Cover

MVME197BUG
Diagnostic Firmware
(Part 2 of 2)

34 pages
1/8″ spine

® ™

36 - 84 pages
3/16″ & 1/4″ spine

® ™

86 - 100 pages
5/16″ spine

® ™

102 - 180 pages
3/8″ - 1/2″ spine

® ™

182 - 308 pages
5/8″ - 1 1/8″ spine
2 lines allowed

MVME197BUG Diagnostic Firmware (Part 2 of 2)

MVME197BUG
Diagnostic Firmware
(Part 2 of 2)

V197DIAA2/UM1

Notice

Restricted Rights Legend

Motorola, Inc.
Computer Group
2900 South Diablo Way
Tempe, Arizona 85282-9602

Preface

This manual provides general information and a description of the diagnostic firmware for the MVME197BUG (197Bug) Debugging Package.

Use of the MVME197 debugger, the debugger command set, use of the one-line assembler/ disassembler, and system calls for the Debugging Package are all described in the *MVME197BUG (197Bug) Debugging Package User's Manual*.

Note

This document is bound in two parts:

Part 1 (V197DIAA1/UM1) contains Chapters 1 and 2, and the first half of Chapter 3 (pages 3-1 through 138, covering test suites RAM, RAMCDIx, RAMCD, ECDM, DCAM, BSW, XCAx, and MK48T0x).

Part 2 (*this volume*, *V197DIAA2/UM1*) contains the rest of Chapter 3 (pages 3-139 through 3-302, covering test suites PPC2, CD2401, VME2, LANC, and NCR), Chapter 4, and Appendix A.

The table of contents and index are duplicated in Parts 1 and 2.

This manual is intended for anyone who wants to design OEM systems, supply additional capability to an existing compatible system, or work in a lab environment for experimental purposes. A basic knowledge of computers and digital logic is assumed.

To use this manual, you should be familiar with the publications listed in the *Related Documentation* section found in the following pages.

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:

$	dollar	specifies a hexadecimal number
%	percent	specifies a binary number
&	ampersand	specifies a decimal number

For example, "12" is the decimal number twelve, and "$12" is the decimal number eighteen. 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 two-byte is 16 bits, numbered 0 through 15, with bit 0 being the least significant. For the MVME197 and other RISC modules, this is called a *half-word*.
- A four-byte is 32 bits, numbered 0 through 31, with bit 0 being the least significant. For the MVME197 and other RISC modules, this is called a *word*.
- An eight-byte is 64 bits, numbered 0 through 63, with bit 0 being the least significant. For the MVME197 and other RISC modules, this is called a *double-word*.

Throughout this document, it is assumed that the MPU on the MVME197/MVME297 module series is always programmed with *big-endian byte ordering*, as shown below. Any attempt to use *small-endian byte ordering* will immediately render the MVME197Bug debugger unusable.

BIT 63 ... 56	**55 ... 48**	**47 ... 40**	**39 ... 32 BIT**
ADR0	ADR1	ADR2	ADR3
31 ... 24	**23 ... 16**	**15 ... 08**	**07 ... 00**
ADR4	ADR5	ADR6	ADR7

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

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 Carriage Return or Enter key.

CTRL

represents the Control key. Execute control characters by holding down the control key while pressing the letter key, e.g., **CTRL-d**.

Related Documentation

The following publications are applicable to the MVME197 module series and may provide additional helpful information. If not shipped with this product, they may be purchased by contacting your Motorola sales office.

Document Title	Motorola Publication Number
MVME197LE Single Board Computer User's Manual	MVME197LE
MVME197LE Single Board Computer Support Information	SIMVME197LE
MVME197DP and MVME197SP Single Board Computers User's Manual	MVME197
MVME197DP and MVME197SP Single Board Computers Support Information	SIMVME197
MVME197LE, MVME197DP, and MVME197SP Single Board Computers Programmer's Reference Guide	MVME197PG
MVME197BUG 197Bug Debugging Package User's Manual	MVME197BUG
MVME712M Transition Module and P2 Adapter Board User's Manual	MVME712M
MVME712-12, MVME712-13, MVME712A, MVME712AM, and MVME712B Transition Module and LCP2 Adapter Board User's Manual	MVME712A
MC88110 Second Generation RISC Microprocessor User's Manual	MC88110UM
MC68040 Microprocessor User's Manual	MC68040UM
MC88410 Secondary Cache Controller User's Manual	MC88410UM

Notes

1. **The support information manuals (SIMVME197LE and SIMVME197) contain: the connector interconnect signal information, parts lists, and the schematics for the specific board indicated.**

2. **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 "/D2" (the second revision of a manual); a supplement bears the same number as the manual but has a suffix such as "/A1" (the first supplement to the manual).**

To further assist your development effort, Motorola has collected user's manuals for each of the peripheral controllers used on the MVME197 module series and other boards from the suppliers. This bundle includes manuals for the following:

68-1X7DS for use with the MVME197 series of Single Board Computers.

NCR 53C710 SCSI Controller Data Manual and Programmer's Guide
Intel i82596 Ethernet Controller User's Manual
Cirrus Logic CD2401 Serial Controller User's Manual
SGS-Thompson MK48T08 NVRAM/TOD Clock Data Sheet

The following non-Motorola publications may also be of interest and may be obtained from the sources indicated. The VMEbus Specification is contained in ANSI/IEEE Standard 1014-1987.

ANSI/IEEE Std 1014-1987 Versatile Backplane Bus: VMEbus	The Institute of Electrical and Electronics Engineers, Incorporated Publication and Sales Department 345 East 47th Street New York, New York 10017-2633 Telephone: 1-800-678-4333
ANSI Small Computer System Interface-2 (SCSI-2), Draft Document X3.131-198X, Revision 10c	Global Engineering Documents P.O. Box 19539 Irvine, California 92713-9539 Telephone (714) 979-8135

Printed in the United States of America

August 1995

Contents

CHAPTER 2 DIAGNOSTIC FIRMWARE

CHAPTER 4 197BUG GENERAL INFORMATION

APPENDIX A MVME197 FAMILY DRAM MEMORY LINE MAPPING

Figures

Tables

Peripheral Channel Controller (PCC2) Tests

These sections describe the individual PCCchip2 (**PCC2**) tests.

Entering **PCC2** without parameters causes all **PCC2** tests to execute in the order shown in the following table.

To run an individual test, add that test name to the **PCC2** command.

The individual tests are described in alphabetical order on the following pages. The error message displays following the explanation of a **PCC2** test pertain to the test being discussed.

Table 3-10. PCC2 Test Group

Mnemonic	Description
REGA	Device Access
REGB	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
TMR2C	Timer 2 Clear On Compare
TMR2D	Timer 2 Overflow Counter
TMR2E	Timer 2 Interrupts
ADJ	Prescaler Clock Adjust
PCLK	Prescaler Clock

3

Table 3-10. PCC2 Test Group (Continued)

Mnemonic	Description
GPIO	GPIO Interrupts
LANC	LANC Interrupts
PRNTA	Printer ACK Interrupts
PRNTB	Printer FAULT Interrupts
PRNTC	Printer SEL Interrupts
PRNTD	Printer PE Interrupts
PRNTE	Printer BUSY Interrupts
MIEN	MIEN Bit
FAST	FAST Bit
VBR	Vector Base Register

PCC2 Configuration Parameters

CF PCC2

Command Input

```
197-Diag>cf PCC2
```

Description

User configurable test parameters are available for the **PCC2** test group. Refer to Chapter 2 for information on using the **CF** command to set configuration parameters.

Note

There are no configuration parameters for the PCC2 test group.

Prescaler Clock Adjust - ADJ

PCC2 ADJ

Command Input

3

```
197-Diag>PCC2 ADJ
```

Description

Verifies that the Prescaler Clock Adjust Register can vary the period of the Tick Timer input clock. This is accomplished by setting the Clock Adjust Register to zero and allowing Tick Timer 1 to free-run for a small software delay, this will establish a reference count. Next a 1 is walked through the Clock Adjust Register and the timer is allowed to run for the same delay period, the resulting count should be greater than the last count.

Higher level software will always initialize the prescaler prior to calling the test, so a check of the register with a result of zero will be treated as a failure.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     ADJ: Prescaler Clock Adjust................. Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     ADJ: Prescaler Clock Adjust................. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     ADJ: Prescaler Clock Adjust................. Running ---> FAILED

PCC2/ADJ Test Failure Data:
(error message)
```

PCC2 ADJ

Here, (error message) is one of the following:

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

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

3

FAST Bit - FAST

PCC2 FAST

Command Input

```
197-Diag>PCC2 FAST
```

Description

Verifies the FAST/SLOW access time to BBRAM by using Tick Timer 1 to measure the time it takes to access BBRAM. To ensure a stable timer count the BBRAM is accessed 2K times.

1. Tick Timer 1 is initialized to zero and set for free-run.
2. “FAST” bit is set.
3. Timer is started.
4. BBRAM is accessed.
5. Timer is stopped and count is saved.
6. “FAST” bit is cleared.
7. Timer is initialized to zero.
8. Timer is started.
9. BBRAM is accessed.
10. Timer is stopped and count is saved.
11. Slow count is checked against fast count.
12. Error if fast count not less than slow count.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     FAST: ‘FAST’ bit........................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     FAST: ‘FAST’ bit........................... Running ---> PASSED
```

PCC2 FAST

If any part of the test fails, then the display appears as follows:

```
PCC2     FAST: 'FAST' bit............................ Running ---> FAILED

PCC2/FAST Test Failure Data:

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

3

GPIO Interrupts - GPIO

PCC2 GPIO

Command Input

```
197-Diag>PCC2 GPIO
```

Description

Verifies that level 0 interrupts will not generate an interrupt, but will set the appropriate status. Then verifies that all interrupts (1 through 7) can be generated and received and that the appropriate status is set.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     GPIO: GPIO Interrupts...................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     GPIO: GPIO Interrupts...................... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     GPIO: GPIO Interrupts...................... Running ---> FAILED

PCC2/GPIO Test Failure Data:
(error message)
```

PCC2 GPIO

Here, (error message) is one of the following:

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

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

PCC2 GPIO

```
Access Fault Information:
  Address________
  Data________
  Access Size___
  Access Type___
  Address Space Code___
  Vector Number___

Unsolicited Exception:
  Instruction Pointer________
  Vector Number___
  Status Register____
  Interrupt Level___
```

LANC Interrupts - LANC

PCC2 LANC

Command Input

```
197-Diag>PCC2 LANC
```

Description

Verifies that level 0 interrupts will not generate an interrupt, but will set the appropriate status. Then verifies that all interrupts (1 through 7) can be generated and received and that the appropriate status is set.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     LANC: LANC Interrupts...................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     LANC: LANC Interrupts...................... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     LANC: LANC Interrupts...................... Running ---> FAILED

PCC2/LANC Test Failure Data:
(error message)
```

PCC2 LANC

Here, (error message) is one of the following:

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

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

PCC2 LANC


```
Access Fault Information:
 Address________
 Data________
 Access Size___
 Access Type___
 Address Space Code___
 Vector Number___

Unsolicited Exception:
 Instruction Pointer________
 Vector Number___
 Status Register____
 Interrupt Level___
```

3

MIEN Bit - MIEN

PCC2 MIEN

Command Input

```
197-Diag>PCC2 MIEN
```

Description

Uses the General Purpose I/O Interrupt Control to generate and service a level 7 interrupt with the “MIEN” bit set. The bit is then cleared and a level 7 interrupt is generated and checked for interrupt not serviced.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     MIEN: ‘MIEN’ bit............................ Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     MIEN: ‘MIEN’ bit............................ Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     MIEN: ‘MIEN’ bit............................ Running ---> FAILED

PCC2/MIEN Test Failure Data:
(error message)
```

PCC2 MIEN

Here, (error message) is one of the following:

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

'MIEN' bit did not disable interrupts
Status: Expected =__, Actual =__
Vector: Expected =__, Actual =__
State : IRQ Level =_, VBR =__

Access Fault Information:
  Address________
  Data________
  Access Size___
  Access Type___

  Address Space Code___
  Vector Number___

Unsolicited Exception:
  Instruction Pointer________
  Vector Number___
  Status Register____
  Interrupt Level___
```

Prescaler Clock - PCLK

PCC2 PCLK

3

Command Input

```
197-Diag>PCC2 PCLK
```

Description

Verifies the accuracy of the Prescaler Clock, by using a constant time source and allowing Tick Timer 1 to free-run for one second and comparing the accumulated timer count with the expected count for the time period.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     PCLK: Prescaler Clock...................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     PCLK: Prescaler Clock...................... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     PCLK: Prescaler Clock...................... Running ---> FAILED

PCC2/PCLK Test Failure Data:
(error message)
```

PCC2 PCLK

Here, (error message) is one of the following:

```
Illegal prescaler calibration:
Expected EF, EC, E7, or DF, Actual = __

RTC is stopped, invoke SET command.

RTC is in write mode, invoke SET command.

RTC is in read mode, invoke SET command.

RTC seconds register didn't increment

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

Printer ACK Interrupts - PRNTA

PCC2 PRNTA

Command Input

```
197-Diag>PCC2 PRNTA
```

Description

Verifies that level 0 interrupts will not generate an interrupt, but will set the appropriate status. Then verifies that all interrupts (1 through 7) can be generated and received and that the appropriate status is set.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     PRNTA: Printer ‘ACK’ Interrupts............. Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     PRNTA: Printer ‘ACK’ Interrupts............. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     PRNTA: Printer ‘ACK’ Interrupts............. Running ---> FAILED

PCC2/PRNTA Test Failure Data:
(error message)
```

PCC2 PRNTA

Here, (error message) is one of the following:

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

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

PCC2 PRNTA

```
Access Fault Information:
 Address________
 Data________
 Access Size___
 Access Type___
 Address Space Code___
 Vector Number___

Unsolicited Exception:
 Instruction Pointer________
 Vector Number___
 Status Register____
 Interrupt Level___
```

Printer FAULT Interrupts - PRNTB

PCC2 PRNTB

Command Input

```
197-Diag>PCC2 PRNTB
```

Description

Verifies that level 0 interrupts will not generate an interrupt, but will set the appropriate status. Then verifies that all interrupts (1 through 7) can be generated and received and that the appropriate status is set.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     PRNTB: Printer 'FAULT' Interrupts........... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     PRNTB: Printer 'FAULT' Interrupts........... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     PRNTB: Printer 'FAULT' Interrupts........... Running ---> FAILED

PCC2/PRNTB Test Failure Data:
(error message)
```

PCC2 PRNTB

Here, (error message) is one of the following:

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

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

PCC2 PRNTB


```
Access Fault Information:
  Address________
  Data________
  Access Size___
  Access Type___
  Address Space Code___
  Vector Number___

Unsolicited Exception:
  Instruction Pointer________
  Vector Number___
  Status Register____
  Interrupt Level___
```

Printer SEL Interrupts - PRNTC

PCC2 PRNTC

Command Input

```
197-Diag>PCC2 PRNTC
```

Description

Verifies that level 0 interrupts will not generate an interrupt, but will set the appropriate status. Then verifies that all interrupts (1 through 7) can be generated and received and that the appropriate status is set.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     PRNTC: Printer ‘SEL’ Interrupts............. Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     PRNTC: Printer ‘SEL’ Interrupts............. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     PRNTC: Printer ‘SEL’ Interrupts............. Running ---> FAILED

PCC2/PRNTC Test Failure Data:
(error message)
```

PCC2 PRNTC

Here, (error message) is one of the following:

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

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

PCC2 PRNTC

```
Access Fault Information:
  Address________
  Data________
  Access Size___
  Access Type___
  Address Space Code___
  Vector Number___

Unsolicited Exception:
  Instruction Pointer________
  Vector Number___
  Status Register____
  Interrupt Level___
```

Printer PE Interrupts - PRNTD

PCC2 PRNTD

Command Input

```
197-Diag>PCC2 PRNTD
```

Description

Verifies that level 0 interrupts will not generate an interrupt, but will set the appropriate status. Then verifies that all interrupts (1 through 7) can be generated and received and that the appropriate status is set.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     PRNTD: Printer ‘PE’ Interrupts............. Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     PRNTD: Printer ‘PE’ Interrupts............. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     PRNTD: Printer ‘PE’ Interrupts............. Running ---> FAILED

PCC2/PRNTD Test Failure Data:
(error message)
```

3

PCC2 PRNTD

Here, (error message) is one of the following:

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

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

PCC2 PRNTD


```
Access Fault Information:
 Address________
 Data________
 Access Size___
 Access Type___
 Address Space Code___
 Vector Number___

Unsolicited Exception:
 Instruction Pointer________
 Vector Number___
 Status Register____
 Interrupt Level___
```

Printer BUSY Interrupts - PRNTE

PCC2 PRNTE

3

Command Input

```
197-Diag>PCC2 PRNTE
```

Description

Verifies that level 0 interrupts will not generate an interrupt, but will set the appropriate status. Then verifies that all interrupts (1 through 7) can be generated and received and that the appropriate status is set.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     PRNTE: Printer 'BUSY' Interrupts........... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     PRNTE: Printer 'BUSY' Interrupts........... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     PRNTE: Printer 'BUSY' Interrupts........... Running ---> FAILED

PCC2/PRNTE Test Failure Data:
(error message)
```

PCC2 PRNTE

Here, (error message) is one of the following:

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

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

3

PCC2 PRNTE

```
Access Fault Information:
 Address________
 Data________
 Access Size___
 Access Type___
 Address Space Code___
 Vector Number___

Unsolicited Exception:
 Instruction Pointer________
 Vector Number___
 Status Register____
 Interrupt Level___
```

Device Access - REGA

PCC2 REGA

Command Input

```
197-Diag>PCC2 REGA
```

Description

All the device registers (except the “PIACK” registers) are accessed (read) on 8-, 16-, and 32-bit boundaries (no attempt is made to verify the contents of the registers).

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     REGA: Device Access........................ Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     REGA: Device Access........................ Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     REGA: Device Access........................ Running ---> FAILED

PCC2/REGA Test Failure Data:
(error message)
```

Here, (error message) is one of the following:

```
Access Fault Information:
  Address________
  Data________
  Access Size___
  Access Type___
  Address Space Code___
  Vector Number___

Unsolicited Exception:
  Instruction Pointer________
  Vector Number___
  Status Register____
  Interrupt Level___
```

Register Access - REGB

PCC2 REGB

Command Input

```
197-Diag>PCC2 REGB
```

Description

The device data lines are checked by successive writes and reads to the Tick Timer 1 Compare Register.

1. Checks that the register can be zeroed.
2. Walks a 1 bit through a field of zeros.
3. Walks a 0 bit through a field of ones.

When test is complete, if no error is detected, register is initialized to zero.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     REGB: Register Access....................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     REGB: Register Access....................... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     REGB: Register Access....................... Running ---> FAILED

PCC2/REGB Test Failure Data:
(error message)
```

3

PCC2 REGB

Here, (error message) is one of the following:

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

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

Timer 1 Counter - TMR1A

PCC2 TMR1A

Command Input

```
197-Diag>PCC2 TMR1A
```

Description

Verifies the Tick Timer Counter Register write/read ability and functionality.

1. Checks that the register can be zeroed.
2. Walks a 1 bit through a field of zeroes.
3. Walks a 0 bit through a field of ones.
4. Verifies that the Counter Register value increments when the counter is enabled.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     TMR1A: Timer 1 Counter...................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     TMR1A: Timer 1 Counter...................... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     TMR1A: Timer 1 Counter...................... Running ---> FAILED

PCC2/TMR1A Test Failure Data:
(error message)
```

Here, (error message) is one of the following:

```
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 =________
```

Timer 1 Free-Run - TMR1B

PCC2 TMR1B

Command Input

```
197-Diag>PCC2 TMR1B
```

Description

Verifies the Compare Register write/read ability and the functionality of the Tick Timer Free-run mode, i.e., that the Clear On Compare is disabled.

1. Checks that the register can be zeroed.
2. Walks a 1 bit through a field of zeros.
3. Walks a 0 bit through a field of ones.
4. Verifies that the Counter value exceeds the Compare value.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     TMR1B: Timer 1 Free-run..................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     TMR1B: Timer 1 Free-run..................... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     TMR1B: Timer 1 Free-run..................... Running ---> FAILED

PCC2/TMR1B Test Failure Data:
(error message)
```

PCC2 TMR1B

Here, (error message) is one of the following:

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

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

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

Timer 1 Clear On Compare - TMR1C

PCC2 TMR1C

Command Input

```
197-Diag>PCC2 TMR1C
```

Description

Verifies the Clear On Compare functionality by setting the Compare and Count Registers and letting the timer run until software timeout or error if Counter exceeds Compare.

Starts with a compare value of $FFFF and on each loop fills next higher bit position with a 1 until value rolls over to a one.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     TMR1C: Timer 1 Clear on Compare............. Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     TMR1C: Timer 1 Clear on Compare............. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     TMR1C: Timer 1 Clear on Compare............. Running ---> FAILED

PCC2/TMR1C Test Failure Data:
   Count did not zero on Compare
   Address =________, Expected =________, Actual =________
```

Timer 1 Overflow Counter - TMR1D

PCC2 TMR1D

Command Input

```
197-Diag>PCC2 TMR1D
```

Description

Verifies the Overflow Counter functionality by performing the following:

1. Checks Overflow Counter for clear condition.
2. Verifies that the Overflow Counter increments by setting the Compare Register to $FFFF, the Count Register to zero and letting the timer run until the counter exceeds the compare value or error (software timeout).
3. Verifies that the Overflow Counter can be cleared (zeroed).
4. Verifies the Overflow Counter by setting the Compare Register to $FF, the Count Register to zero and letting the timer run until all the Overflow Counter Register bits have been set to a one or error (software timeout). Starting with an overflow count of 1 each bit is verified as it is set.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     TMR1D: Timer 1 Overflow Counter............. Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     TMR1D: Timer 1 Overflow Counter............. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     TMR1D: Timer 1 Overflow Counter............. Running ---> FAILED

PCC2/TMR1D Test Failure Data:
(error message)
```

PCC2 TMR1D

Here, (error message) is one of the following:

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

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

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

Timer 1 Interrupts - TMR1E

PCC2 TMR1E

Command Input

```
197-Diag>PCC2 TMR1E
```

Description

Verifies that level 0 interrupts will not generate an interrupt, but will set the appropriate status. Then verifies that all interrupts (1 through 7) can be generated and received and that the appropriate status is set.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     TMR1E: Timer 1 Interrupts.................. Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     TMR1E: Timer 1 Interrupts.................. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     TMR1E: Timer 1 Interrupts.................. Running ---> FAILED

PCC2/TMR1E Test Failure Data:
(error message)
```

PCC2 TMR1E

Here, (error message) is one of the following:

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

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

PCC2 TMR1E

3

```
Access Fault Information:
  Address________
  Data________
  Access Size___
  Access Type___
  Address Space Code___
  Vector Number___

Unsolicited Exception:
  Instruction Pointer________
  Vector Number___
  Status Register____
  Interrupt Level___
```

Timer 2 Counter - TMR2A

PCC2 TMR2A

Command Input

```
197-Diag>PCC2 TMR2A
```

Description

Verifies the Tick Timer Counter Register write/read ability and functionality.

1. Checks that the register can be zeroed.
2. Walks a 1 bit through a field of zeros.
3. Walks a 0 bit through a field of ones.
4. Verifies that the Counter Register value increments when the counter is enabled.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     TMR2A: Timer 2 Counter...................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     TMR2A: Timer 2 Counter...................... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     TMR2A: Timer 2 Counter...................... Running ---> FAILED

PCC2/TMR2A Test Failure Data:
(error message)
```

PCC2 TMR2A

Here, (error message) is one of the following:

```
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 =________
```

Timer 2 Free-Run - TMR2B

PCC2 TMR2B

Command Input

```
197-Diag>PCC2 TMR2B
```

Description

Verifies the Compare Register write/read ability and the functionality of the Tick Timer Free-run mode, i.e., that the Clear On Compare is disabled.

1. Checks that the register can be zeroed.
2. Walks a 1 bit through a field of zeros.
3. Walks a 0 bit through a field of ones.
4. Verifies that the Counter value exceeds the Compare value.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     TMR2B: Timer 2 Free-run..................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     TMR2B: Timer 2 Free-run..................... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     TMR2B: Timer 2 Free-run..................... Running ---> FAILED

PCC2/TMR2B Test Failure Data:
(error message)
```

PCC2 TMR2B

Here, (error message) is one of the following:

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

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

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

Timer 2 Clear On Compare - TMR2C

PCC2 TMR2C

Command Input

```
197-Diag>PCC2 TMR2C
```

Description

Verifies the Clear On Compare functionality by setting the Compare and Count Registers and letting the timer run until software timeout or error if Counter exceeds Compare.

Starts with a compare value of $FFFF and on each loop fills next higher bit position with a 1 until value rolls over to a one.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     TMR2C: Timer 2 Clear on Compare............. Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     TMR2C: Timer 2 Clear on Compare............. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     TMR2C: Timer 2 Clear on Compare............. Running ---> FAILED

PCC2/TMR2C Test Failure Data:

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

3

Timer 2 Overflow Counter - TMR2D

PCC2 TMR2D

Command Input

```
197-Diag>PCC2 TMR2D
```

Description

Verifies the Overflow Counter functionality by performing the following:

1. Checks Overflow Counter for clear condition.
2. Verifies that the Overflow Counter increments by setting the Compare Register to $FFFF, the Count Register to zero and letting the timer run until the counter exceeds the compare value or error (software timeout).
3. Verifies that the Overflow Counter can be cleared (zeroed).
4. Verifies the Overflow Counter by setting the Compare Register to $FF, the Count Register to zero and letting the timer run until all the Overflow Counter Register bits have been set to a one or error (software timeout). Starting with an overflow count of 1 each bit is verified as it is set.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     TMR2D: Timer 2 Overflow Counter............. Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     TMR2D: Timer 2 Overflow Counter............. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     TMR2D: Timer 2 Overflow Counter............. Running ---> FAILED

PCC2/TMR2D Test Failure Data:
(error message)
```

PCC2 TMR2D

Here, (error message) is one of the following:

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

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

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

Timer 2 Interrupts - TMR2E

PCC2 TMR2E

3

Command Input

```
197-Diag>PCC2 TMR2E
```

Description

Verifies that level 0 interrupts will not generate an interrupt, but will set the appropriate status. Then verifies that all interrupts (1 through 7) can be generated and received and that the appropriate status is set.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     TMR2E: Timer 2 Interrupts.................. Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     TMR2E: Timer 2 Interrupts.................. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     TMR2E: Timer 2 Interrupts.................. Running ---> FAILED

PCC2/TMR2E Test Failure Data:
(error message)
```

PCC2 TMR2E

Here, (error message) is one of the following:

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

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

PCC2 TMR2E

```
Access Fault Information:
 Address________
 Data________
 Access Size___
 Access Type___
 Address Space Code___
 Vector Number___

Unsolicited Exception:
 Instruction Pointer________
 Vector Number___
 Status Register____
 Interrupt Level___
```

Vector Base Register - VBR

PCC2 VBR

Command Input

```
197-Diag>PCC2 VBR
```

Description

Uses the General Purpose I/O Interrupt Control to generate and service level 1 interrupts testing every iteration of the Vector Base Register.

Response/Messages

After the command has been issued, the following line is printed:

```
PCC2     VBR: Vector Base Register................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
PCC2     VBR: Vector Base Register................... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
PCC2     VBR: Vector Base Register................... Running ---> FAILED

PCC2/VBR Test Failure Data:
(error message)
```

PCC2 VBR

Here, (error message) is one of the following:

```
Write/read error to VBR
Address =________, Expected =__, Actual =__

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

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

Access Fault Information:
 Address________
 Data________
 Access Size__
 Access Type_
 Address Space Code_
 Vector Number___

Unsolicited Exception:
 Instruction Pointer________
 Vector Number___
 Status Register____
 Interrupt Level_
```

CD2401 Serial Port (ST2401) Tests

These tests check the **CD2401** serial ports.

Entering **ST2401** without parameters causes all **ST2401** tests to execute in the order shown in the table below.

To run an individual test, add that test name to the **ST2401** command.

The individual tests are described in alphabetical order on the following pages.

Table 3-11. ST2401 Test Group

Mnemonic	Description
POLL	Internal Loopback (ASYNC) Polled
INTR	Internal Loopback (ASYNC) Interrupt
DMA	Internal Loopback (ASYNC) DMA
BAUD	Baud Rates Internal Loopback (ASYNC)

ST2401 Configuration Parameters CF ST2401

Command Input

```
197-Diag>cf st2401
ST2401 Configuration Data:
Port Mask =0000000F ?
Chip A base =FFF45000 ?
Chip B base =00000000 ?
Base Intr. Vector =00000040 ?
197-Diag>
```

Description

User configurable test parameters are available for the **ST2401** test group. Refer to Chapter 2 for information on using the **CF** command to set configuration parameters.

The **ST2401** test parameters are listed in the command input block above and described below.

```
Port Mask =0000000F ?
```

This parameter is a mask, used to select which ports are tested, with each bit selecting a port. Bit 0 selects port 0, bit 1 selects port 1, bit 2 selects port 2, bit 3 selects port 3, etc.

```
Chip A base =FFF45000 ?
```

This parameter is the base address of Chip A.

```
Chip B base =00000000 ?
```

This parameter is the base address of Chip B.

```
Base Intr. Vector =00000040 ?
```

This parameter is the base interrupt vector which will encode the interrupt source and port.

Baud Rates, Async, Internal Loopback - BAUD **ST2401 BAUD**

Command Input

```
197-Diag>ST2401 BAUD
```

Description

This test verifies that the selected ports will operate at:

- ❑ 38400,
- ❑ 9600, and
- ❑ 1200

baud. It does so by configuring each selected port with the Local Loopback Mode enabled, then sending and (hopefully) receiving an incrementing pattern of data. The time required to receive 100 characters is measured to the nearest microsecond. If this time is within a tolerance of 0.5%, and the data is successfully sent and received, then the test passes.

If the test passes, the word `PASSED` is displayed, otherwise the word `FAILED` is displayed along with an error message describing the nature of the failure (unless the “non-verbose” mode has been chosen).

The ports tested are initially configured as follows:

- ❑ asynchronous
- ❑ DMA, 38.4 Kbaud
- ❑ eight bits
- ❑ one stop bit
- ❑ no parity, and
- ❑ no in-band flow control

The PCCchip2 is also configured to allow the chip to provide interrupt vectors directly (as opposed to auto-vectoring) during interrupt acknowledge cycles. During this test, these interrupts are permitted by the PCCchip2.

MC68000 family microprocessors automatically perform interrupt acknowledge cycles to obtain the interrupt vector.

The MC88110 does not do this, requiring the interrupt to be acknowledged manually via special logic within the PCCchip2.

3

ST2401 BAUD

After the 38.4 Kbaud operation has been verified the port being tested is reconfigured for 9600 baud and tested again. Following this, it is configured for 1200 baud and tested once more. The acceptable ranges for the time to receive 100 characters are shown in the following table.

Table 3-12. Baud Rates

Baud Rate	Low Value (ms)	High Value (ms)
38,400	25911	26172
9600	103646	104686
1200	829127	837500

Regardless of the outcome of the testing, ports 0 and 1 are returned to their original configuration afterward. Ports 2 and 3 are left disabled.

Response/Messages

After the command has been issued, the following line is printed:

```
ST2401   BAUD: Baud Rates, Async, Internal Loopback.. Running --->
```

If all selected ports send and receive the test data successfully, then the test passes:

```
ST2401   BAUD: Baud Rates, Async, Internal Loopback.. Running ---> PASSED
```

If an error occurs while configuring, transmitting, receiving, or reconfiguring, then the test fails:

```
ST2401   BAUD: Baud Rates, Async, Internal Loopback.. Running ---> FAILED
```

(error description follows unless non-verbose mode chosen)

Refer to the *ST2401 Error Messages* section for a list of the error messages and their meaning.

DMA I/O, Async, Internal Loopback - DMA **ST2401 DMA**

Command Input

```
197-Diag>ST2401 DMA
```

Description

DMA mode refers to a mode where the Direct Memory Access feature of the CD2401 is utilized. In the previous test, received characters generated interrupts and were moved from the CD2401 device to memory by the microprocessor. Likewise, the CD2401 indicated its readiness to accept transmit characters by generating an interrupt, which the microprocessor responded to by moving the characters from memory to the CD2401 device.

In the DMA mode, received characters and characters to be transmitted are directly moved to and from memory by the CD2401, with transmit and receive data interrupts being issued only when the buffers that hold these characters need emptying (receive case) or refilling (transmit case). This mode of operation greatly reduces the involvement of the microprocessor.

This test verifies that the selected ports will operate in DMA mode. It does so by configuring each selected port with the Local Loopback Mode enabled, then sending and (hopefully) receiving an incrementing pattern of data. The test passes if the entire sequence of data is successfully send and received.

If the test passes, the word `PASSED` will be displayed, otherwise the word `FAILED` will be displayed along with an error message describing the nature of the failure (unless the “non-verbose” mode has been chosen).

The ports tested are configured as follows:

- asynchronous
- 38.4 Kbaud
- eight bits
- one stop bit
- no parity, and
- no in-band flow control

The PCCchip2 is also configured to allow the chip to provide interrupt vectors directly (as opposed to auto-vectoring) during interrupt acknowledge cycles. During this test, these interrupts are permitted by the PCCchip2.

3

ST2401 DMA

MC68000 family microprocessors automatically perform interrupt acknowledge cycles to obtain the interrupt vector.

The MC88110 does not do this, requiring the interrupt to be acknowledged manually via special logic within the PCCchip2.

Regardless of the outcome of the testing, ports 0 and 1 are returned to their original configuration afterward. Ports 2 and 3 are left disabled.

Response/Messages

After the command has been issued, the following line is printed:

```
ST2401   DMA: DMA I/O, Async, Internal Loopback...... Running --->
```

If all selected ports send and receive the test data successfully, then the test passes:

```
ST2401   DMA: DMA I/O, Async, Internal Loopback...... Running ---> PASSED
```

If an error occurs while configuring, transmitting, receiving, or reconfiguring, then the test fails:

```
ST2401   DMA: DMA I/O, Async, Internal Loopback...... Running ---> FAILED
```

(error description follows unless non-verbose mode chosen)

Refer to the *ST2401 Error Messages* section for a list of the error messages and their meaning.

Interrupt I/O, Async, Internal Loopback - INTR

ST2401 INTR

Command Input

```
197-Diag>ST2401 INTR
```

Description

Interrupt mode refers to a mode of operation that is characterized by the CD2401 interrupting the microprocessor to indicate one or more of the following conditions:

- Data has been received
- A readiness to accept data to be transmitted
- A change in state of the modem signals, or
- The receiver has status to report

This is a mode commonly used by operating systems. It does not involve direct memory access, which will be used in subsequent tests.

This test verifies that the selected ports will operate in interrupt mode. It does so by configuring each selected port with the Local Loopback Mode enabled, then sending and (hopefully) receiving an incrementing pattern of data. The test passes if the entire sequence of data is successfully send and received.

If the test passes, the word `PASSED` will be displayed, otherwise the word `FAILED` will be displayed along with an error message describing the nature of the failure (unless the “non-verbose” mode has been chosen).

The ports tested are configured as follows:

- asynchronous
- 38.4 Kbaud
- eight bits
- one stop bit
- no parity, and
- no in-band flow control

The PCCchip2 is also configured to allow the chip to provide interrupt vectors directly (as opposed to auto-vectoring) during interrupt acknowledge cycles. During this test, these interrupts are permitted by the PCCchip2.

3

ST2401 INTR

MC68000 family microprocessors automatically perform interrupt acknowledge cycles to obtain the interrupt vector.

The MC88110 does not do this, requiring the interrupt to be acknowledged manually via special logic in the PCCchip2.

Regardless of the outcome of the testing, ports 0 and 1 are returned to their original configuration afterward. Ports 2 and 3 are left disabled.

Response/Messages

After the command has been issued, the following line is printed:

```
ST2401  INTR: Interrupt I/O, Async, Internal Loopback.. Running --->
```

If all selected ports send and receive the test data successfully, then the test passes:

```
ST2401  INTR: Interrupt I/O, Async, Internal Loopback.. Running ---> PASSED
```

If an error occurs while configuring, transmitting, receiving, or reconfiguring, then the test fails:

```
ST2401  INTR: Interrupt I/O, Async, Internal Loopback.. Running ---> FAILED
```

(error description follows unless non-verbose mode chosen)

Refer to the *ST2401 Error Messages* section for a list of the error messages and their meaning.

Polled I/O, Async, Internal Loopback - POLL **ST2401 POLL**

Command Input

```
197-Diag>ST2401 POLL
```

Description

Polled mode refers to a mode of operation that prevents interrupts from reaching the microprocessor. This mode is used by the debugger. The CD2401 does generate interrupts while in the polled mode, but they are "masked" by the PCCchip2. This device provides a feature that permits the microprocessor to "poll" for interrupt status and simulate the taking of interrupts, which is required to operate the CD2401.

This test verifies that the ports selected by the Port Mask parameter will operate in polled mode. It does so by configuring each selected port with the Local Loopback Mode enabled, then sending and (hopefully) receiving an incrementing pattern of data. The test passes if the entire sequence of data is successfully sent and received.

If the test passes, the word `PASSED` is displayed; otherwise the word `FAILED` is displayed along with an error message describing the nature of the failure unless the "non-verbose" mode has been chosen.

The ports tested are configured as follows:

- ❏ asynchronous
- ❏ 38.4 Kbaud
- ❏ eight bits
- ❏ one stop bit
- ❏ no parity, and
- ❏ no in-band flow control

The PCCchip2 is also configured to allow the chip to provide interrupt vectors directly (as opposed to auto-vectoring) during interrupt acknowledge cycles. Some form of vectoring is required, as the CD2401 generates interrupts even during polled operation. During this test, these interrupts are masked by the PCCchip2 and acknowledged manually via special logic in the PCCchip2.

Regardless of the outcome of the testing, ports 0 and 1 are returned to their original configuration afterward. Ports 2 and 3 are left disabled.

Response/Messages

After the command has been issued, the following line is printed:

```
ST2401   POLL: Polled I/O, Async, Internal Loopback.. Running --->
```

If all selected ports send and receive the test data successfully, then the test passes:

```
ST2401   POLL: Polled I/O, Async, Internal Loopback.. Running ---> PASSED
```

If an error occurs while configuring, transmitting, receiving, or reconfiguring, then the test fails:

```
ST2401   POLL: Polled I/O, Async, Internal Loopback.. Running ---> FAILED
```

(error description follows unless non-verbose mode chosen)

Refer to the *ST2401 Error Messages* section for a list of the error messages and their meaning.

ST2401 Common Test Error Messages

The **ST2401** test group error messages generally take the following form:

```
ST2401  POLL: Polled I/O, Async, Internal Loopback.... Running ---> FAILED
ST2401/POLL Test Failure Data:
Port #$00: Timed-out, expecting RX IRQ
```

First there is a header message that describes which test was executing and announcing the "Test Failure Data". Following this, a single line of information is displayed, which in turn identifies the port being tested (0 through 3) and the failure symptom.

The "symptoms" are listed below:

Error Message	Symptom or Cause
`Interrupt, IACK'd Vector $XX`	Indicates occurrence of unexpected interrupt
`Exception, Vector $XX`	Indicates occurrence of unexpected exception
`Rx: IACK'd Vector $XX`	Unexpected vector read from PCCchip2 SCC Receiver pseudo-IACK register
`Tx: IACK'd Vector $XX`	Unexpected vector read from PCCchip2 SCC Transmitter pseudo-IACK register
`Modem IRQ unexpected`	Interrupt from modem signal change unexpected
`Timed-out, expecting RX IRQ`	Expected receive data interrupt, time expired first
`BREAK detect status`	Receiver indicates BREAK detected
`Framing Error status`	Receiver indicates a framing error occurred

3

Error Message	Symptom or Cause
`Overrun Error status`	Receiver indicates a data overrun occurred
`Parity Error status`	Receiver indicates a parity error occurred
`RX data corrupted, address a, expected e, read r1`	Received data differs from that transmitted; address shown is for the received character
`Chars follow EOT`	Extra characters follow test message
`Timed-out before TX FIFO empty`	Time-out expired waiting for transmit FIFO to empty
`b baud, 100 chars took t usec, expected x-y`	Time to receive 100 characters fails 0.5% criterion (expected range shown)
`CF error - no such device`	User selected port other than those supported by hardware; port mask should be in range $01-$0F
`can't idle device before test`	Time ran out waiting for CD2401 to indicate idle condition prior to configuring for test
`can't idle device after test`	Time ran out waiting for CD2401 to indicate idle condition after completion of testing
`can't write Chan. Cmd Reg - busy`	One second elapsed without Channel Command Register indicating readiness to accept next command (register contents remained nonzero)

VME Interface ASIC (VME2) Tests

These sections describe the individual VME2 tests.

Entering **VME2** without parameters executes all **VME2** tests in the order shown in the next table.

To run an individual test, add that test name to the **VME2** command.

The individual tests are listed in the next table and described in alphabetical order on the following pages.

Table 3-13. VME2 Test Group

Mnemonic	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)

Table 3-13. VME2 Test Group (Continued)

Mnemonic	Description
SWIB	Software Interrupts (Processor Interrupt Mode)
SWIC	Software Interrupts Priority

VME2 Configuration Parameters **CF VME2**

Command Input

```
197-Diag>cf vme2
VME2 Configuration Data:
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 ?
197-Diag>
```

Description

User configurable test parameters are available for the **VME2** test group. Refer to Chapter 2 for information on using the **CF** command to set configuration parameters.

The VME2 test parameters are listed and described below.

```
Prescaler Clock Adjust Timeout =00FF0000 ?
```

This parameter is not used.

```
tmr_cmp(): counter reg mask =FFFFFFF0 ?
```

This parameter selects the significant bits of each counter to be tested. Used by the TMRD through TMRG tests.

```
User defined Aux ROM base address Enable [Y/N] =N ?
```

This parameter is used for factory tests and should not be modified.

```
User defined Aux ROM base address =00080000 ?
```

This parameter is used for factory tests and should not be modified.

```
Master Decoder default select =00000001 ?
```

This parameter is not used.

```
Master Write Post Interrupt level =00000001 ?
```

This parameter is used for factory tests and should not be modified.

```
Master Decoder Trans. test: AUX slave select =00000001 ?
```

This parameter is used for factory tests and should not be modified.

Register Access - REGA

VME2 REGA

Command Input

```
197-Diag>VME2 REGA
```

Description

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.

Response/Messages

After the command has been issued, the following line is printed:

```
VME2     REGA: Register Access...................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
VME2     REGA: Register Access...................... Running ---> PASSED
```

VME2 REGA

If any part of the test fails, then the display appears as follows:

```
VME2     REGA: Register Access....................... Running ---> FAILED

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
```

Notes

1. **All data is displayed as hexadecimal values.**
2. **The Access Fault Information is only displayed if the exception was an Access Fault (Bus Error).**
3. **Access size is displayed in bytes.**
4. **Access type is 0 or 1 for write or read, respectively.**
5. **The address space code message uses the following codes: 1, 2, 5, 6, and 7 for user data, user program, supervisor data, supervisor program, and MPU space, respectively. All address space codes listed above may not be applicable to any single microprocessor type.**

Register Walking Bit - REGB

VME2 REGB

Command Input

```
197-Diag>VME2 REGB
```

Description

This test verifies that certain bits in the VME2 ASIC user registers can be set independently of other bits in the VME2 ASIC user registers. The test also assures that the VME2 ASIC user registers can be written without a Data Fault (Bus Error).

The VME2 register walking bit test is implemented by first saving the initial state of the Local Control and Status Registers (LCSR). All eligible bits are then initialized to zero. This initialization is verified. A one is walked through the LCSR bit array and the entire register bit field is verified after each write. All eligible bits are then initialized to one. This initialization is then verified. A zero is walked through the LCSR bit array and the entire register bit field is verified after each write. The initial state of the LCSR is restored except for the LCSR Prescaler Counter register.

Response/Messages

After the command has been issued, the following line is printed:

```
VME2     REGB: Register Walking Bit.................. Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
VME2     REGB: Register Walking Bit.................. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
VME2     REGB: Register Walking Bit.................. Running ---> FAILED

(error message)
```

VME2 REGB

Here, (error message) is one of the following:

If a bit in the LCSR cannot be initialized:

```
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:

```
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:

```
VME2/REGB Test Failure Data:

Unsolicited Exception:
  Exception Time PC/IP_____
  Vector_
Access Fault Information:
  Address________
  Data________
  Access Size_
  Access Type_
  Address Space Code__
```

Software Interrupts (Polled Mode) - SWIA

VME2 SWIA

Command Input

```
197-Diag>VME2 SWIA
```

Description

This test verifies that all software interrupts (1 through 7) can be generated and that the appropriate status is set.

Response/Messages

After the command has been issued, the following line is printed:

```
VME2     SWIA: Software Interrupts Polled............ Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
VME2     SWIA: Software Interrupts Polled............ Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
VME2     SWIA: Software Interrupts Polled............ Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

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. If any bits are set:

```
Interrupt Status Register is not initially cleared
Status: Expected =00000000, Actual =________
```

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:

```
Interrupt Status Register is not clear
Status: Expected =________, Actual =________
State : IRQ Level =__, SWI__, VBR =__
```

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, else the following message:

```
Unexpected status set in Interrupt Status Register
Status: Expected =________, Actual =________
State : IRQ Level =__, SWI__, VBR =__
```

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:

```
Interrupt Status Bit did not clear
Status: Expected =________, Actual =________
State : IRQ Level =__, SWI__, VBR =__
```

Software Interrupts (Processor Interrupt Mode) - SWIB VME2 SWIB

Command Input

```
197-Diag>VME2 SWIB
```

Description

This test verifies that all software interrupts (levels 1 through 7) can be generated and received and that the appropriate status is set.

Response/Messages

After the command has been issued, the following line is printed:

```
VME2     SWIB: Software Interrupts Interrupt......... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
VME2     SWIB: Software Interrupts Interrupt......... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
VME2     SWIB: Software Interrupts Interrupt......... Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

The interrupt enable register is cleared and status bits are read to verify that none are true:

```
Interrupt Status Register is not initially cleared
Status: Expected =________, Actual =________
```

VME2 SWIB

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:

```
Interrupt Status Register is not clear
Status: Expected =________, Actual =________
State : IRQ Level =__, SWI__, VBR =__
```

If the MPU is an M88000 family processor, the exception vector number is checked to make sure that the exception received was that of the interrupt (exception number 1):

```
Incorrect Vector type
Vector: Expected =____, Actual =____
Status: Expected =____, Actual =____
State : IRQ Level =___, SWI__, VBR =___
```

If the received interrupt vector is not that of the programmed interrupt vector:

```
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:

```
Incorrect Interrupt Level
Level : Expected =____, Actual =____
State : IRQ Level =____, SWI__, VBR =____
```

If the programmed interrupt did not occur:

```
Software Interrupt did not occur
Status: Expected =____, Actual =____
State : IRQ Level =___, SWI__, VBR =___
```

VME2 SWIB

The VMEchip2 Interrupt Status Register is checked for the proper interrupt status bit to be active:

```
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:

```
Interrupt Status Bit did not clear
Status: Expected =____, Actual =____
State : IRQ Level =___, SWI__, VBR =___
```

Software Interrupts Priority - SWIC

VME2 SWIC

Command Input

```
197-Diag>VME2 SWIC
```

Description

This test verifies that all software interrupts (1 through 7) occur in the priority set by the hardware.

Response/Messages

After the command has been issued, the following line is printed:

```
VME2     SWIC: Software Interrupts Priority.......... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
VME2     SWIC: Software Interrupts Priority.......... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
VME2     SWIC: Software Interrupts Priority.......... Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

The interrupt enable register is cleared and status bits are read to verify that none are true:

```
Interrupt Status Register is not initially cleared
Status: Expected =________, Actual =________
```

VME2 SWIC

If the MPU is an M88000 family processor, the exception vector number is checked to make sure that the exception received was that of the interrupt (exception number 1):

```
Incorrect Vector type
Vector: Expected =__, Actual =__
Status: Expected =________, Actual =________
State : IRQ Level =___, SWI__, VBR =__
```

If the received interrupt vector is not that of the programmed interrupt vector:

```
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:

```
Incorrect Interrupt Level
Level : Expected =__, Actual =__
State : IRQ Level =___, SWI__, VBR =__
```

If the programmed interrupt did not occur:

```
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:

```
Unexpected status set in Interrupt Status Register
Status: Expected =________, Actual =________
State : IRQ Level =___, SWI__, VBR =__
```

3

VME2 SWIC

If, after receiving an interrupt, the interrupt status cannot be negated by writing the interrupt clear register:

```
Interrupt Status Bit did not clear
Status: Expected =________, Actual =________
State : IRQ Level =___, SWI__, VBR =__
```

Timer Accuracy Test - TACU

VME2 TACU

Command Input

```
197-Diag>VME2 TACU
```

Description

This test verifies the VMEChip2 ASIC timer and prescaler circuitry using the on-board Real Time Clock (RTC) as a timing reference.

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.

The prescaler calibration register is checked to verify that it contains one of four legal MPU clock calibration values.

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.

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.

Response/Messages

After the command has been issued, the following line is printed:

```
VME2     TACU: Timer Accuracy........................ Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
VME2     TACU: Timer Accuracy........................ Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
VME2     TACU: Timer Accuracy........................ Running ---> FAILED

(error message)
```

VME2 TACU

Here, (error message) is one of the following:

If the RTC is stopped:

```
RTC is stopped, invoke SET command.
```

If the RTC is in the write mode:

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

If the RTC is in the read mode:

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

If the prescaler calibration register does not contain one of four legal MPU clock calibration values:

```
Illegal prescaler calibration:
Expected EF, EC, E7, or DF, Actual =__
```

If tick timer accuracy is out of tolerance:

```
Timer counter register read (greater/less) than expected
Address =________, Expected =________, Actual =________
```

If prescaler counter register accuracy is out of tolerance, the prescaler counter address and expected and actual difference counts are displayed:

```
Prescaler delta was (greater/less) than expected
Address =________, Expected =________, Actual =________
```

If the RTC seconds register does not increment during the test:

```
RTC seconds register didn't increment
```

Timer Increment - TMRA, TMRB

VME2 TMRA/TMRB

Command Input

```
197-Diag>VME2 TMRA
```

or

```
197-Diag>VME2 TMRB
```

Description

This test verifies that Timer x Counter Register (x = 1 or 2) can be set to 0, and, that Timer x Counter Register value increments when enabled. The Timer is initialized by writing 0 to the Tick Timer Counter Register. The Clear On Compare mode is disabled by writing the COCx bit in the Tick Timer Control Register. The Timer is enabled by the ENx bit in the Tick Timer Control Register. The MPU executes a time delay loop, then disables Tick Timer x. The Tick Timer Control Register is read to see if it incremented from its initial value of 0. **TMRA** specifies Tick Timer 1. **TMRB** specifies Tick Timer 2.

Response/Messages

Note that in all responses shown below, the response "TMRx: `Timer n`" is `TMRA: Timer 1` or `TMRB: Timer 2`, depending upon which test set is being performed.

After the command has been issued, the following line is printed:

```
VME2     TMRx: Timer n Increment..................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
VME2     TMRx: Timer n Increment..................... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
VME2     TMRx: Timer n Increment..................... Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

```
Tick Timer _ Counter did not clear.

Tick Timer _ Counter did not increment.
```

Prescaler Clock Adjust - TMRC

VME2 TMRC

Command Input

```
197-Diag>VME2 TMRC
```

Description

This test verifies that the Prescaler Clock Adjust register can vary the period of the tick timer input clock. The test fails if the Prescaler Clock Adjust register has not been previously initialized to a nonzero value. Two MPU timing loops are executed, the first with a "low" Prescaler Clock Timer 1 of the VMEchip2 is used for reference in this test. 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. The Prescaler Clock Adjust register value is restored upon correct test execution.

Response/Messages

After the command has been issued, the following line is printed:

```
VME2     TMRC: Prescaler Clock Adjust................ Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
VME2     TMRC: Prescaler Clock Adjust................ Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
VME2     TMRC: Prescaler Clock Adjust................ Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

If Prescaler Clock Adjust register was = 0:

```
Prescaler Clock Adjust reg was not initialized
```

A non-incrementing timer gives the following for first loop timeouts:

```
Low value:Timed out waiting for compare (ITIC1) ________ to assert.
```

A non-incrementing timer gives the following for last loop timeouts:

```
High value:Timed out waiting for compare (ITIC1) ________ to assert.
```

If the Prescaler Clock Adjust did not vary tick period:

```
Prescaler Clock Adjust did not vary tick period.
Loop1=________, Loop2=________.
```

Tick Timer No Clear On Compare - TMRD, TMRE **VME2 TMRD/TMRE**

Command Input

```
197-Diag>VME2 TMRD
```

or

```
197-Diag>VME2 TMRE
```

Description

This test verifies the Tick Timers No Clear On Compare mode. The Timer is initialized by writing 0 to the Tick Timer Counter Register. The Clear On Compare mode is disabled by writing the COCx bit in the Tick Timer Control Register. The compare value is initialized by writing $55aa to the Tick Timer Compare Register. The Timer is enabled by the ENx bit in the Tick Timer Control Register.

After starting the timer, the MPU enters a time delay loop while testing for Tick Timer compare. Tick Timer compare is sensed by reading the TICx bit in the Local Bus Interrupter Status Register. The Timer is stopped when Timer Compare is sensed, or an MPU loop counter register decrements to 0 (timeout). 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.

TMRD specifies Tick Timer 1. **TMRE** specifies Tick Timer 2.

Response/Messages

After the command has been issued, the following line is printed:

```
VME2     TMRD: Timer 1 No Clear On Compare........... Running --->
```

or

```
VME2     TMRE: Timer 2 No Clear On Compare........... Running --->
```

VME2 TMRD/TMRE

If all parts of the test are completed correctly, then the test passes.

```
VME2     TMRD: Timer 1 No Clear On Compare........... Running ---> PASSED
```

or

```
VME2     TMRD: Timer 2 No Clear On Compare........... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
VME2     TMRD: Timer 1 No Clear On Compare........... Running ---> FAILED

(error message)
```

or

```
VME2     TMRD: Timer 2 No Clear On Compare........... Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

```
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)
```

Tick Timer Clear On Compare - TMRF, TMRG

VME2 TMRF/TMRG

Command Input

```
197-Diag>VME2 TMRF
```

or

```
197-Diag>VME2 TMRG
```

Description

This test verifies the Tick Timers Clear On Compare mode. The Timer is initialized by writing 0 to the Tick Timer Counter Register. The Clear On Compare mode is enabled by writing the COCx bit in the Tick Timer Control Register. The compare value is initialized by writing $55aa to the Tick Timer Compare Register. The Timer is enabled by the ENx bit in the Tick Timer Control Register.

After starting the timer, the MPU enters a time delay loop while testing for Tick Timer compare. Tick Timer compare is sensed by reading the TICx bit in the Local Bus Interrupter Status Register. The Timer is stopped when Timer Compare is sensed, or an MPU loop counter register decrements to 0 (timeout). If the MPU loop counter did not time out, the Timer Counter Register is read to make sure that it was cleared on compare.

TMRF specifies Tick Timer 1. **TMRG** specifies Tick Timer 2.

Response/Messages

After the command has been issued, the following line is printed:

```
VME2     TMRF: Timer 1 Clear On Compare.............. Running --->
```

or

```
VME2     TMRG: Timer 2 Clear On Compare.............. Running --->
```

VME2 TMRF/TMRG

If all parts of the test are completed correctly, then the test passes.

```
VME2     TMRF: Timer 1 Clear On Compare.............. Running ---> PASSED
```

or

```
VME2     TMRG: Timer 2 Clear On Compare.............. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
VME2     TMRF: Timer 1 Clear On Compare.............. Running ---> FAILED

(error message)
```

or

```
VME2     TMRG: Timer 2 Clear On Compare.............. Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

```
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)
```

Overflow Counter - TMRH, TMRI

VME2 TMRH/TMRI

Command Input

```
197-Diag>VME2 TMRH
```

or

```
197-Diag>VME2 TMRI
```

Description

This test verifies that a count of timer overflow is accumulated when the overflow counter is enabled.

The COVF bit in the timer control register is asserted and OVF bit is verified to be clear. The timer counter register is set to zero, the timer compare register is loaded with the value $55aa, and the timer is enabled. When TIC(1/2) becomes true, the timer is disabled and the timer overflow counter register is checked to see that the resultant overflow was counted.

TMRH specifies Tick Timer 1 Overflow Counter. **TMRI** specifies Tick Timer 2 Overflow Counter.

Response/Messages

After the command has been issued, the following line is printed:

```
VME2     TMRH: Timer 1 Overflow Counter.............. Running --->
```

or

```
VME2     TMRI: Timer 2 Overflow Counter.............. Running --->
```

VME2 TMRH/TMRI

If all parts of the test are completed correctly, then the test passes.

```
VME2     TMRH: Timer 1 Overflow Counter.............. Running ---> PASSED
```

or

```
VME2     TMRI: Timer 2 Overflow Counter.............. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
VME2     TMRH: Timer 1 Overflow Counter.............. Running ---> FAILED

(error message)
```

or

```
VME2     TMRI: Timer 2 Overflow  Counter............. Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

```
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 = ________
```

Watchdog Timer Counter - TMRJ

VME2 TMRJ

Command Input

```
197-Diag>VME2 TMRJ
```

Description

Checks the functionality of the watchdog timer at all programmable timing values. The test also checks watchdog timer clear status and timeout functions.

The following is done for all programmable watchdog timeouts:

- ❑ Check for linear timeout period with respect to previous timeout
- ❑ Verify that timeout status can be cleared

Response/Messages

After the command has been issued, the following line is printed:

```
VME2     TMRJ: Watchdog Timer Counter................ Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
VME2     TMRJ: Watchdog Timer Counter................ Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
VME2     TMRJ: Watchdog Timer Counter................ Running ---> FAILED

(error message)
```

VME2 TMR

Here, (error message) is one of the following:

```
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
```

Watchdog Timer Board Fail - TMRK

VME2 TMRK

Command Input

```
197-Diag>VME2 TMRK
```

Description

Tests 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. The test verifies BRFLI (both negated and asserted states) for WDBFE.

Response/Messages

After the command has been issued, the following line is printed:

```
VME2     TMRK: Watchdog Timer Board Fail............ Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
VME2     TMRK: Watchdog Timer Board Fail............ Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
VME2     TMRK: Watchdog Timer Board Fail............ Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

```
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
```

LAN Coprocessor for Ethernet (LANC) Tests

This section describes the individual Local Area Network Coprocessor (i82596) for Ethernet (**LANC**) tests. The terms **LANC** and 82596 are used interchangeably in the following **LANC** test group explanation text.

The 82596, as an intelligent, high-performance LAN coprocessor, executes high-level commands, command chaining, and interprocessor 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. This includes the following:

- ❑ 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/s.

Entering **LANC** without parameters causes all **LANC** tests, except those noted otherwise, to execute in the order shown in the table below.

To run an individual test, add that test name to the **LANC** command.

Table 3-14. LANC Test Group

Mnemonic	Description
FUSE	+12VDC Fuse
CST	Chip Self Test
BERR	Bus Error
IRQ	Interrupt Request
DUMP	Dump Configuration/Registers
DIAG	Diagnose Internal Hardware

Table 3-14. LANC Test Group (Continued)

Mnemonic	Description
ILB	Internal Loopback
ELBT	External Loopback Transceiver
Executed only when specified:	
ELBC	External Loopback Cable
MON	Monitor (Incoming Frames) Mode
TDR	Time Domain Reflectometry

The individual tests are described in alphabetical order on the following pages. The error message displays following the explanation of a **LANC** test pertain to the test being discussed.

Following the descriptions of each test in the **LANC** test group is a list of additional error message displays which pertain to all tests within the group.

You can use the **CF** command to configure some parameters for these tests. Use the command **CF LANC** to change the transmit to receive loop count (32) for the **ELBC**, **ELBT**, and **ILB** tests.

LANC Configuration Parameters — CF LANC

Command Input

```
197-Diag>CF LANC
LANC Configuration Data:
Instruction Cache Enable [Y/N] =Y ?
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 ?
197-Diag>
```

Description

User configurable test parameters are available for the **LANC** test group. Refer to Chapter 2 for information on using the **CF** command to set configuration parameters.

The **LANC** test parameters are listed and described below.

```
Instruction Cache Enable [Y/N] =Y ?
```

This parameter enables the external instruction cache if set. Otherwise, the instruction cache is disabled.

```
Control Memory Base Address Override [Y/N] =N ?
```

This parameter is the LANC management control buffer base address override. This parameter should not be modified by the user.

```
Control Memory Base Address =00000000 ?
```

This parameter is the LANC management control buffer base address. This parameter should not be modified by the user.

CF LANC


```
Self Test Results Block Address =00000000 ?
```

This parameter defines the base address of the self test results data block in memory.

```
System Configuration Pointer =00000000 ?
```

This parameter is the system configuration pointer block pointer. This parameter should not be modified by the user.

```
Intermediate System Configuration Pointer =00000000 ?
```

This parameter indicates the location of the System Control Block (SCB). The CPU loads the address of the SCB into this pointer.

```
System Control Block Address =00000000 ?
```

This parameter is the system command block pointer. This parameter should not be modified by the user.

```
Configuration Command Block Address =00000000 ?
```

This parameter is the configuration command block address pointer. This parameter should not be modified by the user.

```
Individual Address Command Block Address =00000000 ?
```

This parameter is the individual address command block address pointer. This parameter should not be modified by the user.

```
Diagnose/NOP Command Block Address =00000000 ?
```

This parameter indicates the location of the Diagnose/NOP Command Block. The Diagnose/NOP Command Block triggers an internal self-test procedure that checks the 82596 serial subsystem. Refer to the *Diagnose Internal Hardware* section for additional information.

```
Dump Configuration/Registers Address =00000000 ?
```

This parameter indicates the address in memory for the output of the 82596 Dump command, which is configuration and register state from the Channel Interface Module.

CF LANC

```
TDR Command Block Address =00000000 ?
```

This parameter indicates the address of the Time Domain Reflectometry Command Block. The TDR Command Block is used to detect opens or shorts on the link and their distance from the diagnosing station.

```
Number Transmit/Receive Loopback Packets =00000020 ?
```

This parameter sets the number of loopback packets to be transmitted and received.

```
Ethernet Address (Source) =000000000000 ?
```

This parameter indicates the source ethernet address.

```
Ethernet Address (Destination) =000000000000 ?
```

This parameter indicates the destination ethernet address.

Bus Error - BERR

LANC BERR

Command Input

```
197-Diag>LANC BERR
```

Description

This test activates the LANC Dump Registers function, which is a mechanism to test the PCC2 or MCC LANC Error Status register. The dump area is pointed to a known local bus timeout (bus error). Currently, only the "Local Bus Timeout" bus error is tested. This function conforms to the CTMI test module specification and is called directly by the CTMI Host.

Response/Messages

After the command has been issued, the following line is printed:

```
LANC     CST: Chip Self Test........................ Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
LANC     CST: Chip Self Test........................ Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
LANC     CST: Chip Self Test........................ Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

If the pre-interrupt register state does not indicate polled interrupt mode and enabled interrupts then the following diagnostic is displayed:

```
LANC Bus Error Interrupt Control/Status Register Error:
Expected = 08, Actual = 13
```

LANC BERR

If the post-interrupt register state does not indicate that an interrupt occurred in polled interrupt mode then the following diagnostic is displayed:

```
LANC Bus Error Interrupt Control/Status Register Error:
Expected = 08, Actual = 13
```

If the LANC error status register indicates a local timeout then the following diagnostic is displayed:

```
LANC Error Status Register Error:
Expected = 02, Actual = 13
```

Chip Self Test - CST

LANC CST

Command Input

```
197-Diag>LANC CST
```

Description

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.

3

LANC CST

Response/Messages

After the command has been issued, the following line is printed:

```
LANC     CST: Chip Self Test......................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
LANC     CST: Chip Self Test......................... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
LANC     CST: Chip Self Test......................... Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

If the expected results do not match (equal) the actual results of the 82596 self-test command results:

```
LANC Chip Self-Test Error: Expected =00000000, Actual =10040000
```

Diagnose Internal Hardware - DIAG

LANC DIAG

Command Input

```
197-Diag>LANC DIAG
```

Description

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.

Phase 1:

1. Simultaneously resets all counters and shift registers.
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”.

LANC DIAG

6. Phase 1 is successful if the 10 least significant bits (when applicable) of all four counters are 'All Zeros'.

Phase 2:

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, then a “Passed” status is returned.
5. If Step 3 is not successful (a 0), then a “Failed” status is returned, and Step 3 is repeated.

Response/Messages

After the command has been issued, the following line is printed:

```
LANC     DIAG: Diagnose Internal Hardware............ Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
LANC     DIAG: Diagnose Internal Hardware............ Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
LANC     DIAG: Diagnose Internal Hardware............ Running ---> FAILED

(error message)
```

LANC DIAG

Here, (error message) is one of the following:

This failure is the result of the Diagnose command failing:

```
DIAGNOSE Command Completion Status Error:
OK-Bit =0, F(ail)-Bit =1
```

Dump Configuration/Registers - DUMP

LANC DUMP

Command Input

```
197-Diag>LANC DUMP
```

Description

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 expires, 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.

Response/Messages

After the command has been issued, the following line is printed:

```
LANC     DUMP: Dump Configuration/Registers.......... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
LANC     DUMP: Dump Configuration/Registers.......... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
LANC     DUMP: Dump Configuration/Registers.......... Running ---> FAILED

(error message)
```

LANC DUMP

Here, (error message) is one of the following:

This failure is the result of the Dump command failing:

```
Dump Status Error: Expected =A006, Actual =8006
```

External Loopback Cable - ELBC

LANC ELBC

3

Command Input

```
197-Diag>LANC ELBC
```

Description

Verifies that the 82596 can be operated in the “External Loopback with the LPBK pin not activated” mode.

The 82596 has three modes of loopback:

- ❏ Internal Loopback
- ❏ External Loopback with the LPBK pin activated, and
- ❏ External Loopback with the LPBK pin not activated

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.

In Internal Loopback mode 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.

In External Loopback mode 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.

The test sets up a data packet (incrementing data pattern) to be transmitted, transmits it, and waits for the reception of the data. Once the data is received, the data is verified to the data transmitted. This transmit to receive loop is performed 32 times. You can change the loop count with the **CF** command (**CF LANC**).

Note that this test does not execute when the **LANC** test group is executed (**LANC** with no arguments). This test is supplied only for diagnostic purposes. It requires a properly set up Ethernet network (cable).

LANC ELBC

Response/Messages

After the command has been issued, the following line is printed:

```
LANC     ELBC: External Loopback Cable............... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
LANC     ELBC: External Loopback Cable............... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
LANC     ELBC: External Loopback Cable............... Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

Once the data packet has been set up to be transmitted, the test instructs the 82596 (through the Command Unit) to transmit the data packet. This failure is the result of the 82596 completing with a transmit data error. The status bits of the error message display indicate the source of the problem:

```
TRANSMIT Command Completion Status Error:
OK-Bit =0, ABORT-Bit =0, STATUS-Bits =0010
```

STATUS-Bits Breakdown:

Bit #6 Late collision. A late collision (a collision after the slot time elapsed) is detected.

Bit #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).

LANC ELBC

3

Bit #4 Transmission unsuccessful (stopped) due to Loss of Clear to Send signal.

Bit #3 Transmission unsuccessful (stopped) due to DMA Underrun, i.e., the system did not supply data for transmission.

Bit #2 Transmission Deferred, i.e., transmission was not immediate due to previous link activity.

Bit #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.

Bit #0 Transmission attempt was stopped because the number of collisions exceeded the maximum allowable number of retries.

Once the data packet is transmitted successfully, the test waits for four seconds for the receipt of the data. This failure results if the time-out expires:

```
RECEIVE Data Time-Out:
```

Once the transmitted data has been received, the test verifies the status of the receive data packet. This failure results if the receive data packet was received in error:

```
RECEIVE Status Error:
COMPLETE-Bit =1, OK-Bit=0, STATUS-Bits =0000
```

STATUS-Bits Breakdown:

Bit #12 Length error if configured to check length.

Bit #11 CRC error in an aligned frame.

Bit #10 Alignment error (CRC error in misaligned frame).

Bit #9 Ran out of buffer space - no resources.

Bit #8 DMA Overrun failure to acquire the system bus.

Bit #7 Frame to short.

Bit #6 No EOP flag (for Bit stuffing only).

Bit #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 does not match the individual address. For example, a multicast or broadcast address sets this bit to a one.

Bit #0 Receive collision. A collision is detected during reception.

Once the data packet has been received and the receive data status verified, the test verifies that the number of bytes received equals the number of bytes transmitted. This failure results if the receive data count and the transmit data count were not equal:

```
RECEIVE Data Transfer Count Error:
Expected =05EA, Actual =003C
```

Upon completion of all the status checks, the test now verifies the received data to the transmitted data. This failure results if the data does not verify (compare):

```
Receive Data Miscompare Error:
Address =0000E2C0, Expected =3E3F, Actual =3E3E
```

3

External Loopback Transceiver - ELBT

LANC ELBT

Command Input

```
197-Diag>LANC ELBT
```

Description

Verifies that the 82596 can be operated in the “External Loopback with the LPBK pin activated” mode.

The 82596 has three modes of loopback:

- ❏ Internal Loopback
- ❏ External Loopback with the LPBK pin activated, and
- ❏ External Loopback with the LPBK pin not activated

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.

In Internal Loopback mode 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.

In External Loopback mode 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.

The test sets up a data packet (incrementing data pattern) to be transmitted, transmits it, and waits for the reception of the data. Once the data is received, the data is verified to the data transmitted. The test performs the transmit to receive loop 32 times. You can change the loop count with the **CF** command (**CF LANC**).

Response/Messages

After the command has been issued, the following line is printed:

```
LANC     ELBT: External Loopback Transceiver......... Running --->
```

LANC ELBT

If all parts of the test are completed correctly, then the test passes.

```
LANC     ELBT: External Loopback Transceiver......... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
LANC     ELBT: External Loopback Transceiver......... Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

Once the data packet has been set up to be transmitted, the test instructs the 82596 (through the Command Unit) to transmit the data packet. This failure is the result of the 82596 completing with a transmit data error. The status bits of the error message display indicate the source of the problem:

```
TRANSMIT Command Completion Status Error:
OK-Bit =0, ABORT-Bit =0, STATUS-Bits =0010
```

STATUS-Bits Breakdown:

Bit #6 Late collision. A late collision (a collision after the slot time elapsed) is detected.

Bit #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).

Bit #4 Transmission unsuccessful (stopped) due to Loss of Clear to Send signal.

Bit #3 Transmission unsuccessful (stopped) due to DMA Underrun, i.e., the system did not supply data for transmission.

Bit #2 Transmission Deferred, i.e., transmission was not immediate due to previous link activity.

Bit #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.

Bit #0 Transmission attempt was stopped because the number of collisions exceeded the maximum allowable number of retries.

Once the data packet is transmitted successfully, the test waits for four seconds for the receipt of the data. This failure is the result of the time-out (four seconds) expiring:

```
RECEIVE Data Time-Out:
```

Once the transmitted data has been received, the test verifies the status of the receive data packet. This failure is the result of the receive data packet having been received in error:

```
RECEIVE Status Error:
COMPLETE-Bit =1, OK-Bit=0, STATUS-Bits =0000
```

STATUS-Bits Breakdown:

Bit #12 Length error if configured to check length.

Bit #11 CRC error in an aligned frame.

Bit #10 Alignment error (CRC error in misaligned frame).

Bit #9 Ran out of buffer space - no resources.

Bit #8 DMA Overrun failure to acquire the system bus.

Bit #7 Frame to short.

Bit #6 No EOP flag (for Bit stuffing only).

LANC ELBT

Bit #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.

Bit #0 Receive collision. A collision is detected during reception.

Once the data packet has been received and the receive data status verifies, the test verifies that the number of bytes received equals the number of bytes transmitted. This failure results if the receive data count and the transmit data count were not equal:

```
RECEIVE Data Transfer Count Error:
Expected =05EA, Actual =003C
```

Upon completion of all the status checks, the test now verifies the received data to the transmitted data. This failure results if the data does not verify (compare):

```
Receive Data Miscompare Error:
Address =0000E2C0, Expected =3E3F, Actual =3E3E
```

+12VDC Fuse - FUSE

LANC FUSE

3

Command Input

```
197-Diag>LANC FUSE
```

Description

Verifies that the +12VDC Fuse indicator (via the VMEChip2) is true (fuse present). The MVME197 supplies the +12VDC power to the Ethernet transceiver interface through a fuse. The green +12VDC (LAN power) LED (part of DS3) lights when power is available to the transceiver interface.

Response/Messages

After the command has been issued, the following line is printed:

```
LANC     FUSE: +12VDC Fuse.......................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
LANC     FUSE: +12VDC Fuse.......................... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
LANC     FUSE: +12VDC Fuse.......................... Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

This failure is the result of the fuse indicator (via the VMEChip2) being false (fuse not present or blown):

```
FUSE (+12VDC) Status Bit Error: Expected =0, Actual =1
```

Internal Loopback - ILB

LANC ILB

Command Input

```
197-Diag>LANC ILB
```

Description

Verifies that the 82596 can be operated in the "Internal Loopback" mode.

The 82596 has three modes of loopback:

- ❏ Internal Loopback
- ❏ External Loopback with the LPBK pin activated, and
- ❏ External Loopback with the LPBK pin not activated

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.

In Internal Loopback mode 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.

In External Loopback mode 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.

The test sets up a data packet (incrementing data pattern) to be transmitted, transmits it, and waits for the reception of the data. Once the data is received, the data is verified to the data transmitted. The test performs this transmit to receive loop 32 times. You can change the loop count with the **CF** command (**CF LANC**).

Response/Messages

After the command has been issued, the following line is printed:

```
LANC     ILB: Internal Loopback...................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
LANC     ILB: Internal Loopback...................... Running ---> PASSED
```

3

LANC ILB

If any part of the test fails, then the display appears as follows:

```
LANC     ILB: Internal Loopback..................... Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

Once the data packet has been set up to be transmitted, the test instructs the 82596 (through the Command Unit) to transmit the data packet. This failure is the result of the 82596 completing with a transmit data error. The status bits of the error message display indicate the source of the problem:

```
TRANSMIT Command Completion Status Error:
OK-Bit =0, ABORT-Bit =0, STATUS-Bits =0010
```

STATUS-Bits Breakdown:

Bit #6 Late collision. A late collision (a collision after the slot time elapsed) is detected.

Bit #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).

Bit #4 Transmission unsuccessful (stopped) due to Loss of Clear to Send signal.

Bit #3 Transmission unsuccessful (stopped) due to DMA Underrun, i.e., the system did not supply data for transmission.

Bit #2 Transmission Deferred, i.e., transmission was not immediate due to previous link activity.

LANC ILB

Bit #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.

Bit #0 Transmission attempt was stopped because the number of collisions exceeded the maximum allowable number of retries.

Once the data packet is transmitted successfully, the test waits for four seconds for the receipt of the data. This failure is the result of the time-out (four seconds) expiring:

```
RECEIVE Data Time-Out:
```

Once the transmitted data has been received, the test verifies the status of the receive data packet. This failure result if the receive data packet was received in error:

```
RECEIVE Status Error:
COMPLETE-Bit =1, OK-Bit=0, STATUS-Bits =0000
```

STATUS-Bits Breakdown:

Bit #12 Length error if configured to check length.

Bit #11 CRC error in an aligned frame.

Bit #10 Alignment error (CRC error in misaligned frame).

Bit #9 Ran out of buffer space - no resources.

Bit #8 DMA Overrun failure to acquire the system bus.

Bit #7 Frame to short.

Bit #6 No EOP flag (for Bit stuffing only).

Bit #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.

Bit #0 Receive collision. A collision is detected during reception.

Once the data packet has been received and the receive data status verified, the test verifies that the number of bytes received equals the number of bytes transmitted. This failure results if the receive data count and the transmit data count were not equal:

```
RECEIVE Data Transfer Count Error:
Expected =05EA, Actual =003C
```

Upon completion of all the status checks, the test now verifies the received data to the transmitted data. This failure results if the data does not verify (compare):

```
Receive Data Miscompare Error:
Address =0000E2C0, Expected =3E3F, Actual =3E3E
```

Interrupt Request - IRQ

LANC IRQ

Command Input

```
197-Diag>LANC IRQ
```

Description

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.

Response/Messages

After the command has been issued, the following line is printed:

```
LANC     IRQ: Interrupt Request..................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
LANC     IRQ: Interrupt Request..................... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
LANC     IRQ: Interrupt Request..................... Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

Prior to the 82596 initialization sequence launch, the interrupt control register in the PCC2 is verified against the pretest expected results. This failure is the result of the register contents not verifying against the expected pretest results:

```
LANC Interrupt Control/Status Register Error:
Expected =50, Actual =70
```

LANC IRQ

Upon completion of the initialization sequence of the 82596, the test verifies the interrupt control register for interrupt status. This failure is the result of the register contents not verifying against the expected post test results, i.e., interrupt status bit not set:

```
LANC Interrupt Control/Status Register Error:
Expected =70, Actual =50
```

Once the interrupt status is verified, the interrupt status is cleared via the ICLR bit in the interrupt control register in the PCC2. This failure is the result of the interrupt status bit (INT) in the interrupt control register not clearing:

```
LANC Interrupt Control/Status Register Error:
Expected =50, Actual =70
```

Monitor (Incoming Frames) Mode - MON

LANC MON

Command Input

```
197-Diag>LANC MON
```

Description

Monitors all incoming (receive data) frames.

This test is subclassed as a utility test. It does not execute when the **LANC** test group is executed. Also, there is no PASS/FAIL message associated with it. The utility is provided for diagnostic purposes only. Note that 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).

Response/Messages

```
CRCE=0000000 AE=0000000 SF=0000000 RC=0000000 TGB=0000000 TG=0000000
```

Where:

CRCE	This 32 bit count specifies the number of aligned frames discarded because of a CRC error.
AE	This 32 bit count specifies the number of frames that are both misaligned, i.e., CRS deasserts on a non-octet boundary and contain a CRC error.
SF	This 32 bit count specifies the number of received frames that are shorter than the minimum length.
RC	This 32 bit count specifies the number of collisions detected during frame reception.
TGB	This 32 bit count specifies the number of good and bad frames received.
TG	This 32 bit count specifies the number of good frames received.

The Short Frame counter has priority over CRC, Alignment, and RX Collision counters. Only one of these counters is incremented per frame. For example, if a received frame is both short and collided, only the Short Frame counter is incremented.

Time Domain Reflectometry - TDR

LANC TDR

Command Input

```
197-Diag>LANC TDR
```

Description

Verifies that Time Domain Reflectometry (TDR) can be executed, and that an error free completion status is returned.

This test activates the TDR feature of the 82596, which is a mechanism to detect open or shorts on the link and their distance from the diagnosing station. The maximum length of the TDR frame is 2048 bits. If the 82596 senses collision while transmitting the TDR frame, it transmits the jam pattern and stops the transmission. The 82596 then triggers an internal timer (STC); the timer is reset at the beginning of transmission and reset if CRS is returned. 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:

1. 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.
2. 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.
3. The Collision Detect signal goes active before the counter expires. This means that the link is not properly terminated (an open).
4. 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.

LANC TDR

The distance to the cable failure can be calculated as follows:

Distance = TIME X (Vs / (2 X Fs))

where:

Vs = wave propagation speed on the link (M/s)

Fs = serial clock frequency (Hz)

Accuracy is plus/minus Vs / (2 X Fs)

Note that this test does not execute when the **LANC** test group is executed (**LANC** with no arguments). This test is supplied only for diagnostic purposes. It requires a properly set up Ethernet network (cable).

Response/Messages

After the command has been issued, the following line is printed:

```
LANC     TDR: Time Domain Reflectometry.............. Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
LANC     TDR: Time Domain Reflectometry.............. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
LANC     TDR: Time Domain Reflectometry.............. Running ---> FAILED

(error message)
```

Here, (error message) is one of the following:

This failure is the result of TDR command executing with error status:

```
TDR Command Completion Status Error:
OK-Bit =0
```

LANC TDR

Once the TDR command has completed successfully, the LINK-OK bit is checked in the TDR command packet. This failure is the result of the LINK-OK bit being false (problem with link). The various diagnostic parameters also are displayed with an error message:

```
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
```

Additional Error Messages

The following error messages, and descriptions for each, may apply to any or all of the tests within the **LANC** test group.

If the amount memory found during the diagnostics subsystem initialization does not meet the amount of memory needed by the **LANC** test group:

```
Test Initialization Error:
Not Enough Memory, Need =00010000, Actual =000087F0
```

If the control memory address specified by the **LANC** test group configuration parameters is not 16 byte aligned:

```
Test Initialization Error:
Control Memory Address Not 16 Byte Aligned =0000E008
```

The ISCP (Intermediate System Configuration Pointer) indicates the location of the SCB (System Control Block). The CPU loads the SCB address into the ISCP and asserts CA (Channel Attention). 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. This failure is the result of the busy byte in the ISCP not becoming clear after one tenth of a second from the issue of the channel attention:

```
LANC Initialization Error:
SCB Read Failure (Channel Attention Signal)
```

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. This failure is the result of the 82596 command queue not accepting the command:

```
LANC Initialization Error:
LANC Command Unit Command Acceptance Time-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. This failure is the result of the command timing out from the issue of the command. The time out value is set to one second:

```
LANC Initialization Error:
LANC Command Unit Interrupt Acknowledge Command Completion Time-Out
```

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. This failure is the result of any bus error condition:

```
LANC Error Status Register (DMA Bits) Not Clear =02
```

Prior to issuing a command to the Command Unit of the 82596, the command execution function verifies that the command unit is idle. This failure is the result of the command unit not being in the idle state:

```
LANC Command Unit Not Idle (Busy)
```

Prior to issuing a command to the Receive Unit of the 82596, the receive command execution function verifies that the receive unit is idle. This failure is the result of the receive unit not being in the idle state:

```
LANC Receive Unit Not Idle (Busy)
```

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. This failure is the result of the command unit having pending interrupt requests:

```
LANC Command Unit Interrupt(s) Pending
```

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. This failure is the result of the one second time-out expiring:

```
LANC Command Unit Command Acceptance Time-Out
```

Once a command has been completed by the 82596, the command execution functions 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. This failure is the result of the one second time-out expiring:

```
LANC Command Unit Interrupt Status Time-Out
```

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. This failure is the result of the one second time-out expiring:

```
LANC Command Unit Interrupt Acknowledge Command Completion Time-Out
```

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. This failure is the result of the one second time-out expiring:

```
LANC Receive Unit Command Acceptance Time-Out
```

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. This failure is the result of the one second time-out expiring:

```
LANC Receive Unit Interrupt Acknowledge Command Completion Time-Out
```

Upon completion of the Configure with Operating Parameters command, the command completion status is verified that it was successful. This failure is the result of an error condition in the completion of the command:

```
Configure Command Completion Status Error:
OK-Bit =0, ABORT-Bit =0
```

Upon completion of the Individual Address Setup command, the command completion status is verified that it was successful. This failure is the result of an error condition in the completion of the command:

```
Individual Address Setup Command Completion Status Error:
OK-Bit =0, ABORT-Bit =0
```

SCSI I/O Processor (NCR) Tests

These sections describe the individual NCR 53C710 (SCSI I/O Processor) tests.

Entering **NCR** without parameters executes all **NCR** tests that do not require operator input.

To run an individual test, append that test name to the **NCR** command.

The individual tests are listed in the next table and described in alphabetical order on the following pages.

The error message(s) displayed following the explanation of an **NCR** test pertain to the test being discussed.

Table 3-15. NCR Test Group

Mnemonic	Description
DMA	DMA SCRIPTS Utility
ACC1	Device Access
ACC2	Register Access
SFIFO	SCSI FIFO
DFIFO	DMA FIFO
LPBK	Loopback
SCRIPTS	SCRIPTs Processor
IRQ	Interrupts

You can configure some parameters that these tests use with the **CF** command. Use **CF NCR** to change the "Test Memory Base Address" parameter for the **DMA, IRQ** and **SCRIPTS** tests and the "Memory Move Addresses and Byte Count" parameter for the **DMA** and **SCRIPTS** tests.

NCR Configuration Parameters CF NCR

Command Input

```
197-Diag>cf ncr
NCR Configuration Data:
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 ?
Memory Move BURST Enable [Y/N] =Y ?
197-Diag>
```

Description

User configurable test parameters are available for the **NCR** test group. Refer to Chapter 2 for information on using the **CF** command to set configuration parameters.

The **NCR** test parameters are listed and described below.

```
Test Memory Base Address Override [Y/N] =N ?
```

This parameter allows the user to specify an alternative test memory base.

```
Test Memory Base Address =00000000 ?
```

This parameter contains an alternative test memory base. It is used only if Test Memory Base Override is set to Y (yes).

```
Diagnostic Base Address =00000000 (READ ONLY) ?
```

This parameter points to the memory-mapped SIOP registers. It is always equal to the Test Memory Base Address, regardless of whether the default Test Memory Base Address was used or overridden.

```
SCRIPTs Buffer Base Address =00000000 (READ ONLY) ?
```

This parameter is the pointer to the SCRIPTS buffer in which diagnostic SCRIPTS are created dynamically based on the current configuration values. A SCRIPT consists of contiguous words in memory that define an operation to be performed and its operands. Since the **DMA** and **SCRIPTS** tests can use configuration data set via **CF**, they use this temporary buffer to build their commands. This buffer immediately follows the memory-mapped SIOP registers, which begin at the Diagnostic Base Address.

```
Memory Move Address (Source) =00000000 ?
```

This parameter is the source address for the memory move operation. A block of memory is copied from the Memory Move Source Address to the Memory Move Destination Address. Both the source and the destination addresses must start with the same address alignment (A(1-0) must be the same).

```
Memory Move Address (Destination) =00000000 ?
```

This parameter is the destination address for the memory move operation. A block of memory is copied from the Memory Move Source Address to the Memory Move Destination Address. Both the source and the destination addresses must start with the same address alignment (A(1-0) must be the same).

```
Memory Move Byte Count =00002000 ?
```

This parameter contains the number of bytes to be copied from the source address to the destination address. The default is 0x2000, or decimal 8192 (8K). The maximum value of this parameter is 0xFFFFFF, or decimal 16777215 (16MB), since it is contained in a 24-bit register on the NCR 53C710. Setting this value to zero can cause errors.

```
Memory Move BURST Enable [Y/N] =Y ?
```

This parameter enables burst mode operation between the NCR 53C710 and local memory. This parameter is only used by the **SCRIPTS** test and the **DMA** utility. If this parameter is set to N (no), then two 32-bit words are transferred per bus grant. The default is enabled.

Device Access - ACC1 **NCR ACC1**

Command Input

```
197-Diag>NCR ACC1
```

Description

Tests the 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 then the NCR device is reset; otherwise the device is left in the test state.

If a device register read fails, a device access error occurs. Next, a scratch register is cleared in order to hold the walking bit mask. If the scratch register is then not read back as zero, a “scratch register” error is issued. Finally, the walking bit patterns are written and then read. If the written and the read values do not match, a device access error occurs.

Response/Messages

After the command has been issued, the following line is printed:

```
NCR      ACC1: Device Access........................ Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
NCR      ACC1: Device Access........................ Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
NCR      ACC1: Device Access........................ Running ---> FAILED

NCR/ACC1 Test Failure Data:
(error message)
```

Here, (error message) is one of the following:

```
SCRATCH Register is not initially cleared

Device Access Error:
Address =________, Expected =________, Actual =________

Device Access Error:
Address =________, Expected =________, Actual =________

Access Fault Information:
  Address________
  Data________
  Access Size__
  Access Type_
  Address Space Code_
  Vector Number___

Unsolicited Exception:
  Instruction Pointer________
  Vector Number___
  Status Register____
  Interrupt Level_
```

Notes

1. **All error message data is displayed as hexadecimal values.**
2. **The Unsolicited Exception information is only displayed if the exception was not an Access Fault.**
3. **Access Size is displayed in bytes.**
4. **Access Type is: 0 (write), or 1 (read).**
5. **The Address Space Code is: 1 (user data), 2 (user program), 5 (supervisor data), 6 (supervisor program), or 7 (MPU space).**
6. **The Vector Number will represent the fact that a Bus Error occurred.**

Register Access - ACC2 **NCR ACC2**

Command Input

```
197-Diag>NCR ACC2
```

Description

Tests the ability to access the basic 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 with a walking 1 through a field of zeros. If no errors are detected, then the NCR device is reset; otherwise the device is left in the test state.

The status registers tested are ISTAT, SSTAT0, SSTAT1, and SSTAT2. The writable registers checked are SIEN, SDID, SODL, SXFER, SCID, DSA, TEMP, DCMD/DBC, and DNAD. If any of the status registers are not zero, an appropriate error message will be displayed. If any data pattern written to the writable messages does not equal what is read back, the corresponding error message will be displayed. If a bus error occurs, a register access error will be displayed.

Response/Messages

After the command has been issued, the following line is printed:

```
NCR      ACC2: Register Access...................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
NCR      ACC2: Register Access...................... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
NCR      ACC2: Register Access...................... Running ---> FAILED

NCR/ACC1 Test Failure Data:
(error message)
```

Here, (error message) is one of the following:

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

NCR ACC2


```
Access Fault Information:
  Address________
  Data________
  Access Size__
  Access Type_
  Address Space Code_
  Vector Number___

Unsolicited Exception:
  Instruction Pointer________
  Vector Number___
  Status Register____
  Interrupt Level_
```

Notes

1. **All error message data is displayed as hexadecimal values.**
2. **The Unsolicited Exception information is only displayed if the exception was not an Access Fault.**
3. **Access Size is displayed in bytes.**
4. **Access Type is: 0 (write), or 1 (read).**
5. **The address space code is: 1 (user data), 2 (user program), 5 (supervisor data), 6 (supervisor program), or 7 (MPU space).**
6. **The Vector Number will represent the fact that a Bus Error occurred.**

DMA FIFO - DFIFO

NCR DFIFO

Command Input

```
197-Diag>NCR DFIFO
```

Description

Tests the ability to write data into the DMA FIFO and retrieve it in the same order as written. The DMA FIFO is checked for an empty condition following a software reset; then the FBL2 bit is set and verified. The FIFO is then filled with 16 bytes of data in each of the four byte lanes, verifying that before the first write to the FIFO or the byte lanes occurs, the FIFO and byte lanes are marked as empty. After the first write to the FIFO and/or byte lanes, the FIFO and the appropriate byte lanes are verified to be non-empty. After all sixteen byte lanes have been filled, the byte lanes and the FIFO are verified to be full.The process reverses itself by reading the FIFO; in addition to verifying the correct empty/non-empty/full state of the FIFO and byte lanes, the data read from the FIFO is verified for accuracy. If no errors are detected, then the NCR device is reset; otherwise the device is left in the test state. The “byte control error” message described below refers to the fact that a byte lane is not in the proper empty/non-empty/full state; all other error messages are self-explanatory.

Response/Messages

After the command has been issued, the following line is printed:

```
NCR      DFIFO: DMA FIFO............................. Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
NCR      DFIFO: DMA FIFO............................. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
NCR      DFIFO: DMA FIFO............................. Running ---> FAILED

NCR/DFIFO Test Failure Data:
(error message)
```

NCR DFIFO

Here, (error message) is one of the following:


```
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 _
```

DMA SCRIPTs Utility - DMA

NCR DMA

3

Command Input

```
197-Diag>NCR DMA
```

Description

This is a utility which utilizes the **SCRIPTS** capability of the NCR 53C710 to perform memory move operations as specified in the NCR configuration parameters.It initializes the test structures and makes use of the diagnostic registers for testing.

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 **CF NCR** command. If a parameter is changed, the only check for validity is the “Byte Count” during test structures initialization. This utility is essentially a subset of the **SCRIPTS** diagnostics, except that the source buffer is not initialized prior to the operation.

The “Memory Move” SCRIPT copies the specified number of bytes from the source address to the destination address. The data is verified after the move operation, and if incorrect, the destination address, expected data, and received data are printed.

Response/Messages

After the command has been issued, the following line is printed:

```
NCR      DMA: NCR 53C710 SCRIPTs DMA Utility....... Running --->
```

If all parts of the utilitt are completed correctly, then the utility passes.

```
NCR      DMA: NCR 53C710 SCRIPTs DMA Utility....... Running ---> PASSED
```

If any part of the utility fails, then the display appears as follows:

```
NCR      DMA: NCR 53C710 SCRIPTs DMA Utility....... Running ---> FAILED

NCR/DMA Test Failure Data:
(error message)
```

NCR DMA

Here, (error message) is one of the following:

```
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 =__

Test Timeout during: Memory Move SCRIPTs Test

Address =________, Expected =__, Actual =__

"SIR" not detected during: Memory Move SCRIPTs Test
Address =________, Expected =__, Actual =__
```

3

Interrupts - IRQ

NCR IRQ

Command Input

```
197-Diag>NCR IRQ
```

Description

Verifies that level 0 interrupts will not generate an interrupt if interrupts are not enabled, but will set the appropriate status. The test then verifies that all interrupts (1 through 7) can be generated and received and that the appropriate status is set. The test begins by attempting to generate a SCSI Gross Error (SGE). This error should set the SIP (SCSI Interrupt Pending) bit in the ISTAT register. The SGE and SIP bits are then cleared. Next,the onboard SCSI Interrupt Control Register is verified to be clear. Then SGE interrupts are enabled, and interrupts are enabled in the onboard SCSI interrupt controller. An SGE interrupt is generated; then interrupts are disabled in the SIEN and another SGE interrupt is generated; the SIP bits should be set and cleared respectively. Then interrupt levels 1-7 are tested; an SGE is generated, and the onboard interrupt controller is checked to ensure that it indicates an interrupt at the proper level. Each interrupt is cleared. At the end of the test, the SSTAT0 and onboard interrupt status registers are cleared, and SCSI interrupts are disabled via SIEN.

Response/Messages

After the command has been issued, the following line is printed:

```
NCR      IRQ: NCR 53C710 Interrupts.................. Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
NCR      IRQ: NCR 53C710 Interrupts.................. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
NCR      IRQ: NCR 53C710 Interrupts.................. Running ---> FAILED

NCR/IRQ Test Failure Data:
(error message)
```

NCR IRQ

Here, (error message) is one of the following:

```
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 =__
```

NCR IRQ

```
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 =__
```

NCR IRQ


```
Access Fault Information:
  Address________
  Data________
  Access Size__
  Access Type_
  Address Space Code_
  Vector Number___

Unsolicited Exception:
  Instruction Pointer________
  Vector Number___
  Status Register____
  Interrupt Level_
```

3

Loopback - LPBK

NCR LPBK

Command Input

```
197-Diag>NCR LPBK
```

Description

Checks the Input and Output Data Latches and performs a selection, with the 53C710 executing initiator instructions and the host CPU implementing the target role by asserting and polling the appropriate SCSI signals. The SCSI bus data lines are verified, and the DNAD/DBC register is incremented and decremented in both one- and four-byte increments and verified for correct data. The data is incremented and decremented via setting the ADCK and BBCK bits (respectively) in the CTEST5 register. When these bits are set by the test software, the DNAD/DBC register is incremented/decremented. Then the ADCK or BBCK bit should automatically clear itself.

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.

If no errors are detected, then the NCR device is reset; otherwise the device is left in the test state.

Response/Messages

After the command has been issued, the following line is printed:

```
NCR      LPBK: Loopback.............................. Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
NCR      LPBK: Loopback.............................. Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
NCR      LPBK: Loopback.............................. Running ---> FAILED

NCR/LPBK Test Failure Data:
(error message)
```

NCR LPBK

Here, (error message) is one of the following:

```
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 Processor - SCRIPTS

NCR SCRIPTS

Command Input

```
197-Diag>NCR SCRIPTS
```

Description

Initializes the test structures and makes use of the diagnostic registers for testing, as follows:

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

Sets SCSI outputs in high impedance state, disables interrupts using MIEN, and sets NCR device for Single Step Mode.

The address of a simple “INTERRUPT instruction” SCRIPT is loaded into the DMA SCRIPTs Pointer register. The SCRIPTs processor is started by setting the “STD” bit in the DMA Control Register.

Single Step is checked by verifying that ONLY the first instruction was executed and that the correct status bits are set. Single Step Mode is then turned off and the SCRIPTs processor started again. The “INTERRUPT instruction” should then be executed and a check for the correct status bits set is made.

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.

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 “cnfg” command. If a parameter is changed, the only check for validity is the “Byte Count” during test structures initialization.

NCR SCRIPTS

The “Memory Move” SCRIPT copies the specified number of bytes from the source address to the destination address. The data is verified, and if incorrect, the destination address, expected data, and received data are printed.

The SIR (SCRIPTS interrupt instruction received) and SSI (single step interrupt) bits are contained in DIEN (DMA Interrupt Enable), and are checked for validity. The DIP (DMA interrupt pending, found in ISTAT) bit is also checked for validity.

Response/Messages

After the command has been issued, the following line is printed:

```
NCR      SCRIPTS: NCR 53C710 SCRIPTs Processor....... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
NCR      SCRIPTS: NCR 53C710 SCRIPTs Processor....... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
NCR      SCRIPTS: NCR 53C710 SCRIPTs Processor....... Running ---> FAILED

NCR/SCRIPTS Test Failure Data:
(error message)
```

NCR SCRIPTS

Here, (error message) is one of the following:

```
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 =__
```

NCR SCRIPTS


```
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 "True", and Compare = False; Jump not taken

Test Timeout during: Memory Move SCRIPTs Test

Jump if "False", and Compare = True; Jump taken

Address =________, Expected =__, Actual =__

"SIR" not detected during: Memory Move SCRIPTs Test
Address =________, Expected =__, Actual =__
```

SCSI FIFO - SFIFO

NCR SFIFO

3

Command Input

```
197-Diag>NCR SFIFO
```

Description

Tests the ability to write data into the SCSI FIFO and retrieve it in the same order as written. The SCSI FIFO is checked for an empty condition following a software reset, then the SFWR bit in CTEST4 (SCSI FIFO write enable) is set and verified. The FIFO is then filled with 8 bytes of data, verifying the byte count in the FIFO with each write. Next, the SFWR bit is cleared and the FIFO read verifying the byte count and the data with each read. If no errors are detected, then the NCR device is reset; otherwise the device is left in the test state.

Response/Messages

After the command has been issued, the following line is printed:

```
NCR      SFIFO: SCSI FIFO........................... Running --->
```

If all parts of the test are completed correctly, then the test passes.

```
NCR      SFIFO: SCSI FIFO........................... Running ---> PASSED
```

If any part of the test fails, then the display appears as follows:

```
NCR      SFIFO: SCSI FIFO........................... Running ---> FAILED

NCR/SFIFO Test Failure Data:
(error message)
```

NCR SFIFO

Here, (error message) is one of the following:

```
SCSI FIFO is not initially empty

SCSI FIFO writes not enabled

SCSI FIFO Count Error:
Address =________, Expected =__, Actual =__

SCSI FIFO Error:
Address =________, Expected =__, Actual =__
```

197BUG GENERAL INFORMATION 4

Introduction

The user can use 197Bug to configure certain parameters contained in the Non-Volatile RAM (NVRAM), also known as Battery Backup RAM (BBRAM). Use the **CNFG** command to change operating parameters of the hardware that are contained in the NVRAM board information block. Use the **ENV** command to change configurable parameters in NVRAM.

The **CNFG** and **ENV** commands are both described in the *MVME197BUG 197Bug Debugging Package User's Manual*. Refer to that manual for general information about their use and capabilities.

This chapter presents information about **CNFG** and **ENV** that is specific to 197Bug and describes the parameters that can be configured with the **ENV** command.

Configure Board Information Block - CNFG

This command is used to display and configure the board information block, which is resident within the Non-Volatile RAM (NVRAM). The board information block contains various elements detailing specific operation parameters of the hardware. The **CNFG** command does *not* describe the elements and their use. The board information block contents are checksummed for validation purposes. This checksum is the last element of the block.

```
197-Bug>cnfg
Board (PWB) Serial Number   = "0000000xxxxx"
Board Identifier            = "MVME197LE       "
Artwork (PWA) Identifier    = "01-W3869B01A    "
MPU Clock Speed             = "5000"
Ethernet Address            = 08003E21EG7A
Local SCSI Identifier       = "07"
Optional Board 1 Artwork (PWA) Identifier     = "0           "
Optional Board 1 (PWB) Serial Number          = "0           "
Optional Board 2 Artwork (PWA) Identifier     = "0           "
Optional Board 2 (PWB) Serial Number          = "0           "
197-Bug>
```

Note that the parameters that are quoted are left-justified character (ASCII) strings padded with space characters, and the quotes (“) are displayed to indicate the size of the string. Parameters that are not quoted are considered data strings, and data strings are right-justified. The data strings are padded with zeros if the length is not met.

Refer to the board specific MVME197 User’s Manual for the actual location and other information about the board information block. Refer to the *MVME197BUG 197Bug Debugging Package User’s Manual* for a description of **CNFG** and examples.

Set Environment to Bug/Operating System - ENV

The **ENV** command allows you to view and/or configure all debugger operational parameters that are stored in the Non-Volatile RAM (NVRAM).

Refer to the *MVME197BUG 197Bug Debugging Package User’s Manual* for a description of the use of **ENV**. Listed and described below are the parameters that you can configure using **ENV**.

Configuring Parameters with ENV

The parameters that can be configured using **ENV** are:

```
Bug or System environment [B/S] = S?
```

B Bug is the mode where no system type of support is displayed. However, system-related items are still available.

S System is the standard mode of operation, and is the one defaulted to if NVRAM should fail. This mode is defined in Appendix A (*MVME197Bug System Mode Operation*) of the *MVME197BUG 197Bug Debugging Package User’s Manual*. (Default)

```
Field Service Menu Enable [Y/N] = Y?
```

Y Display the field service menu. (Default)

N Do not display the field service menu.

`Remote Start Method Switch [G/M/B/N] = B?`

The Remote Start Method Switch is used when the MVME197 is cross-loaded from another VME-based CPU, to start execution of the cross-loaded program.

G Use the Global Control and Status Register (GCSR) in the VMEchip2 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 methods to pass and start execution of cross-loaded program. (Default)

N Do not use any Remote Start Method.

`Probe System for Supported Disk/Tape Controllers [Y/N] = Y?`

Y Accesses will be made to the VMEbus to determine the presence of supported controllers. (Default)

N Accesses will not be made to 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 the bug command monitor. (Default)

`Local SCSI Bus Reset on Debugger Setup [Y/N] = Y?`

Y The local SCSI bus is reset on the debugger setup.

N The local SCSI bus is not reset on the debugger setup.

`Local SCSI Bus Negotiations Type [A/S/N] = A?`

A Use asynchronous negotiations on the local SCSI bus.

Y Use synchronous negotiations on the local SCSI bus.

N (None). Do not precede the SCSI data transfer with a type negotiation. Do all data transfers in asynchronous mode.

`Ignore CFGA Block on a Hard Disk Boot [Y/N] = Y?`

Y Enable the ignorance of the Configuration Area (CFGA) Block (hard disk only).

N Do not enable the ignorance of the Configuration Area (CFGA) Block.

`Auto Boot Enable [Y/N] = N?`

Y The auto boot function is enabled.

N The auto boot function is disabled. (Default)

`Auto Boot at power-up only [Y/N] = Y?`

Y Auto Boot is attempted at power up reset only. (Default)

N Auto Boot is attempted at any reset.

`Auto Boot Controller LUN = 00?`

Refer to Appendix E (*Disk/Tape Controller Data*) of the *MVME197BUG 197Bug Debugging Package User's Manual* for a listing of disk/tape controller modules currently supported by the Bug. The default for this parameter is $0.

`Auto Boot Device LUN = 00?`

Refer to Appendix E (*Disk/Tape Controller Data*) of the *MVME197BUG 197Bug Debugging Package User's Manual* for a listing of disk/tape devices currently supported by the Bug. The default for this parameter is $0.

`Auto Boot Abort Delay = 15?`

This is the time in seconds that the Auto Boot sequence will delay before starting the boot. The purpose for the delay is to allow you the option of stopping the boot by use of the Break key. The time value is from 0 through 255 seconds.

`Auto Boot Default String [NULL for an empty string] = <none>`

You may specify a string (filename) which is passed on to the code being booted. The maximum length of this string is 16 characters. The default for this parameter is the null string.

`ROM Boot Enable [Y/N] = N?`

Y	The ROM Boot function is enabled.
N	The ROM Boot function is disabled. (Default)

`ROM Boot at power-up only [Y/N] = Y?`

Y	ROM Boot is attempted at power up only. (Default)
N	ROM Boot is attempted at any reset.

`ROM Boot Enable search of VMEbus [Y/N] = N?`

Y	VMEbus address space will be searched for a ROM Boot module in addition to the usual areas of memory.
N	VMEbus address space will not be accessed by ROM Boot. This is the default mode.

`ROM Boot Abort Delay = 0?`

This is the time in seconds that the ROM Boot sequence will delay before starting the boot. The purpose for the delay is to allow you the option of stopping the boot by use of the Break key. The time value is from 0 through 255 seconds.

`ROM Boot Direct Starting Address = FF800000?`

This is the first location tested when the Bug searches for a ROM Boot Module. This is the start of the Flash memory address space. Default is $FF800000.

`ROM Boot Direct Ending Address = FFBFFFFC?`

This is the last location tested when the Bug searches for a ROM Boot Module. This the end of the Flash memory address space. Default is $FFBFFFFC.

`Network Auto Boot Enable [Y/N] = N?`

Y The Network Auto Boot function is enabled.

N The Network Auto Boot function is disabled.

`Network Auto Boot at power-up only [Y/N] = Y?`

Y The Network Auto Boot is attempted at power-up reset only (if enabled).

N The Network Auto Boot is attempted at any reset (if enabled).

`Network Auto Boot Controller LUN = 00?`

The Logical Unit Number (LUN) of a disk/tape controller module currently supported by the Bug. Refer to Appendix E (*Disk/Tape Controller Data*) of the *MVME197BUG 197Bug Debugging Package User's Manual* for a listing of disk/tape controller modules. Default is $0.

`Network Auto Boot Device LUN = 00?`

The Logical Unit Number (LUN) of a disk/tape device currently supported by the Bug. Refer to Appendix E (*Disk/Tape Controller Data*) of the *MVME197BUG 197Bug Debugging Package User's Manual* for a listing of disk/tape controller modules. Default is $0.

`Network Auto Boot Delay = 5?`

This is the time in seconds that the Network Boot sequence will delay before starting the boot. The purpose of the delay is to allow the user the option of stopping the boot by use of the Break key. The time value is from 0 through 255 seconds.

`Network Auto Boot Configuration Parameter = Pointer (NVRAM) = 00000000?`

This is the address where the network interface configuration parameters are to be saved/retained in NVRAM: these parameters are the necessary parameters to perform an unattended network boot.

`Memory Search Starting Address = 00000000?`

This is 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 the debugger work page, modulo $10000 (64KB). In a multi-197 environment, each MVME197 board could be set to start its work page at a unique address so as to allow multiple debuggers to operate simultaneously. The default Memory Search Starting Address is $00000000.

`Memory Search Ending Address = 02000000?`

This is the top limit of the Bug's search for a work page.

`Memory Search Increment Size = 00010000?`

This must be a multiple of the debugger work page, modulo $10000 (64KB).

`Memory Search Delay Enable [Y/N] = N?`

Y There will be a delay before the Bug begins its search for a work page. This delay could be used to allow time for some other MVME197 in the system to configure its address decoders.

N There will be no delay before the Bug begins its search for a work page. This is the default selection.

`Memory Search Delay Address = FFFFD00F?`

The default address is $FFFFD00F. This is the MVME197 GCSR (global control and status register) GPCSR5 as accessed through VMEbus A16 space and assumes the MVME197 GRPAD (group address) and BDAD (board address within group) switches are set to “on”.

This byte-wide value is initialized to $FF by MVME197 hardware after a System or Power-on Reset. In a multi-197 environment, where the work pages of several Bugs are to reside in the memory of the primary (first) MVME197, the non-primary CPUs will wait for the data at the Memory

Search Delay Address to be set to $00, $01, or $02 (refer to the *Memory Requirements* section in the *197Bug General Information* chapter of the *MVME197BUG 197Bug Debugging Package User's Manual* for the definition of these values) before attempting to locate their work page in the memory of the primary CPU.

`Memory Size Enable [Y/N] = Y?`

Y Memory will be sized for Self Test diagnostics. This is the default.

N Memory will not be sized for Self Test diagnostics.

`Memory Size Starting Address = 00000000?`

The default Starting Address is $0.

`Memory Size Ending Address = 02000000?`

The default Ending Address is the calculated size of the local memory.

`Base Address of Local Memory = 00000000?`

This is the beginning address of the Local Memory. It must be a multiple of the Local Memory board size, starting with 0. The Bug will set up the hardware address decoders so that the Local Memory resides as one contiguous block at this address. Default is $0.

`Size of Local Memory = 02000000?`

The default is the calculated size of the local memory.

Configuring the VMEbus Interface

ENV asks the following series of questions to set up the VMEbus interface for the MVME197 series modules.

You should have a working knowledge of the VMEchip2 as given in the *MVME197LE, MVME197DP, and MVME197SP Single Board Computers Programmer's Reference Guide* in order to perform this configuration.

The slave address decoders are used to allow another VMEbus master to access a local resource of the MVME197. There are two slave address decoders set.

They are set up as follows.

`Slave Enable #1 [Y/N] = Y?`

Y   Yes, set up and enable the Slave Address Decoder #1. (Default)

N   Do not set up and enable the Slave Address Decoder #1.

`Slave Starting Address #1 = 00000000?`

This is the base address of the local resource that is accessible by the VMEbus. (Default is the base of the local memory, $0).

`Slave Ending Address #1 = 01FFFFFF?`

This is the ending address of the local resource that is accessible by the VMEbus. (Default is the end of calculated memory).

`Slave Address Translation Address #1 = 00000000?`

This register will allow the VMEbus address and the local address to be different. The value in this register is the base address of the local resource that is associated with the starting and ending address selection from the previous questions. (Default is 0).

`Slave Address Translation Select #1 = 00000000?`

This register defines which bits of the address are significant. A logical one "1" indicates significant address bits, logical zero "0" is non-significant. (Default is 0).

`Slave Control #1 = 01FF?`

This defines the access restriction for the address space defined with this slave address decoder. The default is $01FF.

`Slave Enable #2 [Y/N] = N?`

Y   Yes, set up and enable the Slave Address Decoder #2.

N   Do not set up and enable the Slave Address Decoder #2.(Default)

`Slave Starting Address #2 = 00000000?`

This is the base address of the local resource that is accessible by the VMEbus.

`Slave Ending Address #2 = 00000000?`

This is the ending address of the local resource that is accessible by the VMEbus.

`Slave Address Translation Address #2 = 00000000?`

This register will allow the VMEbus address and the local address to be different. The value in this register is the base address of the local resource that is associated with the starting and ending address selection from the previous questions. (Default is 0).

`Slave Address Translation Select #2 = 00000000?`

This register defines which bits of the address are significant. A logical one "1" indicates significant address bits, logical zero "0" is non-significant. (Default is 0).

`Slave Control #2 = 0000?`

This defines the access restriction for the address space defined with this slave address decoder.

`Master Enable #1 [Y/N] = Y?`

Y	Yes, set up and enable the Master Address Decoder #1. (Default)
N	Do not set up and enable the Master Address Decoder #1.

`Master Starting Address #1 = 02000000?`

This is the base address of the VMEbus resource that is accessible from the local peripheral bus. (Default is the end of calculated local memory).

`Master Ending Address #1 = EFFFFFFF?`

This is the ending address of the VMEbus resource that is accessible from the local peripheral bus.

`Master Control #1 = 0D?`

This defines the access characteristics for the address space defined with this master address decoder.

`Master Enable #2 [Y/N] = Y?`

Y Yes, set up and enable the Master Address Decoder #2. (This is the default if the board contains version 1 of the VMEchip).

N Do not set up and enable the Master Address Decoder #2. (This is the default for boards containing version 2 of the VMEchip).

`Master Starting Address #2 = FF000000?`

This is the base address of the VMEbus resource that is accessible from the local peripheral bus. If enabled, default is $FF000000, otherwise $00000000.

`Master Ending Address #2 = FF7FFFFF?`

This is the ending address of the VMEbus resource that is accessible from the local peripheral bus. If enabled, default is $FF7FFFFF, otherwise $00000000.

`Master Control #2 = 0D?`

This defines 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] = N?`

Y Yes, set up and enable the Master Address Decoder #3. (This is the default if the board contains less than 16MB of calculated RAM).

N Do not set up and enable the Master Address Decoder #3. (This is the default for boards containing at least 16MB of calculated RAM).

`Master Starting Address #3 = 00000000?`

This is the base address of the VMEbus resource that is accessible from the local peripheral bus. (If enabled, the value is calculated as 1 less than the calculated size of memory. If not enabled, default is $00000000).

`Master Ending Address #3   = 00000000?`

This is the ending address of the VMEbus resource that is accessible from the local peripheral bus. (If enabled, the default is $00FFFFFF, otherwise $00000000).

`Master Control #3 = 00?`

This defines 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	Yes, set up and enable the Master Address Decoder #4.
N	Do not set up and enable the Master Address Decoder #4. (Default)

`Master Starting Address #4 = 00000000?`

This is the base address of the VMEbus resource that is accessible from the local peripheral bus. (Default is $0).

`Master Ending Address #4 = 00000000?`

This is the ending address of the VMEbus resource that is accessible from the local peripheral bus. (Default is $0).

`Master Address Translation Address #4 = 00000000?`

This register will allow the VMEbus address and the local address to be different. The value in this register is the base address of VMEbus resource that is associated with the starting and ending address selection from the previous questions. (Default is 0).

`Master Address Translation Select #4 = 00000000?`

This register defines which bits of the address are significant. A logical one "1" indicates significant address bits, logical zero "0" is non-significant. (Default is 0).

`Master Control #4 = 00?`

This defines the access characteristics for the address space defined with this master address decoder. The default is $00.

```
Short I/O (VMEbus A16) Enable [Y/N] = Y?
```

Y Yes, enable the Short I/O Address Decoder. (Default)

N Do not enable the Master Address Decoder.

```
Short I/O (VMEbus A16) Control = 01?
```

This defines the access characteristics for the address space defined with the Short I/O address decoder. The default is $01.

```
F-Page (VMEbus A24) Enable [Y/N] = Y?
```

Y Yes, enable the F-Page Address Decoder. (Default)

N Do not enable the F-Page Address Decoder.

```
F-Page (VMEbus A24) Control = 02?
```

This defines the access characteristics for the address space defined with the F-Page address decoder. The default is $02.

```
ROM Speed Bank A Code = 03?
ROM Speed Bank B Code = 03?
```

These parameters are used to set up the ROM speed. (Default $03 = 165 nanoseconds).

```
PCC2 Vector Base = 05?
VMEC2 Vector Base #1 = 06?
VMEC2 Vector Base #2 = 07?
```

These parameters are the base interrupt vector for the component specified. (Default: PCCchip2 = $05, VMEchip2 Vector 1 = $06, VMEchip2 Vector 2 = $07).

```
VMEC2 GCSR Group Base Address = D0?
```

This parameter specifies the group address ($FFFFXX00) in Short I/O for this board. (Default = $D0).

```
VMEC2 GCSR Board Base Address = 00?
```

This parameter specifies the base address ($FFFFCEXX) in Short I/O for this board. (Default = $00).

```
VMEbus Global Time Out Code = 01?
```

This parameter controls the VMEbus timeout when this board is the systems controller. (Default $01 = 64 microseconds).

```
Local Peripheral Bus Time Out Code = 01?
```

This parameter controls the local peripheral bus timeout. (Default $00 =64 microseconds).

```
VMEbus Access Time Out Code = 02?
```

This parameter controls the local peripheral bus to VMEbus access timeout. (Default $02 = 32 milliseconds).

MVME197 FAMILY DRAM MEMORY LINE MAPPING

A

This appendix provides information which will be useful when translating RAM and RAMCD(I) test failure data to physical devices and pathways. It also provides information on line bit numbering and ECC sub-system bit numbering imposed by the ECDM test structure.

For RAM error data translation, Figure A-1 correlates address offsets within a single cache line ($20 bytes) to a specific ECDM and further to a particular data segment (RA-RD) of the ECDM. Each row of the table represents 64 bits of the cache line which correlate to a set of 16 DRAMs each of which contain 4 data bits. The bits of each DRAM are shared among the four ECDM devices in the sub-system so that if an entire device fails, only a single-bit error is seen by each ECDM (ECC sub- system). The shaded boxes on the left side of the table map the lower 5 bits of the failure address to a specific ECDM data segment (RA-RD). The byte offset ($00 - $1F) found in the small shaded boxes in the upper left corner of the squares in the body of the table will map the failure to a particular ECDM. The numbers in the ECDM header boxes map to the ECDM section data channels. Additionally, the ECC line bit numbers on the bottom row of the table correlate to the MC88110 data bus channel numbers. For example, a failure at address 01579430 with “expected” data of $55555555 and “actual” data of $55555575 would map to ECDM1 section RB channel 5 (line bit #101).

The other numbers in the table sections in Figure A-1 may be used to correlate ECDM test failure information to a particular ECDM or to DRAM devices. The numbers in the center of the boxes making up the body of the table indicate the channel numbers describing the data bits of each ECDM ECC sub-system. The numbers at the bottom of the boxes attach bit numbers to all the data bits in an entire line. Similar to the RAM test failure example above, the position of the bits in the table can be correlated to specific ECDM data segments and channels. Board schematics can then be used to match this information to specific devices.

The table also indicates the mapping of check byte data for each ECDM and can be used to correlate check bit failure information to the ECDM associated with the failure. The byte offsets are shown with black backgrounds.

	ECDM 0		ECDM 1		ECDM 2		ECDM 3	
	15 8	7 0	15 8	7 0	15 8	7 0	15 8	7 0
$X0 RD	0 63 56 255 248	1 55 48 247 240	2 63 56 239 232	3 55 48 231 224	4 63 56 223 216	5 55 48 215 208	6 63 56 207 200	7 55 48 199 192
$X8 RC	8 47 40 191 184	9 39 32 183 176	A 47 40 175 168	B 39 32 167 160	C 47 40 159 152	D 39 32 151 144	E 47 40 143 136	F 39 32 135 128
$10 RB	10 31 25 127 120	11 24 16 119 112	12 31 25 111 104	13 24 16 103 96	14 31 25 65 88	15 24 16 87 80	16 31 25 79 72	17 24 16 71 64
$18 RA	18 15 8 63 56	19 7 0 55 48	1A 15 8 47 40	1B 7 0 39 32	1C 15 8 31 24	1D 7 0 23 16	1E 15 8 15 8	1F 7 0 7 0

Double Word Offset (DRAM Set) ECDM Data Group	Byte Offset Within Line ECDM ECC Bits ECDM Test Line Bit #'s

NOTES: Each row represents 16 DRAM devices.

Black Background denotes check byte data position.

Figure A-1. MVME197 Family DRAM Memory Line Mapping

Index

INDEX

INDEX