.
- Checks the Interrupt Status and Sense CSRs for the expected status.
- Verifies the contents of the input and output data buffers.

Test 19 WSC - Burst Contention LWord Address Xfrs & MCbus Busy (VMEBUS)

This test causes concurrent activities within the transmitter and receiver hardware. It also causes contention between the MEMORY CHANNEL Adapter Mezzanine RX port and the EDRAM VME port used to request EDRAM memory arbitration.

The test causes the TX FIFO to fill to its maximum level, just prior to Stall, and then enables the FIFO output to provide a steady data burst through the MCbus TX and RX logic into the RX Buffer FIFO and Receive FIFO. The RX Receive FIFO is disabled from writing data to EDRAM and forces a Global Busy or MCbus Busy condition once its maximum level is reached. The test continues to fill the output buffer to force an MCbus Busy condition and then reads the Sense CSR to verify that Global Busy and Node Busy status is present. The RX Receive FIFO is then enabled to allow the MCA to write data to EDRAM. The input buffer is checked to verify that all data was transferred.

The test performs longword data transfers across the VMEbus.

Note – The MCbus Multiple Grant Request bit (MCMULT) in the MC Control CSR is exercised by the MCbus Interface when the TX FIFO is half full.

The test uses a byte binary progression data pattern.

This test is bypassed if you disabled running the MCbus Busy Tests during program initialization.

This test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the TX and RX Windows for the memory test area.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Disables the TX and RX FIFOs.
- Writes to the first 450 locations of the test area to force a FIFO Full condition.
- Enables the TX FIFO and disables the RX FIFO.
- Writes to the next 230 locations in the test area to force a Global Busy condition.
- Verifies that the Sense CSR indicates that the MCA is online and Global Busy and Node Busy are both present.
- Writes to the next 440 locations in the test area to fill the TX FIFO and create a maximum burst time period.
- Enables the RX FIFO to start writing data to EDRAM.
- Writes to the next 928 locations in the test area to complete an 8KB data buffer.
- Waits for the data transfers to complete.

- Checks the Interrupt Status and Sense CSRs for the expected status.
- Verifies that reading the Sense CSR clears the busy conditions.
- Verifies the contents of the input and output data buffers.

Test 20 WSC - Burst Contention Mix Address Xfrs & MCbus Busy (VMEBUS)

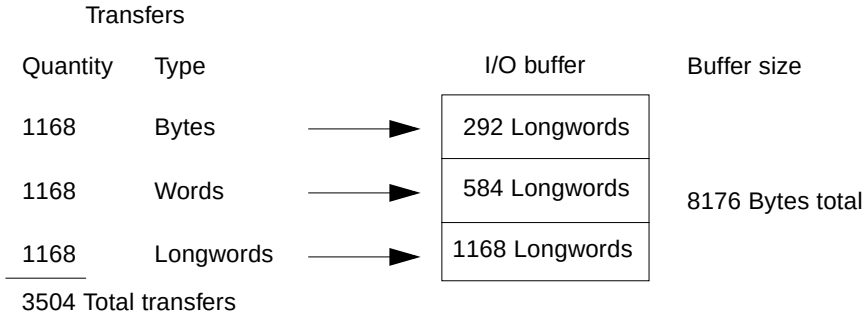
This test causes concurrent activities within the transmitter and receiver hardware and causes contention between the MEMORY CHANNEL Adapter Mezzanine RX port and the EDRAM VME port used to request EDRAM memory arbitration.

The test causes the TX FIFO to fill to its maximum level, just prior to Stall, and then enables the FIFO output to provide a steady data burst through the MCbus TX and RX logic into the RX Buffer FIFO and Receive FIFO. The RX Receive FIFO is disabled from writing data to EDRAM and forces a Global Busy or MCbus Busy condition once its maximum level is reached. The test continues to fill the output buffer to force an MCbus Busy condition and then reads the Sense CSR to verify that Global Busy and Node Busy status is present. The RX Receive FIFO is then enabled to allow the MCA to write data to EDRAM. The input buffer is checked to verify that all data has been transferred.

The test performs a mixture of byte, word, and longword data transfers across the VMEbus. Every other data transfer is a different address type. The data pattern used is unique to the address type. The following data patterns are used during testing:

Address Type	Data Pattern
byte	`0xcc`
word	`0x6666`
longword	`0x33333333`

The test partitions the output buffer into three sections: one for each type of data (byte, word, longword). A byte, word, and longword are written in sequence into their respective buffer areas. This sequence is repeated 1,168 times for a total of 3,504 data transfers. The I/O buffer allocation is contiguous as follows.



After all 3,504 transfers are completed, the input buffer is checked to verify that all the data was transferred.

Note – The MCbus Multiple Grant Request bit (MCMULT) in the MC Control CSR is exercised by the MCbus Interface when the TX FIFO is half full.

The test uses a byte binary progression data pattern.

This test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the TX and RX Windows for the memory test area.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Disables the TX and RX FIFOs.
- Writes the first 450 data type transfers to the test area to force a FIFO Full condition. This is 150 write transfers for each data type.
- Enables the TX FIFO and disables the RX FIFO.
- Writes to the next 231 locations in the test area to force a Global Busy condition. This is 77 write transfers for each data type.
- Verifies that the Sense CSR indicates that the MCA is online and Global Busy, Node Busy, and MC_TX_ACTV are present.
- Writes to the next 441 locations in the test area to fill the TX FIFO and create a maximum burst time period. This is 147 write transfers for each data type.
- Enables the RX FIFO to start writing data to EDRAM.
- Writes to the next 2,382 locations in the test area to complete an 8,176-byte data buffer. This is 794 write transfers for each data type.

- Waits for the data transfers to complete.
- Checks the Interrupt Status and Sense CSRs for the expected status.
- Verifies that reading the Sense CSR clears the busy conditions.
- Verifies the contents of the input and output data buffers.

Test 21 WSC - Reflecting From Multiple CPUs each using Address in Data

This test provides an increased write data rate for reflecting data by using four CPUs writing concurrently to four different output buffers. An Address-in-address data pattern is used for data transfers. The test contains the following subtests:

Subtest	Subtest Name
a	Reflecting from Multiple CPUs (Local Bus)
b	Reflecting from Multiple CPUs (VME Bus)

Note – Subtest a is bypassed if you selected the VME ONLY bus configuration during program initialization.

Once started, the test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate TX and RX Windows.

Each subtest performs the following operations:

- Initializes the input and output data buffers.
- Starts the data transfer.
- Waits for the write operation to complete.
- Verifies that the data transferred correctly.

Test 22 DMA - Stuck and Tied Data Lines Transfer (VMEbus)

This test verifies the ability of the MCA to transmit and receive four doubleword data patterns: all ones, all fives, all A's, all zeros. The test verifies the basic transfer control for transmit and receive hardware and verifies that no bits in the data path are stuck or tied.

Once started, the test performs the following operations:

- Checks for any unexpected status.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes all TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that no unexpected Sense status is present.
- Starts the data transfer.
- Waits for the transfer to complete.
- Verifies that the correct status is received.
- Verifies that the Source address counter is bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 23 DMA - Stuck and Tied Address Lines Single Transfer (VMEbus)

This test verifies the DMA Source and Destination address lines from the DMA address counters. The Source address is used to read EDRAM. The Destination address is set to the TX FIFO and transmitted to the MCbus interface, onto the receiver, and back to the EDRAM receiver path used for writing data into the input buffer. Four different doubleword addresses are used to verify that there are no stuck or tied address lines. The Source and Destination addresses are the one's complement of each other. Therefore, a Source address of ones results in a Destination address of zeros. The test also verifies the ability to access the first and last doubleword memory addresses and to set and reset the V, L, and R bits in the Least Significant Destination Address (LSDA) CSR. In addition, the test verifies the DMACI field and DMA.CMPLT status in the Interrupt Status (INT.STATUS) CSR and the DMA.DONE and DMA.Q.DONE flags in the Byte Count (BC) CSR. The following Source and Destination addresses are used during testing: all ones, all zeros, all fives, all A's. The test uses a byte binary progression data pattern.

Once started, the test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that no unexpected Sense status is present.
- Starts the data transfer.
- Waits for the transfer to complete.
- Checks the Interrupt Status and Sense CSRs for the expected DMACI value, DMA.CMPLT, and online status.
- Verifies that the Source Address (SA) counter is bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.
- Reinitializes the input and output buffers and proceeds with the next data transfer.

Test 24 DMA - Variable Transfer Count (VMEBUS)

This test verifies that the DMA can handle sixteen different transfer counts. The transfer counts are derived by passing a floating one pattern through the 16-bit byte count field which results in counts from one doubleword (8 bytes) to 32K doublewords (256K bytes). This test verifies the ability of the DMA to subdivide the overall byte count into smaller subblocks for transmission across the MCbus.

Note – Subblock sizes of 8, 16, or 32 are jumper-selectable on the ATB board.

The test uses a byte binary progression data pattern when executing the following subtests:

Subtest	Subtest Name		
a	DMA - Variable Transfer Count =	1	(VMEbus)
b	DMA - Variable Transfer Count =	2	(VMEbus)
c	DMA - Variable Transfer Count =	4	(VMEbus)
d	DMA - Variable Transfer Count =	8	(VMEbus)
e	DMA - Variable Transfer Count =	16	(VMEbus)
f	DMA - Variable Transfer Count =	32	(VMEbus)
g	DMA - Variable Transfer Count =	64	(VMEbus)

Subtest	Subtest Name		
h	DMA - Variable Transfer Count =	128	(VMEbus)
i	DMA - Variable Transfer Count =	256	(VMEbus)
j	DMA - Variable Transfer Count =	512	(VMEbus)
k	DMA - Variable Transfer Count =	1K	(VMEbus)
l	DMA - Variable Transfer Count =	2K	(VMEbus)
m	DMA - Variable Transfer Count =	4K	(VMEbus)
n	DMA - Variable Transfer Count =	8K	(VMEbus)
o	DMA - Variable Transfer Count =	16K	(VMEbus)
p	DMA - Variable Transfer Count =	32K	(VMEbus)
q	DMA - Variable Transfer Count =	65528	(VMEbus)

Note – Subtest q uses the maximum transfer count of 64K minus one doubleword (0x7FFF8).

Examples:

- Transfer Count = 1, Byte Count = 0x00008
- Transfer Count = 2, Byte Count = 0x00010
- Transfer Count = 4, Byte Count = 0x00020
- Transfer Count = 32K, Byte Count = 0x40000

Once started, the test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Initializes the input and output data buffers.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the TX Windows.
- Turns the MCA board online.
- Verifies that the board is online and that no unexpected Sense status is present.

Each subtest performs the following operations:

- Starts the data transfer.
- Waits for the data transfer to complete.
- Verifies that the expected DMACI value, DMA.CMPLT, and online status is present.
- Verifies that the SA counter is bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Tests 25 - 27 DMA - Data Pattern Tests (VMEBUS)

These tests verify the ability of the DMA to transmit the following data patterns correctly:

Test	Data Pattern
25	All fives
26	All A's
27	Alternating

Note – The alternating data pattern uses alternating doublewords of all zeros and 0xfefefefe.

Each test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that the Sense CSR indicates that the MCA is online and no unexpected status is present.
- Starts the data transfer.
- Waits for the data transfer to complete.
- Verifies that the Interrupt Status and Sense CSRs contain the expected DMACI value, DMA.CMPLT status, and online status.
- Verifies that the SA counter is bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 28 DMA - Random Data & 64b ECC Sequenced (VMEBUS)

This test verifies the ability of the MEMORY CHANNEL Adapter to generate and check 256 combinations of ECC code. An output data buffer of 64-bit doublewords is provided to make the transmitter EDC chip generate a 64-bit ECC code sequence from 0 to 0xff. This is considered a random data pattern. A total of 8K bytes are transferred. The random pattern repeats every 256 doubleword transfers. This test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that no unexpected Sense status is present.
- Starts the data transfer.
- Waits for the data transfer to complete.
- Checks the Interrupt Status and Sense CSRs for the expected DMACI, DMA.CMPLT, and online status.
- Verifies that the SA counter is bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

TABLE 3-2 contains the data patterns and ECC codes used during testing.

TABLE 3-2 Test 28 Data Patterns and ECC Codes

Data Pattern	ECC	Data Pattern	ECC
0x8ac83333 fff91fdc	0x00	0x3afb2d85 b02c1a2e	0x01
0x9ab3cdf0 0fe4ba99	0x02	0x332fe024 a860cccd	0x03
0x2e9e8f5c a3cf7c05	0x04	0x2b3192c6 a0627f6f	0x05
0x7f5ce93f f48dd5e8	0x06	0x82c9e5d5 f7fad27e	0x07
0x27b39631 9ce482da	0x08	0x16b4a741 8be593ea	0x09
0x5b0562fc d0364fa5	0x0a	0xcde39abd 43148766	0x0b
0x554fbe02 ca80aaab	0x0c	0xa3b46f82 18e55c2b	0x0d
0xe384da74 58b5c71d	0x0e	0x83dd3a08 f90e26b1	0x0f
0xeca77c04 61d868ad	0x10	0xaa9f68ad 1fd05556	0x11
0x146bfedd 899ceb86	0x12	0x1f8248d6 94b3357f	0x13
0x46667779 bb976422	0x14	0x69cca987 defd9630	0x15

TABLE 3-2 Test 28 Data Patterns and ECC Codes *(Continued)*

Data Pattern	ECC	Data Pattern	ECC
0x8ab73334 ffe81fdd	0x16	0xda7338e3 4fa4258c	0x17
0x369cdcba abcdc963	0x18	0x7e38950d f36981b6	0x19
0x2c33e6fa a164d3a3	0x1a	0x6417048d d947f136	0x1b
0x3e8a2a19 b3bb16c2	0x1c	0x220ef136 973fdddf	0x1d
0x9cfc7654 122d62fd	0x1e	0xe6bed70d 5befc3b6	0x1f
0xdee289ad 54137656	0x20	0xe13c3210 566d1eb9	0x21
0x10ff0247 862feef0	0x22	0x0ec759e2 83f8468b	0x23
0xfb6ec28f 709faf38	0x24	0x72cd4b19 e7fe37c2	0x25
0x541a69d1 c94b567a	0x26	0x906cd82e 059dc4d7	0x27
0x6cf5a621 e22692ca	0x28	0xb5f9b2a2 2b2a9f4b	0x29
0xadfb6544 232c51ed	0x2a	0x5d3d0b61 d26df80a	0x2b
0x17d8fb73 8d09e81c	0x2c	0x80813d71 f5b22a1a	0x2d
0x6f714e82 e4a23b2b	0x2e	0x431b7ae1 b84c678a	0x2f
0x7893f012 edc4dcbb	0x30	0x60aa07f7 d5daf4a0	0x31
0x5e615f93 d3924c3c	0x32	0xae0c6543 233d51ec	0x33
0x8739369f fc6a2348	0x34	0xa7326c17 1c6358c0	0x35
0xb3b10a3e 28e1f6e7	0x36	0x3d32d5ea b263c293	0x37
0x0c7eb17e 81af9e27	0x38	0x68a85555 ddd941fe	0x39
0x18fd4fa5 8e2e3c4e	0x3a	0x1b45f809 9076e4b2	0x3b
0xd93de4b2 4e6ed15b	0x3c	0x875b369d fc8c2346	0x3d
0x3b1d2d83 b04e1a2c	0x3e	0x07ed60b6 7d1e4d5f	0x3f
0x89a3df01 fed4cbaa	0x40	0x551cbe05 ca4daaae	0x41
0xc94149f6 3e72369f	0x42	0xb29db60b 27cea2b4	0x43
0x5c18b72f d149a3d8	0x44	0x28d7ea63 9e08d70c	0x45
0x7d2540da f2562d83	0x46	0x11100246 8640eeef	0x47
0xfa4a6e5d 6f7b5b06	0x48	0x886e8ad0 fd9f7779	0x49
0xa2a11b4f 17d207f8	0x4a	0x61ce5c29 d6ff48d2	0x4b
0x0b5a5d4c 808b49f5	0x4c	0xf24c20ff 677d0da8	0x4d
0x6c0451ec e1353e95	0x4e	0xa18dc71c 16beb3c5	0x4f
0x1c6a4c3b 919b38e4	0x50	0x3fae7e4b b4df6af4	0x51

TABLE 3-2 Test 28 Data Patterns and ECC Codes *(Continued)*

Data Pattern	ECC	Data Pattern	ECC
0xba8b036a 2fbbf013	0x52	0x23334568 98643211	0x53
0x0febae14 851c9abd	0x54	0xfdb76af3 72e8579c	0x55
0x59d00ecb cf00fb74	0x56	0x59e10eca cf11fb73	0x57
0xc71aa190 3c4b8e39	0x58	0x6d39a61d e26a92c6	0x59
0xf003789b 65346544	0x5a	0x7e49950c f37a81b5	0x5b
0x1a21a3d7 8f529080	0x5c	0xd274eb85 47a5d82e	0x5d
0xc91f49f8 3e5036a1	0x5e	0xd4bd93e9 49ee8092	0x5f
0x1b23f80b 9054e4b4	0x60	0xf3707531 68a161da	0x61
0x67730124 dca3edcd	0x62	0x3556888a aa877533	0x63
0x540969d2 c93a567b	0x64	0x67840123 dcb4edcc	0x65
0xedbad037 62ebbce0	0x66	0xd1509753 468183fc	0x67
0xbbaf579c 30e04445	0x68	0xbde70001 3317ecaa	0x69
0xe12b3211 565c1eba	0x6a	0xdef389ac 54247655	0x6b
0x443fcf13 b970bbbc	0x6c	0x916f2c62 06a0190b	0x6d
0x38c38520 adf471c9	0x6e	0x3545888b aa767534	0x6f
0xfedbbf25 740cabce	0x70	0x377d30f0 acae1d99	0x71
0x9849258e 0d7a1237	0x72	0x34543456 a98520ff	0x73
0x1d8ea06d 92bf8d16	0x74	0x50ad6d3b c5de59e4	0x75
0x81a591a3 f6d67e4c	0x76	0x30d637c1 a607246a	0x77
0x9f451eb8 14760b61	0x78	0x4f9a1908 c4cb05b1	0x79
0xc5f64d5e 3b273a07	0x7a	0xf22a2101 675b0daa	0x7b
0x819491a4 f6c57e4d	0x7c	0x5ae362fe d0144fa7	0x7d
0xd94ee4b1 4e7fd15a	0x7e	0xb3c20a3d 28f2f6e6	0x7f
0x6d28a61e e25992c7	0x80	0xf7f0c5fa 6d21b2a3	0x81
0x1fb548d3 94e6357c	0x82	0xb13561dd 26664e86	0x83
0xeb6127d4 6092147d	0x84	0x23444567 98753210	0x85
0xd4ac93ea 49dd8093	0x86	0x93c8d4c5 08f9c16e	0x87
0x69aaa989 dedb9632	0x88	0x8f5983fb 048a70a4	0x89
0xda2f38e7 4f602590	0x8a	0x9e20ca86 1351b72f	0x8b
0xea5ed3a0 5f8fc049	0x8c	0x56741234 cba4fedd	0x8d

TABLE 3-2 Test 28 Data Patterns and ECC Codes *(Continued)*

Data Pattern	ECC	Data Pattern	ECC
0xeb7227d3 60a3147c	0x8e	0x1590530f 8ac13fb8	0x8f
0xd02c4321 455d2fca	0x90	0xeb8327d2 60b4147b	0x91
0x47accba9 bcddb852	0x92	0x58abba99 cddca742	0x93
0x2c55e6f8 a186d3a1	0x94	0xa2b21b4e 17e307f7	0x95
0xa15ac71f 168bb3c8	0x96	0x0237bbbc 7768a865	0x97
0x7084a2b5 e5b58f5e	0x98	0xca659e28 3f968ad1	0x99
0x368bdcbb abbcc964	0x9a	0xc709a191 3c3a8e3a	0x9b
0x034b0fef 787bfc98	0x9c	0xc83ef5c2 3d6fe26b	0x9d
0x257bedcc 9aacda75	0x9e	0x8e352fc9 03661c72	0x9f
0xe9297f6f 5e5a6c18	0xa0	0x4d5170a4 c2825d4d	0xa1
0xb5e8b2a3 2b199f4c	0xa2	0xabb2bce0 20e3a989	0xa3
0x8bec8765 011d740e	0xa4	0xc5e54d5f 3b163a08	0xa5
0xefd0789e 65016547	0xa6	0xc27850c9 37a93d72	0xa7
0x1fd748d1 9508357a	0xa8	0xfc7116c3 71a2036c	0xa9
0xf6dd71c7 6c0e5e70	0xaa	0x1d7da06e 92ae8d17	0xab
0x12345678 87654321	0xac	0x38d4851f ae0571c8	0xad
0xd91be4b4 4e4cd15d	0xae	0xd5e1e81b 4b12d4c4	0xaf
0x3c4181b5 b1726e5e	0xb0	0x91a22c5f 06d31908	0xb1
0xa05872eb 15895f94	0xb2	0xb0110dab 2541fa54	0xb3
0x0a36091a 7f66f5c3	0xb4	0x0a25091b 7f55f5c4	0xb5
0x4e53c4d8 c384b181	0xb6	0x24689999 99998642	0xb7
0xa06972ea 159a5f93	0xb8	0x4e31c4da c362b183	0xb9
0xe8162b3c 5d4717e5	0xba	0x48d11fdb be020c84	0xbb
0x45532346 ba840fef	0xbc	0xa5ec17e7 1b1d0490	0xbd
0x1eb2f49f 93e3e148	0xbe	0xb8315b07 2d6247b0	0xbf
0x2a0d3e94 9f3e2b3d	0xc0	0x8f3783fd 046870a6	0xc1
0xf494c963 69c5b60c	0xc2	0x41f726af b7281358	0xc3
0x764b47ae eb7c3457	0xc4	0xe4a92ea6 59da1b4f	0xc5
0xf9261a2b 6e5706d4	0xc6	0xc4d1f92c 3a02e5d5	0xc7
0x28e8ea62 9e19d70b	0xc8	0xcb9af259 40cbdf02	0xc9

TABLE 3-2 Test 28 Data Patterns and ECC Codes *(Continued)*

Data Pattern	ECC	Data Pattern	ECC
0x2c44e6f9 a175d3a2	0xca	0x0593b853 7ac4a4fc	0xcb
0x776f9be0 eca08889	0xcc	0x4655777a bb866423	0xcd
0x221ff135 9750ddde	0xce	0x035c0fee 788cfc97	0xcf
0x83ee3a07 f91f26b0	0xd0	0x887f8acf fdb07778	0xd1
0x05a4b852 7ad5a4fb	0xd2	0xb4e65e6f 2a174b18	0xd3
0x94fe28f6 0a2f159f	0xd4	0xf127cccd 6658b976	0xd5
0xf801c5f9 6d32b2a2	0xd6	0x73f19f4b e9228bf4	0xd7
0x58bcba98 cdeda741	0xd8	0x7f6de93e f49ed5e7	0xd9
0x7095a2b4 e5c68f5d	0xda	0x1c594c3c 918a38e5	0xdb
0xbcd3abce 32049877	0xdc	0xbf1c5432 344d40db	0xdd
0x56631235 cb93fede	0xde	0x6c1551eb e1463e94	0xdf
0x6af0fdb9 e021ea62	0xe0	0xb0330da9 2563fa52	0xe1
0x35788888 aaa97531	0xe2	0x62f2b05b d8239d04	0xe3
0x57986666 ccc9530f	0xe4	0xaf30b975 2461a61e	0xe5
0x986b258c 0d9c1235	0xe6	0x12235679 87544322	0xe7
0xace81111 2218fdba	0xe8	0xb71e06d4 2c4ef37d	0xe9
0xc02fa865 3560950e	0xea	0x905bd82f 058cc4d8	0xeb
0x542b69d0 c95c5679	0xec	0x804e3d74 f57f2a1d	0xed
0xc39ca4fb 38cd91a4	0xee	0xe5cd82d8 5afe6f81	0xef
0xb60ab2a1 2b3b9f4a	0xf0	0x57876667 ccb85310	0xf1
0xa3d66f80 19075c29	0xf2	0xd7063c4d 4c3728f6	0xf3
0xfc9316c1 71c4036a	0xf4	0x1d6ca06f 929d8d18	0xf5
0xddbe357b 52ef2224	0xf6	0x45642345 ba950fee	0xf7
0x1b34f80a 9065e4b3	0xf8	0xd13f9754 467083fd	0xf9
0x37b030ed ace11d96	0xfa	0x96227d28 0b5369d1	0xfb
0xddad357c 52de2225	0xfc	0x0c4bb181 817c9e2a	0xfd
0x907dd82d 05aec4d6	0xfe	0x9746d15a 0c77be03	0xff

Test 29 DMA - DMA Throttled by TX & RX FIFOs (VMEbus)

This test verifies the throttling control of the DMA engine. The first part of the test verifies that the half-full state of the TX FIFO stops DMA from reading EDRAM at a precise transfer count. The second half of the test verifies that the RX FIFO half-full state stops DMA activity and also verifies that a back-to-back burst subblock can be transferred correctly. This part of the test causes concurrent operation between DMA and the receiver arbitrating for the Mezzanine port. This test also verifies the DMACI field and DMA.CMPLT status in the Interrupt Status (INT.STATUS) CSR, the DMA.DONE flag in the Byte Count (BC) CSR, and that the Source Address Error (SAE) CSR reflects the correct source address.

Note – The Multiple Bus Grant Request bit (MCMULT) in the MC Control CSR is exercised by the MCbus interface when the TX FIFO is half-full.

The test uses an Address-in-address data pattern and performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Checks for unexpected Sense status.
- Disables the TX and RX FIFOs.

The first part of the test verifies that the DMA stops when the TX FIFO is half-full by performing the following operations:

- Starts the data transfer.
- Waits for a DMA Stall condition.
- Verifies that the Interrupt Status and Sense CSRs report the correct status.
- Calculates the expected Source address and verifies that the Source Address Error (SAE) CSR contains the correct address.

The second part of the test verifies that the DMA stops when the RX FIFO is half-full and also verifies that a back-to-back burst subblock transfers correctly. The following operations are performed:

- Enables Local Loopback mode.
- Enables the TX FIFO and disables the RX FIFO.
- Waits for the RX FIFO to reach the half-full state.
- Enables Local Loopback mode.
- Enables the TX and RX FIFOs.

- Waits for the data transfer to complete.
- Verifies that the Interrupt Status and Sense CSRs report the correct status.
- Verifies that the SA counter is bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 30 DMA - Source & Destination Address Cross 4K Boundary (VMEBUS)

This test verifies that the DMA can detect its Source and Destination addresses crossing a 4K boundary and can properly terminate a burst read and restart a new burst read subblock. The test contains the following subtests:

Subtest	Subtest Name
a	DMA - Source & Destination Address Cross 4K Boundary 1-1-5 (VMEbus)
b	DMA - Source & Destination Address Cross 4K Boundary 2-1-4 (VMEbus)
c	DMA - Source & Destination Address Cross 4K Boundary 4-3 (VMEbus)
d	DMA - Source & Destination Address Cross 4K Boundary 5-1-1 (VMEbus)

Each subtest verifies that the hardware can handle different combinations of boundary crossings. Every subtest except c forces the Source address to cross and then forces the Destination address to cross. Subtest c forces both the Source and Destination addresses to cross at the same point.

The sequence number appended to the subtest name describes how the DMA's overall byte count is divided into subblocks. For example, 1-1-5 is appended to Subtest a. This indicates a division of three subblocks with an overall byte count of seven. Using seven for the byte count allows the same subblock division to be used for 8, 16, and 32 subblock selections. The 1-1-5 value is further defined as follows:

1	The first subblock has one doubleword terminated by the Sense address crossing.
1	The second subblock has one doubleword terminated by the Destination address crossing.
5	The third subblock has five doublewords terminated when the DMA byte count decrements to zero.

Once started, the test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the TX Windows.

- Turns the MCA board online.
- Initializes the input and output data buffers. The test uses an Address-in-address data pattern.
- Checks for unexpected Sense status.

Each subtest performs the following operations:

- Starts the data transfer.
- Waits for the data transfer to complete.
- Verifies that the Interrupt Status and Sense CSRs report the correct status.
- Verifies that the SA counter is bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 31 DMA - Command Queuing (VMEBUS)

This test verifies the ability of the DMA command FIFO to queue multiple commands. The test contains the following subtests:

Subtest	Subtest Name
a	DMA - Command Queue Full Verification (VMEbus)
b	DMA - Command Issuance & Queuing Concurrent with MCbus I/O (VMEbus)
c	DMA - Command Queue Overflow Verification (VMEbus)

Each subtest provides a different level of contention while command queuing is performed. A subtest issues 133 commands with an output buffer length of 64 doublewords. The buffers are concatenated to create one large contiguous I/O buffer. Subtest c issues an additional command to cause command overflow error status.

Before the first command is issued, the I/O buffer is initialized to minimize software overhead time between commands. The objective is to stress the hardware by issuing commands at the highest possible frequency.

Note – The Multiple Bus Grant Request bit (MCMULT) in the MC Control CSR is exercised by the MCbus interface when the TX FIFO is half-full.

Subtest a

This subtest verifies that the maximum number of commands can be queued and also provides concurrent activities of MCbus I/O and the receiver writing data back to the EDRAM input buffer. The subtest issues 133 commands with the transmit FIFO output disabled to cause the command queue to reach the full state. The full state occurs at 133 commands instead of 128 since the first five commands are dequeued by the hardware DMA. This is because data for the first four commands is stored and data for the fifth command is partially stored in the transmitter output FIFO when it reaches the half-full state. After all the commands are queued, the transmit FIFO output is enabled to allow MCbus transfers. This subtest performs the following operations:

- Resets the MCA board and verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers. The subtest uses an Address-in-address data pattern.
- Checks for unexpected Sense status.
- Disables the transmit FIFO.
- Issues 133 commands, changing the source and destination addresses accordingly.
- Verifies that the DMACI value advanced by four commands.
- Verifies that no unexpected status is present.
- Enables the transmit FIFO.
- Waits for the data transfers to complete.
- Checks for the expected DMACI value.
- Verifies that no unexpected status was received.
- Verifies that the SA counter was bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Subtest b

This subtest provides the concurrent activities of issuing commands, MCbus I/O, and the receiver writing data back to the EDRAM input buffer. The subtest issues 90 commands with the transmit output FIFO disabled to queue commands in the command FIFO and fill the transmitter output FIFO to half-full. The subtest then enables the transmit FIFO output to allow MCbus I/O transfers and issues the remaining 43 commands. This subtest performs the following operations:

- Resets the MCA board and verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.
- Turns the MCA board online.

- Initializes the input and output data buffers. The subtest uses an Address-in-address data pattern.
- Checks for unexpected Sense status.
- Disables the transmit FIFO.
- Issues 90 commands, changing the source and destination addresses.
- Verifies that the DMACI value advanced by four commands.
- Verifies that no unexpected status is present.
- Enables the transmit FIFO.
- Issues 43 commands.
- Waits for the data transfers to complete.
- Checks for the expected DMACI value.
- Verifies that no unexpected status was received.
- Verifies that the SA counter was bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Subtest c

This subtest verifies the operation of the command queue overflow detection logic. The subtest issues 134 commands while the transmit FIFO output is disabled to cause a command FIFO overflow condition. By the time the 134th command is issued, the transmitter output FIFO has already reached its half-full state. This allows the subtest to check the state of the TX.HALTED status and verify the DMACI count is advanced by four doublewords. The subtest then enables the transmit FIFO output and verifies that the TX.HALTED state prohibits data from being transferred. This subtest performs the following operations:

- Resets the MCA board and verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers. The subtest uses an Address-in-address data pattern.
- Checks for unexpected Sense status.
- Disables the transmit FIFO.
- Issues 134 commands, changing the source and destination addresses accordingly.
- Enables the transmit FIFO.
- Verifies that the DMACI value advanced by four commands.
- Verifies that command queue overflow (CMD.OVFL) status is present.
- Verifies that TX.HALTED status is present.
- Calculates the expected Source Address Error byte address.
- Verifies that no data was transferred.
- Resets the MCA board.

Test 32 DMA - Command Queuing Address Cross 4K Boundary (VMEBUS)

This test verifies the ability of the DMA command FIFO to queue multiple commands. The test contains the following subtests:

Subtest	Subtest NAme
a	DMA - Command Queue Full Verification (VMEbus)
b	DMA - Command Issuance & Queuing Concurrent with MCbus I/O (VMEbus)
c	DMA - Command Queue Overflow Verification (VMEbus)

Each subtest provides a different level of contention while command queuing is performed. A subtest issues 133 commands with an output buffer length of 64 doublewords. The buffers are concatenated to create one large contiguous I/O buffer. Subtest c issues an additional command to cause command overflow error status. The DMA source and destination starting addresses are offset to cause the transmission of a combination of partial and full subblocks due to crossing a 4K byte address boundary.

Before the first command is issued, the I/O buffer is initialized to minimize software overhead time between commands. The objective is to stress the hardware by issuing commands at the highest possible frequency.

Note – The Multiple Bus Grant Request bit (MCMULT) in the MC Control CSR is exercised by the MCbus interface when the TX FIFO is half-full.

Subtest a

This subtest verifies that the maximum number of commands can be queued and also provides concurrent activities of MCbus I/O and the receiver writing data back to the EDRAM input buffer. The subtest issues 133 commands with the transmit FIFO output disabled to cause the command queue to reach the full state. The full state occurs at 133 commands instead of 128 since the first five commands are dequeued by the hardware DMA. This is because data for the first four commands is stored and data for the fifth command is partially stored in the transmitter output FIFO when it reaches the half-full state. After all the commands are queued, the transmit FIFO output is enabled to allow MCbus transfers. This subtest performs the following operations:

- Resets the MCA board and verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.

- Closes the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers. The subtest uses an Address-in-address data pattern.
- Checks for unexpected Sense status.
- Disables the transmit FIFO.
- Issues 133 commands, changing the source and destination addresses.
- Verifies that the DMACI value advanced by four commands.
- Verifies that no unexpected status is present.
- Enables the transmit FIFO.
- Waits for the data transfers to complete.
- Checks for the expected DMACI value.
- Verifies that no unexpected status was received.
- Verifies that the SA counter was bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Subtest b

This subtest provides the concurrent activities of issuing commands, MCbus I/O, and the receiver writing data back to the EDRAM input buffer. The subtest issues 90 commands with the transmit output FIFO disabled to queue commands in the command FIFO and fill the transmitter output FIFO to half-full. The subtest then enables the transmit FIFO output to allow MCbus I/O transfers and issues the remaining 43 commands. This subtest performs the following operations:

- Resets the MCA board and verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers. The subtest uses an Address-in-address data pattern.
- Checks for unexpected Sense status.
- Disables the transmit FIFO.
- Issues 90 commands, changing the source and destination addresses.
- Verifies that the DMACI value advanced by four commands.
- Verifies that no unexpected status is present.
- Enables the transmit FIFO.
- Issues 43 commands.
- Waits for the data transfers to complete.
- Checks for the expected DMACI value.
- Verifies that no unexpected status was received.
- Verifies that the SA counter was bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Subtest c

This subtest verifies the operation of the command queue overflow detection logic. The subtest issues 134 commands while the transmit FIFO output is disabled to cause a command FIFO overflow condition. By the time the 134th command is issued, the transmitter output FIFO has already reached its half-full state. This allows the subtest to check the state of the TX.HALTED status and verify the DMACI count is advanced by four doublewords. The subtest then enables the transmit FIFO output and verifies that the TX.HALTED state prohibits data from being transferred. This subtest performs the following operations:

- Resets the MCA board and verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers. The subtest uses an Address-in-address data pattern.
- Checks for unexpected Sense status.
- Disables the transmit FIFO.
- Issues 134 commands, changing the source and destination addresses.
- Enables the transmit FIFO.
- Verifies that the DMACI value advanced by four commands.
- Verifies that command queue overflow (CMD.OVFL) status is present.
- Verifies that TX.HALTED status is present.
- Calculates the expected Source Address Error byte address.
- Verifies that no data was transferred.
- Resets the MCA board.

Test 33 DMA - Receiver Closes Odd Page Window Addresses (VMEbus)

This test verifies that the receiver can handle DMA burst transfers and accept or purge the data depending on the state of the Receive Window RAM's open or close bit. Occasionally, the DMA will contend with the receiver for the EDRAM mezzanine port. For example, when a Receive Window is closed, the receive Buffer FIFO purges the data which idles the receiver FIFO. The DMA is issued a single command with a byte count of 0x40000 (32768 doublewords), starting at the beginning of the memory test area, which is an even page Window. The receiver is programmed only with even Window addresses. Once started, the test performs the following operations:

- Resets the MCA board and verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.

- Turns the MCA board online.
- Initializes the input and output data buffers. The test uses an Address-in-address data pattern.
- Checks for unexpected Sense status.
- Issues a DMA command.
- Verifies that the command executed successfully.
- Verifies that the source address advanced by four doublewords.
- Verifies the contents of the input and output data buffers.

Test 34 DMA - Access EDRAM Memory while DMA Transfer is in Progress

This test verifies that the EDRAM memory array can be accessed concurrently by both the MCA mezzanine port and the Local Bus or VMEbus ports. The DMA is issued a single command with a byte count of 0x40000 (32768 doublewords) to keep the mezzanine port busy. In the meantime, the test performs CPU writes and reads to an unused Page Window of the memory test area. The CPU writes cause the MCA to handle Snoop traffic concurrently with the DMA and receiver activities. The Snoop transfer will be purged because all the Transmit Windows are closed. The CPU writes and reads cause the EDRAM port arbiter to handle two ports concurrently. This test contains the following subtests:

Subtest	Subtest Name
a	DMA - Access EDRAM Memory while DMA Transfer is in Progress (LBUS)
b	DMA - Access EDRAM Memory while DMA Transfer is in Progress (VMEbus)

Note – Subtest a is bypassed if you selected the VME ONLY testing configuration during program initialization.

During Subtest a, the CPU is used to access EDRAM across the Local Bus while the mezzanine port is busy with the MCA DMA engine and MCA receiver. During Subtest b, the CPU is used to access EDRAM across the VMEbus while the mezzanine port is busy with the MCA DMA engine and MCA receiver. Each subtest uses a byte binary progression data pattern to perform the following operations:

- Resets the MCA board and verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the even RX Windows and closes the odd RX Windows.
- Closes all the TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.

- Executes the DMA command.
- While the DMA transfer is in progress, the CPU performs the following operations:
  - Writes one 4K byte Page Window in an unused test area.
  - Reads the data back and compares it for errors.
  - If the DMA transfer is not complete, the CPU continues to write and read the same page with a different data pattern until the data transfer is complete or a timeout error occurs.
- Verifies that the correct status and Sense values are present.
- Verifies that the correct Source Address Error value is present.
- Verifies the contents of the input and output data buffers.

Test 35 DMA - Access Window RAMs while DMA Transfer is in Progress (VMEbus)

This test verifies that the MCA can handle concurrent activities of DMA read EDRAM, receiver write EDRAM, and TX and RX Translation Window RAM CSR accesses. The test causes sufficient contention to momentarily suspend the DMA and the receiver while the Window RAM is accessed. When the DMA is inactive, the mezzanine STALL function is exercised whenever the Window RAM is accessed. The test uses a byte binary progression data pattern and performs the following operations:

- Resets the MCA board and verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes all the TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Checks for unexpected Sense status.
- Writes the RAM address into the TWRA, RWSRA and RWPRA. The RAM address must point outside the memory test area.
- Executes the DMA command.
- While the DMA transfer is in progress, the test loops on RAM accesses.
- Verifies that the DMA command executed successfully.
- Verifies that the correct status and Sense values are present.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 36 WSC/DMA - Alternating WSC & DMA Subblock Transfers (VMEBUS)

This test verifies the ability of the MCA to transmit and receive alternating WSC (Snoop) and DMA (burst) transfers. The transmit FIFO is disabled and alternately filled with a longword Snoop transfer and a DMA burst transfer of three doublewords. These commands are issued 64 times to half-fill the transmit FIFO. The FIFO is then enabled causing the transmission of back-to-back single and burst transfers through the MCbus interface and through the receiver into the EDRAM input buffer.

The WSC I/O buffers are contiguous and are allocated to the first Window Page of the memory test area. The DMA I/O buffers are also contiguous and are allocated to the second Window Page of the memory test area. WSC transfers use an all ones data pattern and DMA transfers use a byte binary progression data pattern.

Note – This test does not verify simultaneous operation of the DMA, Snoop transfer, and receiver at the mezzanine interface. This feature is checked in other tests.

Once started, this test performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Opens the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input data buffer.
- Initializes the output data buffer for DMA only.
- Checks for unexpected Sense status.
- Disables the TX FIFO.
- Repeats the following steps 64 times to create 64 WSC sequential memory writes alternating with 64 DMA burst transfers:
  - Issues a WSC Snoop transfer.
  - Issues a DMA burst transfer.
- Enables the TX FIFO.
- Waits for DMA input data transfer to complete.
- Verifies that the correct status and Sense values were received.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 37 WSC/DMA - Byte Transfer Concurrent with DMA Transfer

This test verifies the ability of the MCA to transmit and receive WSC (Snoop) byte transfers concurrent with DMA (burst) transfers. The test verifies the simultaneous operation of the DMA, Snoop transfer, and receiver at the mezzanine interface. This test contains the following subtests:

Subtest	Subtest Name
a	WSC/DMA - Byte Transfer Concurrent with DMA Transfer (LBUS)
b	WSC/DMA - Byte Transfer Concurrent with DMA Transfer (VMEbus)

Note – Subtest a is bypassed if you selected the VME ONLY testing configuration during program initialization.

Subtest a performs WSC writes across the Local Bus port and Subtest b performs WSC writes across the VMEbus port. Each port provides different WSC write rates to the MCA.

The WSC I/O buffer is a 4K byte contiguous buffer allocated to the first Window Page of the memory test area. The DMA I/O buffer is a 32,768 doubleword contiguous buffer allocated to start at the fifth Window Page of the memory test area. Buffers are selected to cause the WSC write transfer period to overlap the DMA transfer period. WSC transfers use an all ones data pattern and DMA transfers use a byte binary progression data pattern.

Once started, each subtest performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Opens the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input data buffer.
- Initializes the output data buffer for DMA only.
- Checks for unexpected Sense status.
- Issues a DMA burst transfer.
- Issues a WSC Snoop transfer.
- Waits for DMA input data transfer to complete.
- Verifies that the correct status and Sense values were received.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 38 WSC/DMA - Longword Transfer Concurrent with DMA Transfer

This test verifies the ability of the MCA to transmit and receive WSC (Snoop) longword transfers concurrent with DMA (burst) transfers. The test verifies the simultaneous operation of the DMA, Snoop transfer, and receiver at the mezzanine interface. This test contains the following subtests:

Subtest	Subtest Name
a	WSC/DMA - Longword Transfer Concurrent with DMA Transfer (LBUS)
b	WSC/DMA - Longword Transfer Concurrent with DMA Transfer (VMEbus)

Note – Subtest a is bypassed if you selected the VME ONLY testing configuration during program initialization.

Subtest a performs WSC writes across the Local Bus port and Subtest b performs WSC writes across the VMEbus port. Each port provides different WSC write rates to the MCA.

The WSC I/O buffer is a 16K byte contiguous buffer allocated to the first four Window Pages of the memory test area. The DMA I/O buffer is a 32,768 doubleword contiguous buffer allocated to start at the fifth Window Page of the memory test area. Buffers are selected to cause the WSC write transfer period to overlap the DMA transfer period. WSC transfers use an all ones data pattern and DMA transfers use a byte binary progression data pattern.

Once started, each subtest performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Opens the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input data buffer.
- Initializes the output data buffer for DMA only.
- Checks for unexpected Sense status.
- Issues a DMA burst transfer.
- Issues a WSC Snoop transfer.
- Waits for DMA input data transfer to complete.
- Verifies that he correct status and Sense values were received.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 39 WSC/DMA - WSC Transfer while Command Queuing (VMEBUS)

This test verifies that WSC Snoop transfers can be transferred concurrently with DMA data transfers while the DMA command FIFO is queued to its maximum level. The test checks the following conditions:

- When the transmit FIFO is half-full, it suspends DMA reads, but allows Snoop transfers to be loaded until it is seven-eighths full.
- The MCA can handle a variety of transfer combinations. Testing starts with four DMA commands using partial and full subblocks of burst data, followed by multiple Snoop single subblock transfers. These transfers are followed with a mixture of DMA burst transfers and Snoop single transfers.
- The DMA can fetch queued commands, arm the DMA, and transfer DMA data into the transmit FIFO while Snoop transfers are passing over the mezzanine port and being processed by the TX Translation RAM and loaded into the TX FIFO. While this occurs, the receiver contends with the DMA and Snoop activities for writing loopback data to EDRAM.

The DMA source and destination addresses are purposely offset to cause a combination of partial and full subblocks to be transferred because a 4K byte address boundary is crossed. The entire DMA I/O buffer is initialized before the first command is issued to minimize the software overhead time between commands.

This test contains the following subtests:

Subtest	Subtest Name
a	WSC/DMA - WSC Longword Transfer while Command Queuing (VMEbus)
b	WSC/DMA - WSC Byte Transfer while Command Queuing (VMEbus)

Note – The Multiple Bus Grant Request bit (MCMULT) in the MC Control CSR is exercised by the MCbus interface when the TX FIFO is half-full.

The WSC I/O buffer used for Subtest a is an 8K byte contiguous buffer allocated to the first and second Window Pages of the memory test area. The WSC I/O buffer used for Subtest b is an 4K byte contiguous buffer allocated to the first Window Page of the memory test area. The DMA I/O buffer is an 8,512 doubleword contiguous buffer allocated to start at the fifth Window Page of the memory test area. Buffers are selected to cause the WSC write transfer period to overlap the DMA transfer period. WSC transfers use an all ones data pattern and DMA transfers use a byte binary progression data pattern.

Once started, each subtest performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Opens the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input data buffer.
- Initializes the output data buffer for DMA only.
- Checks for unexpected Sense status.
- Disables the TX FIFO.
- Issues 133 DMA transfer commands.
- Issues WSC Snoop transfers until the TX FIFO is seven-eighths full.
- Verifies that the expected status is present.
- Enables the TX FIFO.
- Waits until DMA starts up from the suspended state (TX FIFO drops below half-full).
- Verifies that the status indicates the fifth command is completed.
- Issues additional Snoop writes until the entire output buffer has been written.
- Waits for the DMA input data transfer to complete.
- Verifies that the correct status and Sense values were received.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 40 EDRAM VME32 Block Transfer Mode (SCSI & MC4)

This test verifies that the EDRAM VME port accepts 32-bit read and write transfers in VME32 block mode. The SCSI is used to create memory traffic in VME32 block transfer mode. The test performs the following subtests:

Subtest	Transfer Count
a	Single Block Transfer (VMEBus Boundary 0:0)
b	Single Block Transfer (VMEBus Boundary 0:4)
c	Single Block Transfer (VMEBus Boundary 4:4)
d	Single Block Transfer (VMEBus Boundary 4:8)
e	Single Block Transfer (VMEBus Boundary 8:8)
f	Single Block Transfer (VMEBus Boundary 8:c)
g	Single Block Transfer (VMEBus Boundary c:c)
h	Single Block Transfer (VMEBus Boundary c:0)
i	Multiple Block Transfer (VMEBus Cross Boundary 0:0)

Subtest	Transfer Count
j	Multiple Block Transfer (VMEBus Cross Boundary 0:4)
k	Multiple Block Transfer (VMEBus Cross Boundary 4:4)
l	Multiple Block Transfer (VMEBus Cross Boundary 4:8)
m	Multiple Block Transfer (VMEBus Cross Boundary 8:8)
n	Multiple Block Transfer (VMEBus Cross Boundary 8:c)
o	Multiple Block Transfer (VMEBus Cross Boundary c:c)
p	Multiple Block Transfer (VMEBus Cross Boundary c:0)

This test performs the following operations for each word boundary combination and 256 / 48,992 byte count:

- Clears, initializes, and verifies the register map.
- Executes a `SCSI Reset` command and verifies the SCSI status received.
- Executes a `Restart SCSI` command and verifies the SCSI status received.
- Issues a `SCSI Reinitialize Device` command and verifies the SCSI status received.
- Issues a `Start-Stop` command to the SCSI.
- Initializes the input buffer to zeros.
- Fills the output buffer with an incrementing data pattern.
- Issues a `Write Data Buffer` command to the SCSI.
- If an error occurs, the subtest issues `Test SCSI Device` and `SCSI Request Sense` commands and verifies the SCSI status received from each command.
- Issues a `Read Data Buffer` command to the SCSI.
- Compares the output buffer to the input buffer.
- Issues a `Start-Stop` command to the SCSI.

Note – This test is bypassed if no SCSI disk drive is configured on the VME Bus. Refer to Chapter 1 for the supported controllers and devices.

Test 41 EDRAM VME D64 Block Transfer Mode (SCSI & MC4)

This test verifies that the EDRAM VME port accepts 64-bit read and write transfers in VME D64 block mode. The SCSI is used to create memory traffic in VME D64 block transfer mode. The test performs the following subtests:

Subtest	Transfer Count
a	Single Block Transfer (VMEBus Boundary 0:0)
b	Single Block Transfer (VMEBus Boundary 0:8)
c	Single Block Transfer (VMEBus Boundary 8:0)
d	Single Block Transfer (VMEBus Boundary 8:8)
e	Multiple Block Transfer (VMEBus Cross Boundary 0:0)
f	Multiple Block Transfer (VMEBus Cross Boundary 0:8)
g	Multiple Block Transfer (VMEBus Cross Boundary 8:0)
h	Multiple Block Transfer (VMEBus Cross Boundary 8:8)

This test performs the following operations for each doubleword boundary combination and 2048 / 48,992 byte count:

- Clears, initializes, and verifies the register map.
- Executes a `SCSI Reset` command and verifies the SCSI status received.
- Executes an `Initialize SCSI Controller` command and verifies the SCSI status received.
- Executes a `Restart SCSI` command and verifies the SCSI status received.
- Issues a `SCSI Reinitialize Device` command and verifies the SCSI status received.
- Issues a `Start-Stop` command to the SCSI.
- Initializes the input buffer to zeros.
- Fills the output buffer with an incrementing data pattern.
- Issues a `Write Data Buffer` command to the SCSI.
- If an error occurs, the subtest issues `Test SCSI Device` and `SCSI Request Sense` commands and verifies the SCSI status received from each command.
- Issues a `Read Data Buffer` command to the SCSI.
- Compares the output buffer to the input buffer.
- Issues a `Start-Stop` command to the SCSI.

Note – This test is bypassed if no SCSI disk drive is configured on the VME Bus. Refer to Chapter 1 for the supported controllers and devices.

Test 42 DMA - VME Interrupts

This test verifies the ability of the MCA to assert a programmable VME interrupt and return the programmable interrupt vector during the interrupt acknowledge cycle. The interrupt level and vector are programmed into the VIC CSR and the interrupt is enabled with the ECI bit in the MC.CTL CSR. This test contains the following subtests:

Subtest	Subtest Name
a	DMA - VME Interrupt Levels 1-7
b	DMA - VME Interrupt Going From a DCI to ECI State
c	DMA - VME Interrupt Asserted Inhibits INT.STATUS Clearing
d	DMA - VME Interrupt or ECI State Write Protects VIC CSR

The test uses all ones, fives, A's, and zeros data patterns while performing testing operations. Once started, this test performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that no unexpected status is present.

Subtest a

This subtest verifies VME interrupt levels 1 through 7. An all fives data pattern is used as the interrupt vector value. Once started, the subtest performs the following operations:

- Enables Local Loopback mode and Continue on Read Error, and disables the channel interrupt.
- Sets the interrupt level and interrupt vector.
- Enables the channel interrupt.
- Starts the data transfer.
- Waits for the interrupt to occur.
- Acknowledges the interrupt and verifies that the correct interrupt vector was returned.
- Verifies that the expected status is present and that no unexpected status was received.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.
- Repeats the subtest for each interrupt level.

Subtest b

This subtest verifies that the interrupt under test is disabled while interrupt status is pending, and then enabled. An all A's data pattern is used as the interrupt vector value. Once started, the subtest performs the following operations:

- Enables Local Loopback mode and Continue on Read Error, and disables the channel interrupt.
- Sets the interrupt level and interrupt vector.
- Starts the data transfer.
- Verifies that no interrupt was received.
- Enables the channel interrupt.
- Waits for the interrupt to occur.
- Acknowledges the interrupt and verifies that the correct interrupt vector was returned.
- Verifies that the expected status is present and that no unexpected status was received.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Subtest c

This subtest verifies that the INT.STATUS CSR is not cleared by reading while an interrupt is asserted. An all A's data pattern is used as the interrupt vector value. Once started, the subtest performs the following operations:

- Enables Local Loopback mode and Continue on Read Error, and disables the channel interrupt.
- Sets the interrupt level and interrupt vector.
- Enables the channel interrupt.
- Starts the data transfer.
- Waits for the interrupt to occur.
- Verifies that the expected status was received and that no unexpected status is present.
- Verifies that reading does not clear the status.
- Acknowledges the interrupt and verifies that the correct interrupt vector was returned.
- Verifies that the expected status is present and that no unexpected status was received.
- Verifies that the status was cleared.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Subtest d

This subtest verifies that the VIC CSR is write protected while interrupts are enabled and an interrupt is asserted. An all A's data pattern is used as the interrupt vector value. Once started, the subtest performs the following operations:

- Enables Local Loopback mode and Continue on Read Error, and disables the channel interrupt.
- Sets the interrupt level and interrupt vector.
- Enables the channel interrupt.
- Attempts to change the interrupt level and vector.
- Verifies that the interrupt level and vector cannot be changed when an interrupt is enabled.
- Starts the data transfer.
- Waits for the interrupt to occur.
- Enables Local Loopback mode and Continue on read error, and disables the channel interrupt.
- Attempts to change the interrupt level and interrupt vector.
- Verifies that the interrupt level and vector cannot be changed when the channel interrupt is disabled and an interrupt is asserted.
- Acknowledges the interrupt and verifies that the correct interrupt vector was returned.
- Verifies that the expected status is present and that no unexpected status was received.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 43 DMA - Force ECC & Framing Errors (VMEBUS)

This test verifies the ability of the MCA to detect single and multiple bit ECC errors, and subblock count framing errors across the MCbus. The transmitter is programmed to generate various single and multiple bit ECC errors and the receiver's ECC checker is programmed to correct or not correct the ECC errors. In addition, the receiver is checked for proper error recovery by following the subblock in error with an error-free subblock. All subblocks are a maximum of eight doublewords. Each subtest uses specific addresses or data to force ECC errors. This test contains the following subtests:

Subtest	Subtest Name
a	DMA - Single Bit ECC Error & Correction (VMEBUS)
b	DMA - Frame Error via Subblk Count Received < Data Received (VMEBUS)
c	DMA - Frame Error via Subblk Count Received > Data Received (VMEBUS)
d	DMA - Frame Error via Zero Subblk Count Received (VMEBUS)
e	DMA - Address Phase Multi-Bit ECC Error (VMEBUS)
f	DMA - Data Phase Multi-Bit ECC Error on 1st DbWord (VMEBUS)
g	DMA - Data Phase Multi-Bit ECC Error on 4th DbWord (VMEBUS)
h	DMA - Data Phase Multi-Bit ECC Error on 8th DbWord (VMEBUS)

Subtest a

This subtest verifies that the MCA can detect and correct single-bit ECC errors. The transmitter is programmed to transmit eight doublewords with bit 50 on the MCbus inverted for both address and data phases. This causes the receiver to Sense the RX.ECC.ERR status for single-bit errors. The input buffer is checked for the corrected data. This subtest uses an address-in-address data pattern. Once started, the subtest performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.
- Initializes the input and output data buffers.
- Turns the MCA board online.
- Verifies that no unexpected status is present.
- Starts the data transfer.
- Waits for the data transfer to complete.
- Verifies that the expected status is present.
- Verifies that the SA counter bumped to the next doubleword address.

- Verifies the contents of the input and output data buffers.

Subtest b

This subtest verifies that the MCA can detect and recover from a frame error caused when the subblock count received is less than the quantity of data received. The transmitter is programmed to transmit two subblocks. A frame error is forced in the count field of the first subblock. The transfer is programmed for a count of seven doublewords, but the hardware forces the count to three doublewords. The second subblock is error-free to verify the receiver recovers from the error. Bit 50 on the MCbus is inverted for the address and data phases. The second subblock uses an address-in-address data pattern. The following data pattern is used for the first subblock:

1st doubleword	0x00040000	0x00000001
2nd doubleword	0x00000002	0x00000003
3rd doubleword	0x00040004	0x00000005
4th doubleword	0x00000006	0x00000007
5th doubleword	0x00040008	0x00000009
6th doubleword	0x0000000a	0x0000000b
7th doubleword	0x0004000c	0x0000000d

Once started, this subtest performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that no unexpected status is present.
- Turns the MCA board offline.
- Programs the MC.CTL CSR to force a frame error and disables RX ECC correction.
- Turns the MCA board online.
- Starts the data transfer.
- Waits for the error to occur.
- Verifies that a frame error occurred and that the expected status is present.
- Starts the second data transfer with no frame error.
- Waits for the data transfer to complete.
- Verifies that the expected status is present.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Subtest c

This subtest verifies that the MCA can detect and recover from a frame error caused when the subblock count received is greater than the quantity of data received. The transmitter is programmed to transmit three subblocks. A frame error is forced in the count field of the first subblock. The transfer is programmed for a count of three doublewords, but the hardware forces the count to seven doublewords. The second and third subblocks are error-free to verify the receiver recovers from the error. Bit 50 on the MCbus is inverted for the address and data phases. The second and third subblocks use an address-in-address data pattern. The data pattern used for the first subblock follows:

1st doubleword	0x00040000	0x00000001
2nd doubleword	0x00000002	0x00000003
3rd doubleword	0x00040004	0x00000005
4th doubleword	0x00000006	0x00000007
5th doubleword	0x00040008	0x00000009
6th doubleword	0x0000000a	0x0000000b
7th doubleword	0x0004000c	0x0000000d

Once started, this subtest performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that no unexpected status is present.
- Turns the MCA board offline.
- Programs the MC.CTL CSR to force a frame error and disables RX ECC correction.
- Turns the MCA board online.
- Starts the data transfer.
- Waits for the error to occur.
- Verifies that a frame error occurred and that the expected status is present.
- Starts the second data transfer.
- Waits for the error to occur.
- Verifies that a frame error occurred and that the expected status is present.
- Starts the third data transfer.
- Waits for the data transfer to complete.
- Verifies that the expected status is present.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Subtest d

This subtest verifies that the MCA can detect and recover from a frame error caused when a zero subblock count is received. The transmitter is programmed to transmit two subblocks. A frame error is forced in the count field of the first subblock. The transfer is programmed for a count of four doublewords, but the hardware forces the count to zero doublewords. The second subblock is error-free to verify the receiver recovers from the error. Bit 50 on the MCbus is inverted for the address and data phases. The second subblock uses an address-in-address data pattern. The data pattern used for the first subblock follows:

1st doubleword	0x00040000	0x00000001
2nd doubleword	0x00000002	0x00000003
3rd doubleword	0x00040004	0x00000005
4th doubleword	0x00000006	0x00000007
5th doubleword	0x00040008	0x00000009
6th doubleword	0x0000000a	0x0000000b
7th doubleword	0x0004000c	0x0000000d

Once started, this subtest performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that no unexpected status is present.
- Turns the MCA board offline.
- Programs the MC.CTL CSR to force a frame error and disables RX ECC correction.
- Turns the MCA board online.
- Starts the data transfer.
- Waits for the error to occur.
- Verifies that a frame error occurred and that the expected status is present.
- Starts the second data transfer with no frame error.
- Waits for the data transfer to complete.
- Verifies that the expected status is present.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Subtest e

This subtest verifies that the MCA can detect and recover from a multi-bit error occurring in the address phase. The transmitter is programmed to issue a DMA command that will produce two subblocks. The first subblock forces a multi-bit error in the address phase and the second subblock is error-free to verify the receiver recovers from the error. The receiver should detect RX.A.MBE status and Sense RX.ECC.ERR. The transmitter produces two subblocks by starting the Destination address at 0xff8. This causes a 4K byte address crossing which yields a new subblock. The multi-bit ECC error is forced by the hardware which zero-fills MCbus Bit 3 and inverts Bit 50. This subtest uses an address-in-address data pattern.

Once started, this subtest performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that no unexpected status is present.
- Turns the MCA board offline.
- Programs the MC.CTL CSR to force a multi-bit error.
- Turns the MCA board online.
- Starts the data transfer.
- Waits for the data transfer to complete.
- Verifies that a multi-bit error occurred and that the expected status is present.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Subtest f

This subtest verifies that the MCA can detect and recover from a multi-bit error occurring in the data phase. The transmitter is programmed to issue a DMA command that will produce two subblocks. The first subblock forces a multi-bit error on the first data doubleword. The second subblock is error-free to verify the receiver recovers from the error. The receiver should detect RX.D.MBE status and Sense RX.ECC.ERR. The transmitter produces two subblocks by starting the Destination address at 0xfc0. This causes a 4K byte address crossing which yields a new subblock. The multi-bit ECC error is forced by the hardware which zero-fills MCbus Bit 3 and inverts Bit 50. The following data pattern is used:

Doubleword			Comment	Address Offset
1st	0x00000000	0x00000008	force error	0xfc0
2nd	0x11111111	0x00000000		
3rd	0x22222222	0x00000000		

Doubleword			Comment	Address Offset
4th	0x33333333	0x00000000		
5th	0x44444444	0x00000000		
6th	0x55555555	0x00000000		
7th	0x66666666	0x00000000		
8th	0x77777777	0x00000000		
9th	0x88888888	0x00000000		0x1000
10th	0x99999999	0x00000000		
11th	0xaaaaaaaa	0x00000000		
12th	0xbbbbbbbb	0x00000000		
13th	0xcccccccc	0x00000000		
14th	0xdddddddd	0x00000000		
15th	0xeeeeeeee	0x00000000		
16th	0xffffffff	0x00000000		

Once started, this subtest performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that no unexpected status is present.
- Programs the MC.CTL CSR to force a multi-bit error in the first doubleword.
- Starts the data transfer.
- Waits for the transfer to complete.
- Verifies that a multi-bit error occurred in the data phase and that the expected status is present.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Subtest g

This subtest verifies that the MCA can detect and recover from a multi-bit error occurring in the data phase. The transmitter is programmed to issue a DMA command that will produce two subblocks. The first subblock forces a multi-bit error on the fourth data doubleword. The second subblock is error-free to verify the receiver recovers from the error. The receiver should detect RX.D.MBE status and Sense RX.ECC.ERR. The transmitter produces two subblocks by starting the Destination address at 0xfc0. This causes a 4K byte address crossing which yields a new subblock. The multi-bit ECC error is forced by the hardware which zero-fills MCbus Bit 3 and inverts Bit 50. The following data pattern is used:

Doubleword			Comment	Address Offset
1st	0x00000000	0x00000000		0xfc0
2nd	0x11111111	0x00000000		
3rd	0x22222222	0x00000000		
4th	0x33333333	0x00000008	force error	0xfd8
5th	0x44444444	0x00000000		
6th	0x55555555	0x00000000		
7th	0x66666666	0x00000000		
8th	0x77777777	0x00000000		
9th	0x88888888	0x00000000		0x1000
10th	0x99999999	0x00000000		
11th	0xaaaaaaaa	0x00000000		
12th	0xbbbbbbbb	0x00000000		
13th	0xcccccccc	0x00000000		
14th	0xdddddddd	0x00000000		
15th	0xeeeeeeee	0x00000000		
16th	0xffffffff	0x00000000		

Once started, this subtest performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that no unexpected status is present.
- Programs the MC.CTL CSR to force a multi-bit error in the fourth doubleword.
- Starts the data transfer.

- Waits for the transfer to complete.
- Verifies that a multi-bit error occurred in the data phase and that the expected status is present.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Subtest h

This subtest verifies that the MCA can detect and recover from a multi-bit error occurring in the data phase. The transmitter is programmed to issue a DMA command that will produce two subblocks. The first subblock forces a multi-bit error on the eighth data doubleword. The second subblock is error-free to verify the receiver recovers from the error. The receiver should detect RX.D.MBE status and Sense RX.ECC.ERR. The transmitter produces two subblocks by starting the Destination address at 0xfc0. This causes a 4K byte address crossing which yields a new subblock. The multi-bit ECC error is forced by the hardware which zero-fills MCbus Bit 3 and inverts Bit 50. The following data pattern is used:

Doubleword			Comment	Address Offset
1st	0x00000000	0x00000000		0xfc0
2nd	0x11111111	0x00000000		
3rd	0x22222222	0x00000000		
4th	0x33333333	0x00000000		
5th	0x44444444	0x00000000		
6th	0x55555555	0x00000000		
7th	0x66666666	0x00000000		
8th	0x77777777	0x00000008	force error	0xff8
9th	0x88888888	0x00000000		0x1000
10th	0x99999999	0x00000000		
11th	0xaaaaaaaa	0x00000000		
12th	0xbbbbbbbb	0x00000000		
13th	0xcccccccc	0x00000000		
14th	0xdddddddd	0x00000000		
15th	0xeeeeeeee	0x00000000		
16th	0xffffffff	0x00000000		

Once started, this subtest performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that no unexpected status is present.
- Programs the MC.CTL CSR to force a multi-bit error in the first doubleword.
- Starts the data transfer.
- Waits for the transfer to complete.
- Verifies that a multi-bit error occurred in the data phase and that the expected status is present.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 44 TMI Local Loopback... Control/Status Register

This test verifies the Transition Module Interface (TMI) Control/Status Register in TMI Local Loopback mode. The test contains two subtests.

Note – This test is applicable only to the TMI Local Loopback testing mode.

Subtest a

This subtest walks a bit through the Control/Status Register. It also tests the TMI lock semaphore.

Subtest b

This subtest verifies auto status reporting by the TMI.

Test 46 TMI Local Loopback... Programming to Operational Mode

This test verifies the ability of the Transition Module Interface (TMI) to change from Programming mode to Operational mode. The test then verifies the ability to write and read the Control/Status Register and Maps in Operational mode. This test contains the following subtests:

Subtest	Subtest Name
a	Enabled Loopback Mode
b	Writes & Reads Window RAM in Loopback Mode
c	Enabled Operational Mode
d	Writes & Reads Window RAM in Operational Mode

Note – This test is applicable only to the TMI Local Loopback testing mode.

Subtest a

This subtest verifies the ability to enable Loopback mode.

Subtest b

This subtest verifies the ability to write and read the Window RAM while in Loopback mode.

Subtest c

This subtest verifies the ability to enable Operational mode.

Subtest d

This subtest verifies the ability to write and read the Window RAM while in Operational mode.

Test 47 TMI Local Loopback... Data Transfer

This test verifies the ability of the Transition Module Interface (TMI) to transmit and receive various data patterns in TMI Local Loopback mode. The test contains the following subtests:

Subtest	Data Pattern	
a	Floating Zero	(0xfffffffe, 0xfffffffd)
b	A's & 5's	(0xaaaaaaaa, 0x55555555)
c	Ones & Zeros	(0xffffffff, 0x00000000)
d	Rotating	(0x12345678, 0x2468acf0)
e	Progression	(0x03020100, 0x07060504)
f	Incrementing	(0x00000000, 0x00000004)
g	Segment check	Verifies all 2048 segments
h	FIFO Full check	Verifies UFC and DFC status
i	Mailbox check	Verifies mailbox interrupt status

Note – This test is applicable only to the TMI Local Loopback testing mode.

Test 52 TMI/WSC - Single Data Transfer Verification (VMEbus)

This test verifies the ability of the Adapter to transmit and receive single data transfers.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

The test contains the following subtests:

Subtest	Subtest Name
a	Long Transfer (VMEbus)
b	Byte Transfer (VMEbus)

Subtest a uses a longword data transfer to verify basic transfer control. Subtest b uses a byte transfer to verify that the MEMORY CHANNEL Adapter receiver can perform a basic read-modify-write operation to EDRAM. Both subtests use a binary progression data pattern. Each subtest performs the following operations:

- Resets the MCA board.
- Reads and verifies the Interrupt Status CSR.
- Enables Local Loopback mode.
- Opens the RX and TX Windows for the first location in the memory test area.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Transfers the data and waits for reflection to complete.
- Checks the Interrupt Status and Sense CSRs for the expected status.
- Checks the TMI status.
- Verifies the contents of the input and output data buffers.

Tests 53-59 TMI/WSC - Data Pattern Tests

These tests verify the ability to transmit and receive various data patterns. Each test contains two subtests. The tests and subtests use different bus access types and data patterns.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

The following subtests are available:

Test	Subtest	Subtest Name
53	a	All Ones Data (LBUS)
	b	All Ones Data (VMEbus)
54	a	All Zeros Data (LBUS)
	b	All Zeros Data (VMEbus)
55	a	All 5's Data (LBUS)
	b	All 5's Data (VMEbus)
56	a	All A's Data (LBUS)
	b	All A's Data (VMEbus)
57	a	Address in Data (LBUS)
	b	Address in Data (VMEbus)

Test	Subtest	Subtest Name
58	a	Alternating Data (LBUS)
	b	Alternating Data (VMEbus)
59	a	Random Data (LBUS)
	b	Random Data (VMEbus)

Note – Subtest a is bypassed if you selected the VME ONLY bus configuration during program initialization.

For Test 57, the address used as data is the output buffer address.

For Test 58, alternating longwords of 0x00000000 and 0xfefefefe are used as data.

Each test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens TX and RX Windows for the test area.
- Turns the MCA board online.
- Initializes the input and output data buffers.

Each subtest performs the following operations:

- Writes the data pattern to the output buffer.
- Waits for the data transfer to complete.
- Checks the Interrupt Status and Sense CSRs for the expected status.
- Checks the TMI status.
- Verifies the contents of the input and output data buffers.

Test 60 TMI/WSC - TX/RX Windows Open/Close Verification (VMEBUS)

This test verifies that the Transmit and Receive Windows open and close properly. Windows 1 through 16 of the Translation Address are used during testing.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

This test contains the following subtests:

Subtest	Subtest Name
a	TX Open, RX Seg Wind 0 Open, Page Winds Open (VMEbus)
b	TX Close, RX Seg Wind 0 Open, Page Winds Open (VMEbus)
c	TX Open, RX Seg Wind 0 Open, Page Winds Close (VMEbus)
d	TX Open, RX Seg Wind 0 Close, Page Winds Open (VMEbus)

Each subtest writes one longword into the first location of each window for a total of 16 writes. Once started, the test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Turns the MCA board online.
- Verifies that the expected status is present.
- Initializes the input and output data buffers.

Subtest a

This subtest performs the following operations:

- Opens the first sixteen TX Windows.
- Opens the first sixteen RX Windows and RX Segment Window 0.
- Verifies that sixteen data longwords were transferred to the input buffer.

Subtest b

This subtest performs the following operations:

- Closes the first sixteen TX Windows.
- Verifies that no data was transferred to the input buffer.

Subtest c

This subtest performs the following operations:

- Opens the first sixteen TX Windows.
- Closes the RX Page Windows.
- Verifies that no data was transferred to the input buffer.

Subtest d

This subtest performs the following operations:

- Closes RX Segment Window 0.
- Opens the RX Page Windows.
- Verifies that no data was transferred to the input buffer.

Test 61 TMI/WSC - Longword Transfer using I/O buffer size equal to TA size

This test verifies that a very large data buffer can be reflected using longword addressing across the Local Bus and VMEbus. The I/O buffer size is determined by the test area (TA) size.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

The test contains the following subtests:

Subtest	Subtest Name
a	TMI/WSC - Longword Transfer (LBUS)
b	TMI/WSC - Longword Transfer (VMEbus)

Note – Subtest a is bypassed if you selected the VME ONLY bus configuration during program initialization.

Each subtest uses a byte binary progression data pattern and performs the following operations:

- Resets the MCA board.
- Verifies that the MCA board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate TX and RX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Waits for the data transfer to complete.
- Checks the Interrupt Status and Sense CSRs for the expected status.
- Checks the TMI status.
- Verifies the contents of the input and output data buffers.

Test 62 TMI/WSC - Word Transfer using I/O buffer size equal to TA size

This test verifies that a very large data buffer can be reflected using word addressing across the Local Bus and VMEbus. The I/O buffer size is determined by the test area (TA) size.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

The test contains the following subtests:

Subtest	Subtest Name
a	TMI/WSC - Word Transfer (LBUS)
b	TMI/WSC - Word Transfer (VMEbus)

Note – Subtest a is bypassed if you selected the VME ONLY bus configuration during program initialization.

Each subtest uses a byte binary progression data pattern and performs the following operations:

- Resets the MCA board.
- Verifies that the MCA board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate TX and RX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Waits for the data transfer to complete.
- Checks the Interrupt Status and Sense CSRs for the expected status.
- Checks the TMI status.
- Verifies the contents of the input and output data buffers.

Test 63 TMI/WSC - Byte Transfer using I/O buffer size equal to TA size

This test verifies that a very large data buffer can be reflected using byte addressing across the Local Bus and VMEbus. The I/O buffer size is determined by the test area (TA) size.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

The test contains the following subtests:

Subtest	Subtest Name
a	TMI/WSC - Byte Transfer (LBUS)
b	TMI/WSC - Byte Transfer (VMEbus)

Note – Subtest a is bypassed if you selected the VME ONLY bus configuration during program initialization.

Each subtest uses a byte binary progression data pattern and performs the following operations:

- Resets the MCA board.
- Verifies that the MCA board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate TX and RX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Waits for the data transfer to complete.
- Checks the Interrupt Status and Sense CSRs for the expected status.
- Checks the TMI status.
- Verifies the contents of the input and output data buffers.

Test 64 TMI/WSC - Reflecting From Multiple CPUs each using Address in Data

This test provides an increased write data rate for reflecting data by using four CPUs writing concurrently to four different output buffers. An Address-in-address data pattern is used for data transfers.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

The test contains the following subtests:

Subtest	Subtest Name
a	Reflecting from Multiple CPUs (Local Bus)
b	Reflecting from Multiple CPUs (VME Bus)

Note – Subtest a is bypassed if you selected the VME ONLY bus configuration during program initialization.

Once started, the test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate TX and RX Windows.

Each subtest performs the following operations:

- Initializes the input and output data buffers.
- Starts the data transfer.
- Waits for the write operation to complete.
- Verifies that the data transferred correctly.

Test 65 TMI/DMA - Stuck and Tied Data Lines Transfer (VMEbus)

This test verifies the ability of the MCA to transmit and receive four doubleword data patterns: all ones, all fives, all A's, all zeros. The test verifies the basic transfer control for transmit and receive hardware and verifies that no bits in the data path are stuck or tied.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

Once started, the test performs the following operations:

- Checks for any unexpected status.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes all TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that no unexpected Sense status is present.
- Starts the data transfer.
- Waits for the transfer to complete.
- Verifies that the correct status is received.
- Verifies that the SA counter is bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 66 TMI / DMA - Stuck and Tied Address Lines Single Transfer (VMEbus)

This test verifies the DMA Source and Destination address lines from the DMA address counters. While the Source address is used to read EDRAM, the Destination address is set to the TX FIFO and transmitted to the MCbus interface, onto the receiver, and back to the EDRAM receiver path used for writing data into the input buffer. Four different doubleword addresses are used to verify that there are no stuck or tied address lines. The Source and Destination addresses are the one's complement of each other. Therefore, a Source address of ones results in a Destination address of zeros. The test also verifies the ability to access the first and last doubleword memory addresses and to set and reset the V, L, and R bits in the Least Significant Destination Address (LSDA) CSR. In addition, the test verifies the DMACI field and DMA.CMPLT status in the Interrupt Status (INT.STATUS) CSR and the DMA.DONE and DMA.Q.DONE flags in the Byte Count (BC) CSR. The following Source and Destination addresses are used during testing: all ones, all zeros, all fives, all A's. The test uses a byte binary progression data pattern.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

Once started, the test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that no unexpected Sense status is present.
- Starts the data transfer.
- Waits for the transfer to complete.
- Checks the Interrupt Status and Sense CSRs for the expected DMACI value, DMA.CMPLT, and Online status.
- Checks the TMI status.
- Verifies that the SA counter is bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.
- Reinitializes the input and output buffers and proceeds with the next data transfer.

Test 67 TMI/DMA - Variable Transfer Count (VMEBUS)

This test verifies that the DMA can handle sixteen different transfer counts. The transfer counts are derived by passing a floating one pattern through the 16-bit byte count field which results in counts from one doubleword (eight bytes) to 32K doublewords (256K bytes). This test verifies the ability of the DMA to subdivide the overall byte count into smaller subblocks for transmission across the MCbus.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

Subblock sizes of 8, 16, or 32 are jumper-selectable on the ATB board.

The test uses a byte binary progression data pattern while executing the following subtests:

Subtest	Subtest Name		
a	TMI/DMA - Variable Transfer Count =	1	(VMEbus)
b	TMI/DMA - Variable Transfer Count =	2	(VMEbus)
c	TMI/DMA - Variable Transfer Count =	4	(VMEbus)
d	TMI/DMA - Variable Transfer Count =	8	(VMEbus)
e	TMI/DMA - Variable Transfer Count =	16	(VMEbus)
f	TMI/DMA - Variable Transfer Count =	32	(VMEbus)
g	TMI/DMA - Variable Transfer Count =	64	(VMEbus)
h	TMI/DMA - Variable Transfer Count =	128	(VMEbus)
i	TMI/DMA - Variable Transfer Count =	256	(VMEbus)
j	TMI/DMA - Variable Transfer Count =	512	(VMEbus)
k	TMI/DMA - Variable Transfer Count =	1K	(VMEbus)
l	TMI/DMA - Variable Transfer Count =	2K	(VMEbus)
m	TMI/DMA - Variable Transfer Count =	4K	(VMEbus)
n	TMI/DMA - Variable Transfer Count =	8K	(VMEbus)
o	TMI/DMA - Variable Transfer Count =	16K	(VMEbus)
p	TMI/DMA - Variable Transfer Count =	32K	(VMEbus)
q	TMI/DMA - Variable Transfer Count =	65528	(VMEbus)

Note – Subtest q uses the maximum transfer count of 64K minus one doubleword (`0x7FFF8`).

Examples:

- Transfer Count = 1, Byte Count = 0x00008
- Transfer Count = 2, Byte Count = 0x00010
- Transfer Count = 4, Byte Count = 0x00020
- Transfer Count = 32K, Byte Count = 0x40000

Once started, the test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Initializes the input and output data buffers.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the TX Windows.
- Turns the MCA board online.
- Verifies that the board is online and that no unexpected Sense status is present.

Each subtest performs the following operations:

- Starts the data transfer.
- Waits for the data transfer to complete.
- Verifies that the expected DMACI value, DMA.CMPLT, and online status is present.
- Checks the TMI status.
- Verifies that the SA counter is bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Tests 68 - 70 TMI/DMA - Data Pattern Tests (VMEBUS)

These tests verify the ability of the DMA to transmit different data patterns correctly.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

The following data patterns are used for testing:

Test	Data Pattern
68	All five's
69	All A's
70	Alternating

Note – The alternating data pattern uses alternating doublewords of all zeros and `0xfefefefe`.

Each test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that the Sense CSR indicates that the MCA is online and that no unexpected status is present.
- Starts the data transfer.
- Waits for the data transfer to complete.
- Verifies that the Interrupt Status and Sense CSRs contain the expected DMACI value, DMA.CMPLT status, and Online status.
- Checks the TMI status.
- Verifies that the SA counter is bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 71 TMI/DMA - Random Data & 64b ECC Sequenced (VMEBUS)

This test verifies the ability of the MEMORY CHANNEL Adapter to generate and check 256 combinations of ECC code. An output data buffer of 64-bit doublewords is used to cause the transmitter EDC chip to generate a 64-bit ECC code sequence from 0 to 0xff. This is also considered a random data pattern. A total of 8K bytes are transferred and the random pattern repeats every 256 doubleword transfers.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

This test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Verifies that no unexpected Sense status is present.
- Starts the data transfer.
- Waits for the data transfer to complete.
- Checks the Interrupt Status and Sense CSRs for the expected DMACI, DMA.CMPLT, and online status.
- Checks the TMI status.
- Verifies that the SA counter is bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

TABLE 3-3 lists the data patterns and ECC codes used during testing.

TABLE 3-3 Test 71 Data Patterns and ECC Codes

Data Pattern	ECC	Data Pattern	ECC
0x8ac83333 fff91fdc	0x00	0x3afb2d85 b02c1a2e	0x01
0x9ab3cdf0 0fe4ba99	0x02	0x332fe024 a860cccd	0x03
0x2e9e8f5c a3cf7c05	0x04	0x2b3192c6 a0627f6f	0x05
0x7f5ce93f f48dd5e8	0x06	0x82c9e5d5 f7fad27e	0x07
0x27b39631 9ce482da	0x08	0x16b4a741 8be593ea	0x09

TABLE 3-3 Test 71 Data Patterns and ECC Codes *(Continued)*

Data Pattern	ECC	Data Pattern	ECC
0x5b0562fc d0364fa5	0x0a	0xcde39abd 43148766	0x0b
0x554fbe02 ca80aaab	0x0c	0xa3b46f82 18e55c2b	0x0d
0xe384da74 58b5c71d	0x0e	0x83dd3a08 f90e26b1	0x0f
0xeca77c04 61d868ad	0x10	0xaa9f68ad 1fd05556	0x11
0x146bfedd 899ceb86	0x12	0x1f8248d6 94b3357f	0x13
0x46667779 bb976422	0x14	0x69cca987 defd9630	0x15
0x8ab73334 ffe81fdd	0x16	0xda7338e3 4fa4258c	0x17
0x369cdcba abcdc963	0x18	0x7e38950d f36981b6	0x19
0x2c33e6fa a164d3a3	0x1a	0x6417048d d947f136	0x1b
0x3e8a2a19 b3bb16c2	0x1c	0x220ef136 973fdddf	0x1d
0x9cfc7654 122d62fd	0x1e	0xe6bed70d 5befc3b6	0x1f
0xdee289ad 54137656	0x20	0xe13c3210 566d1eb9	0x21
0x10ff0247 862feef0	0x22	0x0ec759e2 83f8468b	0x23
0xfb6ec28f 709faf38	0x24	0x72cd4b19 e7fe37c2	0x25
0x541a69d1 c94b567a	0x26	0x906cd82e 059dc4d7	0x27
0x6cf5a621 e22692ca	0x28	0xb5f9b2a2 2b2a9f4b	0x29
0xadfb6544 232c51ed	0x2a	0x5d3d0b61 d26df80a	0x2b
0x17d8fb73 8d09e81c	0x2c	0x80813d71 f5b22a1a	0x2d
0x6f714e82 e4a23b2b	0x2e	0x431b7ae1 b84c678a	0x2f
0x7893f012 edc4dcbb	0x30	0x60aa07f7 d5daf4a0	0x31
0x5e615f93 d3924c3c	0x32	0xae0c6543 233d51ec	0x33
0x8739369f fc6a2348	0x34	0xa7326c17 1c6358c0	0x35
0xb3b10a3e 28e1f6e7	0x36	0x3d32d5ea b263c293	0x37
0x0c7eb17e 81af9e27	0x38	0x68a85555 ddd941fe	0x39
0x18fd4fa5 8e2e3c4e	0x3a	0x1b45f809 9076e4b2	0x3b
0xd93de4b2 4e6ed15b	0x3c	0x875b369d fc8c2346	0x3d
0x3b1d2d83 b04e1a2c	0x3e	0x07ed60b6 7d1e4d5f	0x3f
0x89a3df01 fed4cbaa	0x40	0x551cbe05 ca4daaae	0x41
0xc94149f6 3e72369f	0x42	0xb29db60b 27cea2b4	0x43
0x5c18b72f d149a3d8	0x44	0x28d7ea63 9e08d70c	0x45

TABLE 3-3 Test 71 Data Patterns and ECC Codes *(Continued)*

Data Pattern	ECC	Data Pattern	ECC
0x7d2540da f2562d83	0x46	0x11100246 8640eeef	0x47
0xfa4a6e5d 6f7b5b06	0x48	0x886e8ad0 fd9f7779	0x49
0xa2a11b4f 17d207f8	0x4a	0x61ce5c29 d6ff48d2	0x4b
0x0b5a5d4c 808b49f5	0x4c	0xf24c20ff 677d0da8	0x4d
0x6c0451ec e1353e95	0x4e	0xa18dc71c 16beb3c5	0x4f
0x1c6a4c3b 919b38e4	0x50	0x3fae7e4b b4df6af4	0x51
0xba8b036a 2fbbf013	0x52	0x23334568 98643211	0x53
0x0febae14 851c9abd	0x54	0xfdb76af3 72e8579c	0x55
0x59d00ecb cf00fb74	0x56	0x59e10eca cf11fb73	0x57
0xc71aa190 3c4b8e39	0x58	0x6d39a61d e26a92c6	0x59
0xf003789b 65346544	0x5a	0x7e49950c f37a81b5	0x5b
0x1a21a3d7 8f529080	0x5c	0xd274eb85 47a5d82e	0x5d
0xc91f49f8 3e5036a1	0x5e	0xd4bd93e9 49ee8092	0x5f
0x1b23f80b 9054e4b4	0x60	0xf3707531 68a161da	0x61
0x67730124 dca3edcd	0x62	0x3556888a aa877533	0x63
0x540969d2 c93a567b	0x64	0x67840123 dcb4edcc	0x65
0xedbad037 62ebbce0	0x66	0xd1509753 468183fc	0x67
0xbbaf579c 30e04445	0x68	0xbde70001 3317ecaa	0x69
0xe12b3211 565c1eba	0x6a	0xdef389ac 54247655	0x6b
0x443fcf13 b970bbbc	0x6c	0x916f2c62 06a0190b	0x6d
0x38c38520 adf471c9	0x6e	0x3545888b aa767534	0x6f
0xfedbbf25 740cabce	0x70	0x377d30f0 acae1d99	0x71
0x9849258e 0d7a1237	0x72	0x34543456 a98520ff	0x73
0x1d8ea06d 92bf8d16	0x74	0x50ad6d3b c5de59e4	0x75
0x81a591a3 f6d67e4c	0x76	0x30d637c1 a607246a	0x77
0x9f451eb8 14760b61	0x78	0x4f9a1908 c4cb05b1	0x79
0xc5f64d5e 3b273a07	0x7a	0xf22a2101 675b0daa	0x7b
0x819491a4 f6c57e4d	0x7c	0x5ae362fe d0144fa7	0x7d
0xd94ee4b1 4e7fd15a	0x7e	0xb3c20a3d 28f2f6e6	0x7f
0x6d28a61e e25992c7	0x80	0xf7f0c5fa 6d21b2a3	0x81

TABLE 3-3 Test 71 Data Patterns and ECC Codes *(Continued)*

Data Pattern	ECC	Data Pattern	ECC
0x1fb548d3 94e6357c	0x82	0xb13561dd 26664e86	0x83
0xeb6127d4 6092147d	0x84	0x23444567 98753210	0x85
0xd4ac93ea 49dd8093	0x86	0x93c8d4c5 08f9c16e	0x87
0xeb7227d3 60a3147c	0x8e	0x1590530f 8ac13fb8	0x8f
0xd02c4321 455d2fca	0x90	0xeb8327d2 60b4147b	0x91
0x47accba9 bcddb852	0x92	0x58abba99 cddca742	0x93
0x2c55e6f8 a186d3a1	0x94	0xa2b21b4e 17e307f7	0x95
0xa15ac71f 168bb3c8	0x96	0x0237bbbc 7768a865	0x97
0x7084a2b5 e5b58f5e	0x98	0xca659e28 3f968ad1	0x99
0x368bdcbb abbcc964	0x9a	0xc709a191 3c3a8e3a	0x9b
0x034b0fef 787bfc98	0x9c	0xc83ef5c2 3d6fe26b	0x9d
0x257bedcc 9aacda75	0x9e	0x8e352fc9 03661c72	0x9f
0xe9297f6f 5e5a6c18	0xa0	0x4d5170a4 c2825d4d	0xa1
0xb5e8b2a3 2b199f4c	0xa2	0xabb2bce0 20e3a989	0xa3
0x8bec8765 011d740e	0xa4	0xc5e54d5f 3b163a08	0xa5
0xefd0789e 65016547	0xa6	0xc27850c9 37a93d72	0xa7
0x69aaa989 dedb9632	0x88	0x8f5983fb 048a70a4	0x89
0xda2f38e7 4f602590	0x8a	0x9e20ca86 1351b72f	0x8b
0xea5ed3a0 5f8fc049	0x8c	0x56741234 cba4fedd	0x8d
0x1fd748d1 9508357a	0xa8	0xfc7116c3 71a2036c	0xa9
0xf6dd71c7 6c0e5e70	0xaa	0x1d7da06e 92ae8d17	0xab
0x12345678 87654321	0xac	0x38d4851f ae0571c8	0xad
0xd91be4b4 4e4cd15d	0xae	0xd5e1e81b 4b12d4c4	0xaf
0x3c4181b5 b1726e5e	0xb0	0x91a22c5f 06d31908	0xb1
0xa05872eb 15895f94	0xb2	0xb0110dab 2541fa54	0xb3
0x0a36091a 7f66f5c3	0xb4	0x0a25091b 7f55f5c4	0xb5
0x4e53c4d8 c384b181	0xb6	0x24689999 99998642	0xb7
0xa06972ea 159a5f93	0xb8	0x4e31c4da c362b183	0xb9
0xe8162b3c 5d4717e5	0xba	0x48d11fdb be020c84	0xbb
0x45532346 ba840fef	0xbc	0xa5ec17e7 1b1d0490	0xbd

TABLE 3-3 Test 71 Data Patterns and ECC Codes *(Continued)*

Data Pattern	ECC	Data Pattern	ECC
0x1eb2f49f 93e3e148	0xbe	0xb8315b07 2d6247b0	0xbf
0x2a0d3e94 9f3e2b3d	0xc0	0x8f3783fd 046870a6	0xc1
0xf494c963 69c5b60c	0xc2	0x41f726af b7281358	0xc3
0x764b47ae eb7c3457	0xc4	0xe4a92ea6 59da1b4f	0xc5
0xf9261a2b 6e5706d4	0xc6	0xc4d1f92c 3a02e5d5	0xc7
0x28e8ea62 9e19d70b	0xc8	0xcb9af259 40cbdf02	0xc9
0x2c44e6f9 a175d3a2	0xca	0x0593b853 7ac4a4fc	0xcb
0x776f9be0 eca08889	0xcc	0x4655777a bb866423	0xcd
0x221ff135 9750ddde	0xce	0x035c0fee 788cfc97	0xcf
0x83ee3a07 f91f26b0	0xd0	0x887f8acf fdb07778	0xd1
0x05a4b852 7ad5a4fb	0xd2	0xb4e65e6f 2a174b18	0xd3
0x94fe28f6 0a2f159f	0xd4	0xf127cccd 6658b976	0xd5
0xf801c5f9 6d32b2a2	0xd6	0x73f19f4b e9228bf4	0xd7
0x58bcba98 cdeda741	0xd8	0x7f6de93e f49ed5e7	0xd9
0x7095a2b4 e5c68f5d	0xda	0x1c594c3c 918a38e5	0xdb
0xbcd3abce 32049877	0xdc	0xbf1c5432 344d40db	0xdd
0x56631235 cb93fede	0xde	0x6c1551eb e1463e94	0xdf
0x6af0fdb9 e021ea62	0xe0	0xb0330da9 2563fa52	0xe1
0x35788888 aaa97531	0xe2	0x62f2b05b d8239d04	0xe3
0x57986666 ccc9530f	0xe4	0xaf30b975 2461a61e	0xe5
0x986b258c 0d9c1235	0xe6	0x12235679 87544322	0xe7
0xace81111 2218fdba	0xe8	0xb71e06d4 2c4ef37d	0xe9
0xc02fa865 3560950e	0xea	0x905bd82f 058cc4d8	0xeb
0x542b69d0 c95c5679	0xec	0x804e3d74 f57f2a1d	0xed
0xc39ca4fb 38cd91a4	0xee	0xe5cd82d8 5afe6f81	0xef
0xb60ab2a1 2b3b9f4a	0xf0	0x57876667 ccb85310	0xf1
0xa3d66f80 19075c29	0xf2	0xd7063c4d 4c3728f6	0xf3
0xfc9316c1 71c4036a	0xf4	0x1d6ca06f 929d8d18	0xf5
0xddbe357b 52ef2224	0xf6	0x45642345 ba950fee	0xf7
0x1b34f80a 9065e4b3	0xf8	0xd13f9754 467083fd	0xf9

TABLE 3-3 Test 71 Data Patterns and ECC Codes *(Continued)*

Data Pattern	ECC	Data Pattern	ECC
0x37b030ed ace11d96	0xfa	0x96227d28 0b5369d1	0xfb
0xddad357c 52de2225	0xfc	0x0c4bb181 817c9e2a	0xfd
0x907dd82d 05aec4d6	0xfe	0x9746d15a 0c77be03	0xff

Test 72 TMI/DMA - DMA Throttled by TX & RX FIFOs (VMEbus)

This test verifies the throttling control of the DMA engine. The first part of the test verifies that the half-full state of the TX FIFO stops DMA from reading EDRAM at a precise transfer count. The second half of the test verifies that the RX FIFO half-full state stops DMA activity and also verifies that a back-to-back burst subblock can be transferred correctly. This part of the test causes concurrent operations between DMA and the receiver arbitrating for the Mezzanine port. This test also verifies the DMACI field and DMA.CMPLT status in the Interrupt Status (INT.STATUS) CSR, the DMA.DONE flag in the Byte Count (BC) CSR, and that the Source Address Error (SAE) CSR reflects the correct Source address.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

The Multiple Bus Grant Request bit (MCMULT) in the MC Control CSR is exercised by the MCbus interface when the TX FIFO is half-full.

The test uses an Address-in-address data pattern and performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Checks for unexpected Sense status.
- Disables the TX and RX FIFOs.

The first part of the test verifies that DMA stops when the TX FIFO is half-full by performing the following operations:

- Starts the data transfer.
- Waits for a DMA Stall condition.
- Verifies that the Interrupt Status and Sense CSRs report the correct status.
- Calculates the expected Source address and verifies that the Source Address Error (SAE) CSR contains the correct address.

The second part of the test verifies that DMA stops when the RX FIFO is half-full and also verifies that a back-to-back burst subblock transfers correctly. The following operations are performed:

- Enables Local Loopback mode.
- Enables the TX FIFO and disables the RX FIFO.
- Waits for the RX FIFO to reach the half-full state.
- Enables Local Loopback mode.
- Enables the TX and RX FIFOs.
- Waits for the data transfer to complete.
- Verifies that the Interrupt Status and Sense CSRs report the correct status.
- Checks the TMI status.
- Verifies that the SA counter is bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 73 TMI/DMA - Source & Destination Address Cross 4K Boundary (VMEBUS)

This test verifies that the DMA can detect its Source and Destination addresses crossing a 4K boundary and properly terminate a burst read and restart a new burst read subblock.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

The test contains the following subtests:

Subtest	Subtest Name
a	TMI/DMA - Source & Destination Address Cross 4K Boundary 1-1-5 (VMEbus)
b	TMI/DMA - Source & Destination Address Cross 4K Boundary 2-1-4 (VMEbus)
c	TMI/DMA - Source & Destination Address Cross 4K Boundary 4-3 (VMEbus)
d	TMI/DMA - Source & Destination Address Cross 4K Boundary 5-1-1 (VMEbus)

Each subtest verifies that the hardware can handle different combinations of boundary crossings. Every subtest except c forces the Source address to cross, and then forces the Destination address to cross. Subtest c forces both the Source and Destination addresses to cross at the same point.

The sequence number appended to the subtest name describes how the DMA's overall byte count is divided into subblocks. For example, 1-1-5 is appended to Subtest a. This indicates a division of three subblocks with an overall byte count of seven. Using seven for the byte count allows the same subblock division to be used for 8, 16, and 32 subblock selections. The 1-1-5 value is further defined as follows:

1	The first subblock has one doubleword terminated by the Source address crossing.
1	The second subblock has one doubleword terminated by the Destination address crossing.
5	The third subblock has five doublewords terminated when the DMA byte count decrements to zero.

Once started, the test performs the following operations:

- Resets the MCA board.
- Verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers. The test uses an Address-in-address data pattern.
- Checks for unexpected Sense status.

Each subtest performs the following operations:

- Starts the data transfer.
- Waits for the data transfer to complete.
- Verifies that the Interrupt Status and Sense CSRs report the correct status.
- Checks the TMI Status.
- Verifies that the SA counter is bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 74 TMI/DMA - Receiver Closes Odd Page Window Addresses (VMEbus)

This test verifies that the receiver can handle DMA burst transfers and accept or purge the data depending on the state of the Receive Window RAM's open or close bit. Occasionally, the DMA will contend with the receiver for the EDRAM mezzanine port. For example, when a Receive Window is closed, the receive Buffer FIFO purges the data which idles the receiver FIFO. The DMA is issued a single command with a byte count of 0x40000 (32,768 doublewords). Transfers start at the beginning of the memory test area, which is an even page Window. The receiver is programmed with only even Window addresses.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

Once started, the test performs the following operations:

- Resets the MCA board and verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Closes the appropriate TX Windows.
- Turns the MCA board online.

- Initializes the input and output data buffers. The test uses an Address-in-address data pattern.
- Checks for unexpected Sense status.
- Issues a DMA command.
- Verifies that the command executed successfully.
- Verifies that the Source address advanced by four doublewords.
- Verifies the contents of the input and output data buffers.

Test 75 TMI/DMA - Access EDRAM Memory while DMA Transfer is in Progress

This test verifies that the EDRAM memory array can be accessed concurrently by both the MCA mezzanine port and the Local Bus or VMEbus ports. The DMA is issued a single command with a byte count of 0x40000 (32,768 doublewords) to keep the mezzanine port busy, while the test performs CPU writes and reads to an unused Page Window of the memory test area. The CPU writes cause the MCA to handle Snoop traffic concurrently with the DMA and receiver activities. The Snoop transfer is purged because all the Transmit Windows are closed. The CPU writes and reads cause the EDRAM port arbiter to handle two ports concurrently.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

This test contains the following subtests:

Subtest	Subtest Name
a	TMI/DMA - Access EDRAM Memory while DMA Transfer is in Progress (LBUS)
b	TMI/DMA - Access EDRAM Memory while DMA Transfer is in Progress (VMEbus)

Note – Subtest a is bypassed if you selected the VME ONLY testing configuration during program initialization.

During Subtest a, the CPU is used to access EDRAM across the Local Bus while the mezzanine port is busy with the MCA DMA engine and MCA receiver. During Subtest b, the CPU is used to access EDRAM across the VMEbus while the mezzanine port is busy with the MCA DMA engine and MCA receiver. Each subtest uses a byte binary progression data pattern to perform the following operations:

- Resets the MCA board and verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the even RX Windows and closes the odd RX Windows.
- Closes all the TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Executes the DMA command.
- While the DMA transfer is in progress, the CPU performs the following operations:
  - Writes one 4K byte Page Window in an unused test area.
  - Reads the data and compares it for errors.
  - If the DMA transfer is not complete, the CPU continues to write and read the same page with a different data pattern until either the data transfer is complete or a timeout error occurs.
- Verifies that the correct status and Sense values are present.
- Verifies that the correct Source Address Error value is present.
- Verifies the contents of the input and output data buffers.

Test 76 TMI/DMA - Access Window RAMs while DMA Transfer is in Progress (VMEbus)

This test verifies that the MCA can handle concurrent activities of DMA read EDRAM, receiver write EDRAM, and TX and RX Translation Window RAM CSR accesses. The test causes sufficient contention to momentarily suspend the DMA and the receiver while the Window RAM is accessed. When the DMA is inactive, the mezzanine STALL function is exercised whenever the Window RAM is accessed.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

The test uses a byte binary progression data pattern and performs the following operations:

- Resets the MCA board and verifies that the board reset correctly.
- Enables Local Loopback mode.
- Opens the appropriate RX Windows.

- Closes all the TX Windows.
- Turns the MCA board online.
- Initializes the input and output data buffers.
- Checks for unexpected Sense status.
- Writes the RAM address into the TWRA, RWSRA and RWPRA. The RAM address must point outside the memory test area.
- Executes the DMA command.
- While the DMA transfer is in progress, the test loops on RAM accesses.
- Verifies that the DMA command executed successfully.
- Verifies that the correct status and Sense values are present.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 77 TMI/WSC/DMA - Alternating WSC & DMA Subblock Transfers (VMEBUS)

This test verifies the ability of the MCA to transmit and receive alternating WSC (Snoop) and DMA (burst) transfers. The transmit FIFO is disabled and alternately filled with a longword Snoop transfer and a DMA burst transfer of three doublewords. These commands are issued 64 times to half-fill the transmit FIFO. The FIFO is then enabled causing the transmission of back-to-back single and burst transfers through the MCbus interface and back through the receiver into the EDRAM input buffer.

The WSC I/O buffers are contiguous and are allocated to the first Window Page of the memory test area. The DMA I/O buffers are also contiguous and are allocated to the second Window Page of the memory test area. WSC transfers use an all ones data pattern and DMA transfers use a byte binary progression data pattern.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

This test does not verify simultaneous operation of the DMA, Snoop transfer, and receiver at the mezzanine interface. This feature is checked in other tests.

Once started, this test performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Opens the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input data buffer.

- Initializes the output data buffer for DMA only.
- Checks for unexpected Sense status.
- Disables the TX FIFO.
- Repeats the following steps 64 times to create 64 WSC sequential memory writes alternating with 64 DMA burst transfers:
  - Issues a WSC Snoop transfer.
  - Issues a DMA burst transfer.
- Enables the TX FIFO.
- Waits for the DMA input data transfer to complete.
- Verifies that the correct status and Sense values were received.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 78 TMI/WSC/DMA - Byte Transfer Concurrent with DMA Transfer

This test verifies the ability of the MCA to transmit and receive WSC (Snoop) byte transfers concurrent with DMA (burst) transfers. The test verifies the simultaneous operation of the DMA, Snoop transfer, and receiver at the mezzanine interface.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

This test contains the following subtests:

Subtest	Subtest Name
a	TMI/WSC/DMA - Byte Transfer Concurrent with DMA Transfer (LBUS)
b	TMI/WSC/DMA - Byte Transfer Concurrent with DMA Transfer (VMEbus)

Note – Subtest a is bypassed if you selected the VME ONLY testing configuration during program initialization.

Subtest a performs WSC writes across the Local Bus port and Subtest b performs WSC writes across the VMEbus port. Each port provides different WSC write rates to the MCA.

The WSC I/O buffer is a 4K byte contiguous buffer allocated to the first Window Page of the memory test area. The DMA I/O buffer is a 32768 doubleword contiguous buffer starting at the fifth Window Page of the memory test area. Buffers are selected to cause the WSC write transfer period to overlap the DMA transfer period. WSC transfers use an all ones data pattern and DMA transfers use a byte binary progression data pattern.

Once started, each subtest performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Opens the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input data buffer.
- Initializes the output data buffer for DMA only.
- Checks for unexpected Sense status.
- Issues a DMA burst transfer.
- Issues a WSC Snoop transfer.
- Waits for the DMA input data transfer to complete.
- Verifies that the correct status and Sense values were received.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 79 TMI/WSC/DMA - Longword Transfer Concurrent with DMA Transfer

This test verifies the ability of the MCA to transmit and receive WSC (Snoop) longword transfers concurrent with DMA (burst) transfers. The test verifies the simultaneous operation of the DMA, Snoop transfer, and receiver at the mezzanine interface.

Note – Data is transmitted to the TMI across the MC Bus. The data passes through the TMI which is in either internal (TMI Local Loopback) or remote (TMI Remote Loopback) mode. The TMI then returns the data to the MEMORY CHANNEL IV across the MC Bus.

This test contains the following subtests:

Subtest	Subtest Name
a	TMI/WSC/DMA - Longword Transfer Concurrent with DMA Transfer (LBUS)
b	TMI/WSC/DMA - Longword Transfer Concurrent with DMA Transfer(VMEbus)

Note – Subtest a is bypassed if you selected the VME ONLY testing configuration during program initialization.

Subtest a performs WSC writes across the Local Bus port and Subtest b performs WSC writes across the VMEbus port. Each port provides different WSC write rates to the MCA.

The WSC I/O buffer is a 16K byte contiguous buffer allocated to the first four Window Pages of the memory test area. The DMA I/O buffer is a 32768 doubleword contiguous buffer starting at the fifth Window Page of the memory test area. Buffers are selected to cause the WSC write transfer period to overlap the DMA transfer period. WSC transfers use an all ones data pattern and DMA transfers use a byte binary progression data pattern.

Once started, each subtest performs the following operations:

- Enables Local Loopback mode.
- Opens the appropriate RX Windows.
- Opens the appropriate TX Windows.
- Turns the MCA board online.
- Initializes the input data buffer.
- Initializes the output data buffer for DMA only.
- Checks for unexpected Sense status.
- Issues a DMA burst transfer.
- Issues a WSC Snoop transfer.
- Waits for the DMA input data transfer to complete.
- Verifies that the correct status and Sense values were received.
- Verifies that the SA counter bumped to the next doubleword address.
- Verifies the contents of the input and output data buffers.

Test 90 NR Node Test

This test verifies the ability of the node to receive, execute, and acknowledge commands from the host. The node executes this test while the host executes the Host in Remote (HR) tests.

Note – If a TMI links the host to the node, data and status will pass through that TMI.

This test is applicable only to the Node in Remote (NR) testing mode.

If you selected the VME ONLY bus configuration during program initialization, this test uses the VME Bus. If not, the test uses the Local Bus.

This test performs the following operations:

- Clears the MCA board of any previous error conditions.
- Initializes and verifies the register map.
- Opens the appropriate Receive Windows.
- Waits in an idle state for a command from the host.
- Executes the command and sends a response to the host.
- Restarts by waiting in an idle state for a command from the host.

Test 91 HR WSC - Data Pattern - Address in Data

This test verifies that an address-in-address data pattern can be sent in long transfers to all nodes both sequentially and concurrently. The test writes and reads across either the Local Bus or the VMEbus using WSC data transfers.

Note – If a TMI links the host to the node, data and status will pass through that TMI.

This test contains the following subtests:

Subtest	Subtest Name
a	WSC - Sequential Node Operation
b	WSC - Concurrent Node Operation

Note – This test is applicable only to the Host in Remote (HR) testing mode.

If you selected the VME ONLY bus configuration during program initialization, the test writes and reads across the VMEbus. If not, the Local Bus is used.

This test performs the following operations:

- Clears the MCA board of any previous error conditions.
- Initializes and verifies the register map.
- Opens the appropriate Receive Windows.

Subtest a - Sequential Node Operation

This subtest performs the following operations:

- Fills the host's input buffer with a complemented address-in-address data pattern.
- Sends commands to each node to fill the node's input buffer with a complemented address-in-address data pattern.
- Fills the host's output buffer with an address-in-address data pattern.
- Sends commands to each node to verify that the address-in-address data pattern is received by the node.
- Sends commands to each node to fill the node's output buffer with the address-in-address data pattern.
- Verifies that the address-in-address data pattern is received by the host.

Subtest b - Concurrent Node Operation

This subtest performs the following operations:

- Fills the host's input buffer with a complemented address-in-address data pattern.
- Sends commands to each node to fill the node's input buffer with a complemented address-in-address data pattern.
- Sends commands to each node to fill the node's output buffer with the address-in-address data pattern.
- Starts transferring data to the host concurrently.
- Fills the host's output buffer with the address-in-address data pattern.
- Sends commands to each node to verify that the node received the address-in-address data pattern from the host and from all other nodes.
- Verifies that the host received the address-in-address data pattern from all the nodes.

Test 92 HR WSC - Data Pattern - Alternating Data

This test verifies that an alternating data pattern can be sent in long transfers to all nodes both sequentially and concurrently. The test writes and reads across either the Local Bus or the VMEbus using WSC data transfers.

Note – If a TMI links the host to the node, data and status will pass through that TMI.

The following data pattern is repeated during testing:

```
0x00000000
0x00000000
0xFEFEFEFE
0xFEFEFEFE
```

The test contains the following subtests:

Subtest	Subtest Name
a	Sequential Node Operation
b	Concurrent Node Operation

Note – This test is applicable only to the Host in Remote (HR) testing mode.

If you selected the VME ONLY bus configuration during program initialization, the test writes and reads across the VMEbus. If not, the Local Bus is used.

This test performs the following operations:

- Clears the MCA board of any previous error conditions.
- Initializes and verifies the register map.
- Opens the appropriate Receive Windows.

Subtest a - Sequential Node Operation

This subtest performs the following operations:

- Fills the host's input buffer with a complemented alternating data pattern.
- Sends commands to each node to fill the node's input buffer with a complemented alternating data pattern.
- Fills the host's output buffer with an alternating data pattern.
- Sends commands to each node to verify that the alternating data pattern is received by the node.
- Sends commands to each node to fill the node's output buffer with the alternating data pattern.
- Verifies that the alternating data pattern is received by the host.

Subtest b - Concurrent Node Operation

This subtest performs the following operations:

- Fills the host's input buffer with a complemented alternating data pattern.
- Sends commands to each node to fill the node's input buffer with a complemented alternating data pattern.
- Sends commands to the node to fill the node's output buffer with the alternating data pattern.
- Starts transferring data to the host concurrently.
- Fills the host's output buffer with the alternating data pattern.
- Sends commands to each node to verify that the node received the alternating data pattern from the host and from all other nodes.
- Verifies that the host received the alternating data pattern from all the nodes.

Test 93 HR WSC - VL Transfers (Bus Valid Inhibit & Local Semaphore)

This test uses WSC longword data transfers to verify the operation of the MCbus Bus Valid and Local Semaphore signals. Setting and resetting the VL bits in the TX_trans RAM software enables and disables the MCbus Valid and Local Semaphore signals that are generated by the transmitter hardware. This test allocates four pages of memory. Each page is programmed with a different VL bit combination in the TX_trans RAM. The test interleaves memory writes to each page on a rotating basis to cause alternating VL transfers to the MCbus. Alternating VL transfers perform a thorough test of the hardware timing. The memory write sequence begins at longword address 0 in relative page 0, and is followed by longword address 0 in relative pages 1, 2, and 3. This is followed by longword address 4 in relative pages 0, 1, 2, and 3. The process continues until all 2048 longword writes have been performed in all four pages. The test will check all four pages of the receive buffers for either the expected data or no data if writing was inhibited. For example, a VL setting of 0x10 inhibits data transmission to either the Source or Destination nodes.

TABLE 3-4 lists the data acceptance criteria for the Source and Destination nodes based on the setting of the VL bits.

TABLE 3-4 Test 93 Data Acceptance Criteria for Source and Destination Nodes

VL Bits	Destination Node Receive Data?	Source Node Receive Data?
00	yes	no
01	yes	yes
10	no	no
11	no	yes

Note – If a TMI links the host to the node, data and status will pass through that TMI.

This test uses an address-in-address data pattern. The test contains the following subtests:

Subtest	Subtest Name
a	WSC - Sequential Node Operation
b	WSC - Concurrent Node Operation

Note – This test is applicable only to the Host in Remote (HR) testing mode.

If you selected the VME ONLY bus configuration during program initialization, the test writes and reads across the VMEbus. If not, the Local Bus is used.

This test performs the following operations:

- Clears the MCA board of any previous error conditions.
- Initializes and verifies the register map.
- Opens the appropriate Receive Windows.

Subtest a - Sequential Node Operation

This subtest performs the following operations:

- Fills the host's input buffer with the complement of the data pattern.
- Sends commands to each node to fill the node's input buffer with the complemented data pattern.
- Fills the host's output buffer with the data pattern.
- Sends commands to each node to verify that the data pattern is received by the node.
- Sends commands to each node to fill the node's output buffer with the data pattern.
- Verifies that the data pattern is received by the host.

Subtest b - Concurrent Node Operation

This subtest performs the following operations:

- Fills the host's input buffer with the complement of the data pattern.
- Sends commands to each node to fill the node's input buffer with the complemented data pattern.
- Sends commands to each node to fill the node's output buffer with the data pattern.
- Starts transferring data to the host concurrently.
- Fills the host's output buffer with the data pattern.
- Sends commands to each node to verify that the node received the data pattern from the host and from all other nodes.
- Verifies that the host received the data pattern from all the nodes.

Test 94 HR DMA - Stuck and Tied Data Lines Transfer

This test verifies the ability of the host to transmit and receive four doubleword data patterns: all ones, all fives, all A's, all zeros. The test verifies the basic transfer control for transmit and receive hardware and verifies that no bits in the data path are stuck or tied. Data patterns are sent to all nodes both sequentially and concurrently. The test writes and reads across either the Local Bus or the VMEbus using DMA data transfers.

Note – If a TMI links the host to the node, data and status will pass through that TMI.

This test contains the following subtests:

Subtest	Subtest Name
a	DMA - Sequential Node Operation
b	DMA - Concurrent Node Operation

Note – This test is applicable only to the Host in Remote (HR) testing mode.

If you selected the VME ONLY bus configuration during program initialization, the test writes and reads across the VMEbus. If not, the Local Bus is used.

This test performs the following operations:

- Clears the MCA board of any previous error conditions.
- Initializes and verifies the register map.
- Opens the appropriate Receive Windows.

Subtest a - Sequential Node Operation

This subtest performs the following operations:

- Fills the host's input buffer with a complemented data pattern.
- Sends commands to each node to fill the node's input buffer with a complemented data pattern.
- Fills the host's output buffer with the data pattern.
- Sends commands to each node to verify that the data pattern is received by the node.
- Sends commands to each node to fill the node's output buffer with the data pattern.

- Verifies that the data pattern is received by the host.

Subtest b - Concurrent Node Operation

This subtest performs the following operations:

- Fills the host's input buffer with a complemented data pattern.
- Sends commands to each node to fill the node's input buffer with a complemented data pattern.
- Sends commands to each node to fill the node's output buffer with the data pattern.
- Starts transferring data to the host concurrently.
- Fills the host's output buffer with the data pattern.
- Sends commands to each node to verify that the node received the data pattern from the host and from all other nodes.
- Verifies that the host received the data pattern from all the nodes.

Test 95 HR DMA - Data Pattern - Address in Data

This test verifies that the host can transmit and receive an address-in-address data pattern to and from all nodes both sequentially and concurrently. The test writes and reads across either the Local Bus or the VMEbus using DMA data transfers.

Note – If a TMI links the host to the node, data and status will pass through that TMI.

This test contains the following subtests:

Subtest	Subtest Name
a	DMA - Sequential Node Operation
b	DMA - Concurrent Node Operation

Note – This test is applicable only to the Host in Remote (HR) testing mode.

If you selected the VME ONLY bus configuration during program initialization, the test writes and reads across the VMEbus. If not, the Local Bus is used.

This test performs the following operations:

- Clears the MCA board of any previous error conditions.
- Initializes and verifies the register map.
- Opens the appropriate Receive Windows.

Subtest a - Sequential Node Operation

This subtest performs the following operations:

- Fills the host's input buffer with a complemented address-in-address data pattern.
- Sends commands to each node to fill the node's input buffer with a complemented address-in-address data pattern.
- Fills the host's output buffer with an address-in-address data pattern.
- Sends commands to each node to verify that the address-in-address data pattern is received by the node.
- Sends commands to each node to fill the node's output buffer with the address-in-address data pattern.
- Verifies that the address-in-address data pattern is received by the host.

Subtest b - Concurrent Node Operation

This subtest performs the following operations:

- Fills the host's input buffer with a complemented address-in-address data pattern.
- Sends commands to each node to fill the node's input buffer with a complemented address-in-address data pattern.
- Sends commands to each node to fill the node's output buffer with the address-in-address data pattern.
- Starts transferring data to the host concurrently.
- Fills the host's output buffer with the address-in-address data pattern.
- Sends commands to each node to verify that the node received the address-in-address data pattern from the host and from all other nodes.
- Verifies that the host received the address-in-address data pattern from all the nodes.

Test 96 HR DMA - Variable Transfer Count

This test verifies that the DMA can handle sixteen different transfer counts. The transfer counts are derived by passing a floating one pattern through the 16-bit byte count field which results in counts from one doubleword (eight bytes) to 32K doublewords (256K bytes). This test verifies the ability of the DMA to subdivide the overall byte count into smaller subblocks for transmission across the MCbus between the host and nodes.

Note – If a TMI links the host to the node, data and status will pass through that TMI.

Subblock sizes of 8, 16, or 32 are jumper-selectable on the ATB board.

The test uses a byte binary progression data pattern.

Byte Count Examples:

- Transfer Count = 1, Byte Count = 0x00008
- Transfer Count = 2, Byte Count = 0x00010
- Transfer Count = 4, Byte Count = 0x00020
- Transfer Count = 32K, Byte Count = 0x40000

This test contains the following subtests:

Subtest	Subtest Name
a	DMA - Sequential Node Operation
b	DMA - Concurrent Node Operation

Note – This test is applicable only to the Host in Remote (HR) testing mode.

If you selected the VME ONLY bus configuration during program initialization, the test writes and reads across the VMEbus. If not, the Local Bus is used.

This test performs the following operations:

- Clears the MCA board of any previous error conditions.
- Initializes and verifies the register map.
- Opens the appropriate Receive Windows.

Subtest a - Sequential Node Operation

This subtest performs the following operations:

- Fills the host‘s input buffer with a complemented data pattern.
- Sends commands to each node to fill the node’s input buffer with a complemented data pattern.
- Fills the host’s output buffer with the data pattern.
- Sends commands to each node to verify that the data pattern is received by the node.
- Sends commands to each node to fill the node’s output buffer with the data pattern.
- Verifies that the data pattern is received by the host.

Subtest b - Concurrent Node Operation

This subtest performs the following operations:

- Fills the host’s input buffer with a complemented data pattern.
- Sends commands to each node to fill the node’s input buffer with a complemented data pattern.
- Sends commands to each node to fill the node’s output buffer with the data pattern.
- Starts transferring data to the host concurrently.
- Fills the host’s output buffer with the data pattern.
- Sends commands to each node to verify that the node received the data pattern from the host and all other nodes.
- Verifies that the host received the data pattern from all the nodes.

Test 97 HR DMA - VL Transfers (Bus Valid Inhibit & Local Semaphore)

This test uses DMA longword data transfers to verify the operation of the MCbus Bus Valid and Local Semaphore signals. Setting and resetting the VL bits in the DMA command software enables and disables the MCbus Valid and Local Semaphore signals that are generated by the transmitter hardware. The test disables the TX FIFO output by setting the TFFD bit in the MC.CTL CSR. The test then issues 132 commands to fill the TX FIFO with 256 doublewords and enables the FIFO. The transmitter starts transferring various combinations of VL type transfers at the highest possible frequency. Alternating VL transfers thoroughly test the hardware timing.

The test builds an output buffer of 256 doublewords in contiguous memory locations within one memory page. This allows the receiver buffer to also be located within one memory page. These 256 doublewords are transferred by data chaining 132 DMA commands with various VL settings as follows:

- Commands 1-4

Command	VL Bits	Byte Count	Count
1	000	0x00100	32 doublewords
2	010	0x00100	32 doublewords
3	100	0x00100	32 doublewords
4	110	0x00100	32 doublewords

- The next pair of commands are executed 32 times with their Source and Destination addresses chained together contiguously. This yields 64 doublewords.

Command	VL Bits	Byte Count	Count
5	100	0x00008	1 doubleword
6	000	0x00008	1 doubleword

- The next pair of commands are executed 32 times with their Source and Destination addresses chained together contiguously. This yields 64 doublewords.

Command	VL Bits	Byte Count	Count
69	010	0x00008	1 doubleword
70	000	0x00008	1 doubleword

The test verifies that all 256 doublewords are either received or not received based on the receiver acceptance criteria of the VL transfer type. For example, a VL setting of 0x10 inhibits data transmission to either the Source or Destination nodes.

TABLE 3-5 lists the data acceptance criteria for the Source and Destination nodes based on the setting of the VL bits.

TABLE 3-5 Test 97 Data Acceptance Criteria for Source and Destination Nodes

VL Bits	Destination Node Receive Data?	Source Node Receive Data?
00	yes	no
01	yes	yes
10	no	no
11	no	yes

This test uses an address-in-address data pattern.

Note – If a TMI links the host to the node, data and status will pass through that TMI.

This test is applicable only to the Host in Remote (HR) testing mode.

If you selected the VME ONLY bus configuration during program initialization, the test writes and reads across the VMEbus. If not, the Local Bus is used.

This test performs the following operations:

- Clears the MCA board of any previous error conditions.
- Initializes and verifies the register map.
- Opens the appropriate Receive Windows.
- Fills the host's input buffer with a complemented data pattern.
- Sends commands to each node to fill the node's input buffer with a complemented data pattern.
- Fills the host's output buffer with the data pattern.
- Sends commands to each node to verify that the data pattern is received by the node.
- Sends commands to each node to fill the node's output buffer with the data pattern.
- Verifies that the data pattern is received by the host.

Test 98 HR WSC/DMA - Alternating WSC & DMA Subblock Transfers (VMEbus)

This test verifies the ability of the host to transmit and receive alternating WSC (Snoop) and DMA (burst) transfers. The transmit FIFO is disabled and alternately filled with a longword Snoop transfer and a DMA burst transfer of three doublewords. These commands are issued 64 times to half-fill the transmit FIFO. The FIFO is then enabled which causes the transmission of back-to-back single and burst transfers through the MCbus interface and through the receiver into the EDRAM input buffer.

The WSC I/O buffers are contiguous and are allocated to the first Window Page of the memory test area. The DMA I/O buffers are also contiguous and are allocated to the second Window Page of the memory test area. WSC transfers use an all ones data pattern and DMA transfers use a byte binary progression data pattern.

Note – If a TMI links the host to the node, data and status will pass through that TMI.

This test contains the following subtests:

Subtest	Subtest Name
a	WSC/DMA - Sequential Node Operation
b	WSC/DMA - Concurrent Node Operation

Note – This test is applicable only to the Host in Remote (HR) testing mode.

This test does not verify simultaneous operation of the DMA, Snoop transfer, and receiver at the mezzanine interface. This feature is checked in other tests.

This test performs the following operations:

- Clears the MCA board of any previous error conditions.
- Initializes and verifies the register map.
- Opens the appropriate Receive Windows.

Subtest a - Sequential Node Operation

This subtest performs the following operations:

- Fills the host's input buffers with complemented data patterns.
- Sends commands to each node to fill the node's input buffers with complemented data patterns.
- Fills the host's output buffer with the data pattern for DMA only.
- Disables the TX FIFO in the host.
- Repeats the following steps 64 times to create 64 WSC sequential memory writes alternating with 64 DMA burst transfers:
  - Issues a WSC Snoop transfer.
  - Issues a DMA burst transfer.
- Enables the TX FIFO.
- Waits for the DMA data transfers to complete.
- Sends commands to each node to verify that the data patterns are received by the node.
- Sends commands to each node to fill the node's output buffer with the data pattern for DMA only.
- Verifies that the data pattern is received by the host.

Subtest b - Concurrent Node Operation

This subtest performs the following operations:

- Fills the host's input buffers with complemented data patterns.
- Sends commands to each node to fill the node's input buffers with complemented data patterns.
- Sends commands to each node to fill the node's output buffer with the data pattern for DMA only.
- Disables the TX FIFO.
- Repeats the following steps 64 times to create 64 WSC sequential memory writes alternating with 64 DMA burst transfers:
  - Issues a WSC Snoop transfer.
  - Issues a DMA burst transfer.
- Enables the TX FIFO and starts transferring data to the host concurrently.
- Fills the host's output buffer with the data pattern for DMA only.
- Sends commands to each node to verify that the node received the data pattern from the host and all other nodes.
- Verifies that the host received the data pattern from all the nodes.

Test 99 HR WSC/DMA - Byte Transfer Concurrent with DMA Transfer

This test verifies the ability of the host to transmit and receive WSC (Snoop) byte transfers concurrent with DMA (burst) transfers. The test verifies the simultaneous operation of the DMA, Snoop transfer, and receiver at the mezzanine interface.

Note – If a TMI links the host to the node, data and status will pass through that TMI.

This test contains the following subtests:

Subtest	Subtest Name
a	WSC/DMA - Sequential Node Operation
b	WSC/DMA - Concurrent Node Operation

Note – This test is applicable only to the Host in Remote (HR) testing mode.

If you selected the VME ONLY bus configuration during program initialization, the test writes and reads across the VMEbus. If not, the Local Bus is used.

The WSC I/O buffer is a 4K byte contiguous buffer allocated to the first Window Page of the memory test area. The DMA I/O buffer is a 32768 doubleword contiguous buffer starting at the fifth Window Page of the memory test area. Buffers are selected to cause the WSC write transfer period to overlap the DMA transfer period. WSC transfers use an all ones data pattern and DMA transfers use a byte binary progression data pattern.

This test performs the following operations:

- Clears the MCA board of any previous error conditions.
- Initializes and verifies the register map.
- Opens the appropriate Receive Windows.

Subtest a - Sequential Node Operation

This subtest performs the following operations:

- Fills the host's input buffer with a complemented data pattern.
- Sends commands to each node to fill the node's input buffer with a complemented data pattern.
- Fills the host's output buffer with the data pattern for DMA only.

- Issues a DMA burst transfer.
- Issues a WSC Snoop transfer.
- Sends commands to each node to verify that the data pattern is received by the node.
- Sends commands to each node to fill the node's output buffer with the data pattern for DMA only.
- Verifies that the data pattern is received by the host.

Subtest b - Concurrent Node Operation

This subtest performs the following operations:

- Fills the host's input buffers with complemented data patterns.
- Sends commands to each node to fill the node's input buffers with complemented data patterns.
- Sends commands to each node to fill the node's output buffer with the data pattern for DMA only.
- Issues a DMA burst transfer and WSC Snoop transfer to start transferring data to the host concurrently.
- Fills the host's output buffer with the data pattern for DMA only.
- Sends commands to each node to verify that the node received the data pattern from the host and all other nodes.
- Verifies that the host received the data pattern from all the nodes.

Test 100 HR WSC/DMA - Longword Transfer Concurrent with DMA Transfer

This test verifies the ability of the host to transmit and receive WSC (Snoop) longword transfers concurrent with DMA (burst) transfers. The test verifies the simultaneous operation of the DMA, Snoop transfer, and receiver at the mezzanine interface.

Note – If a TMI links the host to the node, data and status will pass through that TMI.

This test contains the following subtests:

Subtest	Subtest Name
a	WSC/DMA - Sequential Node Operation
b	WSC/DMA - Concurrent Node Operation

Note – This test is applicable only to the Host in Remote (HR) testing mode.

If you selected the VME ONLY bus configuration during program initialization, the test writes and reads across the VMEbus. If not, the Local Bus is used.

The WSC I/O buffer is a 16K byte contiguous buffer allocated to the first four Window Pages of the memory test area. The DMA I/O buffer is a 32768 doubleword contiguous buffer starting at the fifth Window Page of the memory test area. Buffers are selected to cause the WSC write transfer period to overlap the DMA transfer period. WSC transfers use an all ones data pattern and DMA transfers use a byte binary progression data pattern.

This test performs the following operations:

- Clears the MCA board of any previous error conditions.
- Initializes and verifies the register map.
- Opens the appropriate Receive Windows.

Subtest a - Sequential Node Operation

This subtest performs the following operations:

- Fills the host's input buffers with complemented data patterns.
- Sends commands to each node to fill the node's input buffers with complemented data patterns.
- Fills the host's output buffer with the data pattern for DMA only.
- Issues a DMA burst transfer.
- Issues a WSC Snoop transfer.
- Sends commands to each node to verify that the data pattern is received by the node.
- Sends commands to each node to fill the node's output buffer with the data pattern for DMA only.
- Verifies that the data pattern is received by the host.

Subtest b - Concurrent Node Operation

This subtest performs the following operations:

- Fills the host's input buffers with complemented data patterns.
- Sends commands to each node to fill the node's input buffers with complemented data patterns.
- Sends commands to each node to fill the node's output buffer with the data pattern for DMA only.
- Issues a DMA burst transfer and WSC Snoop transfer to start transferring data to the host concurrently.

- Fills the host's output buffer with the data pattern for DMA only.
- Sends commands to each node to verify that the node received the data pattern from the host and all other nodes.
- Verifies that the host received the data pattern from all the nodes.

Test 101 HR DMA - Data Pattern - 0x7E7E7E7E

This test verifies that the host can transmit and receive a data pattern of 0x7E7E7E7E to and from all nodes both sequentially and concurrently. The test writes and reads across either the Local Bus or the VMEbus using DMA data transfers with a word count of 32768.

Note – If a TMI links the host to the node, data and status will pass through that TMI.

This test contains the following subtests:

Subtest	Subtest Name
a	DMA - Sequential Node Operation
b	DMA - Concurrent Node Operation

Note – This test is applicable only to the Host in Remote (HR) testing mode.

If you selected the VME ONLY bus configuration during program initialization, the test writes and reads across the VMEbus. If not, the Local Bus is used.

This test performs the following operations:

- Clears the MCA board of any previous error conditions.
- Initializes and verifies the register map.
- Opens the appropriate Receive Windows.

Subtest a - Sequential Node Operation

This subtest performs the following operations:

- Fills the host's input buffer with a complemented data pattern.
- Sends commands to each node to fill the node's input buffer with a complemented data pattern.
- Fills the host's output buffer with a 0x7E7E7E7E data pattern.
- Sends commands to each node to verify that the data pattern is received by the node.
- Sends commands to each node to fill the node's output buffer with the 0x7E7E7E7E data pattern.
- Verifies that the data pattern is received by the host.

Subtest b - Concurrent Node Operation

This subtest performs the following operations:

- Fills the host's input buffer with a complemented data pattern.
- Sends commands to each node to fill the node's input buffer with a complemented data pattern.
- Sends commands to each node to fill the node's output buffer with the 0x7E7E7E7E data pattern.
- Starts transferring data to the host concurrently.
- Fills the host's output buffer with the 0x7E7E7E7E data pattern.
- Sends commands to each node to verify that the node received the data pattern from the host and from all other nodes.
- Verifies that the host received the data pattern from all the nodes.

Test 102 HR DMA - ATB Test

This test verifies the function of the MEMORY CHANNEL IV node and the Arbitration Termination Board (ATB). The test verifies that a node can perform a DMA Programming transfer to the ATB and the ATB can be enabled to gather and report MEMORY CHANNEL Statistics. When the ATB is enabled, twelve categories of data are counted and written as a structure across the MEMORY CHANNEL Bus to a memory location selected with an MC Bus Programming transfer. The statistics structure in memory is updated every 4096 clocks. This equates to every 0.3072 ms for an MC Bus cycle time of 75 ns or every 0.22528 ms for an MC Bus cycle time of 55 ns.

Note – If a TMI links the host to the node, data and status will pass through that TMI.

This test is applicable only to the Host in Remote (HR) testing mode.

This test is executed only if you entered Y at the ATB PRESENT prompt during the program initialization sequence.

Each of the nodes configured for remote testing will, in turn, verify its ability to operate with the ATB. The host performs its verification, and then sequentially commands each node to perform its verification until all nodes have been verified.

Once started, this test performs the following operations using the host:

- Sends a DMA Pbit transfer to disable ATB statistics reporting. This clears any pending enabled state.
- Verifies that the DMA DMACI completes within five microseconds.
- Fills the input buffer allocated for ATB statistics with zeros.
- Opens an RX Window for the input buffer.
- Sends a DMA Pbit transfer to enable ATB statistics reporting.
- Verifies that the DMA DMACI completes within five microseconds.
- Waits for statistics data to arrive in the input buffer.
- Sends a DMA Pbit transfer to disable ATB statistics reporting.
- Delays one millisecond to verify that statistics reporting is disabled.
- Verifies that the input buffer contains the expected statistics data.
- Repeats the previous steps fifty times.
- Displays the ATB jumper settings.

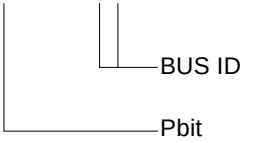
After it completes the testing sequence, the host selects the lowest numbered node to perform the operations and the node repeats the testing sequence. Testing is complete when the host has sequenced through all the nodes.

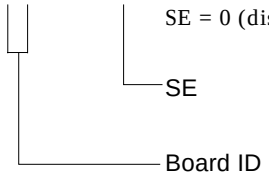
Programming the ATB

The ATB Statistics Reporting function requires a MC Bus Programming transfer for initialization and programming. The initialization and programming operation assigns a 40-bit Statistic Reporting Address and enables the Statistics Counters. The address is used by the Reporting function as the MC Bus address to which the Statistics Structure is written. FIGURE 3-1 and FIGURE 3-2 illustrate the format of the Statistics Reporting Structure. FIGURE 3-3 illustrates the format of the MC Bus Programming transfer.

The following MEMORY CHANNEL IV DMA command enables and disables the ATB:

`.wrt SA =` *address*	The variable (*address*) specifies the source address for the ATB's Statistics Reporting Address. The address format is shown in Figure 4-3
.wrt BC = 0x00000008	Transfer count equals one doubleword or eight bytes
`.wrt MSDA = 0x80000000`	Sets the Pbit, ATB's Target BUS ID = 00
`.wrt LSDA = 0x00000008`	Sets the ATB's Target Board ID = 00, SE = 8 (enable), SE = 0 (disable)





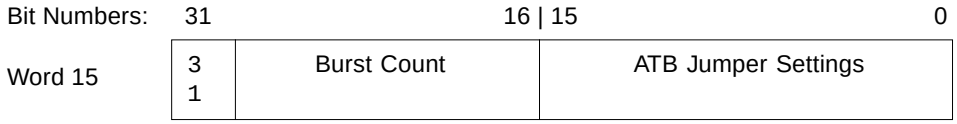
.

Bit Numbers:	31	16 | 15	0
Word 0	[0000]	not used	
Word 1	[0001]	not used	
Word 2	[1xxx] Total Cycle Count	[x]	Idle Cycle Count
Word 3	[x] Turn Around Cycle Count	[x]	Busy Cycle Count
Word 4	[x] Remote Semaphore Count	[x]	Local Semaphore Count
Word 5	[x] Address Phase Count	[x]	Multi Grant Count
Word 6	[x] Node 0 Block Count	[x]	Node 0 Single Count
Word 7	[x] Node 1 Block Count	[x]	Node 1 Single Count
Word 8	[x] Node 2 Block Count	[x]	Node 2 Single Count
Word 9	[x] Node 3 Block Count	[x]	Node 3 Single Count
Word 10	[x] Node 4 Block Count	[x]	Node 4 Single Count
Word 11	[x] Node 5 Block Count	[x]	Node 5 Single Count
Word 12	[x] Node 6 Block Count	[x]	Node 6 Single Count
Word 13	[x] Node 7 Block Count	[x]	Node 7 Single Count
Word 14	[x] Node 8 Single Count	[x]	Node 8 Single Count
Word 15	3| 1| [x] Burst Count	[x]	ATB Jumper Settings see FIGURE 3-1

Note - The memory address of the ATB Statistics Block must be on a doubleword boundary. The expected hexadecimal statistics data is in [ ]. [x] indicates do not care values.

FIGURE 3-1 MEMORY CHANNEL Statistics Structure

.



Note – Word 15 of the Statistics Block displays the current jumper settings for the ATB.

Bit Positions	Contents
5-0	Transfer Count Jumpers 5-0 which define the subblock transfer count.
7-6	Multi-Grant Count Jumpers 1-0 which define the maximum number of consecutive grant sequences for a Multi-Grant Response.
11-8	Turn-Around Count Jumpers 3-0 which define the number of bus cycles required for each arbitration turnaround.
15-12	Burst Requester ID Jumpers 3-0 which define the node ID of the Bus Burst Requester.
31	HSWITCH_IN signal which indicates, when true, that the reporting ATB achieved primary status due to an ATB failover.

FIGURE 3-2 MEMORY CHANNEL Statistics Block - Word 15

Single/Block Transfer - Address Phase

63 56	55 48	47 40	39 32	31 24	23 16	15 3	2 0
Byte | Vld	PU | - CNT -	Node ID | Bus ID	Target Bus ID	Target Board ID	--- Not Used ---	S E	VLR

Single/Block Transfer - Data Phase

63 56	55 48	47 40	39 32	31 24	23 16	15 3	2 0
63---------- Not Used ----------			---------- Statistic Reporting Address ----------0				

FIGURE 3-3 MC Bus Programming Transfer Format

CHAPTER 4

Program Messages

Introduction

The MEMORY CHANNEL IV System Diagnostic displays the following types of messages:

- Monitor
- Start
- Milestone
- Expanded Milestone
- Pass and Error Count
- Error

Monitor Messages

If this diagnostic is configured during automatic testing, the ROM Monitor displays the following program information on the operator's console before calling the diagnostic:

```
Slot n: Loading extended image mc4diag
```

Variable	Description
n	Specifies the slot number of the MEMORY CHANNEL IV module.

Note – The monitor cannot identify the slot associated with an extended diagnostic running in interactive mode.

Start Messages

Once called by the monitor or loaded interactively, the MEMORY CHANNEL IV System Diagnostic displays its start message on the operator's console. The start message is displayed in the following format:

```
Extended Memory Channel IV Diagnostic
Diagnostic Revision: y.y
[Slot: s,] MCA Board CSR Addr: 0xaaaaaaaa - 0xbbbbbbbb
EDRAM Board CSR Addr: 0xcccccccc - 0xdddddddd
[Copyright year Sun Microsystems, Inc.]
```

Variable	Description
y.y	Specifies the current revision of the diagnostic.
s	Specifies the slot number of the MEMORY CHANNEL IV module. The slot number is displayed only when the diagnostic is running in automatic mode.
aaaaaaaa	Specifies the default starting address of the memory address range assigned to the MEMORY CHANNEL IV Adapter board.
bbbbbbbb	Specifies the default ending address of the memory address range assigned to the MEMORY CHANNEL IV Adapter board.
aaaaaaaa	Specifies the default starting address of the memory address range assigned to the EDRAM board.
bbbbbbbb	Specifies the default ending address of the memory address range assigned to the EDRAM board.
year	Specifies the year the program was copyrighted. The copyright message is displayed only when the diagnostic is running in interactive mode.

The start message is followed with the Interactive Mode Options Menu if the diagnostic is running in interactive mode.

If the reset CSR address was changed during program initialization, the program then displays the selected value in the following format:

```
Board CSR Addr: 0xaaaaaaaa - 0xbbbbbbbb
```

Variable	Description
aaaaaaaa	Specifies the selected starting address of the memory address range assigned to this board.
bbbbbbbb	Specifies the selected ending address of the memory address range assigned to this board.

The program displays the following board specific information prior to starting test execution:

```
Board Id: 0xid (partno), Major Rev: 0xrev, Minor Rev: 0xm
EDRAM Id: 0xid, Major Rev: 0xrev, Minor Rev: 0xm

On Board Memory Size: size, Physical Node Id: n
```

Variable	Description
id	Specifies the board identification mnemonic or the value in the ID PROM if the identification number can not be determined.
partno	Specifies the board part number. This field contains ??? if no valid part number is available.
rev	Specifies the board major revision.
m	Specifies the board minor revision.
size	Specifies the size of the on-board memory: 128mb, 256mb or 512mb.
n	Specifies the Node ID number.

Milestone Messages

Milestone messages are displayed only when the diagnostic is running in interactive mode. As each test starts execution, the diagnostic displays a test milestone in the following format:

```
Running Test tt: testname
```

Variable	Description
tt	Specifies the test number.
testname	Specifies the name of the test.

If subtests are associated with individual tests, the subtest milestones display in the following format. Subtest messages are displayed as each subtest starts.

```
Subtest s: name
```

Variable	Description
s	Specifies the letter associated with the current subtest.
name	Specifies the name of the subtest.

If no SCSI disk drive is configured on the VME Bus, the program displays a message in the following format for Tests 40 and 41:

```
Note: Test tt is bypassed if no SCSI Disk Drive is configured.
```

Variable	Description
tt	Specifies the test number.

If Test 41 is selected and no V/SCSI COUGAR drive is configured, the following message is displayed:

```
Test 41 is bypassed when V/SCSI COUGAR is not configured.
```

If the diagnostic is initialized to bypass testing the Arbitration Termination Board (ATB), the following message is displayed for Test 102:

```
Test 102 SKIPPED - ATB Not Present
```

When running in remote mode, the node displays the following message while waiting for the host to start up:

```
Waiting for Host Node
```

When running in remote mode, the node displays a message in the following format to identify the test that the host is executing:

```
Host Executing Test tt
```

Variable	Description
tt	Specifies the number of the Host in Remote (HR) test (91-101).

When running in remote mode, the host displays the following message while waiting for the nodes to start up:

```
Waiting for n Remote Node(s) x x x...
```

Variable	Description
n	Specifies the number of nodes to be started.
x	Specifies the logical node number. If the host is waiting for more than one node to start, each node number is displayed.

When running in remote mode, a node (host or remote) displays the following error message when two remote nodes are using the same logical node ID:

```
Another node found with my Logical Node ID , x
```

Variable	Description
x	Specifies the conflicting logical node ID (0-F).

When running in remote mode, a remote node displays the following error message when the host is using the same logical node ID:

```
Received Host ID = MY ID = x
```

Variable	Description
x	Specifies the conflicting logical node ID (0-F).

Expanded Milestone Messages

If expanded milestone messages were enabled during diagnostic initialization, additional messages are displayed to identify the operations performed during testing. Expanded milestone messages are grouped into the following categories:

- MCA commands
- Window manipulation
- Register displays
- Data manipulation
- Host to node commands
- Miscellaneous

MCA Command Messages

The following milestone messages indicate commands are being sent to the MCA:

- When the diagnostic issues a Software Clear command to the MCA, it displays:

```
Will Issue Software Clear
```

- When the diagnostic issues the first Software Clear command to the MCA, EDRAM is cleared and the following messages are displayed:

```
Clearing 0xnnnnnnnn bytes starting at 0xaddr

EDRAM ERROR COUNTER WAS = 0xoldcnt
EDRAM ERROR COUNTER NOW = 0xnewcnt
```

Variable	Description
nnnnnnnn	Specifies the number of bytes of EDRAM to be cleared.
addr	Specifies the starting EDRAM address.
oldcnt	Specifies the number of ECC errors before clearing is started.
newcnt	Specifies the number of ECC errors after clearing is completed.

- When turning the MCA online or offline, the diagnostic displays a message in the following format:

```
Will Set Online|Offline
```

- When the diagnostic sets up the contents of the MCA Control Register, it displays:

```
Will Set MCA Control Register to 0xaaaaaaaa
```

Variable	Description
aaaaaaaa	Specifies the value to be loaded into the Control Register.

- When a DMA command is sent to the MCA, the following type of message is displayed:

```
Will Send DMA Command = source, bytecnt, upperdest, lowerdest
```

Variable	Description
source	Specifies the contents of the source register.
bytecnt	Specifies the contents of the byte count register.
upperdest	Specifies the contents of the upper destination register.
lowerdest	Specifies the contents of the lower destination register.

- The following message indicates that a special DMA command will be issued:

```
Will Send DMA Command Registers Out of ORDER
```

Window Manipulation Messages

The following types of messages are displayed when the diagnostic is manipulating the Transmit (TX) or Receive (RX) Windows:

- When opening a specified range of RX Windows, the diagnostic displays:

```
Open RX Windows 0xstart to 0xend Seg 0xseg within Testing Range
```

  RX Windows from *start* to *end* will be opened and segment *seg* will be marked open.

- When opening TX Windows, the diagnostic displays:

```
Open TX Windows 0xstart to 0xend within Testing Range
```

  TX Windows from *start* to *end* will be opened.

- When closing windows, the diagnostic displays:

```
Close All TX|RX Windows
```

- When the diagnostic modifies the Local Semaphore or Bus Valid Inhibit bits in a TX Window, it displays:

```
Modify TX Windows 0xaddr eor_val = 0x0000000a
```

Variable	Description	
addr	Specifies the TX Window address.	
a	Specifies one of the following:	
	2	Changes Local Semaphore
	4	Changes Bus Valid Inhibit
	6	Changes both Local Semaphore and Bus Valid Inhibit

- When translating TX Window addresses, the diagnostic displays:

```
TX Translate addr = 0xaaaaaaaa, trans = 0xbbbbbbbb
```

TX address *aaaaaaaa* is translated to the window which includes address *bbbbbbbb*.

- When translating RX Window addresses, the diagnostic displays:

```
RX Translate seg = 0xseg, addr = 0xaaaaaaaa, trans = 0xbbbbbbbb
```

RX address *aaaaaaaa* is translated to the window which includes address *bbbbbbbb* and segment *seg* is marked open.

Register Display Messages

Register display milestone messages describe the actual contents of a register with respect to the expected register contents. The following types of messages are used:

- This message displays the contents of the Interrupt Status register:

```
INT_STATUS Reg exp: 0xeeeeeeee, mask: 0xmask, rcvd: 0xaaaaaaaa
```

Variable	Description
eeeeeeee	Specifies the expected register contents.
mask	Specifies the mask value used by the diagnostic.
aaaaaaaa	Specifies the actual register contents.

- This message displays the contents of the Sense register:

```
SENSE Reg exp: 0xeeeeeeee, mask: 0xmask, rcvd: 0xaaaaaaaa
```

Variable	Description
eeeeeeee	Specifies the expected register contents.
mask	Specifies the mask value used by the diagnostic.
aaaaaaaa	specifies the actual register contents.

- This message displays the contents of the Source Address Exception (SAE) register:

```
SAE Reg exp: 0xeeeeeeee, rcvd: 0xaaaaaaaa
```

Variable	Description
eeeeeeee	Specifies the expected register contents.
aaaaaaaa	Specifies the actual register contents

- This message displays the contents of the RSIDE register:

```
REC SRCIDS ERROR Reg exp: 0xeeeeeeee, mask: 0xmask, rcvd: 0xaaaaaaaa
```

Variable	Description
eeeeeeee	Specifies the expected register contents.
mask	Specifies the mask value used by the diagnostic.
aaaaaaaa	Specifies the actual register contents.

Data Manipulation Messages

The following types of messages are associated with data manipulation:

- This message indicates that memory will be filled with a data pattern:

```
Will Fill Buf 0xaddr, wd_cnt = 0xcount, pat, INV|ACT [, FMT = format]
```

Variable	Description
addr	Specifies the starting address of the data buffer.
count	Specifies the number of long words to be written.
pat	Specifies the data pattern.
INV|ACT	Specifies either the actual (ACT) or inverted (INV) data pattern will be used.
format	Specifies the source of the addresses used with the address in data pattern.

- This message indicates that a data comparison will be made:

```
Will Compare Buf 0xaddr, wd_cnt = 0xcount, pat, INV|ACT [, FMT = format]
```

Variable	Description
addr	Specifies the starting address of the data buffer.
count	Specifies the number of long words to be written.
pat	Specifies the data pattern.
INV|ACT	Specifies either the actual (ACT) or inverted (INV) data pattern will be used.
format	Specifies the source of the addresses used with the address in data pattern.

- This message is used with Test 93 when filling data buffers:

```
Will Fill VL Bufs 0xaddr1, 0xaddr2, 0xaddr3, 0xaddr4, xxxx longwords
```

Variable	Description
addr1-addr4	Specify the starting addresses of the data buffers.
xxxx	Specifies the number of long words written.

Host to Node Command Messages

When sending commands to the node, the copy of the diagnostic running on the host displays messages in the following formats:

- This message indicates that a command is being sent to the nodes:

```
Command Node n, cmd, Command Response = x
```

Variable	Description
n	Specifies the node number.
cmd	Specifies the command to be executed.
x	Specifies the hexadecimal command response value: -1 = command unknown, 1 = command complete.

- This message indicates that a command is being sent to the nodes to place in the queue. Execution is not performed until a command is sent by the host.

```
Queue Command n, cmd
```

Variable	Description
n	Specifies the node number.
cmd	Specifies the command to be queued.

- This message indicates that the host is waiting for the node to complete a queued operation:

```
Wait for Node End n, [Complete]
```

When the specified logical node (*n*) is done, `Complete` is appended to the message.

Miscellaneous Messages

- At the completion of Test 1, the contents of the EDRAM ECC error counter are displayed in the following format. This counter keeps track of the number of ECC errors on the EDRAM board.

```
EDRAM ERROR COUNTER NOW = 0xcount
```

Variable	Description
count	Specifies the number of ECC errors.

- When the diagnostic is waiting for the contents of a memory location to change, it displays:

```
Wait for *addr != pat [COMPLETE]
```

This message indicates successful completion by appending `COMPLETE` to the message when the contents of the specified location (*addr*) change from the specified data pattern (*pat*). If a timeout occurs, `COMPLETE` is not displayed.

- When the Board ID Register (ID PROM) checksum value is read, the following message is displayed:

```
checksum = 0xrrrrrrrr
```

Variable	Description
rrrrrrrr	Specifies the checksum value read from the ID PROM.

Pass and Error Count Messages

If an extended diagnostic is running during the automatic testing sequence, the diagnostic displays PASS when the program executes successfully. If the program fails, the diagnostic displays FAIL.

If the diagnostic is running in interactive mode, the following message is displayed after each pass is completed:

```
Pass Count: pppppppp
```

If an error was detected, the diagnostic displays the pass count and error count in the following format:

```
Pass Count: pppppppp, Error Count: eeeeeeee
```

Variable	Description
pppppppp	specifies the number of program passes completed.
eeeeeeee	specifies the number of errors encountered since the program started execution.

Error Messages

If the ROM Monitor revision is not equal to the required value, the diagnostic displays:

```
MCA:Invalid ROM Version Detected: Requires xxx Version
```

Variable	Description
xxx	Specifies the required ROM Monitor version.

If the ID modifier code is incorrect, the diagnostic displays:

```
MCA:Invalid Id Modifier Code Detected!
```

If the Control/Status Register (CSR) address is not a multiple of 0x800 during the program initialization sequence for automatic testing, the diagnostic displays:

```
Invalid board CSR address
```

If an error is detected during interactive mode testing, the diagnostic displays an error header and message in the following format:

```
Extended Memory Channel IV Diagnostic Failed    Time:hh:mm:ss
Board CSR Addr: 0xaaaaaaaa - 0xbbbbbbbb
VME Base Addr: 0xcccccccc
Local Base Addr: 0xdddddddd, On Board Memory Size: 0xmm, Physical
Node Id: 0xi
Error: Slot b: CPU n: MCA: Test t [s]: test name
error message
```

Variable	Description
hh:mm:ss	Specifies the system elapsed time when the error occurred.
aaaaaaaa	Specifies the starting address of the memory address range assigned to this board.
bbbbbbbb	specIfies the ending address of the memory address range assigned to this board.
cccccccc	Specifies the VME Bus base address.
dddddddd	Specifies the Local Bus base address.
mm	speciFies the size of the on-board memory.
i	Specifies the jumpered physical Node ID number.
b	Specifies the slot number of the MEMORY CHANNEL IV module. When the diagnostic is running in interactive mode, the board slot number is always reported as 0.
n	Specifies the number of the CPU running the diagnostic.
t	Specifies the number of the failing test.
s	Specifies the letter of the failing subtest, if applicable.
test name	Specifies the name of the failing test.
error message	Specifies the type of error message.

The program then displays the pass count and error count and issues the following prompt:

```
MCA: continue on error [cr, d, q]?
```

Press the Return key to continue test execution or enter `q` to return to the Interactive Mode Options Menu. Enter `d` to display the last status and sense information.

Register Error Messages

Test 1 verifies individual registers in the EDRAM control register map. If a register error occurs during Test 1, one of the following error messages is displayed:

- This message indicates an error in the Memory Size Status register:

```
Invalid Memory Size or Slot Number
Data value recd: 0xaaaaaaaa
```

Variable	Description
aaaaaaaa	Specifies the contents of the Memory Size register.

- This message indicates an error in the Local Bus Base address register:

```
Local bus base Reg exp: 0xeeeeeeee, rcvd: 0xaaaaaaaa
```

Variable	Description
eeeeeeee	Specifies the expected contents of the register.
aaaaaaaa	Specifies the actual contents of the register.

- This message indicates an error was detected in the VMEbus Base address register:

```
Vme bus base Reg exp: 0xeeeeeeee, rcvd: 0xaaaaaaaa
```

Variable	Description
eeeeeeee	Specifies the expected contents of the register.
aaaaaaaa	Specifies the actual contents of the register.

- This message indicates that the computed checksum did not match the checksum value read from the ID PROM:

```
Checksum computed = 0xeeeeeeee, read = 0xaaaaaaaa
```

Variable	Description
eeeeeeee	Specifies the checksum value computed by the diagnostic.
aaaaaaaa	Specifies the actual checksum value read from the ID PROM.

- This message indicates that an ATE Pattern error was detected in the ID PROM:

```
ATE Pattern Read = 0xrrrrrrrr
```

Variable	Description
rrrrrrrr	Specifies the ATE Pattern read from the ID PROM.

Test 4 verifies the individual registers in the MEMORY CHANNEL IV Adapter board register map. If a register error is detected during Test 4, one of the following messages is displayed:

- This message indicates that an error was detected in the RX Window registers (Subtests i and h):

```
Data value exp: 0xeeeeeeee, rcvd: 0xaaaaaaaa
RWSRA = 0xssssssss, RWPRA = 0xpppppppp
```

Variable	Description
eeeeeeee	Specifies the expected register value.
aaaaaaaa	Specifies the actual register value.
ssssssss	Specifies the RX Window segment value.
pppppppp	Specifies the RX Window page value.

- This message indicates that an error was detected in the TX Window registers (Subtest e):

```
Data value exp: 0xeeeeeeee, rcvd: 0xaaaaaaaa, TWRA = 0xssssssss
```

Variable	Description
eeeeeeee	Specifies the expected register value.
aaaaaaaa	Specifies the actual register value.
ssssssss	Specifies the TX Window address.

- If an error is detected in subtest b, c, d, f, or g during Test 4, the diagnostic displays:

```
Data value exp: 0xeeeeeeee, rcvd: 0xaaaaaaaa
```

Variable	Description
eeeeeeee	Specifies the expected register value.
aaaaaaaa	Specifies the actual register value.

- This message indicates that an error was detected in the Sense register (Subtest k):

```
SENSE Reg exp ONLINE/TX_HALTED bits: 0x10040, rcvd: 0x0
Data value exp: 0xeeeeeeee, rcvd: 0xaaaaaaaa
```

Variable	Description
eeeeeeee	Specifies the expected register value.
aaaaaaaa	Specifies the actual register value.

This message indicates that an error was detected in the VIC register during Test 42:

```
VIC Reg exp: 0xeeeeeeee, rcvd: 0xaaaaaaaa
```

Variable	Description
eeeeeeee	Specifies the expected VIC register value.
aaaaaaaa	Specifies the actual value read from the VIC register.

This message indicates that the Board ID value in the RSIDE register is incorrect:

```
RSIDE BID exp: 0xexp, rcvd: 0xact
```

Variable	Description
exp	Specifies the expected RSIDE register value.
act	Specifies the actual value read from the RSIDE register.

This message indicates that the Node ID value in the RSIDE register is incorrect:

```
RSIDE NID exp: 0xexp, rcvd: 0xact
```

Variable	Description
exp	Specifies the expected RSIDE register value.
act	Specifies the actual value read from the RSIDE register.

This message indicates that an error was detected in the TMI Address Counter (LSW)/(MSW) Register:

```
TMI- Addr Cntr(LSW) exp: 0xexp, rcvd: 0xact
TMI- Addr Cntr(MSW) exp: 0xexp, rcvd: 0xact
```

Variable	Description
exp	Specifies the expected register value.
act	Specifies the actual value read from the register.

This message indicates that an error was detected in the TMI RAM Data (LSW)/(MSW) Register:

```
TMI- RAM Data(LSW) exp: 0xexp, rcvd: 0xact
TMI- RAM Data(MSW) exp: 0xexp, rcvd: 0xact
```

Variable	Description
exp	Specifies the expected register value.
act	Specifies the actual value read from the register.

This message indicates that an error was detected in the TMI Configuration Register:

```
TMI- Configuration exp: 0xexp, rcvd: 0xact
```

Variable	Description
exp	Specifies the expected register value.
act	Specifies the actual value read from the register.

This message indicates that an error was detected in the TMI Auto Status Register (bits 31-00, 39-32):

```
TMI- Auto Stat Reg (bits 31-00) exp: 0xexp, rcvd: 0xact
TMI- Auto Stat Reg (Bits 39-32) exp: 0xexp, rcvd: 0xact
```

Variable	Description
exp	Specifies the expected status.
act	Specifies the actual status received.

Data Error Messages

If a data error is detected, the following type of message is displayed:

```
Data Errors: x of y longwords

Address       Expected      Actual
0xaddr,       0xeeeeeeee,   0xaaaaaaaa
   .              .             .
   .              .             .
   .              .             .
```

Variable	Description
x	Specifies the number of longwords in error.
y	Specifies the number of longwords transferred.
addr	Specifies the failing address.
eeeeeeee	Specifies the expected data value.
aaaaaaaa	Specifies the actual data value.

If a TMI data error is detected, the diagnostic displays:

```
TMI- Data exp: 0xexp, rcvd: 0xact, at addr: 0xaddr
```

Variable	Description
exp	Specifies the expected data value.
act	Specifies the actual data received.
addr	Specifies the failing address.

Status Error Messages

If a status error is detected, the diagnostic displays the error information in the following format:

```
Status Error: Expected 0xeeeeeeee, Actual 0xaaaaaaaa

MCA: continue on error ;[cr,d,q]?d
```

Variable	Description
eeeeeeee	Specifies the expected status.
aaaaaaaa	Specifies the actual status.

If you enter d at the prompt, the program displays additional status information:

```
Last Status: Actual 0xdddddddd
ENCODED _ERR is errtype
bit 0xbbbbbbbb is unexpected|missing
                .
                .
                .
DMACI = 0xcc
```

Variable	Description
dddddddd	Specifies the last status received before the error.
errtype	Specifies the type of error received.
bit	Specifies the mnemonic of the status bit.
bbbbbbbb	Specifies the hexadecimal value of the status bit.
cc	Specifies the DMACI value.

Each status bit is reported as `unexpected` or `missing`.

If a TMI is present, the program displays the TMI status information in the following format:

```
Last TMI Status: Actual 0xaaaaaaaa, Expected 0xeeeeeeee, Mask 0xmmmmmmmm
bit 0xbbbbbbbb is unexpected|missing
                .
                .
                .
```

Variable	Description
aaaaaaaa	Specifies the last status received before the error.
eeeeeeee	Specifies the expected status value.
mmmmmmmm	Specifies the TMI status mask value.
bit	Specifies the mnemonic of the TMI status bit.
bbbbbbbb	Specifies the hexadecimal value of the TMI status bit.

Each TMI status bit is reported as `unexpected` or `missing`.

Sense Error Messages

If a sense error is detected, the program displays the error information in the following format:

```
Sense Error: Expected 0xeeeeeeee, Actual 0xaaaaaaaa

MCA: continue on error ;[cr,d,q]?d
```

Variable	Description
eeeeeeee	Specifies the expected sense information.
aaaaaaaa	Specifies the actual sense information.

If you enter `d` at the prompt, the program displays additional sense information:

```
Last Sense: Actual 0xdddddddd
NODE ID is id
SUB BLOCK SIZE is size
bit 0xbbbbbbbb is unexpected|missing
                .
                .
                .
```

Variable	Description
dddddddd	Specifies the last sense received before the error.
id	Specifies the node ID number.
size	Specifies the subblock size in use.
bit	Specifies the mnemonic of the sense bit.
bbbbbbbb	Specifies the hexadecimal value of the sense bit.

Each sense bit is reported as `unexpected` or `missing`.

Timeout Error Messages

If a timeout occurs during TMI testing, the diagnostic displays:

```
TMI- Timeout Waiting for SYNC,

CNTRL rcvd: 0xaaaaaaaa, STAT rcvd: 0xbbbbbbbb
```

Variable	Description
aaaaaaaa	Specifies the TMI Configuration register.
bbbbbbbb	Specifies the TMI Status register.