162Bug Diagnostics User's Manual

V162DIAA/UM1

Notice

While reasonable efforts have been made to assure the accuracy of this document, Motorola, Inc. assumes no liability resulting from any omissions in this document, or from the use of the information obtained therein. Motorola reserves the right to revise this document and to make changes from time to time in the content hereof without obligation of Motorola to notify any person of such revision or changes.

It is possible that this publication may contain reference to, or information about Motorola products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that Motorola intends to announce such Motorola products, programming, or services in your country.

Restricted Rights Legend

If the documentation contained herein is supplied, directly or indirectly, to the U.S. Government, the following notice shall apply unless otherwise agreed to in writing by Motorola, Inc.

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

Motorola, Inc.
Computer Group
2900 South Diablo Way
Tempe, Arizona 85282

Preface

The *162Bug Diagnostics User's Manual* provides information on using the 162Bug diagnostics.

This edition (162DIAA/UM1) applies to 162Bug versions 2.2 and up only, and is usable with all versions of the MVME162 series of microcomputers.

Use of the debugger, the debugger command set, use of the one-line assembler/disassembler, and system calls for the Debugging Package are all contained in the *Debugging Package for Motorola 68K CISC CPUs User's Manual (68KBUG1/Dx and 68KBUG2/Dx)*.

This manual is intended for anyone who designs OEM systems, supplies additional capability to an existing compatible system, or uses the 162Bug for experimental purposes. A basic knowledge of computers and digital logic is assumed.

In addition, commands that act on words or longwords over a range of addresses may truncate the selected range so as to end on a properly aligned boundary.

To use this manual, you should be familiar with the publications listed in the *Related Documentation* section in Appendix A of this manual.

Conventions

The following conventions are used in this document:

bold

is used for user input that you type just as it appears. Bold is also used for commands, options and arguments to commands, and names of programs, directories, and files.

italic

is used for names of variables to which you assign values. Italic is also used for comments in screen displays and examples.

courier

is used for system output (e.g., screen displays, reports), examples, and system prompts.

<Return>

represents the Enter or Return key.

CTRL

represents the Control key. Execute control characters by pressing the **CTRL** key and the letter simultaneously, e.g., **CTRL-d**.

Manual Terminology

Throughout this manual, a convention has been maintained whereby data and address parameters are preceded by a character which specifies the numeric format as follows:

$	hexadecimal character
%	binary number
&	decimal number

Unless otherwise specified, all address references are in hexadecimal throughout this manual.

An asterisk (*) following the signal name for signals which are *level significant* denotes that the signal is *true* or valid when the signal is low.

An asterisk (*) following the signal name for signals which are *edge significant* denotes that the actions initiated by that signal occur on high to low transition.

In this manual, *assertion* and *negation* are used to specify forcing a signal to a particular state. In particular, *assertion* and *assert* refer to a signal that is active or *true*; *negation* and *negate* indicate a signal that is inactive or *false*. These terms are used independently of the voltage level (high or low) that they represent.

Data and address sizes are defined as follows:

- A *byte* is eight bits, numbered 0 through 7, with bit 0 being the least significant.
- A *word* is 16 bits, numbered 0 through 15, with bit 0 being the least significant.
- A *longword* is 32 bits, numbered 0 through 31, with bit 0 being the least significant.

Safety Summary
Safety Depends On You

The following general safety precautions must be observed during all phases of operation, service, and repair of this equipment. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the equipment. Motorola, Inc. assumes no liability for the customer's failure to comply with these requirements.

The safety precautions listed below represent warnings of certain dangers of which Motorola is aware. You, as the user of the product, should follow these warnings and all other safety precautions necessary for the safe operation of the equipment in your operating environment.

Ground the Instrument.

To minimize shock hazard, the equipment chassis and enclosure must be connected to an electrical ground. The equipment is supplied with a three-conductor ac power cable. The power cable must either be plugged into an approved three-contact electrical outlet or used with a three-contact to two-contact adapter, with the grounding wire (green) firmly connected to an electrical ground (safety ground) at the power outlet. The power jack and mating plug of the power cable meet International Electrotechnical Commission (IEC) safety standards.

Do Not Operate in an Explosive Atmosphere.

Do not operate the equipment in the presence of flammable gases or fumes. Operation of any electrical equipment in such an environment constitutes a definite safety hazard.

Keep Away From Live Circuits.

Operating personnel must not remove equipment covers. Only Factory Authorized Service Personnel or other qualified maintenance personnel may remove equipment covers for internal subassembly or component replacement or any internal adjustment. Do not replace components with power cable connected. Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, always disconnect power and discharge circuits before touching them.

Do Not Service or Adjust Alone.

Do not attempt internal service or adjustment unless another person capable of rendering first aid and resuscitation is present.

Use Caution When Exposing or Handling the CRT.

Breakage of the Cathode-Ray Tube (CRT) causes a high-velocity scattering of glass fragments (implosion). To prevent CRT implosion, avoid rough handling or jarring of the equipment. Handling of the CRT should be done only by qualified maintenance personnel using approved safety mask and gloves.

Do Not Substitute Parts or Modify Equipment.

Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification of the equipment. Contact your local Motorola representative for service and repair to ensure that safety features are maintained.

Dangerous Procedure Warnings.

Warnings, such as the example below, precede potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed. You should also employ all other safety precautions which you deem necessary for the operation of the equipment in your operating environment.

WARNING

Dangerous voltages, capable of causing death, are present in this equipment. Use extreme caution when handling, testing, and adjusting.

The computer programs stored in the Read Only Memory of this device contain material copyrighted by Motorola Inc., 1995, and may be used only under a license such as those contained in Motorola's software licenses.

The software described herein and the documentation appearing herein are furnished under a license agreement and may be used and/or disclosed only in accordance with the terms of the agreement.

The software and documentation are copyrighted materials. Making unauthorized copies is prohibited by law. No part of the software or documentation may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means without the prior written permission of Motorola, Inc.

Disclaimer of Warranty

Unless otherwise provided by written agreement with Motorola, Inc., the software and the documentation are provided on an "as is" basis and without warranty. This disclaimer of warranty is in lieu of all warranties whether express, implied, or statutory, including implied warranties of merchantability or fitness for any particular purpose.

WARNING

This equipment generates, uses, and can radiate electromagnetic energy. It may cause or be susceptible to electromagnetic interference (EMI) if not installed and used in a cabinet with adequate EMI protection.

Printed in the United States of America
September 1995

Contents

General Information 1

Description of 162Bug

162Bug is a member of the M68000 firmware family. It is implemented on the MVME162 series of MC68040 and MC68LC040-based embedded controllers. 162Bug operates on the MVME162 (MVME162-0*xx*), MVME162LX (MVME162-2*xx*), MVME162FLX (MVME162-4xx), and MVME162FX (MVME162-5*xx*) modules. 162Bug consists of three parts:

- A command-driven, user-interactive software debugger. 162Bug performs its various operations in response to user commands entered at the keyboard. It is described in the *Debugging Package for Motorola 68K CISC CPUs User's Manual*, and is hereafter referred to as the *debugger*.
- A command-driven diagnostic package for the MVME162 modules, described in chapters 2 and 3, and which are hereafter referred to as the *diagnostics*.
- MPU, firmware, and hardware initialization routines, which are described in the *Debugging Package for Motorola 68K CISC CPUs User's Manual*.

Debug and Diagnostic Commands

There are three types of commands: debugger commands, diagnostic commands, and diagnostic tests. In addition, the execution of the diagnostic commands and tests may be modified by using command prefixes. The diagnostic commands and prefixes are described in Chapter 2. The diagnostic tests are described in Chapter 3. The debugger commands are described in the *Debugging Package for Motorola 68K CISC CPUs User's Manual*.

When you are running the diagnostics, the `162-Diag>` prompt appears. You have access to the diagnostics and debugger commands. If you are running the debugger (`162-Bug>` prompt), you have access to the debugger commands only. Switch to the diagnostics by entering the debugger **SD** (Switch directories) command.

162Bug Implementation

On the MVME162-0*xx* modules, 162Bug is installed in the 28F008SA FLASH memory. The FLASH devices provide 512KB (128K longwords) of storage. Optionally, 162Bug can be loaded and executed in a single 27C040 PLCC PROM.

On the MVME162-4*xx* and MVME162-5*xx* modules, 162Bug is installed in 28F008SA FLASH memory devices. The FLASH devices provide 1MB of storage. Optionally, 162Bug can be loaded and executed in a single 27C040 PLCC PROM.

On the MVME162-2*xx* modules, 162Bug is contained in a single 27C040 DIP EPROM installed in socket XU24.

User Interface

The firmware user interface allows users to run commands and tests from the command prompt. The interface reports results to the console terminal. This interface is command line driven and provides input/output facilities, command parsing, error reporting, and interrupt handling. The user interface is similar to those in existing diagnostic packages.

Language

The C programming language is used for most 162Bug modules. The CPU-specific low-level hardware interface code is written in assembly language.

Installation

Set-up and install the MVME162 module per the installation procedures the *Debugging Package for Motorola 68K CISC CPUs User's Manual* and your MVME162 installation manual. This section includes additional information that affects the operation of the 162Bug diagnostics and debugger.

Jumper Settings

162Bug defines the four lower order bits (GPI3 to GPI0) on the General Purpose Readable Header (J11 on MVME162-2*xx* or J22 on MVME162-0*xx*, MVME162-4*xx*, and MVME162-5*xx*), as listed below:

Bit	J11 Pins 2*xx*	J22 Pins 0*xx*, 4*xx*, 5*xx*	Description
0 (GPI0)	1-2	15-16	If this bit is a one (high), the debugger uses local Static RAM for its work page, i.e., variables, stack, vector tables, etc.
1 (GPI1)	3-4	13-14	If this bit is a one (high), the debugger uses the default setup and operation parameters in ROM versus the user setup and operation parameters in Non-Volatile RAM (NVRAM). This is the same as depressing the RESET and ABORT switches at the same time. This feature can be used in the event the user setup is corrupted or does not meet a sanity check. Refer to the **ENV** parameters in Chapter 4 for the ROM defaults.
2 (GPI2)	5-6	11-12	Reserved for future use.
3 (GPI3)	7-8	9-10	If this bit is a zero (low), the debugger executes out of the FLASH memory. If this bit is a one (high), the debugger executes out of the PROM.

Bit	J11 Pins 2*xx*	J22 Pins 0*xx*, 4xx, 5*xx*	Description
4 (GPI4)	9-10	7-8	User defined
5 (GPI5)	11-12	5-6	User defined
6 (GPI6)	13-14	3-4	User defined
7 (GPI7)	15-16	1-2	User defined

The default setting for the MVME162-0*xx*, MVME162-4xx, and MVME162-5*xx* is with all eight jumpers installed. The default setting for the MVME162-2*xx* with all jumpers installed across all pin pairs except pins 7 and 8.

The jumpers can be read as a register (at $FFF4202D) on the Memory Controller Chip (MCchip) ASIC. The bit values are one when the jumper is off and zero when the jumper is on. Jumper block J11 or J22 contains eight bits. Refer also to the *MVME162/MVME162FX/MVME162LX Embedded Controller Programmer's Reference Guide* for more information on the MCchip.

MVME162-2*xx*

Set the jumpers on the EPROM/Flash header J12 connecting pins 5 and 6, 8 and 10, and 9 and 11. This sets it up for 512K x 8 EPROMs.

MVME162-0*xx*, MVME162-4*xx*, and MVME162-5*xx*

If using a PROM version of the 162Bug, install the PROM device in socket U47, if it is not already installed. Be sure that the physical chip orientation is correct, that is, with the notched corner of the PROM aligned with the corresponding portion of the PROM socket on the MVME162 module.

System Console

Connect the 162Bug system console to Serial Port 1 on the front panel of the MVME162. Refer to your MVME162 installation manual for other connection option details.

Set up the terminal as follows:

- Eight bits per character
- One stop bit per character
- Parity disabled (no parity)
- Baud rate 9600 baud (default baud rate of MVME162FX ports at power-up)

You may reconfigure the baud rate of the debug port with the **PF** debugger command.

Note In order for high-baud rate serial communication between 162Bug and the terminal to work, the terminal must do some form of handshaking. If the terminal being used does not do hardware handshaking via the CTS line, then it must do XON/XOFF handshaking. If you get garbled messages and missing characters, then you should check the terminal to make sure XON/XOFF handshaking is enabled.

Start-up

When 162Bug is brought up at either power up or RESET, the following is displayed on the system console:

```
Copyright Motorola Inc. 1988 - 1994, All Rights Reserved

MVME162 Debugger/Diagnostics Release Version x.x - mm/dd/yy
COLD Start

Local Memory Found =00400000 (&4194304)

MPU Clock Speed =50Mhz

162-Bug>
```

At the `162-Bug>` prompt, enter **SD** to switch to the diagnostics prompt (`162-Diag>`).

You may use the **ENV** debugger command to change the environment so the firmware displays the System Menu or the debugger prompt in place of booting the system.

Refer to the *Debugging Package for Motorola 68K CISC CPUs User's Manual* for more information on using the debugger and the Field Service Menu.

The start-up and boot-load sequence is shown in Figure 1-1.

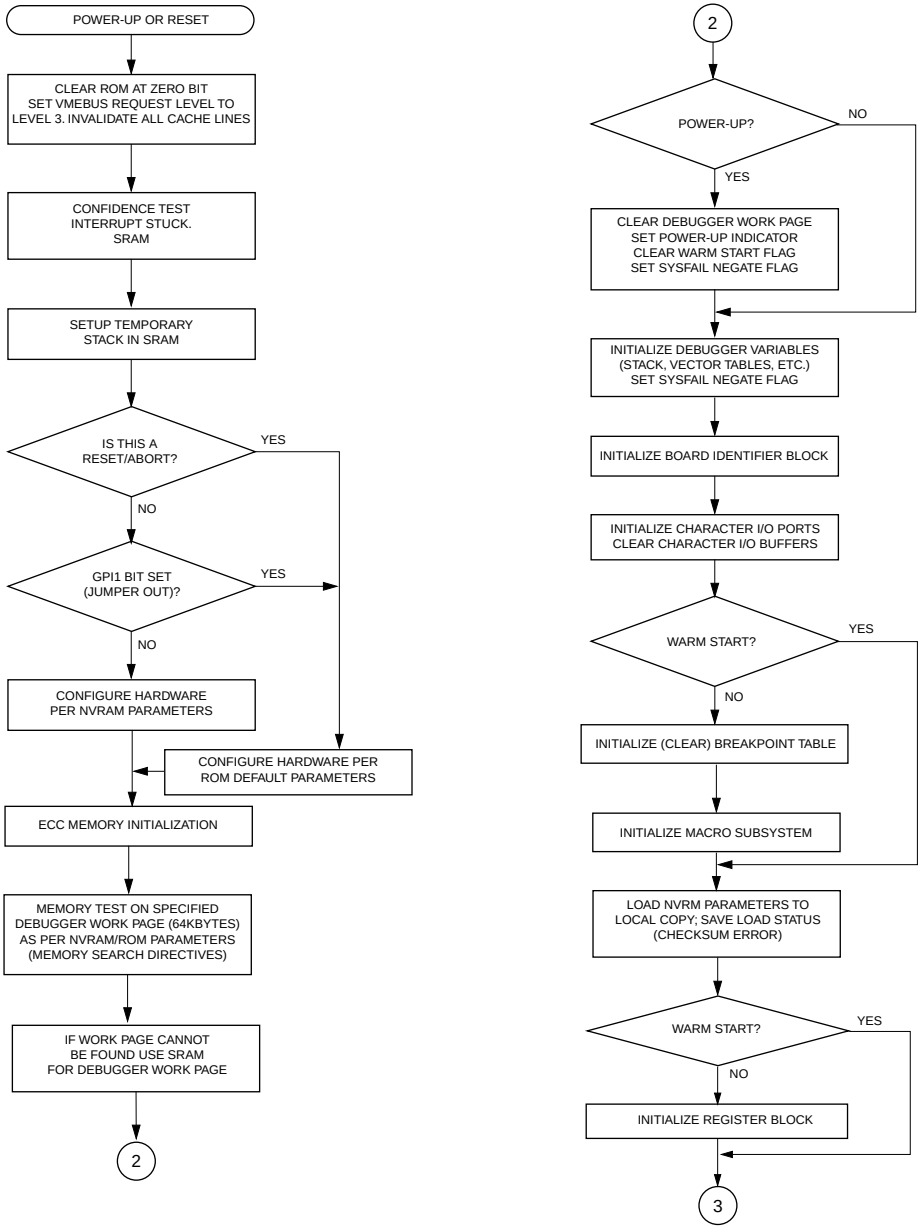

Figure 1-1. 162Bug Start-up Flow (Sheet 1 of 3)

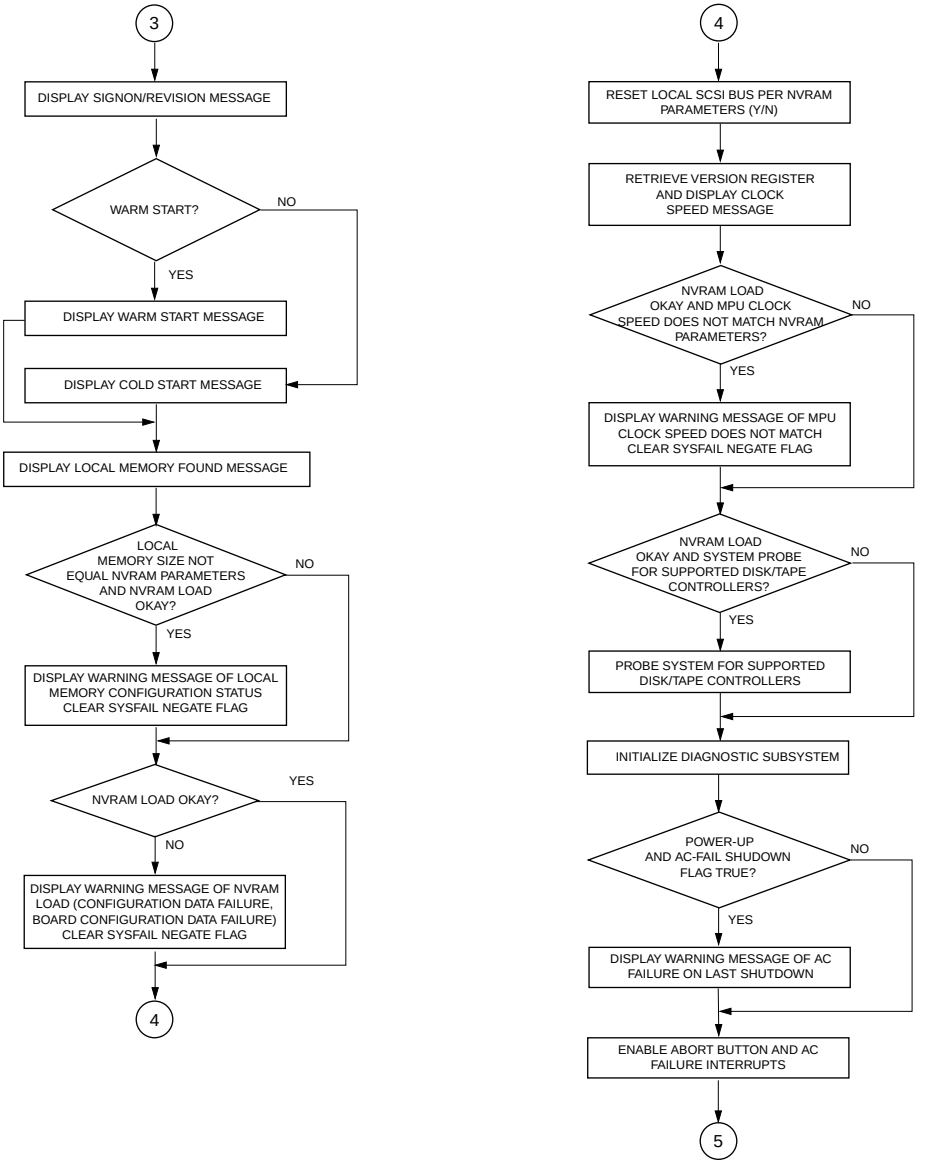

Figure 1-1. 162Bug Start-up Flow (Sheet 2 of 3)

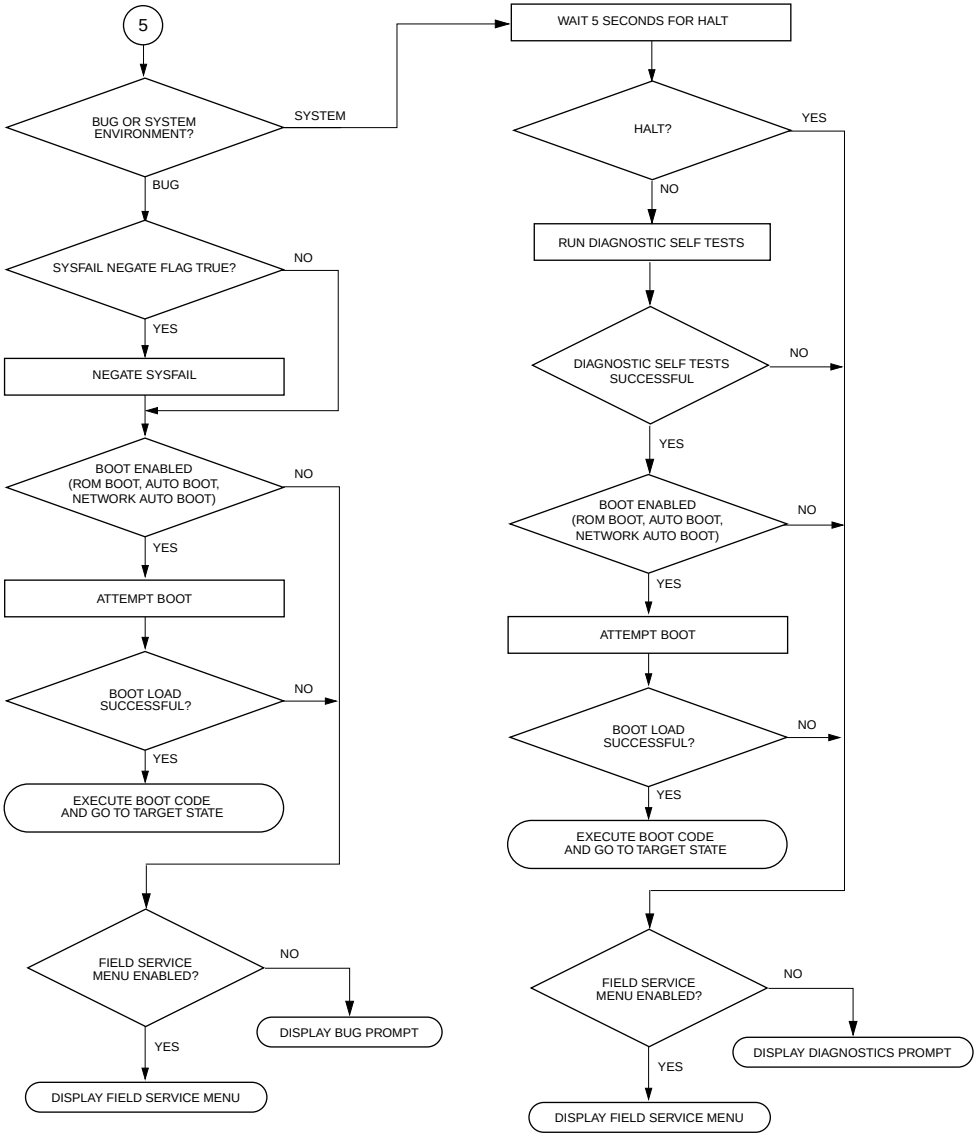

Figure 1-1. 162Bug Start-up Flow (Sheet 3 of 3)

ROMboot

On MVME162-0*xx*, MVME162-4*xx*, and MVME162-5*xx* modules, 162Bug occupies the first half of the FLASH memory when shipped from the factory. This leaves the second half of the FLASH memory and the PROM socket (U47) available for your use. The 162Bug is also available in PROM if your application requires all of the FLASH memory. Contact your Motorola sales office for assistance.

On the MVME162-2*xx* modules, 162Bug occupies an EPROM installed in socket XU24, leaving three sockets available (XU21 - XU23) and the FLASH.

Memory Requirements

162Bug is approximately 512KB of code, which is contained entirely in FLASH or PROM.

162Bug executes from address $FF800000 whether in FLASH or PROM. For MVME162-0*xx*, MVME162-4*xx*, and MVME162-5*xx* modules, a jumper is installed on General Purpose Readable Header J22 pins 9-10. The FLASH memories appear at address $FF800000 and are the parts executed during reset. With this configuration, the PROM socket is mapped to address $FFA00000. If you remove the jumper at J22 pin 9 and 10, the address spaces of the FLASH and PROM are swapped.

For the MVME162-2*xx*, the jumper is absent from General Purpose Readable Header J11 pins 7 and 8. 162Bug operates out of EPROM.

The 162Bug initial stack completely changes 8KB of DRAM memory at addresses offset $C000 from the base address, at power-up or reset. The DRAM and SRAM base addresses are shown in Table 1-1.

DRAM is neither ECC or parity type, but unprotected. DRAM mezzanine is mapped in contiguously starting at zero ($00000000), largest first. With two mezzanines of the same size, ECC type DRAM is first. If both are ECC type, the bottom one is first.

Table 1-1. DRAM and SRAM Base Addresses

Type of Memory	Default DRAM Base Address	Default SRAM Base Address
A single DRAM mezzanine	$00000000	$FFE00000 (onboard SRAM)
A single SRAM mezzanine	N/A	$00000000
A DRAM mezzanine stacked with an SRAM mezzanine	$00000000	$E1000000
Two DRAM mezzanines stacked	$00000000	$FFE00000 (onboard SRAM)

162Bug requires 2KB of NVRAM for storage of board configuration, communication, and booting parameters. This storage area begins at $FFFC16F8 and ends at $FFFC1EF7.

162Bug requires a minimum of 64KB of contiguous read/write memory to operate. The **ENV** command controls where this block of memory is located. Regardless of where the onboard RAM is located, the first 64KB is used for 162Bug stack and static variable space and the rest is reserved as user space.

The following occurs whenever the MVME162 board is reset:

- Target PC is initialized to the address corresponding to the beginning of the user space
- Target stack pointers are initialized to addresses within the user space
- Target Interrupt Stack Pointer (ISP) set to the top of the user space

Using the Diagnostics 2

Introduction

This chapter contains information about entering the 162Bug diagnostic commands and tests. The diagnostic commands and test prefixes are also described in this chapter. The diagnostic tests are described in Chapter 3.

Running Commands

When using 162Bug, you operate the debugger or the diagnostics. If you are in the debugger, the prompt `162-Bug>` is displayed and you have all of the debugger commands at your disposal. If you are in the diagnostics, the prompt `162-Diag>` is displayed and you have all of the diagnostic commands, diagnostic tests, and debugger commands at your disposal. You may switch between the diagnostics and the debugger by using the **SD** command.

Set the parameters that control the operation of all tests in a test group, such as memory range, with the **CF** command.

You may view a list of the debugger or diagnostics commands and test groups by using the **HE** command (when at the diagnostics prompt, **HE** does not list the debugger commands even thought those commands are available).

Command Entry

To execute a command, enter the command at the `162-Diag>` prompt and press the Return key. 162Bug executes the command and the prompt reappears.

You may enter multiple commands on one line. If a command expects parameters and another command is to follow it, separate the two with a semicolon (**;**).

For instance, to invoke the command **RTC CLK** after the command **RAM ADR**, you may enter **RAM ADR ; RTC CLK** on the command line.

Test prefixes are available to modify the execution of a test. Insert a semicolon between the prefix and the test that it modifies. For instance **LF ; RAM** (spaces are not required before or after the semicolon).

Diagnostic Commands

The diagnostic package supports the root-level commands and general commands, which are listed in the table below and described on the following pages.

Table 2-1. Diagnostic Commands

Command	Description
AEM	Append Error Messages Mode
CEM	Clear Error Messages
CF	Test Group Configuration Parameters Editor
DE	Display Error Counters
DEM	Display Error Messages
DP	Display Pass Count
HE	Help
HEX	Help Extended
MASK	Self Test Mask
SD	Switch Directories
ST	Self Test
ZE	Clear (Zero) Error Counters
ZP	Zero Pass Count

AEM - Append Error Messages Mode

The **AEM** command allows you to accumulate error messages in the internal error message buffer of the diagnostics. The **AEM** command sets the internal append error messages flag of the diagnostics. When the internal append error messages flag is clear, the diagnostic error message buffer is erased (cleared of all character data) before each test is executed. The duration of this command is for the life of the command line being parsed by the diagnostics. The default of the internal append error messages flag is clear. The internal flag is not set until it is encountered in the command line by the diagnostics.

CEM - Clear Error Messages

The **CEM** command allows you to clear the internal error message buffer of the diagnostics manually.

CF - Test Group Configuration Parameters Editor

The **CF** command allows you to modify the parameters that control the operation of the diagnostic tests. For example, the **RAM** test group has parameters for the starting address, ending address, and parity enable that you can set with the **CF** command.

The **CF** command prompts you with the parameter and the current value. You may enter a new value for that parameter, or press the Return key leave the parameter unchanged.

You may enter one or more test groups as argument(s) to the **CF** command. Only the parameters for those tests will be displayed. If no test group name is entered, the parameters for all test groups are displayed.

At the time of initial execution of the diagnostic tests, the default configuration parameters are copied from the firmware into the debugger work page.

DE - Display Error Counters

The **DE** command displays all errors in the test error counters. Each test or command in the diagnostics has an individual error counter. As errors are encountered in a particular test, that error counter is incremented. If you were to run a self-test or just a series of tests, the results could be broken down as to which tests passed by examining the error counters.

To view the errors of an individual test, enter the full test name after the **DE** command. For example, to view errors from the test error counter on RAM Code Execution/Copy test routine, enter **DE RAM CODE**.

Only nonzero values are displayed.

DEM - Display Error Messages

The **DEM** command displays the internal error message buffer of the diagnostics.

DP - Display Pass Count

The **DP** command displays a count of the number of passes of tests run in Loop-Continue (**LC**) mode.

HE - Help

The **HE** command displays the available diagnostic commands, test groups, and test prefixes. The character string `(DIR)` appears after a test group name. If there are more entries than fit on the screen, the message `Press "RETURN" to continue` appears.

HE does not list the debugger commands even thought those commands are available from the `162-Diag>` prompt.

To view the tests in a test group, enter the test group name after the **HE** command. For example, to list all the RAM tests, enter **HE RAM**.

To view a description of an individual test, enter the full test name. For example, to view information on the RAM Code Execution/Copy test routine, enter **HE RAM CODE**.

The following is an example of the **HE** command:

```
162-Diag>HE
AEM       Append Error Messages Mode
CEM       Clear Error Messages
CF        Configuration Editor
CMMU      Cache/Memory Management Unit Tests (DIR)
DE        Display Errors
DEM       Display Error Messages
DP        Display Pass Count
FLASH     Flash Memory Tests (DIR)
HE        Help on Tests/Commands
HEX       Help Extended
IPIC      IP Interface Controller (IPIC ASIC) Tests (DIR)
LA        Loop Always Mode
LANC      LAN Coprocessor (Intel 82596) Tests (DIR)
LC        Loop Continuous Mode
LE        Loop on Error Mode
LF        Line Feed Mode
LN        Loop Non-Verbose Mode
MASK      Self Test Mask
MCECC     ECC Memory Board Diagnostics (DIR)
NCR       NCR 53C710 SCSI I/O Processor Test (DIR)
NV        Non-Verbose Mode
RAM       Random Access Memory Tests (DIR)
RTC       MK48T0x Timekeeping (DIR)
Press "RETURN" to continue

SCC       Serial Communication Controller (Z85230) Tests (DIR
SE        Stop on Error Mode
SRAM      Static Random Access Memory Tests (DIR)
ST        Self Test (DIR)
ST2401    CD2401 Serial Self-Tests (DIR)
VME2      VME2Chip2 Tests (DIR)
ZE        Zero Errors
ZP        Zero Pass Count
162-Diag>
```

HEX - Interactive Help

The **HEX** command enters a continuous interactive mode of the **HE** command. When you execute **HEX**, the question mark (?) is displayed as a prompt. You may then enter the name of a test group or diagnostic command. Type **QUIT** to return to the diagnostics prompt.

MASK - Self Test Mask

The **MASK** command enables or disables a test from running as part of the start-up diagnostic self tests or when executing the **ST** command. The **MASK** command "toggles" the test's state. If the specified test is enabled, it will be disabled by running **MASK**; if the is disabled, it will be enabled. The default for a test is the enabled state. The mask values are saved in non-volatile memory.

The syntax is **MASK** *TEST NAME*, where *TEST NAME* is the full name of a diagnostic test. For example, to disable the **RAM CODE** test, enter **MASK RAM CODE**.

If the **MASK** command is invoked with an invalid test group name, an appropriate error message is displayed.

To display the current disabled tests, invoke **MASK** without a test name. A list of disabled (masked) tests is also displayed each time the command is run for a test.

SD - Switch Directories

Use the **SD** command to toggle between the diagnostic and debugger directories. When you are running the diagnostics, the `162-Diag>` prompt appears. All of the debugger and diagnostics commands are available. When you are running the debugger, the prompt is `162-Bug>`, and only the debugger commands are available.

ST - Self Test

The **ST** command runs the system self tests that the bug runs at system start-up. The command **HE ST** lists the test groups that are run with the self tests.

This command is useful for debugging board failures that may require running the test suite while using the debugger. Upon completion of running the test suite, the debugger prompt is displayed.

ZE - Clear Error Counters

The **ZE** command resets all of the error counters to zero. The error counters are initialized with the value of zero. After tests run, it may desirable to reset them to zero.

To clear the error counter for a particular test, enter the test name with the **ZE** command. For example, **ZE VME2 TMRA** clears the error counter for **VME2 TMRA**.

ZP - Zero Pass Count

The **ZP** command resets the pass counter to zero. This is frequently desirable before using the Loop Continue mode.

To reset the counter at each pass of a particular test, enter the **ZP** command on the same line as **LC** and the test. For example, **ZP LC VME2 TMRA**.

Test Prefixes

The tests execution can be modified with the prefixes, which are listed in Table 2-2 and are described on the following pages.

Table 2-2. Diagnostic Command Prefixes

Prefix	Description
LA	Loop Always Mode
LC	Loop-Continue Mode
LE	Loop-On-Error Mode
LF	Line Feed Suppression Mode
LN	Loop Non-Verbose Mode
NV	Non-Verbose Mode
SE	Stop-On-Error Mode

LA - Loop Always

The **LA** prefix causes a failed test or series of failed tests to be re-executed endlessly. To break the loop, press the BREAK key. Certain tests disable the BREAK key interrupt, so it may be necessary to press the ABORT or RESET switches on the MVME162 front panel.

LC - Loop-Continue

The **LC** prefix causes a test or series of tests to be re-executed endlessly. To break the loop, press the BREAK key. Certain tests disable the BREAK key interrupt, so it may be necessary to press the ABORT or RESET switches on the MVME162 front panel.

LE - Loop-On-Error

The **LE** prefix causes a test to be re-executed if the previous execution returns a failure status. To break a loop, press the BREAK key. Certain tests disable the BREAK key interrupt, so it may be necessary to press the ABORT or RESET switches on the MVME162 front panel.

The **LE** prefix is useful to endlessly repeat (loop) a test when an oscilloscope or logic analyzer is in use.

LF - Line Feed Suppression

The **LF** prefix toggles the internal line feed mode flag of the diagnostics. The default state of the internal line feed mode flag is clear which causes the executing test title/status line(s) to be terminated with a line feed character (scrolled). The line feed mode flag is normally used by the diagnostics when executing a system self test.

LN - Loop Non-Verbose

The **LN** prefix causes the test to be re-executed endlessly, and suppresses display of the test title and pass/fail status. This is useful for more rapid execution of the failing test.

NV - Non-Verbose

The **NV** prefix suppresses the display of test status and error data. Only the test name and result (`PASSED` or `FAILED`) are listed.

SE - Stop-On-Error

The **SE** prefix stops a test or series of tests when an error is detected.

Diagnostic Tests 3

Introduction

This chapter contains detailed descriptions of the 162Bug diagnostic tests. The test sets are shown in Table 3-1.

Table 3-1. Diagnostic Test Groups

Test Group	Description
RAM	Local RAM Tests
SRAM	Static RAM Tests
RTC	MK48T0x Real-Time Clock Tests
MCC	Memory Controller Chip Tests
MCECC	Memory Board Tests
CMMU	Cache and Memory Management Unit Tests
VME2	VME Interface ASIC VMEchip2 Tests
LANC	LAN Coprocessor (Intel 82596) Tests
NCR	NCR 53C710 SCSI I/O Processor Tests
IPIC	IndustryPack Interface Chip Tests
SCC	Serial Communication Controller (Z85230) Tests
FLASH	Flash Memory Tests

Running the Tests

The diagnostic test commands consist of a test group name and a test name. To run a test, enter the test group name and the test name on the command line. For instance, **RAM** is a test group, and **ADR** is the name of a test in the group. To invoke the **ADR** test, enter **RAM ADR** on the command line.

To run all tests in a test group, enter the test group name without any test names (the **FLASH** tests must be run individually).

You may enter any number or sequence of tests after the test group name as long as the bug's input buffer size limit is not exceeded.

Upon execution of a test, a status message appears indicating the test name and the current status. For example, the following message appears for the **RAM CODE** test:

```
RAM   CODE: Code Execution/Copy ..... Running --->
```

If all parts of the test pass, `PASSED` appears at the end of the message line. If any part of the test fails, `FAILED` appears at the end of the message line, followed by one or more error messages.

RAM - Local RAM, SRAM - Static RAM

The **RAM** tests check the local RAM and the **SRAM** tests check the Static RAM.

The **RAM** and **SRAM** tests are listed in Table 3-2, and are described in alphabetical order on the following pages. The **RAM** and **SRAM** tests are identical in function.

Enter **RAM** or **SRAM** without a test name to run all tests in the group (**PED** and **REF** do not run with the **SRAM** test group). They will be executed in the order shown in Table 3-2.

Table 3-2. RAM and SRAM Tests

Test	Description
QUIK	Quick Write/Read
ALTS	Alternating Ones/Zeros
PATS	Data Patterns
ADR	Memory Addressing
CODE	Code Execution/Copy
PERM	Permutations
RNDM	Random Data
BTOG	Bit Toggle
PED	Parity Error Detection
REF	Memory Refresh

Configuration Parameters

You may set the following parameters with the **CF** command (the default values are shown):

```
Starting/Ending Address Enable [Y/N] =N ?
Starting Address =00000000 ? (FFE00000 for SRAM)
Ending Address   =01000000 ? (FFE1FFFC for SRAM)
Random Data Seed =12301983 ?
March Address Pattern =00000000 ?
Instruction (Code) Cache Enable [Y/N] =Y ?
Parity Enable [Y/N] =Y ?
MCECC Error Correction Enable [Y/N] =N ?
Parity Interrupt Enable [Y/N] =Y ?
Parity Error Detection Test Address Increment =00001001 ?
Break Key Check Delay Counter =00000100 ?
```

ADR - Memory Addressing

The **ADR** test verifies addressing of memory in the range specified by the configuration parameters for the **RAM** test group. Addressing errors are sought by using a memory location's address as the data for that location. This test is coded to use only 32-bit data entities.

The test runs as follows:

1. A Locations Address is written to its location (*n*).
2. The next location (*n*+4) is written with its address complemented.
3. The next location (*n*+8) is written with the most significant 16 bits and least significant 16 bits of its address swapped with each other.
4. Steps 1, 2, and 3 are repeated throughout the specified memory range.
5. The memory is read and verified for the correct data pattern(s) and any errors are reported.
6. The test is repeated (steps 1 through 5) except that inverted data is used to insure that every data bit is written and verified at both "0" and "1".

Command Input

```
162-Diag>RAM ADR
```

or

```
162-Diag>SRAM ADR
```

Messages

If the test fails, the following message appears:

```
Data Miscompare Error:
Address =________, Expected =________, Actual =________
```

ALTS - Alternating Ones/Zeros

This test verifies addressing of memory in the range specified by the configuration parameters for the **RAM** test group. Addressing errors are sought by using a memory locations address as the data for that location. This test is coded to use only 32-bit data entities.

The test runs as follows:

1. Location (*n*) is written with data of all bits 0.
2. The next location (*n*+4) is written with all bits 1.
3. Steps 1 and 2 are repeated throughout the specified memory range.
4. The memory is read and verified for the correct data pattern(s) and any errors are reported.

Command Input

```
162-Diag>RAM ALTS
```

or

```
162-Diag>SRAM ALTS
```

Messages

If the test fails, the following message appears:

```
Data Miscompare Error:
Address =________, Expected =________, Actual =________
```

BTOG - Bit Toggle

This test toggles the bits in the memory range specified by the configuration parameters for the **RAM** test group. The **RAM** test group configuration parameters also determine the value of the global random data seed used by this test. The global random data seed is incremented after it is used by this test.

This test uses the following test data pattern generation algorithm:

1. The random data seed is copied into a work register.
2. Work register data is shifted right one bit position.
3. The random data seed is added to work register using unsigned arithmetic.
4. Data in the work register may or may not be complemented.
5. Data in the work register is written to current memory location.

If the **RAM** test group configuration parameter for code cache enable equals "Y", the microprocessor code cache is enabled. This test is coded to operate using the 32-bit data size only.

The test runs as follows:

1. The memory locations are written with the test data pattern.
2. The memory locations are written with the complement of the test data pattern complemented.
3. The memory under test is read back to verify that the complement test data is properly retained.
4. The memory locations are written with the test data pattern.
5. The memory under test is read back to verify that the test data is properly retained.

Command Input

```
162-Diag>RAM BTOG
```

or

```
162-Diag>SRAM BTOG
```

Messages

If the test fails, the following message appears:

```
Data Miscompare Error:
Address =________, Expected =________, Actual =________
```

CODE - Code Execution/Copy

This test copies test code to a memory location and executes the code. The test code copies itself to the next higher memory address and executes the new copy. This process is repeated until there is not enough memory, as specified by the configuration parameters, to perform another code copy and execution.

A hardware reset is required to recover if the test fails (MPU irrecoverably crashes).

Command Input

```
162-Diag>RAM CODE
```

or

```
162-Diag>SRAM CODE
```

Messages

If the test passes, the PASSED message appears. If the PASSED message does not appear within a minute of executing the test, the test has failed (the FAILED message does not appear).

PATS - Data Patterns

This test writes and reads a series of test patterns to the test memory range. Each location is filled with all ones ($FFFFFFFF). The test runs eight passes, one for each of the following data patterns:

$00000000
$01010101
$03030303
$07070707
$0F0F0F0F
$1F1F1F1F
$3F3F3F3F
$7F7F7F7F

During each pass of the test, each location is written with the current pattern and the 1's complement of the current pattern. Each write is read back and verified. each location is filled with the data pattern. This test is coded to use only 32-bit data entities.

If the test address range is less than 8 bytes, the test immediately returns pass status. The effective test range end address is reduced to the next lower 8-byte boundary if necessary.

Command Input

```
162-Diag>RAM PATS
```

or

```
162-Diag>SRAM PATS
```

Messages

If the test fails, the following message appears:

```
Data Miscompare Error:
Address =________, Expected =________, Actual =_________
```

PED - Local Parity Memory Error Detection

This test checks memory parity for memory range and address increment is specified by the **RAM** test group configuration parameters.

First, the test verifies a portion of each memory location to be tested by writing and verifying all zeros, and writing an verifying all ones. Each memory location is tested once with parity interrupt disabled, and once with parity interrupt enabled. Parity checking is enabled, and data is written and verified at the test location that causes the parity bit to toggle on and off (verifying that the parity bit of memory is good). Next, data with incorrect parity is written to the test location. The data is read, and if a parity error exception occurs, the fault address is compared to the test address. If the addresses are the same, the test passed and the test location is incremented until the end of the test range has been reached.

Command Input

```
162-Diag>RAM PED
```

or

```
162-Diag>SRAM PED
```

Messages

If a data verification error occurs, the following message appears:

```
RAM/PED Test Failure Data:

Data Miscompare Error:
Address =________, Expected =________, Actual =________
```

If an unexpected exception (parity error detected as the parity bit was being toggled), the following message appears:

```
RAM/PED Test Failure Data:

Unexpected Exception Error, Vector =________
Address Under Test =________
```

If no exception occurred when data with bad parity was read, the following message appears:

```
RAM/PED Test Failure Data:

Parity Error Detection Exception Did Not Occur

Exception Vector =________
Address Under Test =________
```

If the exception address was different from that of the test location, the following message appears:

```
RAM/PED Test Failure Data:

Fault Address Miscompare, Expected =________, Actual =______
```

PERM - Permutations

This test verifies that the memory in the test range can accommodate 8-, 16-, and 32-bit writes and reads in any combination. The test range is the memory range specified by the **RAM** test group configuration parameters for starting and ending address. If the test range is less than 16 bytes, the test immediately returns pass status. The effective test range end address is reduced to the next lower 16-byte boundary if necessary.

This test performs three data size test phases in the following order: 8, 16, and 32 bits. Each test phase writes a 16-byte data pattern (using its data size) to the first 16 bytes of every 256-byte block of memory in the test range. The 256-byte blocks of memory are aligned to the starting address configuration parameter for the **RAM** test group. The test phase then reads and verifies the 16-byte block using 8-, 16-, and 32-bit access modes.

Command Input

```
162-Diag>RAM PERM
```

or

```
162-Diag>SRAM PERM
```

Messages

If the test fails, the following message appears:

```
Data Miscompare Error:
Address =________, Expected =________, Actual =________
```

QUIK - Quick Write/Read

This test writes and reads a pair of test patterns, 0 and $FFFFFFFF, to the test memory range. Each pass of this test fills the test range with a data pattern by writing the current data pattern to each memory location from a local variable and reading it back into that same register. The local variable is verified to be unchanged only after the write pass through the test range.

This test uses only 32-bit data entities.

Command Input

```
162-Diag>RAM QUIK
```

or

```
162-Diag>SRAM QUIK
```

Messages

If the test fails, the following message appears:

```
Data Miscompare Error:
Expected =________, Actual =________
```

REF - Memory Refresh Testing

This test verifies memory locations after a refresh wait cycle. The memory range and address increment is specified by the RAM test group configuration parameters.

The test runs as follows:

1. The real time clock is checked to see if it is functioning properly.
2. Each memory location to be tested has the data portion verified by writing and verifying patterns of all zeros, and all ones.
3. A data pattern is written to the test location.
4. After all the data patterns are filled for all test locations, a refresh wait cycle is executed.
5. The data is read.

If the previously entered data pattern does not match the data pattern read in, a failure occurs. If the data patterns match, the test is passed.

Note **SRAM REF** will not execute because SRAM does not refresh.

Command Input

```
162-Diag>RAM REF
```

Messages

If the real time clock is not functioning properly, one of the following messages appear:

```
RAM/REF Test Failure Data:

RTC is stopped, invoke SET command.
```

or

```
RAM/REF Test Failure Data:

RTC is in write mode, invoke SET command.
```

or

```
RAM/REF Test Failure Data:

RTC is in read mode, invoke SET command.
```

If a data verification error occurs before the refresh wait cycle, the following message appears:

```
RAM/REF Test Failure Data:

Immediate Data Miscompare Error:

Address =________, Expected =________, Actual =________
```

If a data verification error occurs following the refresh wait cycle, the following message appears:

```
RAM/REF Test Failure Data:

Unrefreshed Data Miscompare Error:

Address =________, Expected =________, Actual =________
```

RNDM - Random Data

This test writes and verifies a random test patterns and the complement of the test pattern. The test memory range specified by the **RAM** test group configuration parameters.

The test runs as follows:

1. A random pattern is written throughout the test range.
2. The random pattern complemented is written throughout the test range.
3. The complemented pattern is verified.
4. The random pattern is rewritten throughout the test range.
5. The random pattern is verified.

This test uses only 32-bit data entities. Each time this test is executed, the random seed in the **RAM** test group configuration parameters is post incremented by 1.

Command Input

```
162-Diag>RAM RNDM
```

or

```
162-Diag>SRAM RNDM
```

Messages

If the test fails, the following message appears.

```
Data Miscompare Error:
Address =________, Expected =________, Actual =________
```

RTC - MK48T08 Real Time Clock

The **RTC** tests check the NVRAM, SRAM, and clock portions of the MK48T08 "Zeropower" Real Time Clock (RTC) chip. The tests are listed in Table 3-3, and are described in alphabetical order on the following pages.

Enter **RTC** without a test name to run all tests in the group. They will be executed in the order shown in Table 3-3.

Table 3-3. RTC Tests

Test	Description
CLK	Clock Function
RAM	Battery Backed-Up SRAM
ADR	BBRAM Addressing

Configuration Parameter

You may set the `Restore BBRAM contents on test exit` parameter, used in the **ADR** test, with the **CF** command (the default is Y).

ADR - NVRAM Addressing

This test checks the proper addressing of the MK48T08 NVRAM. The test runs as follows:

1. The NVRAM is filled with data pattern “a.”
2. A single address line of the MK48T0x is set to one, and pattern “b” is written to the resultant address.
3. All other locations in the NVRAM are checked to ensure that they were not affected by this write.
4. The “a” pattern is restored to the resultant address.

All address lines connected to the MK48T0x are tested in this manner.

Since this test overwrites all memory locations in the NVRAM, the NVRAM contents are saved in debugger system memory prior to writing the NVRAM. The RTC test group features a configuration parameter which overrides automatic restoration of the NVRAM contents. The default for this parameter is to restore NVRAM contents upon test completion.

Command Input

```
162-Diag>RTC ADR
```

Messages

If debugger system memory cannot be allocated for use as a save area for the NVRAM contents, the following message appears:

```
RAM allocate

memc.next=________ memc.size=________
```

If the NVRAM cannot be initialized with pattern “a,” the following message appears:

```
Data Verify Error: Address =________, Expected =__, Actual =__
Memory initialization error
```

If a pattern “b” write affects any NVRAM location other than the resultant address, the following message appears:

```
Data Verify Error: Address =________, Expected =__, Actual =__
Memory addressing error - wrote __ to ________
```

CLK - Check Real Time Clock

This test verifies that the RTC is operating. It does not check clock accuracy. This test requires approximately nine seconds to run. At the conclusion of the test, nine seconds are added to the clock time to compensate for the test delay. Because the clock can only be set to the nearest second, this test may induce up to one second of error into the clock time.

Note The Low Battery test only assures Battery OK if the MK48T02 (used on other boards) has not been written since powerup. The Battery test is performed here in case the debugger currently in use does not perform a Low Battery test on powerup. Although the MK48T08 does not support the internal battery voltage check (BOK), the BOK flag status check algorithm is performed by this test on all parts.

The RTC time registers are configured for constant updating by the clock internal counters. The seconds register is read initially and monitored (read) to verify that the seconds value changes. A predetermined number of reads are made of the seconds register.

The RTC time registers are configured for reading. A predetermined number of MPU "do nothing" loops are executed.

Command Input

```
162-Diag>RTC CLK
```

Messages

If the check for low battery fails, the following message appears:

```
RTC low battery
```

If the predetermined number of reads are made before the seconds register changed, the following message appears:

```
RTC not running
```

If the seconds register changes before the full count of MPU loops is executed, the following message appears:

```
RTC did not freeze for reading
```

If the real time clock registers fail the data pattern test:

```
Data Miscompare Error:
Address =________, Expected =________, Actual =________
```

If there is a a programming error, the following message appears:

```
WARNING -- Real Time Clock NOT compensated for test delay.
```

RAM - Battery Backed-Up SRAM

This test performs a data test on each SRAM location of the MK48T08 RAM. RAM contents are unchanged upon completion of test, regardless of pass or fail test return status. This test is coded to test only byte data entities.

The RAM test runs seven passes, once for each the following values: $1, $3, $7, $F, $1F, $3F, and $7F. During each pass:

1. The value is written to each valid byte of the MK48T08.
2. The value is verified.
3. The complement of each value is written to each valid byte of the MK48T08
4. The complement is verified.

Command Input

```
162-Diag>RTC RAM
```

Messages

If the test fails, the following message appears:

```
Data Miscompare Error:
Address =________, Expected =________, Actual =________
```

MCC - Memory Controller Chip

The **MCC** tests check the Memory Controller Chip, which is one of three ASICs that are part of the MVME162. The **MCC** tests are listed in Table 3-4, and are described in alphabetical order on the following pages.

Enter **MCC** to run all tests in the group (except **WDTMRC**). They will be executed in the order shown in Table 3-4.

Table 3-4. MCC Tests

Test Name	Description
ACCESSA	Device Access
ACCESSB	Register Access
TMR1A	Timer 1 Counter
TMR1B	Timer 1 Free-Run
TMR1C	Timer 1 Clear On Compare
TMR1D	Timer 1 Overflow Counter
TMR1E	Timer 1 Interrupts
TMR2A	Timer 2 Counter
TMR2B	Timer 2 Free-Run
TMR2D	Timer 2 Overflow Counter
TMR2E	Timer 2 Interrupts
TMR3A	Timer 2 Counter
TMR3B	Timer 2 Free-Run
TMR3C	Timer 2 Clear On Compare
TMR3D	Timer 2 Overflow Counter
TMR3E	Timer 2 Interrupts
TMR4A	Timer 2 Counter
TMR4B	Timer 2 Free-Run
TMR4C	Timer 2 Clear On Compare
TMR4D	Timer 2 Overflow Counter
TMR4E	Timer 2 Interrupts
ADJ	Prescaler Clock Adjust

Table 3-4. MCC Tests

Test Name	Description
PCLK	Prescaler Clock
MPUCS	MPU Clock Speed
RFRSH	DRAM Refresh Timing
FAST	FAST Bit
WDTMRA	Watchdog Timer Counter
WDTMRB	Watchdog Timer Board Fail
VBR	Vector Base Register
WDTMRC	Watchdog Timer Local Reset

Configuration Parameters

There are no configuration parameters for the **MCC** test group.

Bus Error and Unsolicited Exception Messages

The following error messages may apply to any of the tests in the **MCC** test group:

```
Bus Error Information:
                Address ________
                   Data ________
            Access Size __
            Access Type _
     Address Space Code _
          Vector Number ___

Unsolicited Exception:
        Program Counter ________
          Vector Number __
             Access Size ___
        Status Register ____
        Interrupt Level _
```

ACCESSA - Device Access

This test verifies that the MCC register set can be accessed (read) on byte, word, and long word boundaries (where applicable). No attempt is made to verify the contents of the registers.

Command Input

```
162-Diag>MCC ACCESSA
```

ACCESSB - Register Access

This test checks the device data lines by successive writes and reads to all tick timers compare and counter registers. The test walks a 1 bit through a field of zeros and walks a 0 bit through a field of ones.

Command Input

```
162-Diag>MCC ACCESSB
```

Messages

If the test fails, one of the following error messages appears:

```
Register did not clear
Address =________, Expected =________, Actual =________
```

```
Register access error
Address =________, Expected =________, Actual =________
```

ADJ - Prescaler Clock Adjust

This test verifies that the Prescaler Clock Adjust Register can vary the period of the Tick Timer input clock. The test runs as follows:

1. The Clock Adjust Register is set to zero
2. The Tick Timer 1 free-runs for a small software delay to establish a reference count
3. A 1 is walked through the Clock Adjust Register
4. The timer is allowed to run for the same delay period

The resulting count should be greater than the last (previous) count.

Command Input

```
162-Diag>MCC ADJ
```

Messages

If the test fails, one of the following error messages appears:

```
Prescaler Clock Adjust Register not initialized
Register Address =________, should not be zero

Clock Adjust did not vary tick period correctly
Register Address =________, Adjust Value =__
Test Count     =________, should be greater than
Previous Count =________
```

FAST - FAST Bit

This test verifies the FAST/SLOW access time to the BBRAM. This is accomplished by using Tick Timer #1. The test runs as follows:

1. The tick timer is first used to time 4000 accesses to the BBRAM with the FAST bit set
2. FAST bit is cleared
3. The tick timer is used to time 4000 accesses to the BBRAM

The count measured when the FAST bit is set should be less than the count measured when the FAST bit is cleared.

Command Input

```
162-Diag>MCC FAST
```

Messages

If the test fails, the following error message appears:

```
FAST' bit did not vary access time correctly
Fast access count =________, Slow access count =________
Fast count should be less than Slow count
```

MPUCS - MPU Clock Speed

This test verifies that the calculated MPU clock speed matches both the version register of the MCC and the MCC prescaler initialized value. The MPU clock speed calculation is done by using the RTC (MK48T08) and Tick Timer #1.

Command Input

```
162-Diag>MCC MPUCS
```

Messages

If the test fails, one of the following error messages appears:

```
MPU Clock Speed Calculation failed, RTC not operating

MPU Clock Speed Calculation does not match the MCC Version Register
Calculated MPU Clock Speed =__MHZ (&________HZ), Version Register =__MHZ

MPU Clock Speed Calculation does not match the MCC Prescaler Register
Calculated MPU Clock Speed =__MHZ (&________HZ), Prescaler Register =_MHZ

Unknown Prescaler Adjust Value =__
```

PCLK - Prescaler Clock

This test verifies the accuracy of the Prescaler Clock. This is accomplished by using a constant time source, in this case the MK48T08 RTC, and Tick Timer #1. The test runs as follows:

1. The constant time source is verified for operation
2. The tick timer is initialized and the constant time source is brought to a whole second interval
3. The tick timer is started and the constant time source is polled for the next second to roll over while the tick timer is free-running
4. Upon roll over, the tick timer is stopped
5. The elapsed tick timer count is verified

Acceptance of this count allows for a plus or minus 0.1 percent tolerance.

Command Input

```
162-Diag>MCC PCLK
```

Messages

If the test fails, one of the following error messages appears:

```
Unknown Prescaler Adjust Value =__

RTC seconds register didn't increment

Timer count register greater/less than expected
Address =________, Expected =________, Actual =________
```

DRAM - DRAM Refresh Timing

This test verifies that when the refresh rate is changed via the Bus Clock Register, the total time of access to the Dynamic RAM (DRAM) array also changes. Accesses to the DRAM array should be held off until the refresh cycle is complete.

Tick Timer #1 in the MCC is used to make the necessary elapsed time measurements. The refresh timing logic is tested for both the maximum ($01) and the minimum ($00) refresh periods. The elapsed time of maximum should be larger than elapsed time of minimum.

Command Input

```
162-Diag>MCC RFRSH
```

Messages

If the test fails, the following error message appears:

```
Refresh Period did not vary as expected
Elapsed Time of MAXIMUM should be larger than Elapsed Time of MINIMUM
MAXIMUM Period Value =__, Elapsed Time =________
MINIMUM Period Value =__, Elapsed Time =________
```

TMR*n*A - Timer Counter

These tests verify that the Tick Timer Counters are operational. Each test runs as follows:

1. The Tick Timer counter register is verified for data write/read operability. Both a 1 and 0 bit are walked through the 32-bit register and verified.
2. The Tick Timer counter register is initialized to zero and the counter is enabled. The test verifies that the counter register becomes non-zero (increments).
3. The the Tick Timer counter register is initialized to a predetermined count (i.e., $00000000, $00000001, $00000003, $00000007,..., $7FFFFFFF, $FFFFFFFF) and then enabled. The test waits for the contents of the counter register to be greater than the initialization count.

Command Input

```
162-Diag>MCC TMR1A
```

or

```
162-Diag>MCC TMR2A
```

or

```
162-Diag>MCC TMR3A
```

or

```
162-Diag>MCC TMR4A
```

Messages

If the test fails, one of the following error messages appears:

```
Register did not clear
Address =________, Expected =________, Actual =________
```

```
Register access error
Address =________, Expected =________, Actual =________
```

```
Counter did not increment
Address =________, Expected =________, Actual =________

Timeout waiting for Counter to increment
Address =________, Expected =________, Actual =________

Timeout waiting for Counter to roll over
Address =________, Expected =________, Actual =________
```

TMR*n*B - Timer Free-Run

These test verify that the Tick Timer Compare Registers are operational. Each test runs as follows:

1. The Tick Timer's compare register is verified for data write/read operability. Both a 1 and 0 bit are walked through the 32-bit register and verified.
2. The Tick Timer's counter and compare registers are initialized to a predetermined count (i.e., $00000000, $00000001, $00000003, $00000007,..., $7FFFFFFF, $FFFFFFFF).
3. The counter is enabled and the test waits for the contents of the counter register to exceed the compare register's initial contents. This also verifies that the counter register will not clear when the compare count is met.

Command Input

```
162-Diag>MCC TMR1B
```

or

```
162-Diag>MCC TMR2B
```

or

```
162-Diag>MCC TMR3B
```

or

```
162-Diag>MCC TMR4B
```

Messages

If the test fails, one of the following error messages appears:

```
Register did not clear
Address =________, Expected =________, Actual =________

Register access error
Address =________, Expected =________, Actual =________

Timeout waiting for Count to exceed Compare
Address =________, Expected =________, Actual =________
```

TMR*n*C - Timer Clear on Compare

These tests verify the Tick Timers' Clear on Compare functions. Each test runs as follows:

1. The compare and count registers are enabled.
2. The timer is enabled to run until the software times-out or the counter exceeds the compare (error condition).
3. The counter and compare registers on the Tick Timer are initialized to a predetermined count (i.e. $00000000, $00000001, $00000003, $00000007,..., $7FFFFFFF, $FFFFFFFF)
4. The counter is enabled

The test waits for a software delay or for the contents of the counter register to exceed the compare register's initial contents.

Command Input

```
162-Diag>MCC TMR1C
```

or

```
162-Diag>MCC TMR2C
```

or

```
162-Diag>MCC TMR3C
```

or

```
162-Diag>MCC TMR4C
```

Messages

If the test fails, the following error message appears:

```
Count did not zero on Compare
Address =________, Expected =________, Actual =________
```

TMR*n*D - Timer Overflow Counter

These tests verify the Tick Timers Overflow Counters. Each test runs as follows:

1. The test verifies that the overflow counter can be cleared to zero.
2. The test verifies that the overflow counter will increment from $00 to $10.
3. The test verifies the overflow count can be cleared once set (non-zero).
4. The test verifies that the overflow count can increment from $10 to $F0 (increments of $10).

Command Input

```
162-Diag>MCC TMR1D
```

or

```
162-Diag>MCC TMR2D
```

or

```
162-Diag>MCC TMR3D
```

or

```
162-Diag>MCC TMR4D
```

Messages

If the test fails, one of the following error messages appears:

```
Overflow Counter did not clear
Address =________, Expected =__, Actual =__
```

```
Overflow Counter did not increment
Address =________, Expected =__, Actual =__
```

```
Timeout waiting for Overflow Counter
Address =________, Expected =__, Actual =__
```

TMR*n*E - Timer Interrupts

These tests verify that the Tick Timers can generate interrupts and that the MPU takes the correct vector. The test verifies that:

1. Level 0 interrupts will not generate an interrupt, but will set the appropriate status
2. All interrupts (1-7) can be generated and received
3. The appropriate status is set

Command Input

```
162-Diag>MCC TMR1E
```

or

```
162-Diag>MCC TMR2E
```

or

```
162-Diag>MCC TMR3E
```

or

```
162-Diag>MCC TMR4E
```

Messages

If the test fails, one of the following error messages appears:

```
Interrupt Control Register did not clear
Address =________, Expected =__, Actual =__
```

```
Interrupt Enable bit did not set
Address =________, Expected =__, Actual =__
```

```
Interrupt Status bit did not set
Status: Expected =__, Actual =__
Vector: Expected =__, Actual =__
State : IRQ Level =_, VBR =__
```

```
Incorrect Vector type
Status: Expected =__, Actual =__
Vector: Expected =__, Actual =__
State : IRQ Level =_, VBR =__
```

```
Unexpected Vector taken
Status: Expected =__, Actual =__
Vector: Expected =__, Actual =__
State : IRQ Level =_, VBR =__

Incorrect Interrupt Level
Level: Expected =_, Actual =_
State: IRQ Level =_, VBR =__

Interrupt did not occur
Status: Expected =__, Actual =__
Vector: Expected =__, Actual =__
State : IRQ Level =_, VBR =__

Interrupt Status bit did not clear
Address =________, Expected =__, Actual =__
```

VBR - Vector Base Register

This test verifies that the MCC's Vector Base Register is operational. The register is tested for all possible data patterns. Then the Vector Base Register is tested for vector direction. Vector bases of $40 to $F0 (increments of $10) are used and tested. Vector direction is accomplished by using the LAN Coprocessor Interrupt Control Register (MCC based) as the interrupt source.

Command Input

```
162-Diag>MCC VBR
```

Messages

If the test fails, one of the following error messages appears:

```
Write/Read error on VBR
Address =________, Expected =__, Actual =__
```

```
Unexpected Vector taken
Status: Expected =__, Actual =__
Vector: Expected =__, Actual =__
State : IRQ Level =_, VBR =__
```

```
Interrupt did NOT occur
Expected Vector =__ (&___)
IRQ Level =_, VBR =__, Control/Status Register =__
```

WDTMRA - Watchdog Timer Counter

This test verifies that the Watchdog Timer Counter will count and set the correct status (time-out). The time-out status is verified that it can be cleared. The counter is tested for write/read capability of all settings. The counter selection timeouts are only tested with reasonable settings (4 seconds and under) to keep the total test time short.

Command Input

```
162-Diag>MCC WDTMRA
```

Messages

If the test fails, one of the following error messages appears:

```
Register Write/Read Error
Address =________, Expected =__, Actual =__
```

```
Software Time-Out occurred while waiting for Watchdog Time-Out Status
Time-Out Selection Code =__
```

```
Current Time-Out Selection Code Count was NOT greater than the Previous
Current Count =________, Previous Count =________
```

```
Watchdog Time-Out Status Bit did NOT clear
Watchdog Timer Control Register =__
Time-Out Selection Code         =__
```

WDTMRB - Watchdog Timer Board Fail

This test verifies that the Watchdog Timer will set the Board Fail indicator (FAIL LED and VMEbus SYSFAIL*) when a time-out occurs. The test also verifies that the Board Fail Signal and Indicator can be toggled, both from the MCC and the VMECHIP2.

Command Input

```
162-Diag>MCC WDTMRB
```

Messages

If the test fails, one of the following error messages appears:

```
Board Fail Signal/Indicator is NOT clear
Reset Switch Control Register =__
(Board Fail Signal is driven (negated) via MCC)
```

```
Board Fail Signal/Indicator is NOT clear
Reset Switch Control Register =__
(Board Fail Signal is driven (negated) via VMEC2)
```

```
Board Fail Signal/Indicator is NOT set
Reset Switch Control Register =__
(Board Fail Signal is driven (asserted) via MCC)
```

```
Board Fail Signal/Indicator is NOT set
Reset Switch Control Register =__
(Board Fail Signal is driven (asserted) via VMEC2)
```

```
Board Fail Signal/Indicator is NOT set
Reset Switch Control Register =__
(Board Fail Signal is driven (asserted) via MCC Watchdog Timer)
```

```
Board Fail Signal/Indicator is NOT clear
Reset Switch Control Register =__
(Board Fail Signal is driven (negated) via MCC Watchdog Timer)
```

WDTMRC - Watchdog Timer Local Reset

This test verifies that the Watchdog Timer will generate a local reset upon timing out. If the test returns, it is considered a failure and an appropriate error message is displayed/logged.

This test does not execute when the **MCC** test group is executed. This test is supplied only for diagnostic purposes.

Command Input

```
162-Diag>MCC WDTMRC
```

Messages

If the test fails, one of the following error messages appears:

```
Register Write/Read Error
Address =________, Expected =__, Actual =__
```

```
Watchdog Timer did NOT generate a Local Bus Reset
```

MCECC - ECC Memory Board

The **MCECC** tests check ECC memory devices. The tests are listed in Table 3-5, and are described in alphabetical order on the following pages.

Table 3-5. MCECC Tests

Test	Description
CBIT	Check-Bit DRAM
SCRUB	Scrubbing
SBE	Single-Bit-Error
MBE	Multi-Bit-Error
EXCPTN	Exceptions

Enter **MCECC** without a test name to run all tests (except for **EXCPTN**) in the group. They will be executed in the order shown in the order shown in Table 3-5.

Configuration Parameters

You may change the following parameters with the **CF** command (the default values are shown):

```
Inhibit restore of ECC registers upon test failure (y/n) =n ?
```

This causes the ASICs registers to remain unchanged after a failure. If set to “N” the registers are restored before the diagnostic exits.

```
Verbose messages during execution (y/n) =n ?
```

This displays messages about which portion of the test is currently being executed. Because of the large size of these memory boards, some of the **MCECC** tests can take many minutes to execute; the “verbose” output indicates that the test is still running.

```
Override default starting/ending addresses (y/n) =n ?
```

This overrides the default address ranges for testing, on a per board basis. The default answer “N” means that the **MCECC** diagnostics check the environment, and test all possible memory on every ECC board found in the system.

```
Starting address, 1st memory board (hex,0 - 08000000) =00000000 ?
Ending address, 1st memory board   (hex,0 - 08000000) =00000000 ?
Starting address, 2nd memory board (hex,0 - 08000000) =00000000 ?
Ending address, 2nd memory board   (hex,0 - 08000000) =00000000 ?
```

These are the starting and ending addresses for each memory board. These addresses are relative to the particular board only. Each board address begins at zero, despite where it might be configured in the computer's memory map. If a system is configured with two 32MB ECC memory boards, for purposes of the configuration parameters, each board starts at address 0, and ends at 02000000.

CBIT - Check-Bit DRAM

This test verifies the operation of the check-bit RAM. The test uses the address as the data in the first word, the complement of the address in the second word, and swapped nybbles in the third word. This pattern continues all through the check bit memory. When complete, this process is repeated two more times, but the order of the functions for generating check bit data are rotated until each word has used each of the three types of data-generating functions.

The SBC ECC memory boards are comprised of two ECC ASICs, and DRAM connected to each ASIC. The ASICs have a control bit that may be set, to allow direct reading and writing of check bit memory. In this test, that bit is set, and causes each of the two check bit words to appear in separate bytes of the data word (bits 8 through 15 = lower ECC, bits 24 through 31 = upper ECC). The test data is masked to 8 bits, and copied into bits 8 through 15 and 24 through 31.

All of check bit RAM is written in one pass, followed by a verification pass of all of RAM.

Command Input

```
162-Diag>MCECC CBIT
```

Messages

The status message contains the current address followed by *xnp*, where the *x* is `w` for write or `r` for read, *n* is the memory board number being tested, and *p* is the pass of the test is being executed, either `a`, `b`, or `c`.

```
ECC      CBIT: ECC Check-Bit DRAM..... Running ---> ____ xnp
```

If the scrubber fails during check bit initialization, the following message appears:

```
Timed out waiting for scrubber to start, bd #_ (status __)
Timed out waiting for scrubber to stop, bd #_ (status __)
```

If there is a check bit memory failure, the following message appears:

```
At:  ______, read:  ______, should be:  ______, (lower MCECC)
At:  ______, read:  ______, should be:  ______, (upper MCECC)
```

EXCPTN - Exceptions

This test verifies the ECC board's ability to generate interrupts or bus errors on detecting a memory error. This test plants errors in memory, enables either the interrupt or bus-error, and reads the "faulty" memory location. The proper exception and status is tested, and if received, the test passes.

Command Input

```
162-Diag>MCECC EXCPTN
```

Messages

If there is a scrubber failure during check bit initialization, the following message appears:

```
Timed out waiting for scrubber to start, bd #_ (status __)
Timed out waiting for scrubber to stop, bd #_ (status __)
```

MBE - Multi-Bit-Error

This test verifies the ECC board's ability to detect multi-bit-errors. It fills a memory area with random data containing a “multi-bit-error” in each word. All of the tested memory area is then verified with error correction enabled so that the data errors will be detected during the read operation.

Command Input

```
162-Diag>MCECC MBE
```

Messages

If the scrubber fails during check bit initialization, the following message appears:

```
Timed out waiting for scrubber to start, bd #_ (status __)
Timed out waiting for scrubber to stop, bd #_ (status __)
```

If there is an error-logger test failure, the following message appears:

```
errlog:   logger didn't indicate an error:
          bd #_, addr ____ -__, read ________, actual _________
errlog:   logger didn't indicate error-on-read, bd #_, addr____
errlog:   logger error address wrong: ___, actual: __, board #_
```

If double-bit-errors are not detected properly, the following message appears:

```
mbe:    logger didn't indicate an error:
        bd #_, addr ________, read ________, actual ________
mbe:    logger didn't indicate error-on-read, bd #_, addr _____
mbe:    logger didn't indicate error was multi-bit-error:
        bd #_, addr ______, read _____, actual ____, logger __
mbe:    logger error address wrong: ___, actual: ____, board #_
```

SBE - Single-Bit-Error

This test verifies the ECC board's ability to correct single-bit-errors. It fills a memory area with random data containing a "single-bit-error" in each word. All of the tested memory area is then verified with error correction enabled, so that the data will be "corrected" during the read operation.

Command Input

```
162-Diag>MCECC SBE
```

Messages

The status message contains the current address being accessed followed by *xn*, where the *x* is `w` for write or `r` for read, and `#` is the memory board number being tested.

```
ECC      SBE: ECC Single-Bit-Error....... Running ---> _____ x#
```

If the scrubber fails during check bit initialization, the following message appears:

```
Timed out waiting for scrubber to start, bd #_ (status __)
Timed out waiting for scrubber to stop, bd #_ (status __)
```

If single-bit-errors are not corrected properly, the following message appears:

```
Address=________, Expected=________, Actual=________
```

SCRUB - Scrubbing

This test verifies refresh "scrubbing" of errors from DRAM. It checks the ECC memory board's capability to correct single-bit-errors during normal DRAM refresh cycles. During its operation, the diagnostic displays the current memory board number that it is working on. When the fast-refresh mode is selected, "wait" is displayed, indicating that the test is waiting long enough for fast-refresh to get to every memory location on the board at least once.

The test runs per the following:

1. ECC memory is initiated (the `init` message appears).
2. The error-logger is tested (the `errlog` message appears).
3. Errors are planted in memory, and the first scrub pass runs (the `scrub 1` message appears).
4. The memory is tested with the error-logger.
5. Another pass of the scrubber is run (the `scrub 2` message appears). This scrub pass is checked for zero errors.

Command Input

```
162-Diag>MCECC SCRUB
```

Messages

If the scrubber fails during checkbit initialization, the following message appears:

```
Timed out waiting for scrubber to start, bd #_ (status __)
Timed out waiting for scrubber to stop, bd #_ (status __)
```

If there is an error-logger test failure, the following message appears:

```
errlog:   logger didn't indicate an error:
          bd #_, addr ________, read ________, actual ________
errlog:   logger didn't indicate error-on-read, bd #_, addr ___
errlog:   logger error address wrong: ___, actual: __, board #_
```

If there is a first pass scrubbing failure, the following message appears:

```
Timed out waiting for scrubber to start, bd #_ (status __)
Timed out waiting for scrubber to stop, bd #_ (status __)
single-bit-error at ________ found after scrubbing RAM
multi-bit-error at ________ found after scrubbing RAM
```

If there is a second pass scrubbing failure, the following message appears:

```
Timed out waiting for scrubber to start, bd #_ (status __)
Timed out waiting for scrubber to stop, bd #_ (status __)
After final scrubbing, a single-bit error was found
After final scrubbing, a multi-bit error was found
```

CMMU - Cache and Memory Management Unit

The **CMMU** tests check the Cache and the Memory Management Unit (MMU). The tests are listed in Table 3-6, and are described in alphabetical order on the following pages.

Enter **CMMU** without a test name to run all tests in the group. They will be executed in the order shown in Table 3-6.

In order to test the Cache and MMU it is necessary to build translation tables in memory. The CMMU tests require that memory has been tested and found to be good.

Note All data is hexadecimal.

The Access Fault Information is only displayed if the exception was an Access Fault (Bus Error). Access size is in bytes. Access type is 0 for write and 1 for read.

The address space code message uses the following codes:

1 user data
2 user program
5 supervisor data
6 supervisor program
7 MPU space

All address space codes listed above may not be applicable to any single microprocessor type.

Table 3-6. CMMU Tests

Test	Description
CCHCODE	Cache Code Copy/Execution
CCHCPYB	Cache Copyback
CCHSC	Cache Supervisor Code
CCHSCCI	Cache Supervisor Code Cache Inhibit

Table 3-6. CMMU Tests

Test	Description
CCHSD	Cache Supervisor Data
CCHSDCI	Cache Supervisor Data Cache Inhibit
CCHSDWT	Cache Supervisor Data Write Through
CCHTTM	Translation Table Memory
CCHUC	Cache User Code
CCHUCCI	Cache User Code Cache Inhibit
CCHUD	Cache User Data
CCHUDCI	Cache User Data Cache Inhibit
CCHUDWT	Cache User Data Write Through
MMUMU	MMU Modified/Used Data/Code
MMUSC	MMU Supervisor Code
MMUSD	MMU Supervisor Data
MMUSP	MMU Supervisor Protect Data/Code
MMUSPF	MMU Segment/Page Fault Data/Code
MMUUC	MMU User Code
MMUUD	MMU User Data
MMUWP	MMU Write Protect
TTRSC	TTR Supervisor Code
TTRSD	TTR Supervisor Data
TTRUC	TTR User Code
TTRUD	TTR User Data
TTRWP	TTR Write Protect

Configuration Parameters

You may set the following parameters with the **CF** command (the default values are shown):

```
Starting/Ending Address Enable [Y/N] =N ?
Starting Address =00000000 ?
Ending Address   =003FFFFC ?
```

CCHCODE - Cache Code Copy/Execution

The **CCHCODE** test checks the ability of the MMU to copy or move instruction strings into memory and execute them.

The test runs as follows:

1. An instruction string is written to memory and program control is transferred to it. The instruction string keeps track of its beginning and ending address.
2. The instruction string copies itself to the next higher space in memory and transfers program control to the new copy. Each time the program does this a return address is placed on the stack. As the copies grow toward larger addresses, the stack grows toward smaller addresses. This step is repeated until all selected memory is filled.

   The number of times that the sequence is repeated is determined by the ending address entered in the `Ending Address` **CF** parameter or the amount of memory installed if the user runs with default parameters.
3. The instruction string checks for a "final" address (last copy), and when this is found all the return addresses get pulled from the stack one by one as program control moves back down through each copy of the string.

At the end of the test, the MMU and Cache registers are returned to their original state.

If during the test an unexpected exception occurs, the test will service it and display one or more exception messages.

Command Input

```
162-Diag> CMMU CCHCODE
```

Messages

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

CCHCPYB - Cache Copyback

The **CCHCPYB** test checks the ability of the MMU to operate in the Supervisory mode if Copyback is set in the page descriptor. The test runs as follows:

1. A translation table is built.
2. The end address of the table is found and a “start of region” address is calculated.
3. The start addresses of two regions of memory (regions 1 and 2) are calculated.
4. The number of pages for the MMU is found.
5. Region 1 is filled with a complementing $00000000 pattern.
6. Region 2 is filled with a complementing $55555555 pattern.
7. The MMU is turned on.
8. Region 1 memory is read to cache data, and is filled with a complementing $FFFFFFFF pattern. Data is written to cache “cache is dirty”, not memory.
9. The $FFFFFFFF pattern is copied to the Region 1. Region 2 is read to cache data, and is filled with a complementing $AAAAAAAA pattern (cache is dirty).
10. The data cache is flushed and invalidated (the $AAAAAAAA pattern should be written to the second region).
11. The MMU is turned off.
12. Regions 1 and 2 are verified.

At the end of the test, the MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If during the test an unexpected exception occurs, the test will service it and display one or more exception messages.

Command Input

```
162-Diag> CMMU CCHCPYB
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

If any data verification phase fails, the following message appears:

```
Data Miscompare Error:

Address =________, Expected =________, Actual =________
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                   Address ________
                      Data ________
               Access Size __
               Access Type _
        Address Space Code _
             Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

CCHSC - Cache Supervisor Code

The **CCHSC** test checks the ability of the MMU to execute cached instructions in the Supervisory mode. The test runs as follows:

1. A translation table is built with Copyback mode set in the page descriptor(s) for test memory.
2. The end address of the table is found and a “start of test memory” address is calculated.
3. Instruction string 1 is placed in memory. Both instruction strings do a return from subroutine. The test can determine which instruction string was executed by the value that is returned in MPU data register 0.
4. Memory is verified to hold string 1.
5. The MMU is turned on.
6. Program control is transferred to the instruction string.
7. Upon return, the test verifies that string 1 executed. String 1 has now been cached.
8. Instruction string 2 is written to memory.
9. Memory is verified to hold string 2.
10. Program control is transferred to the instruction string.
11. Upon return, the test verifies that string 1 executed (from cache).
12. The code cache is flushed and invalidated (this should do nothing because this is a code cache, not a data cache).
13. Memory is verified to hold string 2.

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If a string fails to verify, an error message appears. If a string does not execute or the wrong string executes, an error message appears.

If during the test an unexpected exception occurs, the test will service it and display one or more exception messages.

Command Input

```
162-Diag> CMMU CCHSC
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

Memory is filled with the instruction strings. If the instruction strings can not be read back as data, the test aborts and the following is printed, the following message appears:

```
Data Miscompare Error:

Address =________, Expected =________, Actual =________
```

If the wrong instruction strings are executed, the following message appears:

```
Code Execution Status Error:

Address =________, Expected =________, Actual =________
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

CCHSCC - Cache Supervisor Code Cache Inhibit

The **CCHSCC** test checks the ability of the MMU to not execute cached instructions in the Supervisor mode if Cache Inhibit is set in the page descriptor. The test runs as follows:

1. A translation table is built with Cache Inhibit set in the page descriptor(s).
2. The end address of the table is found and a “start of test memory” address is calculated.
3. The code cache is invalidated.
4. An instruction string is written to memory. The instruction string does a return from subroutine and the test can tell if the instruction string was executed by the value that is returned in MPU data register 0.
5. The string is verified.
6. The MMU is turned on.
7. Program control is transferred to the instruction string.
8. Upon return, the test verifies that the string was executed.
9. The Code Cache is turned off.
10. A different instruction string is written to memory and verified.
11. The Code Cache is turned on and program control is transferred to the instruction string.
12. Upon return, the test verifies that the string was executed.

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If the string fails to verify or does not execute, an error message appears. If during the test an unexpected exception occurs, the test will service it and display one or more exception messages.

Command Input

```
162-Diag> CMMU CCHSCCI
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

Memory is filled with the instruction strings. If the instruction strings can not be read back as data, the test aborts and the following is printed, the following message appears:

```
Data Miscompare Error:

Address =________, Expected =________, Actual =________
```

If the wrong instruction strings are executed, the following message appears:

```
Code Execution Status Error:

Address =________, Expected =________, Actual =________
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

CCHSD - Cache Supervisor Data

This test checks the ability of the MMU to write and read cached data in the Supervisory mode. The test runs as follows:

1. A translation table is built with Cache Inhibit set in the page descriptor(s).
2. The end address of the table is found and a “start of test memory” address is calculated.
3. The MMU is turned on with only test memory set for cache enabled in the Write through mode.
4. Test memory is written with data pattern 1.
5. Memory is read so that pattern 1 is cached.
6. The data cache is turned off.
7. Memory is verified that it contains pattern 1.
8. Memory is written with data pattern 2, this should change the memory but not cache.
9. Memory is read and pattern 2 is verified.
10. The data cache is turned on.
11. Memory is read and pattern 1 is verified (read from cache).
12. The test is repeated with pattern 1 an incrementing pattern and pattern 2 a decrementing pattern.

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If pattern 1 (cached) fails to verify or if memory fails to verify, an error message appears. If during the test an unexpected exception occurs, the test will service it and display one or more exception messages.

Command Input

```
162-Diag> CMMU CCHSD
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

If any data verification phase fails, the following message appears:

```
Data Miscompare Error:

Address =________, Expected =________, Actual =________
State: Verifying Memory/Verifying Cache.
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

CCHSDC - Cache Supervisor Data Cache Inhibit

This test checks the ability of the MMU to not write and read the data cache in the Supervisory mode if Cache Inhibit is set in page descriptor. The test runs as follows:

1. A translation table is built with Cache Inhibit set in the page descriptor(s).
2. The end address of the table is found and a “start of test memory” address is calculated.
3. The MMU is turned on.
4. Test memory is written with data pattern 1.
5. Memory is read to try and cache pattern 1.
6. The data cache is turned off.
7. Memory is verified that it contains pattern 1.
8. Memory is written with data pattern 2, this should change the memory but not cache.
9. Memory is read and pattern 2 is verified.
10. The data cache is turned on.
11. Memory is read and pattern 2 is verified (not cached).
12. The test is repeated with pattern 1 an incrementing pattern and pattern 2 a decrementing pattern.

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If pattern 2 (not cached) fails to verify, or if memory fails to verify, an error message appears. If during the test an unexpected exception occurs, the test will service it and display one or more exception messages.

Command Input

```
162-Diag> CMMU CCHSDCI
```

Messages

If the memory range specified is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

If any data verification phase fails, the following message appears:

```
Data Miscompare Error:

Address =________, Expected =________, Actual =________
State: Verifying Memory/Verifying Data not Cached.
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

CCHSDWT - Cache Supervisor Data Write Through

This test checks the ability of the MMU to operate in Supervisory mode if write through mode (memory is always updated upon writes) is set in the page descriptor.

The test runs as follows:

1. A translation table is built with Cache Inhibit set in the page descriptor(s).
2. The end address of the table is found and a “start of test memory” address is calculated.
3. Test memory is written with data pattern 1.
4. The write through mode is set in the page descriptor for test memory.
5. The MMU is turned on.
6. The test memory is read to fill the data cache with pattern 1.
7. Data pattern 2 is written to test memory (should be cached and written to memory).
8. The Data Cache is turned off.
9. The test memory is verified to contain data pattern 2.
10. Data pattern 3 is written to test memory and verified.
11. The Data Cache is turned on.
12. Data pattern 2 is read and verified from test memory (from cache).

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If pattern 2 (cached) fails to verify or if memory fails to verify, an error message appears. If during the test an unexpected exception occurs, the test will service it and display one or more exception messages.

Command Input

```
162-Diag> CMMU CCHSDWT
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

If any data verification phase fails, the following message appears:

```
Data Miscompare Error:

Address =________, Expected =________, Actual =________
State: Verifying Memory/Verifying Cache.
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

CCHTTM - Translation Table Memory

This test checks the memory (RAM) that is used for the translation table. The test runs as follows:

1. The translation table is built. This returns the starting and ending address for the table.
2. A pattern of zeros is written, read back, and verified to each address between start and end.
3. A pattern of Fs is written, read back, and verified to each address between start and end.
4. Memory between start and end is filled with invalid segment/page descriptors.

At the end of the test, the MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If the zero or F pattern fails to verify, an error message appears. If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU CCHTTM
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

If any data verification phase fails, the following message appears:

```
Data Miscompare Error:

Address =________, Expected =________, Actual =________
```

On receipt of an access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

On receipt of an unexpected exception, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

CCHUC - Cache User Code

This test checks the ability of the MMU to execute cached instructions in the User mode. The test runs as follows:

1. A translation table is built with Copyback mode set in the page descriptor(s) for test memory.
2. An exception switch is set for unexpected exceptions and all exceptions are claimed.
3. The end address of the table is found and a “start of test memory” address is calculated.
4. Instruction string 1 is placed in memory. Both instruction strings do a return from subroutine (RTS) and the test can tell which instruction string was executed by the value that is returned in MPU data register 0.
5. Memory is verified to hold string 1.
6. The state of the MPU is saved for return to supervisor mode exceptions.
7. The MMU is turned on.
8. The MPU and the exception switch are set to User mode.
9. Program control is transferred to the instruction string.
10. Upon return, the test verifies that string 1 executed. String 1 has now been cached.
11. Instruction string 2 is written to memory.
12. Memory is verified to hold string 2.
13. Program control is transferred to the instruction string.
14. Upon return, the test verifies that string 1 executed (from cache).
15. A trap always true instruction (vector 7) is executed to return the MPU to the supervisor mode.

16. The code cache is flushed and invalidated (this should do nothing because this is a code cache, not a data cache).
17. Memory is verified to hold string 2.

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If a string fails to verify, or if a string does not execute, or the wrong string executes, an error message appears. If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU CCHUC
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

Test memory is filled with the instruction strings. If the instruction strings can not be read back as data, the test aborts and the following is printed, the following message appears:

```
Data Miscompare Error:

Address =________, Expected =________, Actual =________
```

If the wrong instruction strings are executed, the following message appears:

```
Code Execution Status Error:

Address =________, Expected =________, Actual =________
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

If any exception other than the trap always true exception is taken, the following message appears:

```
Translation failed causing exception.

Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

CCHUCCI - Cache User Code Cache Inhibit

This test checks the ability of the MMU to not execute cached instructions in the User mode when Cache Inhibit is set in the page descriptor. The test runs as follows:

1. A translation table is built with Cache Inhibit mode set in the page descriptor(s).
2. The end address of the table is found and a “start of test memory” address is calculated.
3. Instruction string 1 is written to memory. Both instruction strings do a return from subroutine and the test can tell which instruction string was executed by the value that is returned in MPU data register 0.
4. Memory is verified to hold string 1.
5. The state of the MPU is saved for return to supervisor mode exceptions.
6. The MMU is turned on.
7. The MPU and the exception switch are set to User mode.
8. Program control is transferred to the instruction string.
9. Upon return, the test verifies that string 1 was executed (string 1 should not be cached).
10. Instruction string 2 is written to memory and verified.
11. Program control is transferred to the instruction string.
12. Upon return, the test verifies that string 2 was executed (string 1 was not cached).
13. A trap always true instruction (vector 7) is executed to return the MPU to the supervisor mode.

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If a string fails to verify, or if a string does not execute, or if the wrong string executes, an error message appears. If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU CCHUCCI
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

Test memory is filled with the instruction strings. If the instruction strings can not be read back as data, the test aborts and the following is printed, the following message appears:

```
Data Miscompare Error:

Address =________, Expected =________, Actual =________
```

If the wrong instruction strings are executed, the following message appears:

```
Code Execution Status Error:

Address =________, Expected =________, Actual =________
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

If any exception other than the trap always true exception is taken, the following message appears:

```
Translation failed causing exception.

Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

CCHUD - Cache User Data

This test checks the ability of the MMU to write and read cached data in the User mode. The test runs as follows:

1. A translation table is built with Cache Inhibit mode set in the page descriptor(s).
2. The end address of the table is found and a “start of test memory” address is calculated.
3. The state of the MPU is saved for return to supervisor mode exceptions.
4. Write through mode is set in the page descriptor(s) for test memory.
5. The MMU is turned on.
6. The MPU and the exception switch are set to User mode.
7. Test memory is written with data pattern 1.
8. Memory is read so that pattern 1 is cached.
9. The Data Cache is turned off.
10. Memory is read and verified to contain data pattern 1.
11. Memory is written with data pattern 2 (this should change memory but not cache).
12. Memory is verified that it contains pattern 2.
13. The Data Cache is turned on.
14. Memory is read and pattern 1 is verified (read from cache).
15. The test is repeated with pattern 1 an incrementing pattern and pattern 2 a decrementing pattern.

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

A trap always true instruction (vector 7) is executed to return the MPU to the supervisor mode.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If pattern 1 (cached) fails to verify or if memory fails to verify, an error message appears. If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU CCHUD
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

If any data verification phase fails, the following message appears:

```
Data Miscompare Error:

Address =________, Expected =________, Actual =________
State: Verifying Memory/Verifying Cache.
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

If any exception other than the trap always true exception is taken, the following message appears:

```
Translation failed causing exception.

Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

CCHUDCI - Cache User Data Cache Inhibit

This test checks the ability of the MMU to not write and read the data cache in the User mode when Cache Inhibit is set in the page descriptor. The test runs as follows:

1. A translation table is built with Cache Inhibit mode set in the page descriptor(s).
2. The end address of the table is found and a “start of test memory” address is calculated.
3. The MMU is turned on.
4. The MPU and the exception switch are set to User mode.
5. Test memory is written with data pattern 1.
6. Memory is read to try and cache pattern 1.
7. The data cache is turned off.
8. Memory is verified that it contains pattern 1.
9. Memory is written with data pattern 2, this should change the memory but not cache.
10. Memory is read and pattern 2 is verified.
11. The data cache is turned on.
12. Memory is read and pattern 2 is verified (not cached).
13. The test is repeated with pattern 1 an incrementing pattern and pattern 2 a decrementing pattern.

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

A trap always true instruction (vector 7) is executed to return the MPU to the supervisor mode.

If pattern 2 (not cached) fails to verify, or if memory fails to verify, an error message appears. If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU CCHUDCI
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

If any data verification phase fails, the following message appears:

```
Data Miscompare Error:

Address =________, Expected =________, Actual =________
State: Verifying Memory/Verifying Cache.
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

If any exception other than the trap always true exception is taken, the following message appears:

```
Translation failed causing exception.

Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

CCHUDWT - Cache User Data Write Through

This test checks the ability of the MMU to operate in the User mode, if Write Through mode (memory is always updated upon writes) is set at the page descriptor. The test runs as follows:

1. A translation table is built with Cache Inhibit set in the page descriptor(s).
2. The end address of the table is found and a “start of test memory” address is calculated.
3. The Write through mode is set in the page descriptor(s) for test memory.
4. Test memory is written with data pattern 1.
5. The state of the MPU is saved for return to supervisor mode exceptions.
6. The MMU is turned on.
7. The test memory is read to fill the data cache with pattern 1.
8. Data pattern 2 is written to test memory (should be cached and written to memory).
9. The Data Cache is turned off.
10. The test memory is verified to contain data pattern 2.
11. Data pattern 3 is written to test memory and verified.
12. The Data Cache is turned on.
13. Data pattern 2 is read and verified from test memory (from cache).
14. A trap always true instruction (vector 7) is executed to return the MPU to the supervisor mode.

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If pattern 2 (cached) fails to verify or if memory fails to verify, an error message appears. If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU CCHUDWT
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

If any data verification phase fails:

`Data Miscompare Error`, the following message appears:

```
Address =________, Expected =________, Actual =________
State: Verifying Memory/Verifying Cache.
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                   Address ________
                      Data ________
               Access Size __
               Access Type _
        Address Space Code _
             Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

If any exception other than the trap always true exception is taken, the following message appears:

```
Translation failed causing exception.

Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

MMUMU - MMU Modified/Used Data/Code

This test checks the ability of the MMU to set the Used and Modified bits in the Page Descriptor. There are two parts to the test. In the "Used" portion of the test both code execution and data operations are tested to verify that the "Used" bit can be set with the "Modified" bit remaining clear. The "Modified" portion of the test verifies that both the "Used" and "Modified" bits can be set together.

The test executes the following sequence three times with the only difference being at step 5, and the state of the descriptor bits when verified:

1. A translation table is built with Cache Inhibit set in the page descriptors.
2. The "start of test memory" address is calculated based on the location of the end of the translation table.
3. A return from subroutine instruction (RTS) is placed in memory.
4. The MMU is turned on.
5. During the Used bit Data test a test location is read. During the Used bit Code test the RTS instruction in memory is executed. During the Modified bit test a test location is written.
6. The MMU is turned off.
7. The page descriptor bits are verified.

At the beginning of the tests the state of the MMU and Cache registers is saved and the original state of the MMU and Cache registers is restored when the test completes. All exceptions are claimed and serviced by the test while it is executing.

Command Input

```
162-Diag>CMMU MMUMU
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

If the modified and used bits in the page descriptor are incorrect, the following message appears:

```
State: Used Bit (read/execute) or Modified/Used Bit (write)
Data Miscompare Error:

Address =________, Expected =________, Actual =________
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

MMUSC - MMU Supervisor Code

This test checks the ability of the MMU to execute instructions in Supervisor mode. The test runs as follows:

1. The start of test memory is adjusted to the nearest page address.
2. The data and code caches are turned off.
3. A page list table is built on the stack.
4. The beginning address and number of pages of the code area are written to that element in the page list table.
5. The beginning address and number of pages of the stack area are written to that element in the page list table.
6. The address translation table is built using the page list table.
7. The physical test address is set to follow the translation table.
8. The logical and physical addresses are written to that element in the page list table.
9. The address translation table is rebuilt using the updated page list table.
10. All ATC entries are flushed.
11. A page of test memory is filled with a pattern of Fs.
12. A return from subroutine instruction (RTS) is placed in a single location within the page of test memory.
13. The Supervisor Root Pointer register is initialized.
14. The MMU is turned on.
15. Using the logical address, the instruction in memory is executed.

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If translation doesn't work, an Access Exception occurs.

If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU MMUSC
```

Messages

If the test fails, one of the following messages appears.

If the memory range is less than $32000 byte, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

On receipt of an unexpected access exception, the following message appears:

```
Translation failed causing exception.

Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Translation failed causing exception.

Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

MMUSD - MMU Supervisor Data

This test checks the ability of the MMU to access data in Supervisor mode. The test runs as follows:

1. The start of test memory is adjusted to the nearest page address.
2. The data and code caches are turned off.
3. A page list table is built on the stack.
4. The beginning address and number of pages of the code area are written to that element in the page list table.
5. The beginning address and number of pages of the stack area are written to that element in the page list table.
6. The address translation table is built using the page list table.
7. The physical test address is set to follow the translation table.
8. The logical and physical addresses are written to that element in the page list table.
9. The address translation table is rebuilt using the updated page list table.
10. All ATC entries are flushed.
11. A page of test memory is filled with a pattern of Fs.
12. A data pattern is written to a single location within the page of test memory.
13. The Supervisor Root Pointer register is initialized.
14. The MMU is turned on.
15. Using the logical address, the data pattern is read and verified.

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If translation doesn't work, an Access Exception may occur.

If memory fails to verify, an error message appears. If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU MMUSD
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If while trying to build translation tables, an excessive number of pages were requested, or, if the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

If the data pattern directly placed in a page frame did not match the data read with the MMU enabled, the following message appears:

```
Translation failed causing a data miscompare.
Physical Address = ________, Logical Address = ________
Expected = ________, Actual = ________
```

On receipt of an unexpected access exception, the following message appears:

```
Translation failed causing exception.
```

```
Bus Error Information:
                  Address ________
                     Data ________
               Access Size __
               Access Type _
        Address Space Code _
             Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Translation failed causing exception.

Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

MMUSP - MMU Supervisor Protect Data/Code

This test checks the ability of the MMU to supervisor protect User area memory during Code and Data operations using the appropriate bits in the Page descriptor. The test runs as follows:

1. A translation table is built with Cache Inhibit set in the page descriptor(s).
2. The end address of the table is found and a “start of test memory” address is calculated.
3. An RTS instruction is placed in memory.
4. Supervisor Protect is set in the page descriptor(s) for test memory.
5. The MMU is turned on.
6. The MPU and the exception switch are set to User mode.
7. During the Code test an attempt is made to execute the instruction (an exception should occur).

   During the Data test an attempt is made to read the memory location (an exception should occur).

   The exception should return the MPU to the Supervisor mode. If the exception is not taken, a trap always true instruction (vector 7) is executed to return the MPU to the Supervisor mode.
8. The MMU Fault/Status register is read for proper error status.

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If the Supervisor Protect exception is not taken, an error message appears. If after the exception is received, the descriptor doesn't contain proper status, an error message appears.

If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU MMUSP
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

If the supervisor violation exception is not taken, the following message appears:

```
Code/Data Fault Exception did not occur.
```

If after the exception is received, the descriptor doesn't contain proper status, the following message appears:

```
State: Verifying Code/Data accesses.
Data Miscompare Error:

Address =________, Expected =________, Actual =________
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

If any exception other than the Access Exception is taken, the following message appears:

```
Translation failed causing exception.

Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________

MMU/Fault Status _______, Physical Address ________.
State: Verifying Code/Data accesses.
```

MMUSPF - MMU Segment/Page Fault Data/Code

This test checks the ability of the MMU to deny access to a root, segment, or page by having the descriptors marked as invalid. The test runs as follows:

1. A translation table is built with Cache Inhibit set in the page descriptor(s).
2. The end address of the table is found and a “start of test memory” address is calculated.
3. An RTS instruction is placed in memory.
4. If the test runs at the root/segment level, an unused root/segment descriptor is found and marked invalid.

   If the test runs at the page level, the page descriptor is marked invalid.
5. The MMU is turned on.
6. If the test is for Code, the attempt is made to execute the instruction (an exception should occur).

   If the test is for Data, the attempt is made to read the memory location (an exception should occur).

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If the page violation exception is not taken, an error message appears. If after the exception is received, the descriptor doesn't contain proper status, an error message appears.

If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU MMUSPF
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

If the page violation exception is not taken, the following message appears:

```
Code/Data Fault Exception did not occur.
State: Descriptor Type bits cleared at _______ Descriptor.
```

If after the exception is received, the descriptor doesn't contain proper status, the following message appears:

```
State: Descriptor Type bits cleared at _______ Descriptor.
Data Miscompare Error:

Address =________, Expected =________, Actual =________
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

If any exception other than the Access Exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________

State: Descriptor Type bits cleared at _______ Descriptor.
```

MMUUC - MMU User Code

This test checks the ability of the MMU to execute instructions in User mode. The test runs as follows:

1. The start of test memory is adjusted to the nearest page address.
2. The data and code caches are turned off.
3. A page list table is built on the stack.
4. The beginning address and number of pages of the code area are written to that element in the page list table.
5. The beginning address and number of pages of the stack area are written to that element in the page list table.
6. The address translation table is built using the page list table.
7. The physical test address is set to follow the translation table.
8. The logical and physical addresses are written to that element in the page list table.
9. The address translation table is rebuilt using the updated page list table.
10. All ATC entries are flushed.
11. A page of test memory is filled with a pattern of Fs.
12. A return from subroutine instruction (RTS) is placed in a single location within the page of test memory.
13. The state of the MPU is saved for return to supervisor mode exceptions.
14. The supervisor and user root pointer registers are initialized.
15. The MMU is turned on.
16. The MPU and the exception switch are set to User mode.

17. Using the logical address, the instruction in test memory is executed.
18. A trap always true instruction (vector 7) is executed to return the MPU to the supervisor mode.
19. The MMU Fault/Status register is read for proper error status.

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If translation doesn't work, an Access Exception occurs.

If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU MMUUC
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If while trying to build translation tables, an excessive number of pages were requested, or, if the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

If any exception other than the trap always true exception is taken, the following message appears:

```
Translation failed causing exception.

Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________

MMU/Fault Status _______, Physical Address ________.
```

MMUUD - MMU User Data

This test checks the ability of the MMU to access data in User mode. The test runs as follows:

1. The start of test memory is adjusted to the nearest page address.
2. The data and code caches are turned off.
3. A page list table is built on the stack.
4. The beginning address and number of pages of the code area are written to that element in the page list table.
5. The beginning address and number of pages of the stack area are written to that element in the page list table.
6. The address translation table is built using the page list table.
7. The physical test address is set to follow the translation table.
8. The logical and physical addresses are written to that element in the page list table.
9. The address translation table is rebuilt using the updated page list table.
10. All ATC entries are flushed.
11. A page of test memory is filled with a pattern of Fs.
12. A test data pattern is placed in a single location within the page of test memory.
13. The supervisor and user root pointer registers are initialized.
14. The MMU is turned on.
15. The MPU and the exception switch are set to User mode.
16. Using the logical address, the test memory location is read and verified.

17. A trap always true instruction (vector 7) is executed to return the MPU to the supervisor mode.
18. The MMU Fault/Status register is read for proper error status.

At the end of the test, the MMU is turned off. The MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If translation doesn't work, an Access Exception or a data miscompare occurs. If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU MMUUD
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

On receipt of an unexpected access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
```

If an unexpected exception is taken, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
```

If any exception other than the trap always true exception is taken, the following message appears:

```
Translation failed causing exception.

Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________

MMU/Fault Status _______, Physical Address ________.
```

If the data pattern directly placed in a page frame did not match the data read with the MMU enabled, the following message appears:

```
Translation failed causing a data miscompare.
Physical Address = ________, Logical Address = ________
Expected = ________, Actual = ________
```

MMUWP - MMU Write Protect

This test checks the ability of the MMU to write protect memory using the appropriate bits in the Page and Table descriptors. The test runs as follows:

1. A translation table is built.
2. The end address of the table is found and a “start of region” address is calculated.
3. The “start of region” address is used to find the corrected address for the CMMU test.
4. Test pattern 1 is written to memory.
5. If the test runs at the segment level, an unused segment descriptor is found and made valid and write protected.

   If the test runs at the page level, the page descriptor is write protected.
6. The MMU is turned on.
7. The exception switch is set to handle Write Protect exceptions, and all exceptions are claimed.
8. Test pattern 2 is written to memory. This should cause an access fault.
9. The MMU is turned off.
10. The address is read to verify that test pattern 1 is still there.

At the end of the test, the MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If the test memory location doesn't contain pattern 1, an error message appears. If the access fault exception is not taken, an error message appears.

If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU MMUWP
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

If, while trying to build translation tables, an excessive number of pages were requested, or the end of a requested memory segment would exceed the highest memory address (address zero wrap), the following message appears:

```
Translation Table Build Problem.
```

If pattern 1 is over written by the attempted write of test pattern 2, the following message appears:

```
Data Miscompare Error:

Address =________, Expected =________, Actual =________
State: Write Protect set at _______ Table Descriptor.
```

If test pattern is 2 is written to memory but no data fault exception occurs, the following message appears:

```
Access Fault Exception did not occur
State: Write Protect set at _______ Table Descriptor.
```

If the descriptor doesn't contain proper status, the following message appears:

```
State: _______ Table Descriptor error.
Data Miscompare Error:

Address =________, Expected =________, Actual =________
```

On receipt of an access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
State: Write Protect set at _______ Table Descriptor.
```

On receipt of an unexpected exception, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
State: Write Protect set at _______ Table Descriptor.
```

TTRSC - TTR Supervisor Code

This test checks the ability of the code Transparent Translation Register to execute instructions in supervisor mode. The test runs as follows:

1. The test address is adjusted to the nearest page address.
2. The data and code caches are turned off.
3. A return from subroutine instruction (RTS) is placed in test memory.
4. The Transparent Translation Register is turned on.
5. The instruction in memory is executed.
6. The Transparent Translation Register is turned off.

At the end of the test, the MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If translation doesn't work, an Access Exception will occur.

If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU TTRSC
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

On receipt of an access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
State: ITT_ set for Supervisor Code.
```

On receipt of an unexpected exception, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
State: ITT_ set for Supervisor Code.
```

TTRSD - TTR Supervisor Data

This test checks the ability of the data Transparent Translation Register to access data in Supervisor mode. The test runs as follows:

1. The test address is adjusted to the nearest page address.
2. The state of the MPU is saved for unexpected exceptions. All exceptions are claimed.
3. The data and code caches are turned off.
4. A data pattern is placed in memory.
5. The Transparent Translation Register is turned on.
6. The data pattern is read and verified.
7. The Transparent Translation Register is turned off.

At the end of the test, the MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If translation doesn't work, an Access Exception or a data miscompare occurs.

If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU TTRSD
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

On receipt of an access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
State: DTT_ set for Supervisor Data.
```

On receipt of an unexpected exception, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
State: DTT_ set for Supervisor Data.
```

If the data pattern directly placed in memory did not match the data read with the TTR enabled, the following message appears:

```
Data Miscompare Error:

Address =________, Expected =________, Actual =________
State: DTT_ set for Supervisor Data.
```

TTRUC - TTR User Code

This test checks the ability of the code Transparent Translation Register to execute instructions in User mode. The test runs as follows:

1. The test address is adjusted to the nearest page address.
2. The data and code caches are turned off.
3. A return from subroutine instruction (RTS) is placed in test memory.
4. The state of the MPU is saved for return to supervisor mode exceptions.
5. The Transparent Translation Register is turned on.
6. The MPU and the exception switch are set to User mode.
7. The instruction in memory is executed.
8. A trap always true instruction (vector 7) is executed to return the MPU to the supervisor mode.
9. The Transparent Translation Register is turned off.

At the end of the test, the MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If translation doesn't work, an Access Exception will occur.

If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU TTRUC
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

On receipt of an access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
State: ITT_ set for User Code.
```

On receipt of an unexpected exception, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
State: ITT_ set for User Code.
```

TTRUD - TTR User Data

This test checks the ability of the data Transparent Translation Register to access data in User mode. The test runs as follows:

1. The test address is adjusted to the nearest page address.
2. The data and code caches are turned off.
3. A data pattern is placed in test memory.
4. The state of the MPU is saved for return to supervisor mode exceptions.
5. The Transparent Translation Register is turned on.
6. The MPU and the exception switch are set to User mode.
7. The data pattern is read and verified.
8. A trap always true instruction (vector 7) is executed to return the MPU to the supervisor mode.
9. The Transparent Translation Register is turned off.

At the end of the test, the MMU and Cache registers are returned to their original state.

The memory range specified by the configuration parameters starting address and ending address must be at least $32000 bytes.

If translation doesn't work, an Access Exception or a data miscompare occurs.

If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU TTRUD
```

Messages

If the memory range is less than $32000 bytes, the following message appears:

```
Insufficient Amount of Memory to Perform Test.
```

On receipt of an access exception, the following message appears:

```
Bus Error Information:
                  Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
State: DTT_ set for User Data.
```

On receipt of an unexpected exception, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
State: DTT_ set for User Data.
```

If the data pattern directly placed in memory did not match the data read with the TTR enabled, the following message appears:

```
Data Miscompare Error:

Address =________, Expected =________, Actual =________
State: DTT_ set for User Data.
```

TTRWP - TTR Write Protect - TTR

This test checks the ability to write protect memory using the appropriate bits in the Transparent Translation Registers. The test runs as follows:

1. The test address is set to the second 16MB page address.
2. The data and code caches are turned off and the ATCs are flushed.
3. The Transparent Translation Register is turned on with Write Protect set.
4. The exception switch is set to handle Write Protect exceptions.
5. All exceptions are claimed.
6. A test pattern is written to memory. This should cause an access fault.
7. The Transparent Translation Register is turned off.

At the end of the test, the MMU Fault/Status register is read for proper error status. The MMU and Cache registers are returned to their original state.

If the MMU status information does not indicate a Transparent Translation Register hit, or if the MMU status information does not indicate a Write Protect fault, an error message appears. If the access fault exception is not taken, an error message appears.

If during the test an unexpected exception occurs, the test will service it and display the exception information.

Command Input

```
162-Diag> CMMU TTRWP
```

Messages

If test pattern is written to memory but no data fault exception occurs, the following message appears:

```
Access Fault Exception did not occur
State: DTT_ set for Write Protect.
```

On receipt of an access exception, the following message appears:

```
Bus Error Information:
                Address ________
                     Data ________
              Access Size __
              Access Type _
       Address Space Code _
            Vector Number ___
 Exception Stack Frame ________
State: DTT_ set for Write Protect.
```

On receipt of an unexpected exception, the following message appears:

```
Unsolicited Exception:
        Program Counter ________
          Vector Number ___
        Status Register ____
        Interrupt Level _
 Exception Stack Frame ________
State: DTT_ set for Write Protect.
```

If the access exception was not caused by a Write Protect, the following message appears:

```
MMU/Fault Status ________, Physical Address ________.
State: DTT_ set for Write Protect.
```

VME2 - VME Interface ASICs

The **VME2** tests check the VMEchip2 interface ASICs. The tests are listed in Table 3-7, and are described in alphabetical order on the following page.

Enter **VME2** without a test name to run all tests in the group. They will be executed in the order shown in Table 3-7.

Table 3-7. VME2 Tests

Name	Description
REGA	Register Access
REGB	Register Walking Bit
TMRA	Tick Timer 1 Increment
TMRB	Tick Timer 2 Increment
TMRC	Prescaler Clock Adjust
TMRD	Tick Timer 1 No Clear On Compare
TMRE	Tick Timer 2 No Clear On Compare
TMRF	Tick Timer 1 Clear On Compare
TMRG	Tick Timer 2 Clear On Compare
TMRH	Tick Timer 1 Overflow Counter
TMRI	Tick Timer 2 Overflow Counter
TMRJ	Watchdog Timer Counter
TMRK	Watchdog Timer Board Fail
TACU	Timer Accuracy
SWIA	Software Interrupts (Polled Mode)
SWIB	Software Interrupts (Processor Interrupt Mode)
SWIC	Software Interrupts Priority

Configuration Parameters

You may set the following parameters with the **CF** command (the default values are given):

```
Prescaler Clock Adjust Timeout =00FF0000 ?
tmr_cmp(): counter reg mask =FFFFFFF0 ?
User defined Aux ROM base address Enable [Y/N] =N ?
User defined Aux ROM base address =00080000 ?
Master Decoder default select =00000001 ?
Master Write Post Interrupt level =00000001 ?
Master Decoder Trans. test: AUX slave select =00000001 ?
```

REGA - Register Access

This test verifies that the registers at offsets 0 through 84 can be read accessed. The read access algorithm is performed using eight, sixteen, and thirty-two bit data sizes.

Command Input

```
162-Diag>VME2 REGA
```

Messages

If the test fails, the following message appears:

```
VME2/REGA Test Failure Data:

Unsolicited Exception:
     Exception Time PC/IP _____
                Vector _
Access Fault Information:
           Address ________
           Data ________
           Access Size _
           Access Type _
           Address Space Code __
reg_a:
           Data Width __ bits
```

Note All data is hexadecimal.

The Access Fault Information is only displayed if the exception was an Access Fault (Bus Error). Access size is in bytes. Access type is 0 for write and 1 for read.

The address space code message uses the following codes:

1	user data
2	user program
5	supervisor data
6	supervisor program
7	MPU space

REGB - Register Walking Bit

This test verifies that certain bits in the VMEchip2 ASIC user registers can be set independently of other bits in the VMEchip2 ASIC user registers. This test also assures that the VMEchip2 ASIC user registers can be written without a Data Fault (Bus Error). The test runs as follows:

1. The VMEchip2 register walking bit test is implemented by first saving the initial state of the Local Control and Status Registers (LCSR).
2. All eligible bits are initialized to zero. This initialization is verified.
3. A one is walked through the LCSR bit array and the entire register bit field is verified after each write.
4. All eligible bits are initialized to one. This initialization is verified.
5. A zero is walked through the LCSR bit array and the entire register bit field is verified after each write.
6. The initial state of the LCSR is restored except for the LCSR Prescaler Counter register.

Command Input

```
162-Diag>VME2 REGB
```

Messages

If a bit in the LCSR cannot be initialized, the following message appears:

```
bfverf: Bit Field Initialization Error.
                Address ________
                Read Data ________
          Failing Bit Number __ (&__)
          Expected Bit Value _
          Actual Bit Value _
          Exempt Bits Mask ________
```

If a bit in the LCSR fails to respond properly to the walking bit algorithm, the following message appears:

```
regvrf: bit error:
                    Address ________
                    Read Data ________
              Failing Bit Number __ (&__)
              Expected Bit Value __
              Actual Bit Value __
              Exempt Bits Mask ________

              Written Register ________
              Written Bit Number __ (&__)
                    Written Data __
```

If an unexpected interrupt is received while executing the test, the following message appears:

```
Unsolicited Exception:
      Exception Time PC/IP _____
      Vector _
Access Fault Information:
      Address ________
            Data ________
            Access Size _
            Access Type _
      Address Space Code __
```

SWIA - Software Interrupts (Polled Mode)

This test verifies that all software interrupts (1 through 7) can be generated and that the appropriate status is set.

The VMEchip2 local bus interrupter enable register is cleared and the local bus interrupter status register is read to verify that no interrupt status bits are set.

Prior to asserting any SWI set bit, and with local bus interrupter enable register SWI bits asserted, the local bus interrupter status register is again checked to verify that no status bits became true.

As the different combinations of SWI, interrupt level, and, interrupt vector are asserted, verification is made that the expected SWI interrupt status bit did become true, and only that status bit became true.

After the interrupt is generated, the clear bit for the current SWI interrupter is asserted and a check is made to verify the status bit cleared.

Command Input

```
162-Diag>VME2 SWIA
```

Messages

If any interrupt status bits are set, the following message appears:

```
Interrupt Status Register is not initially cleared
Status: Expected =00000000, Actual =________
```

If any status bits becomes true, the following message appears:

```
Interrupt Status Register is not clear
Status: Expected =________, Actual =________
State: IRQ Level =__, SWI__, VBR =__
```

If an unexpected status bit becomes true, the following message appears:

```
Unexpected status set in Interrupt Status Register
Status: Expected =________, Actual =________
State: IRQ Level =__, SWI__, VBR =__
```

If the interrupt status bit does not clear, the following message appears:

```
Interrupt Status Bit did not clear
Status: Expected =________, Actual =________
State: IRQ Level =__, SWI__, VBR =__
```

SWIB - Software Interrupts (Processor Interrupt Mode)

This test verifies that all software interrupts (levels 1 through 7) can be generated and received and that the appropriate status is set.

The interrupt enable register is cleared and status bits are read to verify that none are true. Prior to asserting any SWI set bit, and with local bus interrupter enable register SWI bits asserted, the local bus interrupter status register is checked to verify that no status bit became true.

Command Input

```
162-Diag>VME2 SWIB
```

Messages

If the Interrupt Status Register is not initially cleared, the following message appears:

```
Interrupt Status Register is not initially cleared
Status: Expected =________, Actual =________
```

If any status bit becomes true, the following message appears:

```
Interrupt Status Register is not clear
Status: Expected =________, Actual =________
State : IRQ Level =__, SWI__, VBR =__
```

If the received interrupt vector is not that of the programmed interrupt vector, the following message appears:

```
Unexpected Vector taken
Vector: Expected =____, Actual =____
Status: Expected =____, Actual =____
State : IRQ Level =___, SWI__, VBR =___
```

If the received interrupt level is not that of the programmed interrupt level, the following message appears:

```
Incorrect Interrupt Level
Level : Expected =____, Actual =____
State : IRQ Level =____, SWI__, VBR =____
```

If the programmed interrupt does not occur, the following message appears:

```
Software Interrupt did not occur:
Status: Expected =____, Actual =____
State : IRQ Level =___, SWI__, VBR =___
```

The VMEchip2 Interrupt Status Register is checked for the proper interrupt status bit to be active. If an unexpected status is set, the following message appears:

```
Unexpected status set in Interrupt Status Register
Status: Expected =____, Actual =____
State : IRQ Level =___, SWI__, VBR =___
```

If, after receiving an interrupt, the interrupt status cannot be negated by writing the interrupt clear register, the following message appears:

```
Interrupt Status Bit did not clear
Status: Expected =____, Actual =____
State : IRQ Level =___, SWI__, VBR =___
```

SWIC - Software Interrupts Priority

This test verifies that all software interrupts (1 through 7) occur in the priority set by the hardware.

Command Input

```
162-Diag>VME2 SWIC
```

Messages

The interrupt enable register is cleared and status bits are read to verify that none are true, the following message appears:

```
Interrupt Status Register is not initially cleared
Status: Expected =________, Actual =________
```

If the received interrupt vector is not that of the programmed interrupt vector, the following message appears:

```
Unexpected Vector taken
Vector: Expected =__, Actual =__
Status: Expected =________, Actual =________
State : IRQ Level =___, SWI__, VBR =__
```

If the received interrupt level is not that of the programmed interrupt level, the following message appears:

```
Incorrect Interrupt Level
Level : Expected =__, Actual =__
State : IRQ Level =___, SWI__, VBR =__
```

If the programmed interrupt did not occur, the following message appears:

```
Software Interrupt did not occur
Status: Expected =________, Actual =________
State : IRQ Level =___, SWI__, VBR =__
```

The VMEchip2 Interrupt Status Register is checked for the proper interrupt status bit to be active. If an unexpected status is set, the following message appears:

```
Unexpected status set in Interrupt Status Register
Status: Expected =________, Actual =________
State : IRQ Level =___, SWI__, VBR =__
```

If, after receiving an interrupt, the interrupt status cannot be negated by writing the interrupt clear register, the following message appears:

```
Interrupt Status Bit did not clear
Status: Expected =________, Actual =________
State : IRQ Level =___, SWI__, VBR =__
```

TACU - Timer Accuracy Test

This test performs a four-point verification of the VMEChip2 ASIC timer and prescaler circuitry using the on-board Real Time Clock (RTC) as a timing reference. The test runs as follows:

1. The RTC seconds register is read and the stop, write, and read bits are verified to be negated to ensure that the RTC is in the correct state for use by the firmware-based diagnostics.
2. The prescaler calibration register is checked to verify that it contains one of four legal MPU clock calibration values.
3. Both 32 bit tick timers are programmed to accumulate count, starting at zero, for a period of time determined by the RTC. The accumulated count is verified to be within a predetermined window.
4. The upper 24 bits of the prescaler counter register is read at two intervals whose timing is determined by the RTC. The difference count is verified to be within a predetermined window.

Command Input

```
162-Diag>VME2 TACU
```

Messages

If the RTC is stopped, the following message appears:

```
RTC is stopped, invoke SET command.
```

If the RTC is in the write mode, the following message appears:

```
RTC is in write mode, invoke SET command.
```

If the RTC is in the read mode, the following message appears:

```
RTC is in read mode, invoke SET command.
```

If the prescaler calibration register does not contain one of four legal MPU clock calibration values, the following message appears:

```
Illegal prescaler calibration:
Expected EF, EC, E7, or DF, Actual =__
```

If tick timer accuracy is out of tolerance, the following message appears:

```
Timer counter register read (greater/less) than expected
Address =________, Expected =________, Actual =________
```

If prescaler counter register accuracy is out of tolerance, the following message appears:

```
Prescaler delta was (greater/less) than expected
Address =________, Expected =________, Actual =________
```

If the RTC seconds register does not increment during the test, the following message appears:

```
RTC seconds register didn't increment
```

TMRA, TMRB - Tick Timer Increment

These tests verify that the Tick Timer 1 (or Timer 2) Counter Register can be set to 0, and, that the register value increments when enabled. Use **TMRA** to test Timer 1 and **TMRB** to test Timer 2. The tests run as follows:

1. The Timer is initialized by writing 0 to the Tick Timer Counter Register.
2. The Clear On Compare mode is disabled by writing the COC1 (or COC2) bit in the Tick Timer Control Register.
3. The Timer is enabled by the EN1 (or EN2) bit in the Tick Timer Control Register.
4. The MPU executes a time delay loop and disables Tick Timer 1 (or Tick Timer 2).
5. The Tick Timer Control Register is read to see if it increments from its initial value of 0.

Command Input

```
162-Diag>VME2 TMRA
```

or

```
162-Diag>VME2 TMRB
```

Messages

If the test fails, one of the following messages appears:

```
Tick Timer _ Counter did not clear.

Tick Timer _ Counter did not increment.
```

TMRC - Prescaler Clock Adjust

This test proves that the Prescaler Clock Adjust register can vary the period of the tick timer input clock. The test runs as follows:

1. Two MPU timing loops are executed, the first with a "low" Prescaler Clock Adjust register value, the second with a "high" value.
2. Timer 1 of the VMEchip2 is used for reference in this test.
3. The first MPU loop count is compared with the second MPU loop count. The first MPU loop count is expected to be smaller than the second.
4. The Prescaler Clock Adjust register value is restored upon correct test execution.

The test fails if the Prescaler Clock Adjust register has not been previously initialized to a nonzero value.

Command Input

```
162-Diag>VME2 TMRC
```

Messages

If Prescaler Clock Adjust register was 0, the following message appears:

```
Prescaler Clock Adjust reg was not initialized
```

If there is a first loop timeout, the following message appears:

```
Low value: Timed out waiting for compare (ITIC1) ____ to assert
```

If there is a last loop timeout, the following message appears:

```
High value: Timed out waiting for compare (ITIC1) ___ to assert
```

If the Prescaler Clock Adjust did not vary the tick period, the following message appears:

```
Prescaler Clock Adjust did not vary tick period.
Loop1=________, Loop2=________.
```

TMRD, TMRE - Tick Timer No Clear On Compare

These tests verify the Tick Timer 1 (or Timer 2) No Clear On Compare mode. Use **TMRD** to test Timer 1 and **TMRE** to test Timer 2. The test runs as follows:

1. The Timer is initialized by writing 0 to the Tick Timer Counter Register.
2. The Clear On Compare mode is disabled by writing the COC1 (or COC2) bit in the Tick Timer Control Register.
3. The compare value is initialized by writing $55AA to the Tick Timer Compare Register.
4. The Timer is enabled by the EN*x* (or EN2) bit in the Tick Timer Control Register.
5. After starting the timer, the MPU enters a time delay loop while testing for Tick Timer compare.
6. Tick Timer compare is sensed by reading the TIC1 (or TIC2) bit in the Local Bus Interrupter Status Register.
7. The Timer is stopped when Timer Compare is sensed, or an MPU loop counter register decrements to 0 (timeout).
8. If the MPU loop counter did not time out, the Timer Counter Register is read to make sure that it was not cleared on compare.

Command Input

```
162-Diag>VME2 TMRD
```

or

```
162-Diag>VME2 TMRE
```

Messages

If the test fails, one of the following messages appears:

```
Tick Timer ___: Counter did not clear.

Timer Counter Register = ________/________ (address/data)

Tick Timer ____: Timed out waiting for compare (ITICn).

Tick Timer ____: Timer cleared on compare.
Timer Counter Register = ________/________ (address/data)
```

TMRF, TMRG - Tick Timer Clear On Compare

These tests verify the Tick Timer 1 (or Timer 2) Clear On Compare mode. Use **TMRF** to test Timer 1 and **TMRG** to test Timer 2. The tests run as follows:

1. The Timer is initialized by writing 0 to the Tick Timer Counter Register.
2. The Clear On Compare mode is enabled by writing the COC1 (or COC2) bit in the Tick Timer Control Register.
3. The compare value is initialized by writing $55AA to the Tick Timer Compare Register.
4. The Timer is enabled by the EN1 (or EN2) bit in the Tick Timer Control Register.
5. After starting the timer, the MPU enters a time delay loop while testing for Tick Timer compare.
6. Tick Timer compare is sensed by reading the TIC1 (or TIC2) bit in the Local Bus Interrupter Status Register.
7. The Timer is stopped when Timer Compare is sensed, or an MPU loop counter register decrements to 0 (timeout).
8. If the MPU loop counter did not time out, the Timer Counter Register is read to make sure that it was cleared on compare.

Command Input

```
162-Diag>VME2 TMRF
```

or

```
162-Diag>VME2 TMRG
```

Messages

If the test fails, one of the following messages appears:

```
Tick Timer ____: Counter did not clear.
Timer Counter Register = ________/________ (address/data)

Tick Timer ____: Timed out waiting for compare (ITIC____).

Tick Timer ____: Timer didn't clear on compare.
Timer Counter Register = ________/________ (address/data)
```

TMRH, TMRI - Overflow Counter

These tests verify that the Tick Timer 1 (or Timer 2) Overflow Counter accumulates a count of timer overflow. Use **TMRH** to test Timer 1 and **TMRI** to test Timer 2. The tests run as follows:

1. The COVF bit in the timer control register is asserted and OVF bit is verified to be clear.
2. The timer counter register is set to zero, the timer compare register is loaded with the value $55AA, and the timer is enabled.
3. When TIC1 (or TIC2) becomes true, the timer is disabled and the timer overflow counter register is checked to see that the resultant overflow was counted.

Command Input

```
162-Diag>VME2 TMRH
```

or

```
162-Diag>VME2 TMRI
```

Messages

If the test fails, one of the following messages appears:

```
Timer ____: Overflow Counter did not clear.
Timer Control Register = ________

Tick Timer ____: Counter did not clear.
Timer Counter Register = ________/________ (address/data)

Tick Timer ____: timeout waiting for ITIC____

Tick Timer ____: Overflow counter did not increment
Timer Control Register = ________
```

TMRJ - Watchdog Timer Counter

This test verifies the watchdog timer to ensure functionality at all programmable timing values. This test also checks watchdog timer clear status and timeout functions. The following is done for all programmable watchdog timeouts:

1. Check for linear timeout period with respect to previous timeout.
2. Verify that timeout status can be cleared.

Command Input

```
162-Diag>VME2 TMRJ
```

Messages

If the test fails, one of the following messages appears:

```
Watchdog failed to timeout: mloops=________

out of tolerance
      time out code ________
      actual loops ________
      expected loops ________
      lower limit ________
      upper limit ________

time out status (WDTO bit) could not be cleared
```

TMRK - Watchdog Timer Board Fail

This test verifies the watchdog timer in board fail mode by setting up a watchdog timeout and verifying the status of the VMEchip2 BRFLI status bit in the Board Control register. This test checks BRFLI for WDBFE both negated and asserted states.

Command Input

```
162-Diag>VME2 TMRK
```

Messages

If the test fails, one of the following messages appears:

```
Watchdog failed to timeout: wdbfe=________, mloops=________

BRFLI (at $________) was High, it should have been Low

BRFLI (at $________) was Low, it should have been High

wdog: time out status (WDTO bit) could not be cleared
```

LANC - LAN Coprocessor

The **LANC** tests check the Local Area Network Coprocessor (Intel 82596). The tests are listed in Table 3-8, and are described in alphabetical order on the following pages.

Enter **LANC** without a test name to run all tests (except **ELBC**, **MON**, and **TDR**) in the group. They will be executed in the order shown in Table 3-8.

Table 3-8. LANC Tests

Test	Description
CST	Chip Self Test
BERR	Bus Error
IRQ	Interrupt Request
DUMP	Dump Configuration/Registers
DIAG	Diagnose Internal Hardware
ILB	Internal Loopback
ELBT	External Loopback Transceiver
ELBC	External Loopback Cable
MON	Monitor (Incoming Frames) Mode
TDR	Time Domain Reflectometry

Following the **LANC** test descriptions is a list of the error messages which pertain to all tests within the group.

The 82596 is an intelligent, high-performance LAN coprocessor. It executes high-level commands, command chaining, and inter-processor communications via shared memory. This relieves the host CPU of many tasks associated with network control. All time-critical functions are performed independently of the CPU, which greatly improves network performance.

The 82596 manages all IEEE 802.3 Medium Access Control and channel interface functions, such as framing, preamble generation and stripping, source address insertion, destination address

checking, short frame detection, and automatic length-field handling. The 82596 supports serial data rates up to 20MB per second.

Configuration Parameters

You may set the following parameters with the **CF** command (the default values are shown):

```
Control Memory Base Address Override [Y/N] =N ?
Control Memory Base Address                =00000000 ?
Self Test Results Block Address            =00000000 ?
System Configuration Pointer               =00000000 ?
Intermediate System Configuration Pointer  =00000000 ?
System Control Block Address               =00000000 ?
Configuration Command Block Address        =00000000 ?
Individual Address Command Block Address   =00000000 ?
Diagnose/NOP Command Block Address         =00000000 ?
Dump Configuration/Registers Address       =00000000 ?
TDR Command Block Address                  =00000000 ?
Number Transmit/Receive Loopback Packets   =00000020 ?
Ethernet Address (Source)                  =000000000000 ?
Ethernet Address (Destination)             =000000000000 ?
```

CST - Chip Self Test

This test verifies that the 82596 self-test mode (command) can be executed, and also verifies that the self-test results (expected results) match the actual results. The 82596 provides the results of the self-test at the address specified by the self-test **PORT** command. The self-test command checks the following blocks (of the 82596):

ROM	The contents of the entire ROM is sequentially read into a Linear Feedback Shift Register (LFSR). The LFSR compresses the data and produces a signature unique to one set of data. The results of the LFSR are then compared to a known good ROM signature. The pass or fail result and the LFSR contents are written into the address specified by the self-test **PORT** command.
Parallel Registers	The micro machine performs write and read operations to all internal parallel registers and checks the contents for proper values. The pass or fail result is then written into the address specified by the self-test **PORT** command.
Bus Throttle Timers	The micro machine performs an extensive test of the Bus Throttle timer cells and decrementation logic. The counters are enabled and the contents are checked for proper values. The pass or fail result is then written to the address specified by the self-test **PORT** command.
Diagnose	The micro machine issues an internal diagnose command to the serial subsystem. The pass or fail result of the Diagnose command is then written into the address specified by the self-test **PORT** command.

Command Input

```
162-Diag>LANC CST
```

Messages

If the expected results do not match (equal) the actual results of the 82596 self-test command results, the following message appears:

```
LANC Chip Self-Test Error: Expected =_______, Actual =_______
```

DIAG - Diagnose Internal Hardware

This test verifies that the Diagnose command of the 82596 can be executed, and that an error-free completion status is returned. The Diagnose command triggers an internal self-test procedure that checks the 82596 hardware, which includes the following:

- Exponential Backoff Random Number Generator (Linear Feedback Shift Register).
- Exponential Backoff Timeout Counter
- Slot Time Period Counter
- Collision Number Counter
- Exponential Backoff Shift Register
- Exponential Backoff Mask Logic
- Timer Trigger Logic

The Channel Interface Module of the 82596 performs the self-test procedure in two phases: Phase 1 tests the counters and Phase 2 tests the trigger logic.

During Phase 1, the Linear Feedback Shift Register (LFSR) and the Exponential Backoff Timer, Slot Timer, and Collision Counters are checked.

During Phase 1, the test runs as follows:

1. All counters and shift registers are reset simultaneously.
2. Starts counting and shifting the registers.
3. The Exponential Backoff Shift Register reaches all ones.
4. Checks the Exponential Backoff Shift Register for all ones when the LFSR content is all ones in its least significant bits.
5. Stops counting when the LFSR (30 bits) reaches a specific state, and Exponential Backoff Counter (10 bits) wraps from all ones to all zeros. Simultaneously, the Slot Time Counter

switches from 01111111111 to 10000000000, and the collision counter (4 bits) wraps from all ones to all zeros.

6. Phase 1 is successful if the 10 least significant bits (when applicable) of all four counters are all zeros.

During Phase 2, the test runs as follows:

1. Resets Exponential Backoff Shift Register and all counters.
2. Temporarily configures Exponential Backoff logic, internally, according to the following:

SLOT-TIME	= $3
LIN-PRIO	= $6
EXP-PRIO	= $3
BOF-MET	= $0

3. Emulates transmission and collision, internally.
4. If the most significant bit of Exponential Backoff Shift Register is 1, a `Passed` status is returned.
5. If Step 4 is not successful (a 0), a `Failed` status is returned, and Step 3 is repeated.

Command Input

```
162-Diag>LANC DIAG
```

Messages

If the **DIAG** test fails, the following message appears:

```
DIAGNOSE Command Completion Status Error:
OK-Bit =0, F(ail)-Bit =1
```

DUMP - Dump Configuration/Registers

This test verifies that the Dump command of the 82596 can be executed, and that an error free completion status is returned. The Dump command instructs the 82596 to transfer the configuration parameters and contents of other registers from the Channel Interface Module via RCV-FIFO by Receive Unit to memory.

The test issues the Dump command to the 82596 and waits for two seconds. Once the delay has expired, the test verifies the command completion status. The 82596 performs the following sequence upon the receipt of the Dump command:

1. Starts Action command.
2. Writes Dump command byte to TX-FIFO.
3. Waits for completion of DUMP.
4. Prepares STATUS word with C=1, B=0, and OK=1.
5. Completes Action command.

Command Input

```
162-Diag>LANC DUMP
```

Messages

If the **DUMP** test fails, the following message appears:

```
Dump Status Error: Expected =A006, Actual =8006
```

ELBC - External Loopback Cable

This test verifies that the 82596 can be operated with the External Loopback and with the LPBK pin not activated.

The test sets up a data packet (incrementing data pattern) to be transmitted, and instructs the 82596 (through the Command Unit) to transmit the data packet. Once the data packet is transmitted successfully, the test waits for four seconds for the receipt of the data.

After the data is received, the test verifies the status of the receive data packet, and verifies that the number of bytes received equals the number of bytes transmitted. Upon completion of all the status checks, the test verifies the received data to the transmitted data.

The transmit to receive loop is performed 32 times (the default for the **CF** command parameter `Number Transmit/Receive Loopback Packets`).

During this test, the 82596 transmits and receives simultaneously at a full rate. This allows checking external hardware as well as the serial link to the transceiver interface.

Note that this test does not execute when the **LANC** test group is executed. This test is supplied only for diagnostic purposes. It requires a properly set up Ethernet network (cable).

Command Input

```
162-Diag>LANC ELBC
```

Messages

If the 82596 completes with a transmit data error, the following message appears:

```
TRANSMIT Command Completion Status Error:
OK-Bit =0, ABORT-Bit =0, STATUS-Bits =0010
```

The STATUS-Bits (the hex value represents a bit sting, big endian) indicate the type of error:

6 A late collision (a collision after the slot time elapsed) is detected.

5 No Carrier Sense signal during transmission. Carrier Sense signal is monitored from the end of Preamble transmission until the end of the Frame Check Sequence for TONOCRS = 1 (Transmit On No Carrier Sense Mode); it indicates that transmission has been executed despite a lack of CRS. For TONOCRS = 0 (Ethernet mode), this bit also indicates unsuccessful transmission (transmission stopped when lack of Carrier Sense has been detected).

4 Transmission unsuccessful (stopped) due to Loss of Clear to Send signal.

3 Transmission unsuccessful (stopped) due to DMA Underrun; i.e., the system did not supply data for transmission.

2 Transmission Deferred, i.e., transmission was not immediate due to previous link activity.

1 Heartbeat Indicator. Indicates that after a previously performed transmission, and before the most recently performed transmission, (Interframe Spacing) the CDT signal was monitored as active. This indicates that the Ethernet Transceiver Collision Detect logic is performing well. The Heartbeat is monitored during Interframe Spacing period.

0 Transmission attempt was stopped because the number of collisions exceeded the maximum allowable number of retries.

If the data receive timeout (four seconds) expires, the following message appears:

```
RECEIVE Data Time-Out
```

If the data packet has been received in error, the following message appears:

```
RECEIVE Status Error:
COMPLETE-Bit =1, OK-Bit=0, STATUS-Bits =0000
```

The STATUS-Bits (the hex value represents a bit sting, big endian) indicate the type of error:

12	Length of error if configured to check length.
11	CRC error in an aligned frame.
10	Alignment error (CRC error in a misaligned frame).
9	Ran out of buffer space - no resources.
8	DMA Overrun. Failure to acquire the system bus.
7	Frame too short.
6	No EOP flag (for Bit stuffing only).
1	IA Match Bit. When it is zero, the destination address of a received frame matches the IA address. When it is one, the destination address of the received frame does not match the individual address. For example, a multicast or broadcast address sets this bit to a one.
0	Receive collision. A collision is detected during reception.

If the receive data count and the transmit data count are not equal, the following message appears:

```
RECEIVE Data Transfer Count Error:
Expected =05EA, Actual =003C
```

If the transmit and receive data do not verify (compare), the following message appears:

```
Receive Data Miscompare Error:
Address =0000E2C0, Expected =3E3F, Actual =3E3E
```

ELBT - External Loopback Transceiver

This test verifies that the 82596 can be operated with the External Loopback and with the LPBK pin activated.

The test sets up a data packet (incrementing data pattern) to be transmitted, and instructs the 82596 (through the Command Unit) to transmit the data packet. Once the data packet is transmitted successfully, the test waits for four seconds for the receipt of the data.

After the data is received, the test verifies the status of the receive data packet, and verifies that the number of bytes received equals the number of bytes transmitted. Upon completion of all the status checks, the test verifies the received data to the transmitted data.

The transmit to receive loop is performed 32 times (the default for the **CF** command parameter `Number Transmit/Receive Loopback Packets`).

The LPBK pin is connected to the accompanying Ethernet Serial Interface (ESI - 82C501AD) chip. The ESI is then connected to the pulse transformer (PE64102), which in turn is connected to the Ethernet Connector.

During the test, the 82596 transmits and receives simultaneously at a full rate. This allows checking external hardware as well as the serial link to the transceiver interface. The LPBK pin is used to inform the external hardware (ESI) of the establishment of a transmit to receive connection.

Command Input

```
162-Diag>LANC ELBT
```

Messages

If the 82596 completes with a transmit data error, the following message appears:

```
TRANSMIT Command Completion Status Error:
OK-Bit =0, ABORT-Bit =0, STATUS-Bits =0010
```

The STATUS-Bits (the hex value represents a bit sting, big endian) indicate the type of error:

6 A late collision (a collision after the slot time elapsed) is detected.

5 No Carrier Sense signal during transmission. Carrier Sense signal is monitored from the end of Preamble transmission until the end of the Frame Check Sequence for TONOCRS = 1 (Transmit On No Carrier Sense Mode); it indicates that transmission has been executed despite a lack of CRS. For TONOCRS = 0 (Ethernet mode), this bit also indicates unsuccessful transmission (transmission stopped when lack of Carrier Sense has been detected).

4 Transmission unsuccessful (stopped) due to Loss of Clear to Send signal.

3 Transmission unsuccessful (stopped) due to DMA Underrun; i.e., the system did not supply data for transmission.

2 Transmission Deferred, i.e., transmission was not immediate due to previous link activity.

1 Heartbeat Indicator. Indicates that after a previously performed transmission, and before the most recently performed transmission, (Interframe Spacing) the CDT signal was monitored as active. This indicates that the Ethernet Transceiver Collision Detect logic is performing well. The Heartbeat is monitored during Interframe Spacing period.

0 Transmission attempt was stopped because the number of collisions exceeded the maximum allowable number of retries.

If the data receive timeout (four seconds) expires, the following message appears:

```
RECEIVE Data Time-Out
```

If the data packet has been received in error, the following message appears:

```
RECEIVE Status Error:
COMPLETE-Bit =1, OK-Bit=0, STATUS-Bits =0000
```

The STATUS-Bits (the hex value represents a bit sting, big endian) indicate the type of error:

12	Length of error if configured to check length.
11	CRC error in an aligned frame.
10	Alignment error (CRC error in a misaligned frame).
9	Ran out of buffer space - no resources.
8	DMA Overrun. Failure to acquire the system bus.
7	Frame too short.
6	No EOP flag (for Bit stuffing only).
1	IA Match Bit. When it is zero, the destination address of a received frame matches the IA address. When it is one, the destination address of the received frame does not match the individual address. For example, a multicast or broadcast address sets this bit to a one.
0	Receive collision. A collision is detected during reception.

If the receive data count and the transmit data count are not equal, the following message appears:

```
RECEIVE Data Transfer Count Error:
Expected =05EA, Actual =003C
```

If the transmit and receive data do not verify (compare), the following message appears:

```
Receive Data Miscompare Error:
Address =0000E2C0, Expected =3E3F, Actual =3E3E
```

ILB - Internal Loopback

This test verifies that the 82596 can be operated in the Internal Loopback mode.

The test sets up a data packet (incrementing data pattern) to be transmitted, and instructs the 82596 (through the Command Unit) to transmit the data packet. Once the data packet is transmitted successfully, the test waits for four seconds for the receipt of the data.

After the data is received, the test verifies the status of the receive data packet, and verifies that the number of bytes received equals the number of bytes transmitted. Upon completion of all the status checks, the test verifies the received data to the transmitted data.

The transmit to receive loop is performed 32 times (the default for the **CF** command parameter `Number Transmit/Receive Loopback Packets`).

During the test, the 82596 disconnects itself from the serial link and logically connects TXD to RXD and TXC to RXC. The TXC frequency is internally divided by four during internal loopback operation.

Command Input

```
162-Diag>LANC ILB
```

Messages

If the 82596 completes with a transmit data error, the following message appears:

```
TRANSMIT Command Completion Status Error:
OK-Bit =0, ABORT-Bit =0, STATUS-Bits =0010
```

The STATUS-Bits (the hex value represents a bit sting, big endian) indicate the type of error:

6 A late collision (a collision after the slot time elapsed) is detected.

5 No Carrier Sense signal during transmission. Carrier Sense signal is monitored from the end of Preamble transmission until the end of the Frame Check Sequence for TONOCRS = 1 (Transmit On No Carrier Sense Mode); it indicates that transmission has been executed despite a lack of CRS. For TONOCRS = 0 (Ethernet mode), this bit also indicates unsuccessful transmission (transmission stopped when lack of Carrier Sense has been detected).

4 Transmission unsuccessful (stopped) due to Loss of Clear to Send signal.

3 Transmission unsuccessful (stopped) due to DMA Underrun; i.e., the system did not supply data for transmission.

2 Transmission Deferred, i.e., transmission was not immediate due to previous link activity.

1 Heartbeat Indicator. Indicates that after a previously performed transmission, and before the most recently performed transmission, (Interframe Spacing) the CDT signal was monitored as active. This indicates that the Ethernet Transceiver Collision Detect logic is performing well. The Heartbeat is monitored during Interframe Spacing period.

0 Transmission attempt was stopped because the number of collisions exceeded the maximum allowable number of retries.

If the data receive timeout (four seconds) expires, the following message appears:

```
RECEIVE Data Time-Out
```

If the data packet has been received in error, the following message appears:

```
RECEIVE Status Error:
COMPLETE-Bit =1, OK-Bit=0, STATUS-Bits =0000
```

The STATUS-Bits (the hex value represents a bit sting, big endian) indicate the type of error:

12	Length of error if configured to check length.
11	CRC error in an aligned frame.
10	Alignment error (CRC error in a misaligned frame).
9	Ran out of buffer space - no resources.
8	DMA Overrun. Failure to acquire the system bus.
7	Frame too short.
6	No EOP flag (for Bit stuffing only).
1	IA Match Bit. When it is zero, the destination address of a received frame matches the IA address. When it is one, the destination address of the received frame does not match the individual address. For example, a multicast or broadcast address sets this bit to a one.
0	Receive collision. A collision is detected during reception.

If the receive data count and the transmit data count are not equal, the following message appears:

```
RECEIVE Data Transfer Count Error:
Expected =05EA, Actual =003C
```

If the transmit and receive data do not verify (compare), the following message appears:

```
Receive Data Miscompare Error:
Address =0000E2C0, Expected =3E3F, Actual =3E3E
```

IRQ - Interrupt Request

This test verifies that the 82596 can assert an interrupt request to the MPU. The 82596 has only one line to signal its interrupt request. The 82596's interrupt request is controlled by the PCC2.

The test issues an initialization sequence of the 82596 to occur. Upon completion of the initialization, the 82596 asserts its interrupt request line to the MPU via the PCC2. The test verifies that the appropriate interrupt status is set in the PCC2 and also that the interrupt status can be cleared.

Prior to the 82596 initialization sequence launch, the interrupt control register in the PCC2 is verified against the pretest expected results. Upon completion of the initialization sequence of the 82596, the test verifies the interrupt control register for interrupt status. Once the interrupt status is verified, the interrupt status is cleared via the ICLR bit in the interrupt control register in the PCC2.

Command Input

```
162-Diag>LANC IRQ
```

Messages

If the register contents do not verify against the expected pretest results, the following message appears:

```
LANC Interrupt Control/Status Register Error:
Expected =50, Actual =70
```

If the register contents do not verify against the expected post test results (i.e., interrupt status bit not set), the following message appears:

```
LANC Interrupt Control/Status Register Error:
Expected =70, Actual =50
```

If the interrupt status bit (INT) in the interrupt control register does not clear, the following message appears:

```
LANC Interrupt Control/Status Register Error:
Expected =50, Actual =70
```

MON - Monitor (Incoming Frames) Mode

This utility monitors activities on the LAN. It instructs the 82596 to monitor all incoming (receive data) frames. No frames are transferred to memory (i.e., 82596 Monitor Mode #3). This utility executes continuously. You must press the BREAK key to exit (abort). No PASS or FAIL message is issued.

This utility does not run when the **LANC** test group is executed.

Command Input

```
162-Diag>LANC MON
```

Messages

The following status message appears while the test is executing:

```
CRCE=0000000 AE=0000000 SF=0000000 RC=0000000 TGB=0000000 TG=0000000
```

where:

CRCE	the number of aligned frames discarded because of a CRC error
AE	the number of frames that are both misaligned (i.e., CRS de-asserts on a non-octet boundary) and contain a CRC error
SF	the number of received frames that are shorter than the minimum length
RC	the number of collisions detected during frame reception
TGB	the number of good and bad frames received
TG	the number of good frames received

Each element is a 32-bit count.

Only one of these counters is incremented per frame. The SF counter has priority over CRCE, AE, and RC counters. For example, if a received frame is both short and collided, only the SF counter is incremented.

TDF -Time Domain Reflectometry

This test verifies that Time Domain Reflectometry (TDR) can be executed, and that an error free completion status is returned. The TDR detects open or shorts on the link and their distance from the diagnosing station. The maximum length of the TDR frame is 2048 bits. The test runs as follows:

1. The TDR is activated.
2. If the 82596 senses collision while transmitting the TDR frame it transmits the jam pattern and stops the transmission.
3. The 82596 triggers the internal timer (STC); the timer is reset at the beginning of transmission and reset if CRS is returned.
4. The timer measures the time elapsed from the start of transmission until an echo is returned. The echo is indicated by Collision Detect going active or a drop in the Carrier Sense signal.

There are four possible results:

- The Carrier Sense signal does not go active before the counter expires. For a Transceiver that should return Carrier Sense during transmission, this means that there is a problem on the cable between the 82596 and the Transceiver. For a Transceiver that should not return Carrier Sense during transmission, this is normal.
- The Carrier Sense signal goes active and then inactive before the counter expires. For a Transceiver that should return Carrier Sense during transmission, this means that there is a short on the link.
- The Collision Detect signal goes active before the counter expires. This means that the link is not properly terminated (an open).
- The Carrier Sense signal goes active but does not go inactive and Collision Detect does not go active before the counter

expires. This is the normal case and indicates that there is no problem on the link.

The distance to the cable failure can be calculated as follows:

Distance = T * (Vs / (2 * Fs))

where:

T = time in seconds
Vs = wave propagation speed on the link (M/s)
Fs = serial clock frequency (Hz)

Accuracy is plus/minus Vs / (2 * Fs).

Once the TDR command has completed successfully, the LINK-OK bit is checked in the TDR command packet.

Note that this test does not run when the **LANC** test group is executed. This test is supplied only for diagnostic purposes. It requires a properly set up Ethernet network (cable).

Command Input

```
162-Diag>LANC TDR
```

Messages

If the TDR command executes with an error status, the following message appears:

```
TDR Command Completion Status Error:
OK-Bit =0
```

If the result of the LINK-OK bit is false (problem with link), the following message appears:

```
TDR Command Results Error:
Transceiver Problem       =TRUE or FALSE
Termination Problem       =TRUE or FALSE
Transmission Line Shorted =TRUE or FALSE
Transmit Clock Cycles     =0 to 7FF
```

LANC Test Group Error Messages

The following error messages may apply to any of the **LANC** tests:

Table 3-9. LANC Error Messages

Message	Cause
`Test Initialization Error:` `Not Enough Memory, Need =00010000, Actual =000087F0`	The amount memory found during the diagnostics subsystem initialization is less than the amount of memory needed by the **LANC** test group.
`Test Initialization Error:` `Control Memory Address Not 16 Byte Aligned =0000E008`	The control memory address specified by the **LANC** test group configuration parameters is not 16-byte aligned.
`LANC Initialization Error:` `SCB Read Failure (Channel Attention Signal)`	The busy byte in the ISCP did not become clear after one tenth of a second from the issue of the channel attention. The Intermediate System Configuration Pointer (ISCP) indicates the location of the System Control Block (SCB). The CPU loads the SCB address into the ISCP and asserts Channel Attention (CA). This Channel Attention signal causes the 82596 to begin its initialization procedure to get the SCB address from the ISCP. The SCB is the central point through which the CPU and the 82596 exchange control and status information.
`LANC Initialization Error:` `LANC Command Unit Command Acceptance Time-Out`	The 82596 command queue is not accepting the interrupt acknowledge command. During the initialization process of the 82596, the **LANC** test group initialization function issues an interrupt acknowledge command to the 82596 to acknowledge the completion of the 82596 initialization.

Table 3-9. LANC Error Messages

Message	Cause
`LANC Initialization Error:` `LANC Command Unit Interrupt Acknowledge Command Completion Time-Out`	The command timed out. During the initialization process of the 82596, the **LANC** test group initialization function issues an interrupt acknowledge command to the 82596 to acknowledge the completion of the 82596 initialization. Once the command is accepted by the 82596, the initialization function waits for the 82596 to post status of the completion of the command.
`LANC Error Status Register (DMA Bits) Not Clear =02`	There is a bus error. At the completion of each test in the **LANC** test group, the **LANC** error status register (PCC2 - $FFF42028) is checked for any possible bus error conditions that may have been encountered by the **LANC** while performing DMA accesses to the local bus.
`LANC Command Unit Not Idle (Busy)`	The command unit is not in the idle state. Prior to issuing a command to the Command Unit of the 82596, the command execution function verifies that the command unit is idle.
`LANC Receive Unit Not Idle (Busy)`	The receive unit is not in the idle state. Prior to issuing a command to the Receive Unit of the 82596, the receive command execution function verifies that the receive unit is idle.

Table 3-9. LANC Error Messages

Message	Cause
`LANC Command Unit Interrupt(s) Pending`	The command unit has pending interrupt requests. Prior to issuing a command to the Command Unit of the 82596, the command execution function verifies that the command unit does not have any outstanding (pending) interrupt requests.
`LANC Command Unit Command Acceptance Time-Out`	The command acceptance timeout expired. When a command is issued to the 82596, the command execution function verifies that the 82596 accepted the command. The command execution function waits for one second for this event to occur.
`LANC Command Unit Command Completion Time-Out`	The command completion timeout expired. Once a command has been accepted by the 82596, the command execution function waits for the command to complete. The command execution function waits for eight seconds for this event to occur.
`LANC Command Unit Interrupt Status Time-Out`	The interrupt status timeout expired. Once a command has been completed by the 82596, the command execution function waits for the appropriate interrupt status to be posted by the 82596. The command execution function waits for one second for this event to occur.

Table 3-9. LANC Error Messages

Message	Cause
`LANC Command Unit Interrupt Acknowledge Command Completion Time-Out`	The interrupt acknowledge timeout expired. Once the appropriate interrupt status is set by the 82596, the command execution function issues an interrupt acknowledge command to the command unit of the 82596. Once this command is issued to the 82596, the command execution function waits for one second for the 82596 to post the completion of the interrupt acknowledge command.
`LANC Receive Unit Command Acceptance Time-Out`	The receive command acceptance timeout expired. When a receive command is issued to the 82596, the receive command execution function verifies that the 82596 accepted the receive command. The receive command execution function waits for one second for this event to occur.
`LANC Receive Unit Interrupt Acknowledge Command Completion Time-Out`	The receive interrupt acknowledge timeout expired. Once the appropriate interrupt status is set by the 82596, the receive command execution function issues an interrupt acknowledge command to the receive command unit of the 82596. Once this command is issued to the 82596, the receive command execution function waits for one second for the 82596 to post the completion of the interrupt acknowledge command.

Table 3-9. LANC Error Messages

Message	Cause
`Configure Command Completion Status Error:` `OK-Bit =0, ABORT-Bit =0`	An error occurred in completing the command. Upon completion of the Configure with Operating Parameters command, the command completion status is verified that it was successful.
`Individual Address Setup Command Completion Status Error:` `OK-Bit =0, ABORT-Bit =0`	An error occurred in completing the command. Upon completion of the Individual Address Setup command, the command completion status is verified that it was successful.

NCR - NCR 53C710 SCSI I/O Processor

The NCR tests check the NCR 53C710 SCSI I/O Processor. The tests are listed in Table 3-10, and are described in alphabetical order on the following pages.

Enter **NCR** without a test name to run all tests in the group. They will be executed in the order shown in Table 3-10.

Table 3-10. NCR Tests

Test	Description
ACC1	Device Access
ACC2	Register Access
SFIFO	SCSI FIFO
DFIFO	DMA FIFO
LPBK	Loopback
SCRIPTS	SCRIPTs Processor
IRQ	Interrupts

Configuration Parameters

You may set the following parameters with the **CF** command (the default values are given):

```
Test Memory Base Address Override [Y/N] =N ?
Test Memory Base Address                =00000000 ?
Diagnostic Base Address                 =00000000 (READ ONLY) ?
SCRIPTs Buffer Base Address             =00000000 (READ ONLY) ?
Memory Move Address (Source)            =00000000 ?
Memory Move Address (Destination)       =00000000 ?
Memory Move Byte Count                  =00002000 ?
```

The Test Memory Base Address parameters are used by the **IRQ** and **SCRIPTS** tests. The Memory Move Address and Byte Count parameters are used by the **SCRIPTS** test.

ACC1 - Device Access

This test verifies the basic ability to access the NCR 53C710 device.

1. All device registers are accessed (read) on 8-bit and 32-bit boundaries. (No attempt is made to verify the contents of the registers.)
2. The device data lines are checked by successive writes and reads to the SCRATCH register, by walking a 1 bit through a field of zeros and walking a 0 bit through a field of ones.

If no errors are detected, the NCR device is reset, otherwise the device is left in the test state.

Command Input

```
162-Diag>NCR ACC1
```

Messages

If any part of the test fails, one of the following messages appears:

```
SCRATCH Register is not initially cleared

Device Access Error:
Address =________, Expected =________, Actual =________

Device Access Error:

Bus Error Information:
            Address ________
            Data ________
            Access Size __
            Access Type _
    Address Space Code _
            Vector Number ___

Unsolicited Exception:
    Program Counter ________
            Vector Number ___
            Status Register ____
            Interrupt Level _
```

Note All data is hexadecimal.

The Access Fault Information is only displayed if the exception was an Access Fault (Bus Error). Access size is in bytes. Access type is 0 for write or 1 for read.

The address space code message uses the following codes:

1 user data
2 user program
5 supervisor data
6 supervisor program
7 MPU space

ACC2 - Register Access

This test verifies the basic ability to access the NCR 53C710 registers by checking the state of the registers from a software reset condition and checking their read/write ability. Status registers are checked for initial clear condition after a software reset. Writable registers are written and read by walking a 1 through a field of zeros. If no errors are detected, the NCR device is reset, otherwise the device is left in the test state.

Command Input

```
162-Diag>NCR ACC2
```

Messages

If any part of the test fails, one of the following messages appears:

```
ISTAT Register is not initially cleared

SSTAT0 Register is not initially cleared

SSTAT1 Register is not initially cleared

SSTAT2 Register is not initially cleared

SIEN Register Error:
Address =________, Expected =__, Actual =__

SDID Register Error:
Address =________, Expected =__, Actual =__

SODL Register Error:
Address =________, Expected =__, Actual =__

SXFER Register Error:
Address =________, Expected =__, Actual =__

SCID Register Error:
Address =________, Expected =__, Actual =__

DSA Register Error:
Address =________, Expected =________, Actual =________

TEMP Register Error:
Address =________, Expected =________, Actual =________
```

```
DMA Next Address Error:
Address =________, Expected =________, Actual =________

Register Access Error:

Bus Error Information:
          Address ________
          Data ________
          Access Size __
          Access Type _
      Address Space Code _
      Vector Number ___

Unsolicited Exception:
      Program Counter ________
          Vector Number ___
      Status Register ____
      Interrupt Level _
```

Note All data is in hexadecimal.

The Unsolicited Exception information is only displayed if the exception was not a Bus Error.

Access Size is in bytes. Access Type is 0 for write or 1 for read.

The address space code message uses the following codes:

1	user data
2	user program
5	supervisor data
6	supervisor program
7	MPU space

DFIFO - DMA FIFO

This tests verifies the ability to write data into the DMA FIFO and retrieve it in the same order as written. The test works as follows:

1. The DMA FIFO is checked for an empty condition following a software reset.
2. The FBL2 bit is set and verified.
3. The FIFO is filled with 16 bytes of data in the four byte lanes verifying the byte lane full or empty with each write.
4. The FIFO is read verifying the data and the byte lane full or empty with each read.
5. If no errors are detected, the NCR device is reset, otherwise the device is left in the test state.

Command Input

```
162-Diag>NCR DFIFO
```

Messages

If any part of the test fails, one of the following messages appears:

```
DMA FIFO is not initially empty

DMA FIFO Byte Control not enabled
Address =_______, Expected =__, Actual =__

DMA FIFO Byte Control Error:
Address =_______, Expected =__, Actual =__

DMA FIFO Empty/Full Error:
Address =_______, Expected =__, Actual =__

DMA FIFO Parity Error:
Address =_______, Expected =__, Actual =__ DMA FIFO Byte Lane _

DMA FIFO Error:
Address =_______, Expected =__, Actual =__ DMA FIFO Byte Lane _
```

IRQ - Interrupts

This test verifies that level 0 interrupts will not generate an interrupt, but will set the appropriate status. The test then verifies that all interrupts (1-7) can be generated and received and that the appropriate status is set.

Command Input

```
162-Diag>NCR IRQ
```

Messages

If any part of the test fails, one of the following messages appears:

```
Test Initialization Error:
Not Enough Memory, Need =________, Actual =________

Test Initialization Error:
Memory Move Byte Count to Large, Max =00ffffff, Requested =____

Test Initialization Error:
Test Memory Base Address Not 32 Bit Aligned =________

SCSI Status Zero “SGE” bit not set
Address =________, Expected =__, Actual =__

Interrupt Status “SIP” bit not set
Address =________, Expected =__, Actual =__

SCSI Status Zero “SGE” bit will not clear
Address =________, Expected =__, Actual =__

Interrupt Status “SIP” bit will not clear
Address =________, Expected =__, Actual =__

Interrupt Control Reg. not initially clear
Address =________, Expected =__, Actual =__

SCSI Interrupt Enable “SGE” bit not set
Address =________, Expected =__, Actual =__

Interrupt Control “IEN” bit not set
Address =________, Expected =__, Actual =__
```

```
Interrupt Status bit did not set
Status: Expected =__, Actual =__
Vector: Expected =__, Actual =__
State : IRQ Level =_, VBR =__

Interrupt Control “INT” bit will not clear
Address =________, Expected =__, Actual =__

SCSI Interrupt Enable Reg. will not mask interrupts
Address =________, Expected =__, Actual =__

Incorrect Vector type
Status: Expected =__, Actual =__
Vector: Expected =__, Actual =__
State : IRQ Level =_, VBR =__

SCSI Interrupt
Status: Expected =__, Actual =__
DMA Interrupt
Status: Expected =__, Actual =__

Unexpected Vector taken
Status: Expected =__, Actual =__
Vector: Expected =__, Actual =__
State : IRQ Level =_, VBR =__

Incorrect Interrupt Level
Level : Expected =_, Actual =_
State : IRQ Level =_, VBR =__

Interrupt did not occur
Status: Expected =__, Actual =__
Vector: Expected =__, Actual =__
State : IRQ Level =_, VBR =__

Interrupt Status bit did not set
Status: Expected =__, Actual =__
Vector: Expected =__, Actual =__
State : IRQ Level =_, VBR =__

Interrupt Control “INT” bit will not clear
Address =________, Expected =__, Actual =__
```

```
Bus Error Information:
            Address ________
            Data ________
            Access Size __
            Access Type _
            Address Space Code _
            Vector Number ___

Unsolicited Exception:
            Program Counter ________
            Vector Number ___
            Status Register ____
            Interrupt Level _
```

LPBK - Loopback

This test checks the Input and Output Data Latches and performs a selection. The 53C710 executes initiator instructions and the host CPU implements the target role by asserting and polling the appropriate SCSI signals. If no errors are detected, the SCSI I/O Processor is reset, otherwise the device is left in the test state.

The 53C710 Loopback Mode in effect lets the chip talk to itself. When the Loopback Enable (SLBE) bit is set in the CTEST4 register, the 53C710 allows control of all SCSI signals.

Command Input

```
162-Diag>NCR LPBK
```

Messages

If any part of the test fails, one of the following messages appears:

```
No Automatic Clear of 'ADCK' bit in 'CTEST5' Register

No Automatic Clear of 'BBCK' bit in 'CTEST5' Register

NCR SCSI Bus Data Lines Error:
Address =________, Expected =__, Actual =__

DMA Next Address Error:
Address =________, Expected =________, Actual =________

DMA Byte Counter Error:
Address =________, Expected =________, Actual =________
```

SCRIPTS - SCRIPTS Processor

This test initializes the test structures and makes use of the diagnostic registers for test. It runs as follows:

1. Verifies that the following registers are initially clear:

SIEN	SCSI Interrupt Enable
DIEN	DMA Interrupt Enable
SSTAT0	SCSI Status Zero
DSTAT	DMA Status
ISTAT	Interrupt Status
SFBR	SCSI First Byte Received

2. Sets SCSI outputs in high impedance state, disables interrupts using the “MIEN”, and sets NCR device for Single Step Mode.
3. The address of a simple “INTERRUPT instruction” SCRIPT is loaded into the DMA SCRIPTs Pointer register. The SCRIPTs processor is started by hitting the “STD” bit in the DMA Control Register.
4. Single Step is checked by verifying that ONLY the first instruction executed and that the correct status bits are set. Single Step Mode is turned off and the SCRIPTs processor started again. The “INTERRUPT instruction” should be executed and a check for the correct status bits set is made.
5. The address of the “JUMP instruction” SCRIPT is loaded into the DMA SCRIPTs Pointer register, and the SCRIPTs processor is automatically started. JUMP “if TRUE” (Compare = True, Compare = False) conditions are checked, then JUMP “if FALSE” (Compare = True, Compare = False) conditions are checked.
6. The “Memory Move instruction” SCRIPT is built in a script buffer to allow the “Source Address”, “Destination Address”, and “Byte Count” to be changed by use of the “config” command. If a parameter is changed, the only check for

validity is the “Byte Count” during test structures initialization.

7. The “Memory Move” SCRIPT copies the specified number of bytes from the source address to the destination address.

Command Input

```
162-Diag>NCR SCRIPTS
```

Messages

If any part of the test fails, one of the following messages appears:

```
Test Initialization Error:
Not Enough Memory, Need =________, Actual =________

Test Initialization Error:
Memory Move Byte Count to Large, Max =00ffffff, Requested =________

Test Initialization Error:
Test Memory Base Address Not 32 Bit Aligned =________

SCSI Interrupt Enable Reg. not initially clear
Address =________, Expected =__, Actual =__

DMA Interrupt Enable Reg. not initially clear
Address =________, Expected =__, Actual =__

SCSI Status Zero Reg. not initially clear
Address =________, Expected =__, Actual =__

DMA Status Reg. not initially clear
Address =________, Expected =__, Actual =__

Interrupt Status Reg. not initially clear
Address =________, Expected =__, Actual =__

SCSI First Byte Received Reg. not initially clear
Address =________, Expected =__, Actual =__

SCSI First Byte Received Reg. not set
Address =________, Expected =__, Actual =__

DMA Status “SSI” bit not set
Address =________, Expected =__, Actual =__

Interrupt Status “DIP” bit not set
Address =________, Expected =__, Actual =__
```

```
SCSI Status Zero Reg. set during single step
Address =________, Expected =__, Actual =__

Test Timeout during: INTERRUPT SCRIPTs Test
Address =________, Expected =__, Actual =__

“SIR” not detected during: INTERRUPT SCRIPTs Test
Address =________, Expected =__, Actual =__

Test Timeout during: JUMP SCRIPTs Test
Address =________, Expected =__, Actual =__

“SIR” not detected during: JUMP SCRIPTs Test
Address =________, Expected =__, Actual =__

Jump if “True”, and Compare = True; Jump not taken

Jump if “True”, and Compare = False; Jump taken

Jump if “False”, and Compare = True; Jump taken

Jump if “True”, and Compare = False; Jump not taken

Test Timeout during: Memory Move SCRIPTs Test
Address =________, Expected =__, Actual =__

“SIR” not detected during: Memory Move SCRIPTs Test
Address =________, Expected =__, Actual =__
```

SFIFO - SCSI FIFO

This procedure tests the basic ability to write data into the SCSI FIFO and retrieve it in the same order as written. The test runs as follows:

1. The SCSI FIFO is checked for an empty condition following a software reset, then the SFWR bit is set and verified.
2. The FIFO is filled with 8 bytes of data verifying the byte count with each write.
3. The SFWR bit is cleared and the FIFO read verifying the byte count with each read.
4. If no errors are detected, the NCR device is reset, otherwise the device is left in the test state.

Command Input

```
162-Diag>NCR SFIFO
```

Messages

If any part of the test fails, one of the following messages appears:

```
SCSI FIFO is not initially empty

SCSI FIFO writes not enabled

SCSI FIFO Count Error:
Address =________, Expected =__, Actual =__

SCSI FIFO Error:
Address =________, Expected =__, Actual =__
```

IPIC - IndustryPack Interface Controller

The **IPIC** tests check the IndustryPack Interface Controller. The **IPIC** tests are listed in Table 3-11, and are described in alphabetical order on the following pages.

Enter **IPIC** to run all tests in the group (except **INRPT**). They will be executed in the order shown in Table 3-11.

Table 3-11. IPIC Tests

Test Name	Description
ACCESSA	Device Access
ACCESSB	Register Access
INRPT	Interrupt Control

The error messages are described in *IPIC Error Messages* on page 3-187.

Configuration Parameter

If more than one IPIC is present, use the **CF** command to select the base address of the IPIC (`IPIC Base Address` parameter) that you want to test. The default is $FFFBC000.

ACCESSA - Read Internal Registers

This test verifies that all of the IPIC ASICs internal registers can be read. It does so by reading the device on byte, word, and long word boundaries.

The data returned by these reads is ignored. The test passes if the entire address space occupied by the chip is successfully read. Regardless of the outcome of the testing, the original configuration is maintained afterward.

Command Input

```
162-Diag>IPIC ACCESSA
```

Messages

See *IPIC Error Messages* on page 3-187 for a list of the error messages.

ACCESSB - Write to Internal Registers

This test verifies that internal registers of the IPIC ASIC can be written to and read from. It does so by executing "walking bit" tests on all IPIC memory base address registers and memory size registers. For this, test the register spaces are accessed on long word (32-bit) boundaries. This test runs as follows:

1. The contents of the registers are saved.
2. Zeroes are written to the registers and read back for verification.
3. Next, a one bit is walked through the field of zeroes.
4. Ones are written to the registers and read back for verification.
5. A zero bit is walked through the field of ones.

The test passes if all data patterns written are successfully read. Regardless of the outcome of the testing, an attempt is made to restore the original configuration afterward.

Command Input

```
162-Diag>IPIC ACCESSB
```

Messages

See *IPIC Error Messages* on page 3-187 for a list of the error messages.

INRPT - Interrupt Control Registers

This test verifies that the bits in the IPIC Interrupt Control Registers are functioning normally. It does so by configuring each register for:

- Level zero
- Interrupts enabled
- Toggling the polarity bit

This sets the INT (interrupt) status bit without generating an interrupt. The same steps are repeated for each interrupt level up to and including level seven, expecting and servicing interrupts for every level above zero.

The test passes if this entire sequence is successful for all eight IPIC interrupt control registers. After testing, IPIC registers are returned to their original configuration.

Note Some IndustryPacks may respond to interrupts generated by the **INRPT** test. Therefore, do not run this test with IndustryPacks installed.

Command Input

```
162-Diag>IPIC INRPT
```

Messages

See *IPIC Error Messages* on page 3-187 for a list of the error messages.

IPIC Error Messages

The following table lists the IPIC test group error messages:

Table 3-12. IPIC Error Messages

Message	Cause
`Register did not clear, address a, expected e, read r`	Read data is not zero, when zero was written.
`Register access error, address a, expected e, read r`	Read data differs from that written.
`Interrupt Control Register did not clear`	Test was unable to write zero to the IL2-IL0 (interrupt level) bits or the IEN (interrupt enable) bit in an IPIC Interrupt Control Register. Or INT (interrupt) did not clear when the ICLR (interrupt clear) bit was set.
`E/L bit did not set`	The test was unable to set (write one) the E/L* (edge/level sensitive) bit in an IPIC Interrupt Control Register.
`Interrupt Enable bit did not set`	The test was unable to set the IEN (interrupt enable) bit in an IPIC Interrupt Control Register.
`Interrupt Status bit did not set`	The INT bit in an IPIC Interrupt Control Register did not set as is the expected result of toggling the PLTY (polarity) bit in the same register.
`Unexpected Vector taken`	An exception occurred with unexpected vector.
`Incorrect Interrupt Level`	Interrupt of unexpected level.
`Interrupt did not occur`	The test gave up waiting for an expected interrupt to occur.
`Interrupt Status bit did not clear`	INT (interrupt) bit in an Interrupt Control Register did not clear when the ICLR (interrupt clear) bit was set.

Table 3-12. IPIC Error Messages

Message	Cause
`Status: Expected =e, Actual =r`	The expected and actual contents of the Interrupt Control Register under test after certain failures occur.
`Vector: Expected =e, Actual =r`	The expected and actual interrupt vector after certain failures of the **IPIC INRPT** test.
`State: IRQ Level =r,`	The level of an interrupt request taken after certain failures of the **IPIC INRPT** test.
`Level: Expected =e, Actual =r`	The expected and actual interrupt level after certain failures of the **IPIC INRPT** test.
`Testing Register Address = a`	The address of the IPIC register being tested when a failure occurred.

SCC - Z85230 Serial Communication Controller

The **SCC** tests check the Z85230 Serial Communication Controller. The **SCC** tests are listed in Table 3-13, and are described in alphabetical order on the following pages.

Enter **SCC** to run the **ACCESS** and **IRQ** tests (these are the only tests that run as part of the test group).

Table 3-13. SCC Tests

Test	Description
ACCESS	Device/Register Access
IRQ	Interrupt Request
BAUDS	Baud Rates
ELPBCK	External Loopback
ILPBCK	Internal Loopback
MDMC	Modem Control

The error messages are described in *SCC Error Messages* on page 3-197.

Configuration Parameters

You may change the following parameters with the **CF** command (the default values are given):

```
SCC Memory Space Base Address          =FFF45000?
```

The is the base address space for the Z85C230 devices. This value is preset and should not be changed.

```
Internal-Loopback/Baud-Rates Port Mask =0000000E? 0A
```

```
External-Loopback/Modem-Control Port Mask=0000000E?
```

The two port selection mask parameters identify which ports are to be tested. The default is to test every port except the console port. The **Internal-Loopback/Baud-Rates Port Mask** is

used for the **BAUDS** and **ILPBCK** test suites. The **External-Loopback/Modem-Control Port Mask** is only used for the **ELPBCK** and **MDMC** test suites.

The mask is a hex value that represents a bit mask. Set bits 0 through 3 (big endian) to select ports 0 through 3 respectively. For example, $02 (0010) selects port 1, $0B (1011) selects ports 0, 1 and 3, and $0F (1111) selects all four ports. $0 (no ports) is not a valid selection.

Note The Z85C230 ports are numbered 0 through 3. The first Z85C230 channel 0 as port 0, the second Z85C230 channel 0 as port 1. MVME162FX-0*xx* and MVME162FX-5*xx* boards have two ports, 0 and 1. MVME162FX -2*xx* boards have four ports: 0, 1, 2 and 3.

ACCESS - SCC Device/Register Access

This test performs a write/read test on two registers in the Z85C230. This test verifies that the device can be both accessed and that the data paths to the device are functioning.

Command Input

```
162-Diag>SCC ACCESS
```

Messages

See *SCC Error Messages* on page 3-197 for a list of the error messages.

IRQ - SCC Interrupt Request

This test verifies that the Z85C230 can generate interrupts to the local processor. This is done using the baud rate zero counter interrupt from the Z85C230.

Command Input

```
162-Diag>SCC IRQ
```

Messages

See *SCC Error Messages* on page 3-197 for a list of the error messages.

BAUDS - SCC Baud Rates

This test transmits 256 characters at various baud rates. The data is received and compared. If any protocol errors are created or the data is not correct when received, the test failed. The bauds tested are:

- 1200
- 2400
- 4800
- 9600
- 19200
- 38,400.

Note Because of the design of the Z85C230, when internal loopback testing is performed, data is still transmitted out of the device on the TXD line. This may cause problems with terminals, modem, printers, and any other device attached.

Command Input

```
162-Diag>SCC BAUDS
```

Messages

See *SCC Error Messages* on page 3-197 for a list of the error messages.

ELPBCK - SCC External Loopback

This test transmits 256 characters at 38,400 baud. The data is received and compared. If any protocol errors are created or the data is not correct when received, the test failed.

Note This test requires an external loopback connector to be installed. For this test TXD and RXD need to be connected in the loopback connector.

Command Input

```
162-Diag>SCC ELPBCK
```

Messages

See *SCC Error Messages* on page 3-197 for a list of the error messages.

ILPBCK - SCC Internal Loopback

This test transmits 256 characters at 38,400 baud. The data is received and compared. If any protocol errors are created or the data is not correct when received, the test failed.

Note Because of the design of the Z85C230, when internal loopback testing is performed, data is still transmitted out of the device on the TXD line. This may cause problems with terminals, modem, printers, and any other device attached.

Command Input

```
162-Diag>SCC ILPBCK
```

Messages

See *SCC Error Messages* on page 3-197 for a list of the error messages.

MDMC - SCC Modem Control

This test verifies that the Z85C230 can negate and assert selected modem control lines and that the appropriate input control functions.

Note This test requires an external loopback connector to be installed. For this test, the DTR must be connected to DCD, and the RTS must be connected to CTS, in the loopback connector.

Command Input

```
162-Diag>SCC MDMC
```

Messages

See *SCC Error Messages* on page 3-197 for a list of the error messages.

SCC Error Messages

The following table lists the SCC test group error messages:

Table 3-14. SCC Error Messages

Message	Cause
`Exception, Vector __`	An unexpected exception occurred
`Data Miscompare Error:` `Address =________, Register Index =__` `Expected =__, Actual =__`	Data write does not match data read.
`Exception Vector Serviced Error:` `Expected =___, Actual =___` `Interrupt Level =_` `SCC Base Address =________, Channel =__`	Incorrect vector taken or provided during interrupt service
`Exception failed to occur, Vector Expected =___` `Interrupt Level =_` `SCC Base Address =________, Channel =__`	During Interrupt testing, no interrupt was generated or received.
`Interrupt Not (Stuck-At) Error:` `Vector =___, Interrupt Level =_` `SCC Base Address =________, Channel =__`	A preexisting interrupt could not be cleared.
`SCC Receiver Error: Status =___` `SCC Base Address =________, Channel =__` `Baud Rate =____` *<Additional error information>*	Data transmission error: possibly framing, parity, or data overrun
`SCC Receiver Error: Status =__` `Break Sequence detected in the RXD stream` `SCC Base Address =________, Channel =__` `Baud Rate =____`	An unexpected break was received during testing.
`Transmit/Receive Character Miscompare Error:` `Expected =__, Actual =__` `SCC Base Address =________, Channel =__` `Baud Rate =____`	The data transmitted does not match data received.
`Transmitter Ready Time-Out` `SCC Base Address =________, Channel =__` `Baud Rate =____`	The selected ports transmitter never indicated ready to transmit.

Table 3-14. SCC Error Messages (Continued)

Message	Cause
Receiver Ready (Character Available) Time-Out SCC Base Address =_____, Channel =__ Baud Rate =____	The receiver has not received a character in the allotted time.
DTR assertion failed to assert DCD SCC Base Address =_____, Channel =__	When DTR was driven, DCD did not follow.
DTR negation failed to negate DCD SCC Base Address =_____, Channel =__	
RTS assertion failed to assert CTS SCC Base Address =_____, Channel =__	When RTS was driven, CTS did not follow.
RTS negation failed to negate CTS SCC Base Address =_____, Channel =__	

FLASH - FLASH Memory Tests

The **FLASH** tests check the Intel 28f008sa FLASHFILE™ FLASH memory devices. The **FLASH** tests must be called individually (you cannot run them as a group) and can be executed only when the Bug resides in PROM.

Note Running a **FLASH** test may be destructive to data stored in the FLASH array. The **FLASH** tests will fail if the Bug is running in FLASH memory.

FLASH memory has a finite life expectancy based on a maximum number of erase cycles. Execution of the FLASH memory tests will perform several erase cycles.

The tests are listed in Table 3-15, and are described in alphabetical order on the following pages.

Table 3-15. FLASH Tests

Test	Description
ERASE	Erase
FILL	Fill
PATS	Patterns

The error messages are listed in *FLASH Test Group Error Messages* on page 3-204.

Configuration Parameters

You may set the following parameters with the **CF** command (the default values are shown):

```
Flash Device Test Mask =0000000F ?
```

The mask is a hex value that represents a bit mask. Set bits 0 through 3 (big endian) to select ports 0 through 3 respectively. For example, $02 (0010) selects port 1, $0B (1011) selects ports 0, 1 and 3, and $0F (1111) selects all four ports. $0 (no ports) is not a valid selection.

```
Flash Test Starting Block =00000000 ?
```

The test range starting block, $0 through $F

```
Flash Test Ending Block =0000000F ?
```

The test range ending block, $0 through $F

```
Save/Restore For PATS Test [Y?N] =Y ?
```

Save the contents of the selected FLASH memory during the patterns test and restore the original data when the test is complete. Not saving and restoring will be equivalent to erasing the FLASH device once the patterns test is complete.

```
Fill Data =000000FF ?
```

The fill pattern, any byte $00 through $FF (used by the **FILL** test).

```
Test Data Increment/Decrement Step =00000001?
```

The value added to the Fill Data at each step (used by the **FILL** test).

ERASE - Erase FLASH Memory

The **ERASE** test erases FLASH memory. This test operates on a single block at a time within a device. Each block is erased and verified. This test does not preserve the contents of the FLASH under test.

Command Input

```
162-Diag>FLASH ERASE
```

Messages

Refer to *FLASH Test Group Error Messages* on page 3-204 for a list of the error messages.

FILL - Fill FLASH Memory

The **FILL** test fills FLASH memory. This test operates on each individual block at a time, within each device. Each block is filled with data, and verified.

This test uses the Fill Data and Test Data Increment/Decrement Step configuration parameters.

Note This test does not preserve the contents of the FLASH under test.

Command Input

```
162-Diag>FLASH FILL
```

Messages

Refer to *FLASH Test Group Error Messages* on page 3-204 for a list of the error messages.

PATS - FLASH Patterns

The **PATS** test writes and reads various data patterns in FLASH memory. This test operates on each individual block at a time, within each device. Each block is filled with patterns and verified. Four patterns are used: $FF, $AA, $55, and $00.

Command Input

```
162-Diag>FLASH PATS
```

Messages

Refer to *FLASH Test Group Error Messages* on page 3-204 for a list of the error messages.

FLASH Test Group Error Messages

The following error messages apply to the **FLASH** tests:

Table 3-16. FLASH Error Messages

Message	Cause
`Flash Memory Device Identifier Codes:` `Manufacturer Code =__ Device Code =__`	The ID codes returned by the device under test.
`Bad Status From Flash Test:` `Device = __ Block =__ Data Pattern =__` `Flash Test Control Status = ________`	The device number, block number, data pattern, and the contents of the control status word in the **FLASH** memory test control packet at the time of a failure.
`Flash Memory Erase Test,` `Error Mapping Starting Address:` `Invalid Address =________`	The test was unable to control the FLASH memory mapping on the board.
`Flash Memory Fill Test,` `Timeout Erasing:` `Address =________`	An erase command to a FLASH memory device failed to complete in the time allowed (**FILL** and **PATS** tests only).
`Flash Memory Erase Test,` `Address Error:` `Address =________`	A function called by the test returned the address range bit set in the control status word of the FLASH memory test control packet.
`Flash Memory Erase Test,` `Error Erasing:` `Address =________`	An erase command to a FLASH memory device failed.
`Flash Memory Erase Test,` `Timeout Writing:` `Address =________`	A write command to a FLASH memory device failed.
`Flash Memory Erase Test, Vpp Error:` `Address =________`	A Vpp error bit was set in the status returned by a FLASH memory device during the test.
`Flash Memory Erase Test, Write Error:` `Address =________`	A write error bit was set in the status returned by a FLASH memory device during the test.
`Flash Memory Patterns Test,` `Verify Error:` `Address =________`	A function called by the test returned the verify bit set in the control status word of the FLASH memory test control packet (**PATS** test only).

Table 3-16. FLASH Error Messages

Message	Cause
`Data Miscompare Error:` `Address =________` `Expected =__ Actual = __`	The data read from a FLASH memory device failed to match the expected data.
`Flash Memory Erase Test,` `Error Saving Flash Contents` `Address =________ Device =__ Block =__`	An error occurred while trying to save the contents of a FLASH device (**PATS** test only).
`Flash Memory Erase Test, Error:` `Not Enough Space Available To Save` `Flash Contents`	The board does not have enough RAM available to save the contents of the FLASH memory during testing (**PATS** test only).
`Flash Memory Erase Test,` `Error While Restoring Flash Contents` `Address =________ Device =__ Block =__`	An error occurred while trying to restore the contents of a block in a FLASH memory device (**PATS** test only).

162Bug Environment 4

Introduction

The parameters that affect board and 162Bug operation are stored in the NVRAM. The board information block operating parameters can be changed with the 162Bug command **CNFG**. 162Bug parameters can be changed with the **ENV** debugger command.

The **CNFG** and **ENV** commands are described in the *Debugging Package for Motorola 68K CISC CPUs User's Manual*. Refer to that manual for general information about their use and capabilities. The following section contain additional information about **CNFG** and **ENV** that is specific to the 162Bug.

CNFG - Configure Board Information Block

The **CNFG** command allows you to view and configure the board information block, which is resident within the NVRAM. The factory fills all fields except the IndustryPack fields.

The board information block parameters are:

```
Board (PWA) Serial Number = "000000061050"
Board Identifier          = "MVME162-513A    "
Artwork (PWA) Identifier  = "01-W3960B01A    "
MPU Clock Speed           = "3200"
Ethernet Address          = 08003E20A867
Local SCSI Identifier     = "07"
Parity Memory Mezzanine Artwork (PWA) Identifier  = "        "
Parity Memory Mezzanine (PWA) Serial Number       = "        "
Static Memory Mezzanine Artwork (PWA) Identifier  = "        "
Static Memory Mezzanine (PWA) Serial Number       = "        "
ECC Memory Mezzanine #1 Artwork (PWA) Identifier  = "        "
ECC Memory Mezzanine #1 (PWA) Serial Number       = "        "
ECC Memory Mezzanine #2 Artwork (PWA) Identifier  = "        "
ECC Memory Mezzanine #2 (PWA) Serial Number       = "        "
Serial Port 2 Personality Artwork (PWA) Identifier   = "        "
Serial Port 2 Personality Module (PWA) Serial Number = "        "
```

```
IndustryPack A Board Identifier         = "        "
IndustryPack A (PWA) Serial Number      = "        "
IndustryPack A Artwork (PWA) Identifier = "        "
IndustryPack B Board Identifier         = "        "
IndustryPack B (PWA) Serial Number      = "        "
IndustryPack B Artwork (PWA) Identifier = "        "
IndustryPack C Board Identifier         = "        "
IndustryPack C (PWA) Serial Number      = "        "
IndustryPack C Artwork (PWA) Identifier = "        "
IndustryPack D Board Identifier         = "        "
IndustryPack D (PWA) Serial Number      = "        "
IndustryPack D Artwork (PWA) Identifier = "        "
```

The parameters in quote marks (“) are left-justified character (ASCII) strings padded with space characters. The quote marks indicate the size of the string. The other parameters are right-justified data strings. The data strings are padded with zeroes.

Refer to your MVME162 installation manual for information about the board information block. Refer to the *Debugging Package for Motorola 68K CISC CPUs User’s Manual* for a description of **CNFG** and examples.

ENV - Set Environment

The **ENV** command allows you to interactively view and configure all 162Bug operational parameters. Refer to the *Debugging Package for Motorola 68K CISC CPUs User's Manual* for a description of the use of **ENV**.

Configuring 162Bug Parameters

The following **ENV** parameters control initialization, booting, and other functions of the debugger firmware:

```
Bug or System environment [B/S] = B?
```

- **B** Do not run self test diagnostics during system start-up. Display the `162-Bug>` prompt.
- **S** Run self test diagnostics during system start-up. Display the `162-Diag>` prompt.

The debugger or diagnostics prompt appears after start-up if no boot mechanism (Auto Boot, ROM Boot, Network Auto Boot) is enabled, if the user aborts the start-up, or if the start-up fails. If the Field Service Menu is enabled, it appears in place of the prompt.

```
Field Service Menu Enable [Y/N] = N?
```

- **Y** Display the Field Service system menu in place of the prompt. The Field Service menu is described in Appendix A of the *Debugging Package for Motorola 68K CISC CPUs User's Manual*.
- **N** Do not display the Field Service Menu.

```
Remote Start Method Switch [G/M/B/N] = B?
```

The method for executing a cross-loaded program when the MVME162 is cross-loaded from another VME-based CPU.

- **G** Use the Global Control and Status Register (GCSR) in the VMEchip2 on to pass and start execution of cross-loaded program.
- **M** Use the Multiprocessor Control Register (MPCR) in shared RAM to pass and start execution of cross-loaded program.

B Use both the GCSR and the MPCR to pass and start execution of cross-loaded program.

N Do not use any remote start method.

```
Probe System for Supported I/O Controllers [Y/N] = Y?
```

Y Access the appropriate system buses (e.g., VMEbus, local MPU bus) to determine the presence of supported controllers.

N Do not access the VMEbus to determine the presence of supported controllers.

```
Negate VMEbus SYSFAIL* Always [Y/N] = N?
```

Y Negate VMEbus SYSFAIL during board initialization.

N Negate VMEbus SYSFAIL after successful completion or entrance into 162Bug.

```
Local SCSI Bus Reset on Debugger Setup [Y/N] = N?
```

Y Reset the Local SCSI bus on debugger set-up.

N Do not reset the Local SCSI bus on debugger set-up.

```
Local SCSI Bus Negotiations Type [A/S/N] = A?
```

A Asynchronous SCSI bus negotiation

S Synchronous SCSI bus negotiation

N No negotiations

```
Industry Pack Reset on Debugger Startup [Y/N] = N
```

Y Reset Industry Packs on debugger start-up.

N Do not reset Industry Packs on debugger start-up.

```
Ignore CFGA Block on a Hard Disk Boot [Y/N] = Y
```

Y Ignore the Configuration Area (CFGA) Block (hard disk only).

N Do not ignore the Configuration Area (CFGA) Block (hard disk only).

```
Auto Boot Enable [Y/N]   = N?
```

Y Enable Auto Boot

N Disable Auto Boot

```
Auto Boot at power-up only [Y/N] = Y?
```

Y Run Auto Boot at power-up only (the prompt or the Field Service Menu appears after a warm start).

N Run Auto Boot at both warm and cold start.

```
Auto Boot Controller LUN = 00?
```

The boot controller Logical Unit Number. Refer to Appendix E in the *Debugging Package for Motorola 68K CISC CPUs User's Manual* for a listing of disk/tape controller modules supported by the Bug.

```
Auto Boot Device LUN     = 00?
```

The boot device Logical Unit Number. Refer to Appendix E in the *Debugging Package for Motorola 68K CISC CPUs User's Manual* for a listing of disk/tape devices supported by the Bug.

```
Auto Boot Abort Delay    = 15?
```

The time in seconds that the start-up sequence waits before starting Auto Boot. During the delay a user may exit to the debugger or diagnostics prompt by pressing the BREAK key. The value may be from 0-255.

```
Auto Boot Default String [Y(NULL String)/(String)] = ?
```

A string (filename) which is passed on to the code being booted. The maximum length of this string is 16 characters.

```
ROM Boot Enable [Y/N] = N?
```

Y Enable ROM Boot

N Disable ROM Boot

```
ROM Boot at power-up only [Y/N] = Y?
```

Y Run ROM Boot at power-up only (the prompt or the Field Service Menu appears after a warm start).

N Run ROM Boot at both warm and cold start.


```
ROM Boot Enable search of VMEbus [Y/N] = N?
```

Y Search the VMEbus address space for a ROM Boot module in addition to the normal areas of memory.

N VMEbus address space will not be accessed by ROM Boot.

```
ROM Boot Abort Delay    = 0?
```

The time in seconds that the start-up sequence waits before starting ROM Boot. During the delay a user may exit to the debugger or diagnostics prompt by pressing the BREAK key. The value may be from 0-255.

```
ROM Boot Direct Starting Address = FF800000?
```

The first location tested when the firmware searches for a ROM Boot module

```
ROM Boot Direct Ending Address   = FFDFFFFC?
```

The last location tested when the firmware searches for a ROM Boot module

```
Network Auto Boot Enable [Y/N]   = N?
```

Y Enable Network Auto Boot

N Disable Network Auto Boot

```
Network Auto Boot at power-up only [Y/N] = Y?
```

Y Run Network Auto Boot at power-up only (the prompt or the Field Service Menu appears after a warm start).

N Run Network Auto Boot at both warm and cold start.

```
Network Auto Boot Controller LUN = 00?
```

The boot controller Logical Unit Number. Refer to Appendix G in the *Debugging Package for Motorola 68K CISC CPUs User's Manual* for a listing of disk/tape controller modules supported by the Bug.

```
Network Auto Boot Device LUN     = 00?
```

The boot device Logical Unit Number. Refer to Appendix G in the *Debugging Package for Motorola 68K CISC CPUs User's Manual* for a listing of disk/tape controller modules supported by the Bug.

```
Network Auto Boot Abort Delay    = 5?
```

The time in seconds that the Network Auto Boot sequence waits before starting the boot. During the delay a user may exit to the debugger or diagnostics prompt by pressing the BREAK key. The value is from 0-255.

```
Network Auto Boot Configuration Parameters Pointer (NVRAM) = 00000000?
```

The address where the network interface configuration parameters are to be saved/retained in NVRAM. These parameters are the parameters necessary to perform an unattended network boot.

```
Memory Search Starting Address   = 00000000?
```

The location where the Bug begins to search for a work page (a 64KB block of memory) to use for vector table, stack, and variables. This must be a multiple of (modulo) the debugger work page size.

In a multi-MVME162 environment, each MVME162 board could be set to start its work page at a unique address so as to allow multiple debuggers to operate simultaneously.

```
Memory Search Ending Address     = 08000000?
```

The top limit of 162Bug's search for a work page. If a contiguous block of memory, 64KB in size, is not found in the range specified by the Memory Search Starting Address and the Memory Search Ending Address parameters, the bug will place its work page in the onboard static RAM on the MVME162.

```
Memory Search Increment Size     = 00010000?
```

The offset to the location of the Bug work page for multi-CPU use. This must be a multiple of (modulo) the debugger work page size ($10000 or 64KB).

Typically, the Memory Search Increment Size is the product of the CPU number and size of the Bug work page. For example, the Memory Search Increment Size for the first CPU would be $0 (0 x $10000), and the second CPU would be $10000 (1 x $10000).


```
Memory Search Delay Enable [Y/N] = N?
```

Y Cause a delay before the Bug begins its search for a work page. The delay could be used to allow time for some other MVME162 in the system to configure its address decoders.

N No delay before the Bug begins its search for a work page.

```
Memory Search Delay Address      = FFFFD20F?
```

The MVME162 GCSR GPCSR0 as accessed through VMEbus A16 space and assumes the MVME162 GRPAD (group address) and BDAD (board address within group) switches are set to “on”. This byte-wide value is initialized to $FF by MVME162 hardware after a System or Power-on Reset.

In a multi-MVME162 environment, the work pages for several CPU boards may be located in memory on the primary (first) MVME162. To accomplish this, the non-primary CPUs must wait for the primary CPU to initialize itself and change the data at the Memory Search Delay Address from the $FF to $00, $01, or $02. By doing this, the primary CPU by indicates to any other CPUs that they may locate their work page in the primary's memory. The *Memory Requirements section* in Chapter 1 defines the minimum memory needed by each MVME162 CPU to operate.

```
Memory Size Enable [Y/N]         = Y?
```

Y Memory is sized for Self Test diagnostics.

N Memory is sized for Self Test diagnostics.

```
Memory Size Starting Address     = 00000000?
```

The Starting Address for memory sizing

```
Memory Size Ending Address       = 08000000?
```

The Ending Address for memory sizing. This is the calculated size of local memory. If the memory start is changed from $0, this parameter would also need to be adjusted.

Note Memory Configuration Defaults:
The default configuration for Dynamic RAM mezzanine boards will position the mezzanine with the largest memory size to start at the address selected with the **ENV** parameter `Base Address of Dynamic Memory`. The Base Address parameter defaults to 0. The smaller sized mezzanine will follow immediately above the larger in the memory map. If mezzanines of the same size and type are present, the first (closest to the board) is mapped to the selected base address. If mezzanines of same size but with different type (parity and ECC) are present, the parity type will be mapped to the selected base address and the ECC type mezzanine will follow. The SRAM does not default to a location in the memory map that is contiguous with Dynamic RAM.

```
Base Address of Dynamic Memory     = 00000000?
```

The beginning address of Dynamic Memory. It must be a multiple of the Dynamic Memory board size, starting with 0. The Bug will set up the hardware address decoders so that the Dynamic Memory resides as one contiguous block at this address.

```
Size of Parity Memory            = 00800000?
```

The size of the parity type dynamic RAM mezzanine, if any. The default is the calculated size of the dynamic memory mezzanine board.

```
Size of ECC Memory Board #0    = 00000000?
```

The size of the first ECC type memory mezzanine. The default is the calculated size of the memory mezzanine.

```
Size of ECC Memory Board #1    = 00000000?
```

The size of the second ECC type memory mezzanine. The default is the calculated size of the memory mezzanine.

```
Base Address of Static Memory    = FFE00000?
```

The beginning address of SRAM. The default for this parameter is FFE00000 for the onboard SRAM (128K on the MVME162-2*xx*, 512K on the MVME162-0*xx* and MVME162-5*xx*), or E1000000 for the 2MB SRAM mezzanine. If only 2MB SRAM is present, it defaults to address 00000000.

```
Size of Static Memory            = 00020000?
```

The size of the SRAM type memory present. The default is the calculated size of the onboard SRAM or an SRAM type mezzanine.

VMEbus Interface Parameters

ENV displays the following prompts to set up the VMEbus interface for the MVME162 modules. Refer to the VMEbus specification and the VMEchip2 information in the *MVME162/MVME162FX/MVME162LX Embedded Controller Programmer's Reference Guide* for configuring these parameters.

The slave address decoders are used to allow another VMEbus master to access a local resource of the MVME162. There are two slave address decoders set.

```
Slave Enable #1 [Y/N] = Y?
```

Y Enable the Slave Address Decoder #1

N Do not enable the Slave Address Decoder #1

```
Slave Starting Address #1 = 00000000?
```

The base address of the local resource that is accessible by the VMEbus (the default $0 is the base of local memory).

```
Slave Ending Address #1   = 07FFFFFF?
```

The ending address of the local resource that is accessible by the VMEbus (the default is the end of calculated memory).

```
Slave Address Translation Address #1 = 00000000?
```

The base address of local resource that is associated with the starting and ending addresses. This allows the VMEbus address and the local address to be different.

```
Slave Address Translation Select #1 = 00000000?
```

A mask that defines which bits of the address are significant. A 1 indicates a significant bit. A 0 indicates a nonsignificant bit.

```
Slave Control #1 = 03FF?
```

The access restriction for the address space defined with this slave address decoder.

```
Slave Enable #2 [Y/N] = N?
```

Y Enable the Slave Address Decoder #2

N Do not enable the Slave Address Decoder #2

```
Slave Starting Address #2 = 00000000?
```

The base address of the local resource that is accessible by the VMEbus.

```
Slave Ending Address #2   = 00000000?
```

The ending address of the local resource that is accessible by the VMEbus.

```
Slave Address Translation Address #2 = 00000000?
```

The base address of local resource that is associated with the slave starting and ending addresses. This allows the VMEbus address and the local address to be different.

```
Slave Address Translation Select #2 = 00000000?
```

A mask that defines which bits of the address are significant. A 1 indicates a significant bit. A 0 indicates a nonsignificant bit.

```
Slave Control #2 = 0000?
```

The access restriction for the address space defined with this slave address decoder.

```
Master Enable #1 [Y/N] = Y?
```

Y Enable the Master Address Decoder #1

N Do not enable the Master Address Decoder #1

```
Master Starting Address #1 = 01000000
```

The base address of the VMEbus resource that is accessible from the local bus. The default is the end of calculated local memory.

```
Master Ending Address #1   = EFFFFFFF?
```

The ending address of the VMEbus resource that is accessible from the local bus (the default is the end of calculated memory)

```
Master Control #1 = 0D?
```

The access characteristics for the address space defined with this master address decoder

```
Master Enable #2 [Y/N] = N?
```

N Enable the Master Address Decoder #2

Y Do not enable the Master Address Decoder #2

```
Master Starting Address #2 = 00000000?
```

The base address of the VMEbus resource that is accessible from the local bus (if enabled, the default is $FF000000, otherwise $00000000).

```
Master Ending Address #2 = 00000000?
```

The ending address of the VMEbus resource that is accessible from the local bus (if enabled, the default is $FF7FFFFF, otherwise $00000000).

```
Master Control #2 = 00?
```

The access characteristics for the address space defined with this master address decoder (if enabled, the default is $0D, otherwise $00).

```
Master Enable #3 [Y/N] = Y?
```

Y Enable the Master Address Decoder #3. Set this to Y if the board contains less than 16MB of calculated RAM.

N Do not enable the Master Address Decoder #3. Set this to N if the default if the board contains at least 16MB of calculated RAM.

```
Master Starting Address #3 = 00800000?
```

The base address of the VMEbus resource that is accessible from the local bus (if enabled, the value is calculated as one less than the calculated size of memory; if not enabled, the default is $00000000)

```
Master Ending Address #3 = 00FFFFFF?
```

The ending address of the VMEbus resource that is accessible from the local bus (if enabled, the default is $00FFFFFF, otherwise $00000000)

```
Master Control #3 = 3D?
```

The access characteristics for the address space defined with this master address decoder (if enabled, the default is $3D, otherwise $00)

```
Master Enable #4 [Y/N] = N?
```

Y Enable the Master Address Decoder #4

N Do not enable the Master Address Decoder #4

```
Master Starting Address #4 = 00000000?
```

The base address of the VMEbus resource that is accessible from the local bus.

```
Master Ending Address #4   = 00000000?
```

The ending address of the VMEbus resource that is accessible from the local bus.

```
Master Address Translation Address #4 = 00000000?
```

The base address of VMEbus resource that is associated with the starting and ending addresses. This allows the VMEbus address and the local address to be different.

```
Master Address Translation Select #4 = 00000000?
```

A mask that defines which bits of the address are significant. A 1 indicates a significant bit. A 0 indicates a nonsignificant bit.

```
Master Control #4 = 00?
```

The access characteristics for the address space defined with this master address decoder.

```
Short I/O (VMEbus A16) Enable [Y/N] = Y?
```

Y Enable the Short I/O Address Decoder

N Do not enable the Master Address Decoder

```
Short I/O (VMEbus A16) Control      = 01?
```

The access characteristics for the address space defined with the Short I/O address decoder.

```
F-Page (VMEbus A24) Enable [Y/N]    = Y?
```

Y Enable the F-Page Address Decoder.

N Do not enable the F-Page Address Decoder.

```
F-Page (VMEbus A24) Control         = 02?
```

The access characteristics for the address space defined with the F-Page address decoder

```
ROM Access Time Code          = 04?
```

The ROM access time code

```
FLASH Access Time Code        = 03?
```

The FLASH access time code

```
MCC Vector Base               = 05?
VMEC2 Vector Base #1          = 06?
VMEC2 Vector Base #2          = 07?
```

The base interrupt vector for the component specified.

```
VMEC2 GCSR Group Base Address = D2?
```

The group address ($FFFFxx00) in Short I/O for the board.

```
VMEC2 GCSR Board Base Address = 00?
```

The base address ($FFFFCCx0) in Short I/O for the board.

```
VMEbus Global Time Out Code   = 01?
```

The VMEbus timeout when systems controller (the default $01 = 64 s)

```
Local Bus Time Out Code       = 02?
```

The local bus timeout

```
VMEbus Access Time Out Code   = 02?
```

The local bus to VMEbus access timeout (the default $02 = 32 ms)

Configuring IndustryPacks

The following **ENV** parameters control the IndustryPacks (IP) on MVME162 modules. The *MVME162/MVME162FX/MVME162LX Embedded Controller Programmer's Reference Guide* describes the base addresses and the IP register settings. Refer to that manual for information on setting base addresses and register bits. Each hexadecimal value represents a four-bit string, big endian.

```
IP A Base Address             = 00000000?
IP B Base Address             = 00000000?
IP C Base Address             = 00000000?
IP D Base Address             = 00000000?
```

These are the base addresses for mapping the IP modules. Only the upper 16 bits are significant.

```
IP D/C/B/A Memory Size        = 00000000?
```

This defines the memory size requirements for the IP modules:

Bits	IP	Register Address
31-24	D	FFFBC00F
23-16	C	FFFBC00E
15-08	B	FFFBC00D
07-00	A	FFFBC00C

```
IP D/C/B/A General Control     = 00000000?
```

This defines the general control requirements for the IP modules:

Bits	IP	Register Address
31-24	D	FFFBC01B
23-16	C	FFFBC01A
15-08	B	FFFBC019
07-00	A	FFFBC018

```
IP D/C/B/A Interrupt 0 Control = 00000000?
```

This defines the interrupt control requirements for the IP modules channel 0:

Bits	IP	Register Address
31-24	D	FFFBC016
23-16	C	FFFBC014
15-08	B	FFFBC012
07-00	A	FFFBC010

```
IP D/C/B/A Interrupt 1 Control = 00000000?
```

This defines the interrupt control requirements for the IP modules channel 1:

Bits	IP	Register Address
31-24	D	FFFBC017
23-16	C	FFFBC015
15-08	B	FFFBC013
07-00	A	FFFBC011

Saving ENV Parameter Settings

Before ENV parameters are saved in the NVRAM, a warning message will appear if the any of environment parameters overlap. Information about each configurable element in the memory map is displayed, showing where any overlaps exist. This will allow the user to quickly identify and correct an undesirable configuration before it is saved.

Below is an example of the error message:

```
WARNING: Memory MAP Overlap Condition Exists

S-Address  E-Address  Enable  Overlap  M-Type  Memory-MAP-Name
$00000000  $FFFFFFFF  Yes     Yes      Master  Local Memory (Dynamic RAM)
$FFE00000  $FFE7FFFF  Yes     Yes      Master  Static RAM
$01000000  $EFFFFFFF  Yes     Yes      Master  VMEbus Master #1
$00000000  $00000000  No      No       Master  VMEbus Master #2
$00000000  $00FFFFFF  Yes     Yes      Master  VMEbus Master #3
$00000000  $00000000  No      No       Master  VMEbus Master #4
$F0000000  $FF7FFFFF  Yes     Yes      Master  VMEbus F Pages (A24/A32)
$FFFF0000  $FFFFFFFF  Yes     Yes      Master  VMEbus Short I/O (A16)
$FF800000  $FFBFFFFF  Yes     Yes      Master  Flash/PROM
$FFF00000  $FFFEFFFF  Yes     Yes      Master  Local I/O
$00000000  $00000000  No      No       Master  Industry Pack A
$00000000  $00000000  No      No       Master  Industry Pack B
$00000000  $00000000  No      No       Master  Industry Pack C
$00000000  $00000000  No      No       Master  Industry Pack D
$00000000  $00000000  No      No       Slave   VMEbus Slave #1
$00000000  $00000000  No      No       Slave   VMEbus Slave #2
```

Related Documentation

The following publications are applicable to 162Bug and may provide additional helpful information. If not shipped with this product, they may be purchased by contacting your local Motorola sales office. Non-Motorola documents may be obtained from the sources listed.

Document Title	Motorola Publication Number
MVME162 Embedded Controller User's Manual	MVME162/D
MVME162 Embedded Controller Programmer's Reference Guide	MVME162PG/D
MVME162 Embedded Controller Installation Guide	MVME162IG/D
MVME162FX Embedded Controller Installation and Use	V162FXA/IH
MVME162FX Embedded Controller Programmer's Reference Guide	V162FXA/PG
MVME162LX Embedded Controller User's Manual	MVME162LX/D
MVME162LX Embedded Controller Installation Guide	MVME162LXIG/D
MVME162LX Embedded Controller Programmer's Reference Guide	MVME162LXPG/D
Single Board Computers SCSI Software User's Manual	SBCSCSI/D
Debugging Package for Motorola 68K CISC CPUs User's Manual	68KBUG1/D and 68KBUG2/D

Note Although not shown in the above list, each Motorola Computer Group manual publication number is suffixed with characters which represent the revision level of the document, such as “2” (the second revision

of a manual). A supplement bears the same number as a manual but has a suffix such as “2A1” (the first supplement to the second revision of the manual).

The following publications are available from the sources indicated.

ANSI Small Computer System Interface-2 (SCSI-2), Draft Document X3.131-198X, Revision 10c; Global Engineering Documents, P.O. Box 19539, Irvine, CA 92714.

Versatile Backplane Bus: VMEbus, ANSI/IEEE Std 1014-1987, The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017 (VMEbus Specification). This is also available as *Microprocessor system bus for 1 to 4 byte data, IEC 821 BUS*, Bureau Central de la Commission Electrotechnique Internationale; 3, rue de Varembé, Geneva, Switzerland.

Index

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

V

W

X

Z