

# **SCRAMNet<sup>®</sup> Network**

## **User Manual for Windows NT<sup>®</sup> Utilities**

Document No. C-T-MU-NTUTIL##-A-0-A3



# FOREWORD

The information in this document has been carefully checked and is believed to be accurate; however, no responsibility is assumed for inaccuracies. Systran reserves the right to make changes without notice.

Systran makes no warranty of any kind with regard to this printed material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

Copyright 1999 Systran Corporation. All rights reserved.



*Systran*<sup>™</sup> is a trademark of Systran, Corporation

SCRAMNet<sup>®</sup> is a registered trademark of Systran Corporation.

Windows<sup>®</sup> and Windows NT<sup>®</sup> are registered trademarks of the Microsoft Corporation.

References to products and/or services of other manufacturers as provided in this document do not constitute endorsement by Systran Corporation.

Revised: June 25, 1999

Systran Corporation  
4126 Linden Avenue  
Dayton, OH 45432-3068 USA  
(937) 252-5601



# TABLE OF CONTENTS

|                                                   |     |
|---------------------------------------------------|-----|
| 1. INTRODUCTION.....                              | 1-1 |
| 1.1 How to Use This Manual .....                  | 1-1 |
| 1.1.1 Purpose .....                               | 1-1 |
| 1.1.2 Scope .....                                 | 1-1 |
| 1.1.3 Style Conventions .....                     | 1-1 |
| 1.2 Related Information .....                     | 1-1 |
| 1.3 Related Documentation .....                   | 1-1 |
| 1.4 Quality Assurance .....                       | 1-2 |
| 1.5 Technical Support .....                       | 1-2 |
| 1.6 Ordering Process .....                        | 1-3 |
| 2. PRODUCT OVERVIEW .....                         | 2-1 |
| 2.1 Overview .....                                | 2-1 |
| 2.2 Network Nodes .....                           | 2-1 |
| 2.2.1 The SCRAMNet Classic Node .....             | 2-1 |
| 2.2.2 The SCRAMNet-LX Node .....                  | 2-1 |
| 2.2.3 The SCRAMNet+ Node .....                    | 2-1 |
| 2.2.4 The SCRAMNet+ <sup>2X</sup> Node .....      | 2-1 |
| 2.3 Network Protocols.....                        | 2-2 |
| 2.3.1 GOLD RING Protocol .....                    | 2-2 |
| 2.3.2 BURST Mode Protocol .....                   | 2-2 |
| 2.3.3 PLATINUM Protocol .....                     | 2-2 |
| 2.3.4 PLUS Mode Protocol.....                     | 2-2 |
| 2.4 Navigating Through the Utility Programs ..... | 2-2 |
| 2.4.1 Menus .....                                 | 2-2 |
| 2.4.2 Dialog Windows .....                        | 2-2 |
| 2.4.3 Editing fields.....                         | 2-2 |
| 2.4.4 WINEPI Check Boxes .....                    | 2-3 |
| 3. WININST OPERATION .....                        | 3-1 |
| 3.1 Overview.....                                 | 3-1 |
| 3.2 Executing the Program .....                   | 3-1 |
| 3.3 Menus.....                                    | 3-1 |
| 3.3.1 Driver Options Menu .....                   | 3-1 |
| 3.3.2 Help .....                                  | 3-2 |
| 3.4 Install .....                                 | 3-2 |
| 3.5 Remove .....                                  | 3-2 |
| 3.6 Edit Registry .....                           | 3-2 |
| 4. WINMON OPERATION .....                         | 4-1 |
| 4.1 Overview.....                                 | 4-1 |
| 4.2 Executing the Program .....                   | 4-1 |
| 4.3 Menus.....                                    | 4-2 |
| 4.3.1 File Menu.....                              | 4-2 |
| 4.3.2 Function Menu.....                          | 4-3 |
| 4.3.3 Options Menu .....                          | 4-4 |
| 4.3.4 Help .....                                  | 4-4 |
| 4.4 Memory Access.....                            | 4-5 |
| 4.4.1 Modify Values .....                         | 4-5 |
| 4.4.2 Refresh Currently Displayed Values.....     | 4-5 |

|                                             |      |
|---------------------------------------------|------|
| 4.5 CSR Registers .....                     | 4-5  |
| 4.6 Functions.....                          | 4-6  |
| 4.6.1 Goto Address .....                    | 4-6  |
| 4.6.2 Clear Memory.....                     | 4-6  |
| 4.6.3 Fill Memory .....                     | 4-7  |
| 4.6.4 Search .....                          | 4-7  |
| 4.6.5 Flash LED.....                        | 4-8  |
| 4.7 Options.....                            | 4-8  |
| 4.7.1 Addressing.....                       | 4-8  |
| 4.7.2 Data Size.....                        | 4-9  |
| 4.8 Board Selection.....                    | 4-11 |
| 5. WINDIAG OPERATION.....                   | 5-1  |
| 5.1 Overview.....                           | 5-1  |
| 5.2 Executing WINDIAG.....                  | 5-1  |
| 5.3 Main Application Description.....       | 5-2  |
| 5.4 Menus.....                              | 5-3  |
| 5.4.1 File Menu.....                        | 5-3  |
| 5.4.2 Tools Menu.....                       | 5-3  |
| 5.4.3 Help .....                            | 5-4  |
| 5.5 Running Memory and Loopback Tests ..... | 5-4  |
| 5.5.1 The Testing Screen .....              | 5-5  |
| 5.6 Running Network Tests.....              | 5-5  |
| 5.7 Board Selection.....                    | 5-6  |
| 5.8 Diagnostic Options.....                 | 5-6  |
| 6. WINEPI OPERATION.....                    | 6-1  |
| 6.1 Overview.....                           | 6-1  |
| 6.2 Executing the Program.....              | 6-1  |
| 6.3 Menus.....                              | 6-2  |
| 6.3.1 File Menu.....                        | 6-3  |
| 6.3.2 Help .....                            | 6-3  |
| 6.4 Editing CSR Values .....                | 6-4  |
| 6.5 Bitwise Editing.....                    | 6-4  |
| 6.6 Restoring Defaults.....                 | 6-4  |

## APPENDICES

|                                               |            |
|-----------------------------------------------|------------|
| APPENDIX A - DIAGNOSTIC OUTPUT FILE.....      | A-1        |
| APPENDIX B - DIAGNOSTIC TEST ALGORITHMS ..... | B-1        |
| GLOSSARY .....                                | GLOSSARY-1 |
| INDEX .....                                   | INDEX-1    |

# FIGURES

|                                                   |      |
|---------------------------------------------------|------|
| Figure 3-1 Main Application Screen .....          | 3-1  |
| Figure 3-2 Driver Options Menu .....              | 3-1  |
| Figure 3-3 About WinInst .....                    | 3-2  |
| Figure 3-4 Registry Edit Dialog .....             | 3-3  |
| Figure 4-1 Main Application Screen .....          | 4-1  |
| Figure 4-2 File Menu.....                         | 4-2  |
| Figure 4-3 Functions Menu .....                   | 4-3  |
| Figure 4-4 Options Menu .....                     | 4-4  |
| Figure 4-5 About Monitor .....                    | 4-4  |
| Figure 4-6 CSR Monitor.....                       | 4-5  |
| Figure 4-7 Goto Memory Address.....               | 4-6  |
| Figure 4-8 Memory Fill Option .....               | 4-7  |
| Figure 4-9 Search Memory Option.....              | 4-7  |
| Figure 4-10 Addressing Sub-menu .....             | 4-8  |
| Figure 4-11 Data Size Selection .....             | 4-9  |
| Figure 4-12 Byte Data Size .....                  | 4-9  |
| Figure 4-13 Word Data Size.....                   | 4-10 |
| Figure 4-14 Longword Data Size .....              | 4-10 |
| Figure 5-1 SCRAMNet+ Node Status Window.....      | 5-1  |
| Figure 5-2 WinDiags Main Application Screen ..... | 5-2  |
| Figure 5-3 File Menu.....                         | 5-3  |
| Figure 5-4 Tools Menu.....                        | 5-3  |
| Figure 5-5 About Diagnostics.....                 | 5-4  |
| Figure 5-6 Loopback Options.....                  | 5-4  |
| Figure 5-7 Test Status Screen.....                | 5-5  |
| Figure 5-8 Options Screen.....                    | 5-6  |
| Figure 6-1 Main Application Screen .....          | 6-2  |
| Figure 6-2 File Sub-menu.....                     | 6-3  |
| Figure 6-3 About EPI .....                        | 6-3  |
| Figure 6-4 Bitwise Edit Screen.....               | 6-4  |

*This page intentionally left blank*

# 1. INTRODUCTION

---

## 1.1 How to Use This Manual

### 1.1.1 Purpose

The *SCRAMNet Network Windows NT Utilities User Manual* describes the basic features of SCRAMNet utilities applications. These include:

- Driver Installation Application (WinInst)
- Network monitor (WINMON)
- Hardware Diagnostic software (WINDIAG)
- EEPROM Initialization Software(WINEPI)

### 1.1.2 Scope

This information is intended for system designers, engineers, programmers and network installation personnel.

You should have at least a system level understanding of general computer processing, memory and hardware operation to effectively use this manual.

Specific knowledge of each SCRAMNet Control/Status Register (CSR) as described in the appropriate *SCRAMNet (bus) Hardware Reference Manual* is necessary for effective use of the applications.

### 1.1.3 Style Conventions

- Hexadecimal values are written with the word hex italicized and one point smaller than the context font following the value. For example: 03FF *hex*.
- Switch, signal and jumper abbreviations are in capital letters. For example: RSW1, J5, etc.
- Register bits and bit ranges are specified by the register identification followed by the bit or range of bits in brackets []. For example: CSR6[4], CSR3[15:0], ACR[1,2]
- Bit values are shown in single-quotes. For example: Set bit 15 to '1'.

## 1.2 Related Information

## 1.3 Related Documentation

*SCRAMNet Network Utilities User Manual (C-T-MU-UTIL)* - A user manual for the SCRAMNet Classic, SCRAMNet-LX, and SCRAMNet+ hardware diagnostic software, SCRAMNet+ EEPROM Initialization software, and the SCRAMNet Network Monitor for UNIX-based systems.

*SCRAMNet Network Windows Utility User Manual (C-T-MU-WNUTIL)* - A user manual for the Windows memory monitor programs **scrmon** and **winmon**.

*SCRAMNet Network Programmer Reference Manual (C-T-MR-PROGREF)* - A reference manual describing the collection of SCRAMNet-interface routines used to assist in UNIX-based application development.

*SCRAMNet Network Windows DLL Reference Guide (C-T-ML-WINDLL)* - A manual that defines a set of user routines that are exported by the Windows Dynamic Link Library (DLL) designed to assist in application development.

*SCRAMNet Network Windows NT DLL Reference Guide (C-T-ML-NTDLL)* - A manual that defines a set of user routines that are exported by the Windows NT Dynamic Link Library (DLL) designed to assist in application development.

## 1.4 Quality Assurance

Systran Corporate policy is to provide our customers with the highest quality products and services. In addition to the physical product, the company provides documentation, sales and marketing support, hardware and software technical support, and timely product delivery. Our quality commitment begins with product concept, and continues after receipt of the purchased product.

Systran's Quality System conforms to the ISO 9001 international standard for quality systems. ISO 9001 is the model for quality assurance in design, development, production, installation and servicing. The ISO 9001 standard addresses all 20 clauses of the ISO quality system and the most comprehensive of the conformance standards.

Our Quality System addresses the following basic objectives:

- Achieve, maintain and continually improve the quality of our products through established design, test, and production procedures.
- Improve the quality of our operations to meet the needs of our customers, suppliers, and other stakeholders.
- Provide our employees with the tools and overall work environment to fulfill, maintain, and improve product and service quality.
- Deliver only the highest quality product or service to our customers and other stakeholders.

The British Standards Institution (BSI), the world's largest and most respected standardization authority, assessed Systran's Quality System. BSI's Quality Assurance division certified we meet or exceed all applicable international standards, and issued Certificate of Registration, number FM 31468, on May 16, 1995. The scope of Systran's registration is: "Design, manufacture and service of high technology hardware and software computer communications products." The registration is maintained under BSI QA's bi-annual quality audit program.

Customer feedback is integral to our quality and reliability program. We encourage customers to contact us with questions, suggestions, or comments regarding any of our products or services. We guarantee professional and quick responses to your questions, comments, or problems.

## 1.5 Technical Support

We provide technical documentation with all of our products. This documentation describes the technology, its performance characteristics, and includes some typical applications. It also includes comprehensive support information, designed to answer any

technical questions that might arise concerning the use of this product. Systran also publishes and distributes technical briefs and application notes that cover a wide assortment of topics. Although we try to tailor the applications to real scenarios, not all possible circumstances are covered.

Although we have attempted to make this document comprehensive, there may be specific problems or issues this document does not satisfactorily cover. If you have any technical or non-technical questions or comments (including software) you can contact us at (937) 252-5601, or send an e-mail message to [support@systran.com](mailto:support@systran.com). Our goal is to offer a combination of products and services that provide complete, easy-to-use solutions for your application.

## 1.6 Ordering Process

To learn more about Systran products or to place an order, please use the following contact information:

- Phone: **(937) 252-5601**
- e-mail address: **info@systran.com**
- Internet address: **<http://www.systran.com/>**

*This page intentionally left blank*

# 2. PRODUCT OVERVIEW

---

## 2.1 Overview

The SCRAMNet Network is a real-time communications network, based on a replicated, shared memory concept. Each computer on the network has access to its own local copy of shared memory which is updated over a high-speed, serial-ring network. It is optimized for the high-speed transfer of data among multiple, real-time computers attached to the network. These computers, with their installed SCRAMNet boards, are called network nodes. Each computer, or node, is solving a portion of the same real-time problem.

## 2.2 Network Nodes

### 2.2.1 The SCRAMNet Classic Node

The SCRAMNet Classic was the first network node produced by Systran. It is a two-board configuration and has GOLD and BURST protocol. This board has byte-swapping capability.

### 2.2.2 The SCRAMNet-LX Node

The “LX” is the first single slot adaptation, and has BURST and BURST PLUS (variable length messages), and no error correction. The “LX” does not have on-board byte-swapping capability.

### 2.2.3 The SCRAMNet+ Node

SCRAMNet+ is an enhancement of the “LX”, and has PLATINUM and PLATINUM PLUS protocol as well as BURST and BURST PLUS. The “SCRAMNet+” board does not have byte-swapping capability.

### 2.2.4 The SCRAMNet+<sup>2X</sup> Node

The SCRAMNet+<sup>2X</sup> interface node board is backwards-compatible with SCRAMNet+ and the original SCRAMNet Classic product with the exception of the GOLD RING communication protocol.

The SCRAMNet+<sup>2X</sup> product uses an enhanced architecture where shared-memory-read operations bypass the SCRAMNet ASIC for improved performance. Although both PIO and DMA operations are improved, DMA operations gives the greatest increase in performance.

## 2.3 Network Protocols

### 2.3.1 GOLD RING Protocol

The GOLD-RING protocol allows as many messages on the network as there are nodes; but only one message from each node on the ring at any given time. Error detection and correction are implemented.

### 2.3.2 BURST Mode Protocol

The BURST mode allows each node to continuously transmit messages onto the network ring without waiting for any message to return for removal by the originating node. Error detection is in force but there is no error correction implemented.

### 2.3.3 PLATINUM Protocol

The PLATINUM mode combines the advantages of GOLD and BURST modes. There may be multiple messages on the network at one time with error correction implemented.

### 2.3.4 PLUS Mode Protocol

The PLUS mode applies to PLATINUM and BURST mode only. The PLUS mode permits variable length messages on the network. Maximum message size may be CSR selected to be 46 bits plus either 256 bytes or 1,024 bytes.

## 2.4 Navigating Through the Utility Programs

The programs are menu and dialog driven. The menus direct the flow of control through the program and the dialog windows are used to gather input from the user. The following is a list of the objects used in the program and how to interact with them.

### 2.4.1 Menus

A menu consists of a list of choices typically depicted in a long rectangular box located under the main windows title bar. The menu is accessed by clicking on the desired choice with the mouse or by pressing an appropriate -key combination. Each entry on the main menu may optionally bring up a sub-menu of additional menu choices.

### 2.4.2 Dialog Windows

A dialog window is a title and frame that can contain various controls for accepting user input. These controls are Editing Fields, Check Boxes, and Buttons. The mouse, arrow keys or tab key are used to move through a dialog box's controls. Pressing  or clicking on the "OK" button in dialog windows accepts the entries and closes the window. Pressing  or clicking on the "Cancel" button in dialog boxes cancels the entries and returns the user to the previous window.

### 2.4.3 Editing fields

An editing field is an area of a dialog box that accepts a string of user input. All editing fields are used to gather numeric data. Numeric editing fields generally have a minimum and maximum value for valid input. Values outside this range are not accepted by the editing field. The arrow keys can be used to move through individual digits in a number.

Editing keys, such as <Backspace>, <Delete>, and <Insert>, can also be used if available.

#### **2.4.4 WINEPI Check Boxes**

When manipulating the CSR bits directly using the bit-wise dialogs, each CSR is displayed in a dialog window as a series of check boxes. The keys used in check box dialogs are the same as for menus except for the addition of the space bar. The <Space Bar> is used to toggle the bit value of the currently selected item. If the item label is “Not Initializable”, that bit is ignored by the EEPROM initialization sequence.

*This page intentionally left blank*

# 3. WININST OPERATION

## 3.1 Overview

The WININST application is used to install and remove the SCRAMNet device driver. In addition, the application allows the editing of registry parameters for each SCRAMNet device found on the system.

## 3.2 Executing the Program

Start the program by double-clicking on the icon or running the program from the File Manager or Start Menu and the SCRAMNet Installation screen displays. This window allows device driver installation and removal, as well as the ability to edit the appropriate registry values.

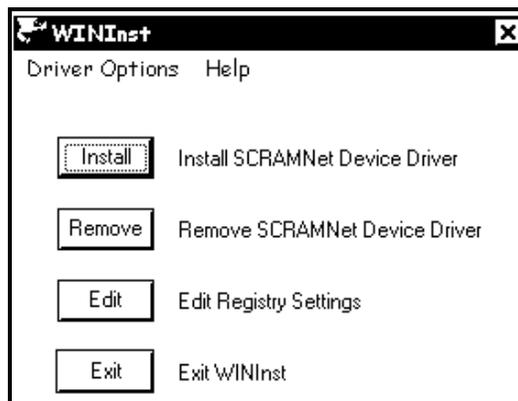


Figure 3-1 Main Application Screen

## 3.3 Menus

There are two menus available from the main window: Driver Options and Help.

### 3.3.1 Driver Options Menu

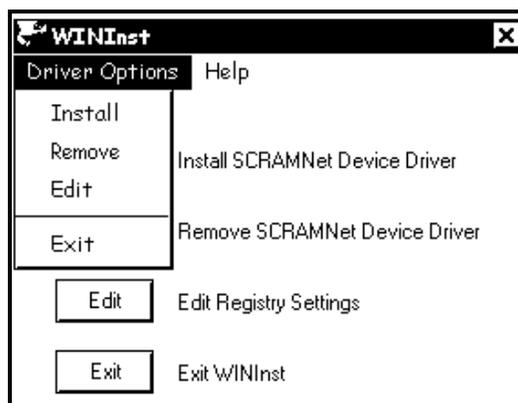


Figure 3-2 Driver Options Menu

Functions include:

- Install**..... Installs the SCRAMNet device driver
- Remove**..... Removes the SCRAMNet device driver.
- Edit** ..... Produces a dialog for editing the registry values.
- Exit**..... Exit the WinInst program.

### 3.3.2 Help

#### ABOUT

Click/select About from the **Help** menu to display a message dialog with information about the SCRAMNet Installation Utility.

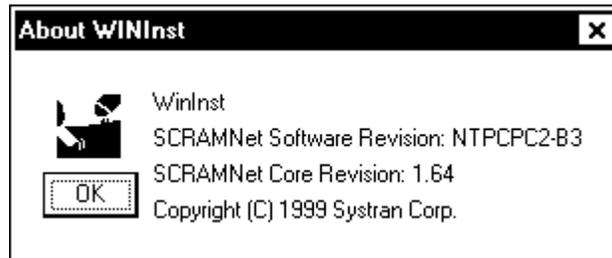


Figure 3-3 About WinInst

### 3.4 Install

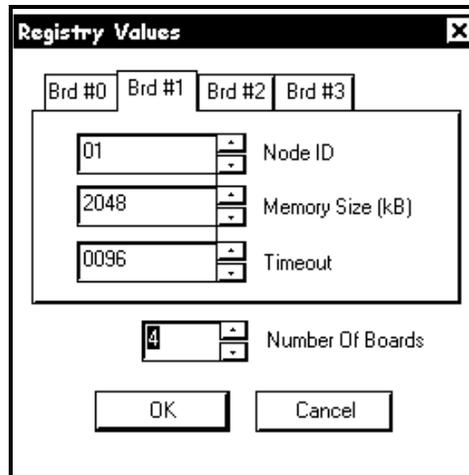
To install the SCRAMNet device driver, simply press the “Install” button, or select **Install** from the **Driver Options** menu. A message box will be displayed if the driver was successfully installed.

### 3.5 Remove

To remove the SCRAMNet device driver, press the “Remove” button, or select **Remove** from the **Driver Options** menu. A message box will be displayed if the driver was successfully removed.

### 3.6 Edit Registry

To edit the registry settings, press the “Edit” button, or select **Edit** from the **Driver Options** menu. This will produce a dialog as shown in Figure 3-4.



**Figure 3-4 Registry Edit Dialog**

The editor dialog provides a tab for each of the SCRAMNet cards identified at installation. Selecting the appropriate tab displays and allows modification of the current Node ID, mapped memory and time-out values stored in the NT registry.

In addition a 'Number Of Boards' option is included. This value defaults to the number of boards found by the driver on the PCI bus.



**NOTE:** Increasing the value of 'Number Of Boards' to a number greater than the actual number of installed devices could produce unpredictable results

Pressing "OK" stores the displayed values; "Cancel" discards the values.

*This page intentionally left blank*

# 4. WINMON OPERATION

## 4.1 Overview

The WINMON application permits viewing and editing memory and CSR locations on the SCRAMNet node. This application is useful during software development to verify that the correct values are being written to SCRAMNet memory and CSRs.

## 4.2 Executing the Program

Start the program by double-clicking on the icon or running the program from the File Manager or Program Manager, and the SCRAMNet Memory Monitor screen displays. This window allows read and write access to any SCRAMNet memory location.

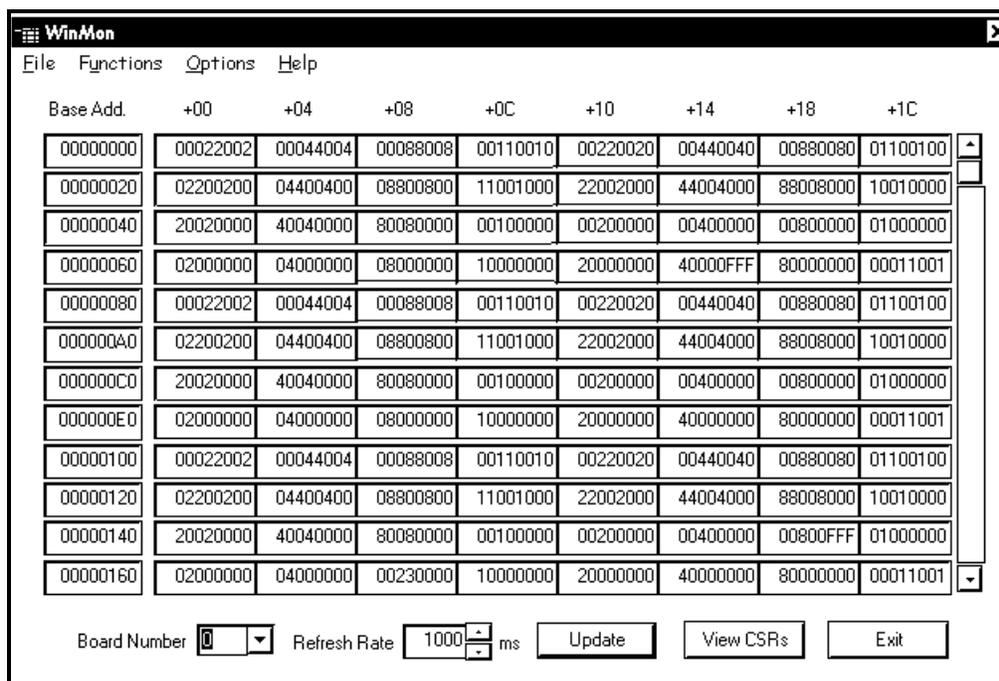


Figure 4-1 Main Application Screen

## 4.3 Menus

There are four menus available from the main window:

- File
- Functions
- Options
- Help.

### 4.3.1 File Menu

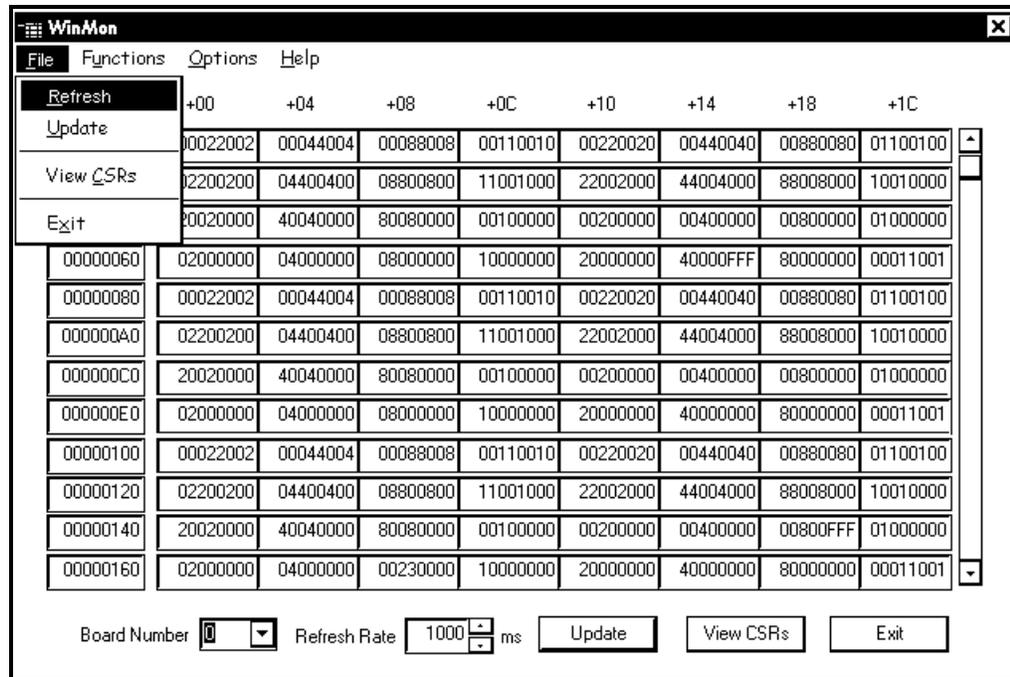


Figure 4-2 File Menu

Functions include:

- Refresh** ..... Re-reads the SCRAMNet memory and update the display. (Page 4-5)
- Update** ..... Updates SCRAMNet memory based on values entered in the display edit boxes. (Page 3-5)
- View CSRs** ..... Toggles the display of the SCRAMNet CSR monitor dialog. (Page 3-5)
- Exit**..... Exit the **WinMon** program.

### 4.3.2 Function Menu

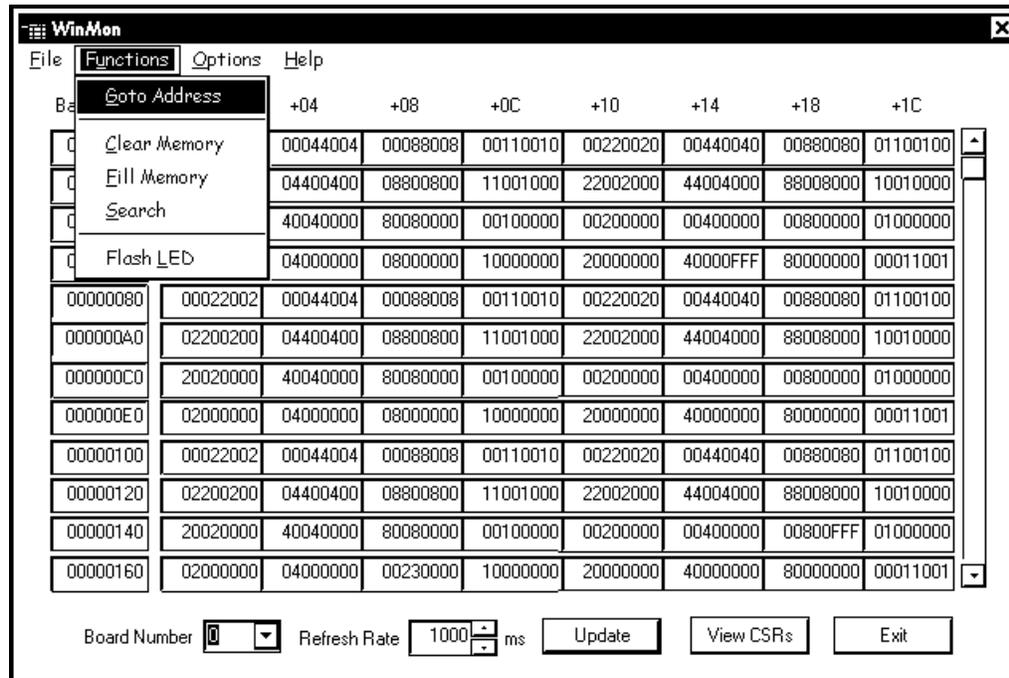


Figure 4-3 Functions Menu

Functions include:

- Goto Address** ..... Permits direct access to a memory address. (Page 4-6)
- Clear Memory** ..... Fills all mapped memory with 0 hex. (Page 4-6)
- Fill Memory** ..... Permits filling a range of memory with a desired value. (Page 4-7)
- Search** ..... Permits search of memory for a specific value. (Page 4-7)
- Flash LED** ..... Permits starting and stopping flashing of the Insert LED. (Page 4-8)

### 4.3.3 Options Menu

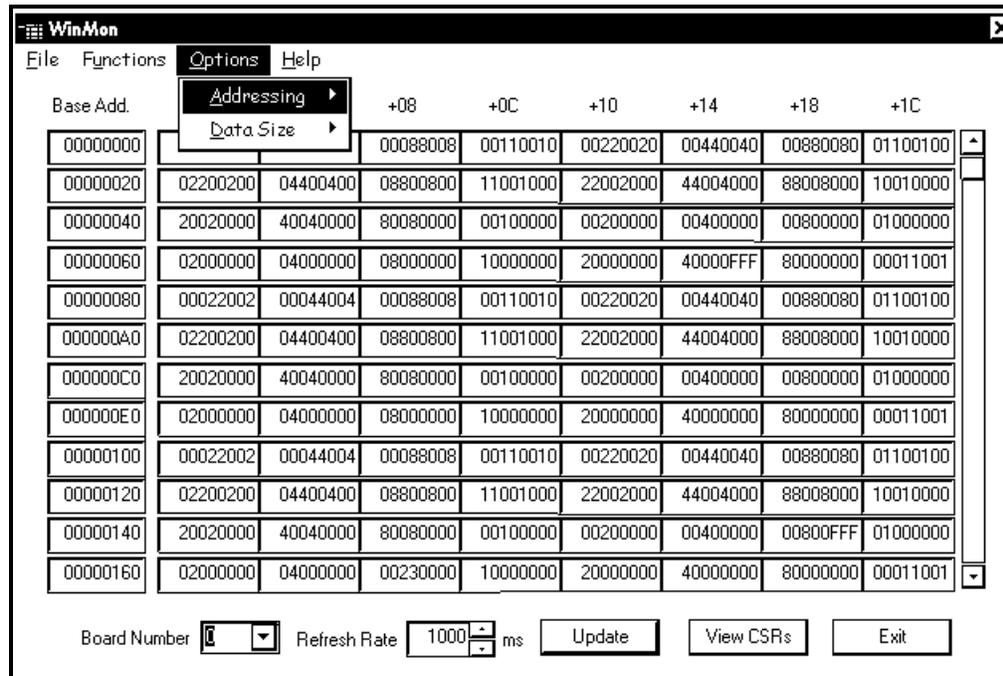


Figure 4-4 Options Menu

The Options Menu includes:

**Addressing** ..... Permits selection of either Relative or Absolute addresses.  
(Page 4-8)

**Data Size**..... Permits selection of memory access modes: Bytes, Words, or Longwords. (Page 4-9)

### 4.3.4 Help

#### ABOUT

Click/select About from the Help menu to display a message dialog with information about the SCRAMNet Memory Monitor Utility.

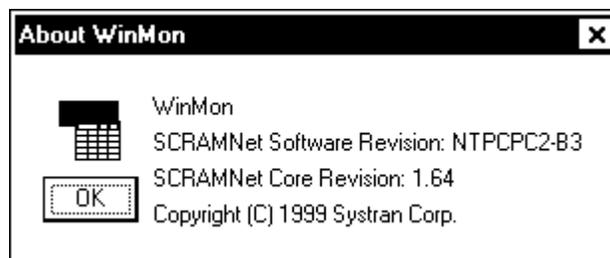


Figure 4-5 About Monitor

## 4.4 Memory Access

The WINMON application allows full read and write capabilities to the mapped segment of SCRAMNet memory

### 4.4.1 Modify Values

To modify a memory location, enter a new value in the edit box corresponding to the appropriate memory address. Press the “Update” button or, select **Update** from the file menu to write the value to memory.

### 4.4.2 Refresh Currently Displayed Values.

In order to refresh the displayed memory values, select **Refresh** from the **File** menu. The displayed SCRAMNet memory will be re-read and the display updated accordingly. On the main application window is an edit box entitled “Refresh Rate”. This is an auto-refresh feature that updates the values automatically over the given time interval in milliseconds. Entering a ‘0’ in this edit box disables the auto-refresh feature.

## 4.5 CSR Registers

The SCRAMNet CSR’s are monitored through the CSR dialog as shown in Figure 4-6.

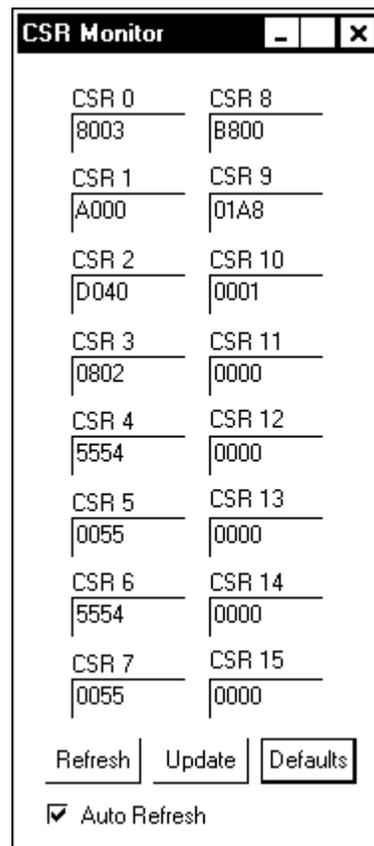


Figure 4-6 CSR Monitor

To enable the CSR Monitor dialog, select **View CSRs** from the file menu, or press the “View CSRs” button on the main application window to toggle the dialog on or off. The

monitor displays the sixteen SCRAMNet CSR's and updates the values based on the refresh-rate in the main application window. Press the "Refresh" button to immediately display new memory values. Click on "Auto Refresh" to toggle the auto-refresh feature on or off.

To update a CSR value, enter its value in the appropriate edit box and press the "Update" button. Press the "Defaults" button to restore a set of pre-programmed CSR values.

## 4.6 Functions

### 4.6.1 Goto Address

A memory location not currently displayed in the window can be accessed using any one of several methods.

- Scroll the screen up or down one row at a time by pressing on the up or down arrow key.
- Use the scroll bar on the right hand side of the window to scroll the displayed offsets.

To go directly to a particular offset, select the **Goto Address** option. A dialog box will be displayed.

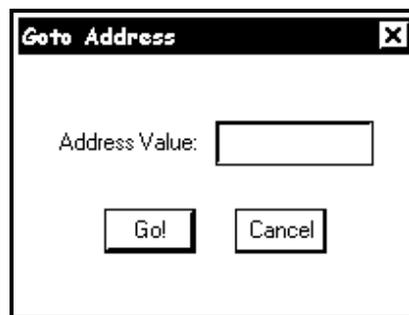


Figure 4-7 Goto Memory Address

Type in the desired address and click/select the "Go!" button to scroll the memory window to the desired address.

### 4.6.2 Clear Memory

Select **Clear Memory** from the functions menu to fill the entire mapped memory space with 0 hex. If a specific range of memory is required to be cleared, the Fill Memory dialog permits specific range access.

### 4.6.3 Fill Memory

Part or all of SCRAMNet memory can be filled with a desired value. To fill memory select the **Fill Memory** option on the **Function** sub-menu. This causes the fill-memory dialog similar to the following to display.

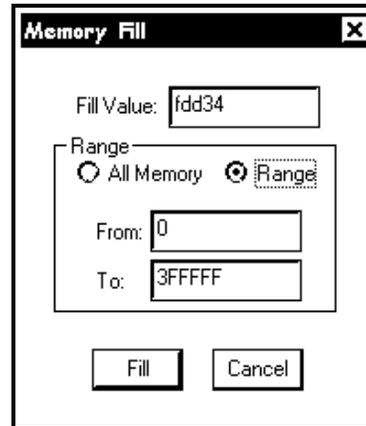


Figure 4-8 Memory Fill Option

Type in any hexadecimal value in the Fill Value box. Then select either "All Memory" or a "Range" of memory. Click/select the "Fill" button to execute.

### 4.6.4 Search

To search memory for a particular value, select the **Search** option on the **Function** menu. The Search Memory dialog box will display.

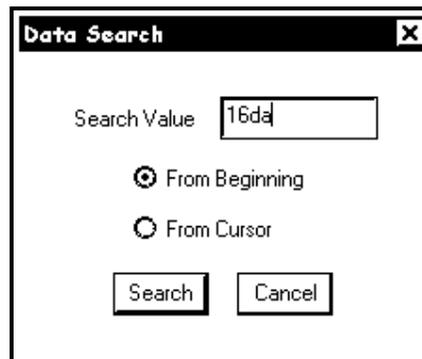


Figure 4-9 Search Memory Option

Type in any hexadecimal value in the Search Value box. Select a starting point, from the beginning or from the cursor location, and click/select the "Search" button to execute.

## 4.6.5 Flash LED

When this toggle option is selected the program will alternatively write the value 8003 *hex* and 0000 *hex* to CSR0 to toggle the insert LED. This process will be constantly repeated with a small delay between each write. Select the option a second time to stop the LED from flashing. This will erase the check mark placed next to the menu item when it was previously selected.



**NOTE:** Check the current value in CSR0 to determine which of the two values is currently in the register and alter the value if needed.

## 4.7 Options

### 4.7.1 Addressing

SCRAMNet memory can be displayed using absolute or relative addresses. In absolute addressing mode the address used to reference each memory location is the physical memory address. In relative addressing mode, each memory address is displayed as a byte offset from the start of SCRAMNet memory.

From the **Options** sub-menu, select **Addressing** and then select either **Relative** or **Absolute**.

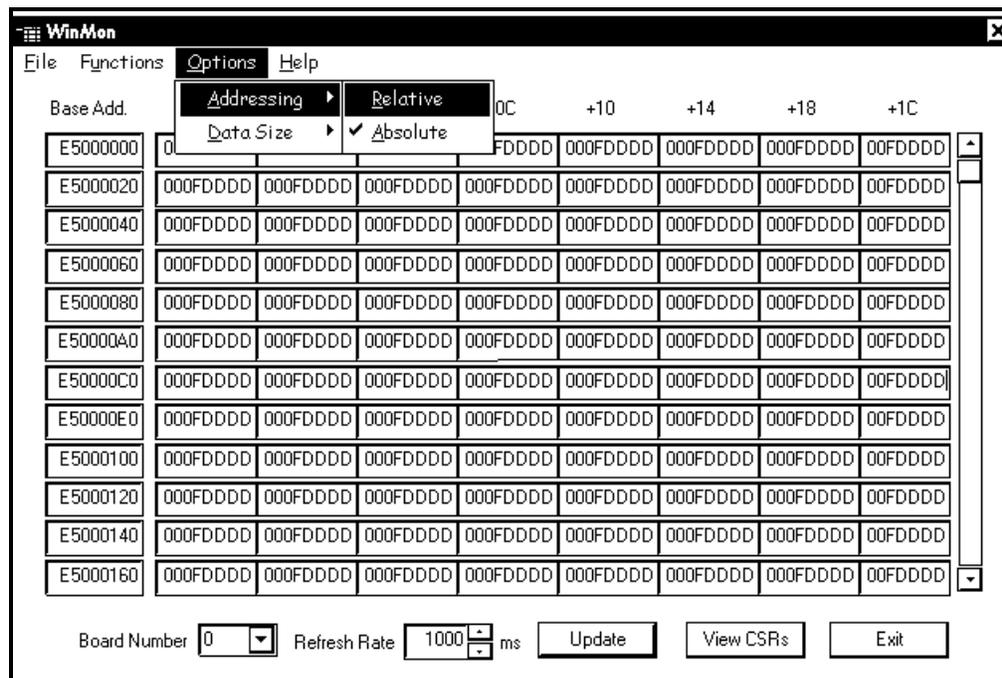


Figure 4-10 Addressing Sub-menu

## 4.7.2 Data Size

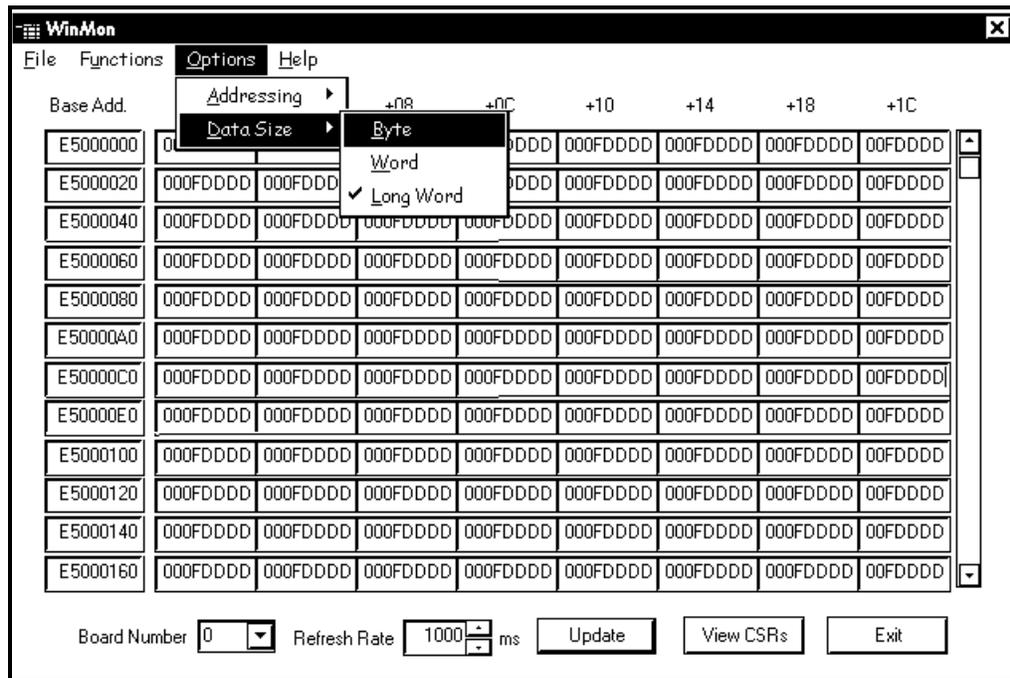


Figure 4-11 Data Size Selection

Memory can be accessed as bytes, words or longwords. Select the **Options** menu and then select the desired size from the **Data Size** sub-menu.

An example of Byte data size:

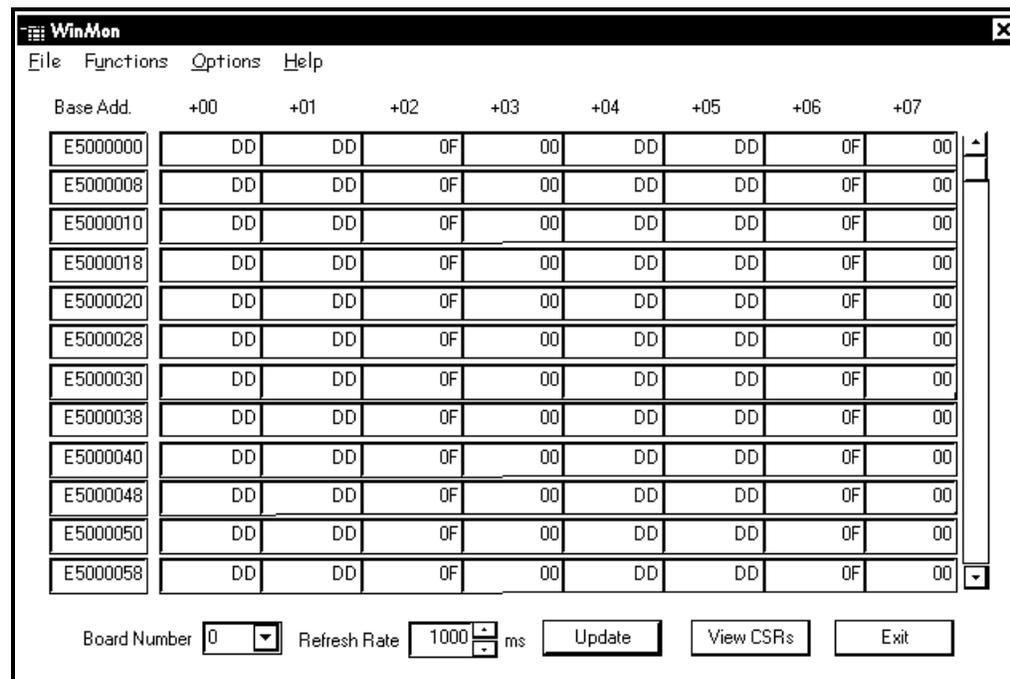


Figure 4-12 Byte Data Size

An example of Word data size:

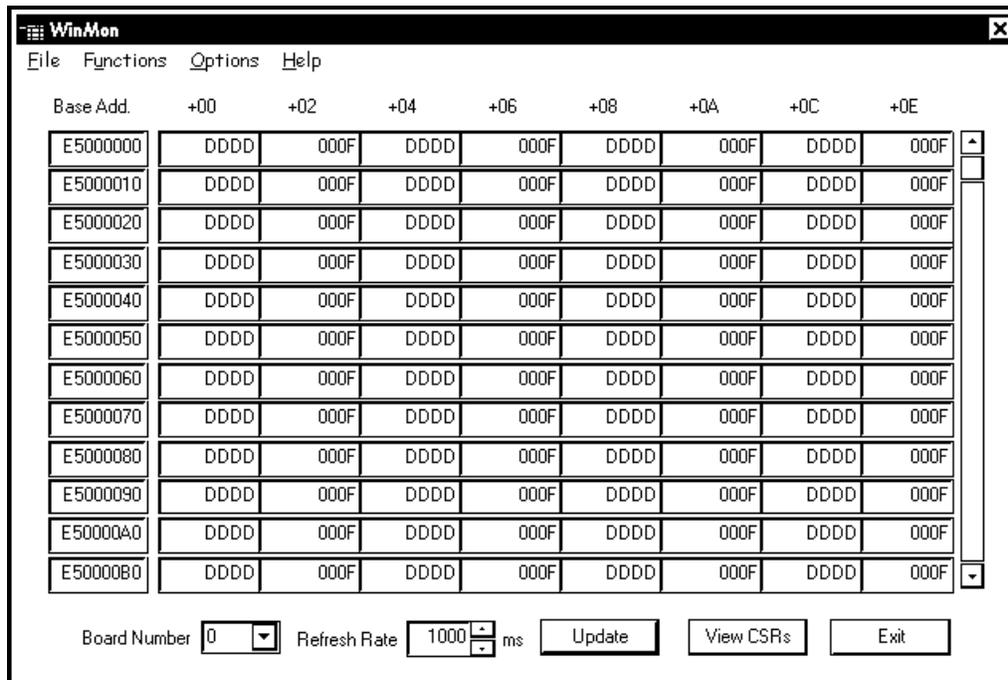


Figure 4-13 Word Data Size

An example of Longword data size:

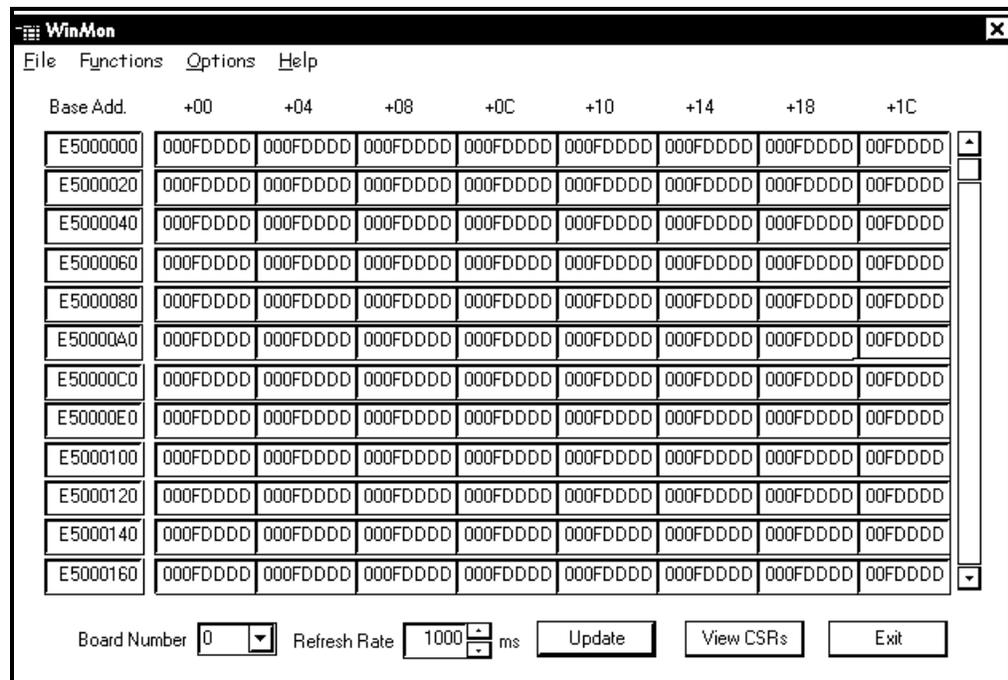


Figure 4-14 Longword Data Size

## 4.8 Board Selection

The main application window has a board-selection option in order to monitor alternate SCRAMNet devices on the PCI bus. Upon selecting another board, the memory and CSR displays are update to reflect the selected board. By running multiple instances of **WinMon**, and setting each to a different SCRAMNet device, multiple SCRAMNet boards can be monitored simultaneously.

*This page intentionally left blank*

# 5. WINDIAG OPERATION

## 5.1 Overview

This section describes how to use the diagnostic software (WINDIAG). To run the diagnostic software package for a SCRAMNet device, the Windows registry must contain a series of software keys describing the configuration data used by the application program and driver. This information is placed into the registry by the **WinInst** utility as described in chapter 3 and the software installation manual. This section describes how to run the various hardware tests. For information on how the tests work, refer to Appendix B: DIAGNOSTIC TEST ALGORITHMS.

## 5.2 Executing WINDIAG

To execute diagnostics, open the SCRAMNet application group and click on the SCRAMNet **Diagnostics** icon. Initially, a SCRAMNet status screen displays the current configuration of board number 0 installed on the system. Between displaying the screens, the SCRAMNet memory and registers are mapped and a quick-existence test is performed to determine whether both the registers and memory are accessible.

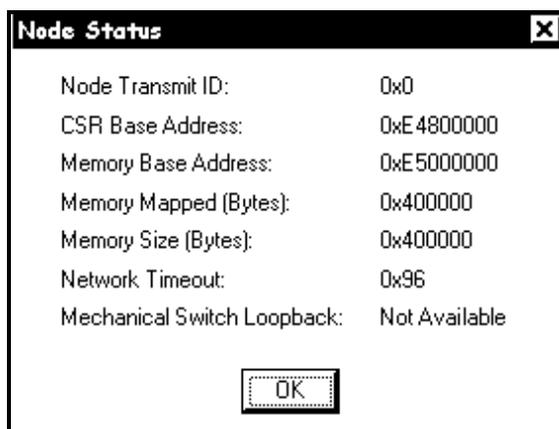


Figure 5-1 SCRAMNet+ Node Status Window

Figure 5-1 shows the Node Status screen for a SCRAMNet+ device. This shows the initial information about the node that is determined by the contents of the CSRs and by the existence test.

Press any key to display the main application screen.

## 5.3 Main Application Description

Figure 5-2 shows the main application screen for the SCRAMNet Diagnostics.

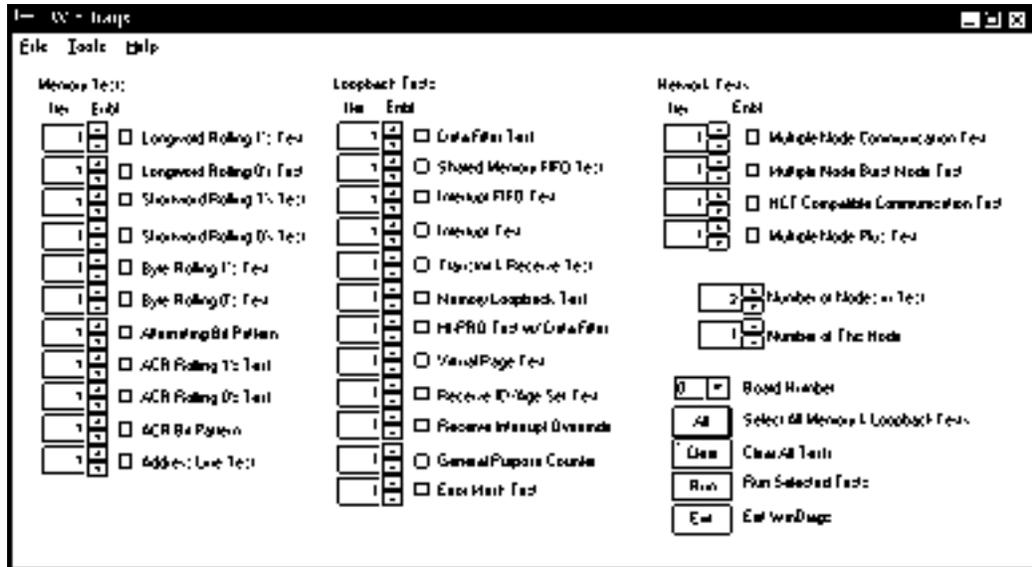


Figure 5-2 WinDiags Main Application Screen

The main application screen displays the various SCRAMNet diagnostic tests to be performed placed into three distinct groups, Memory Tests, Loopback Tests and Network Tests. Each test has an enable (Enbl.) checkbox, as well as an edit box (Iter.) for specifying the number of iterations to be performed.

## 5.4 Menus

There are three menus available from the main window: **File**, **Tools** and **Help**.

### 5.4.1 File Menu

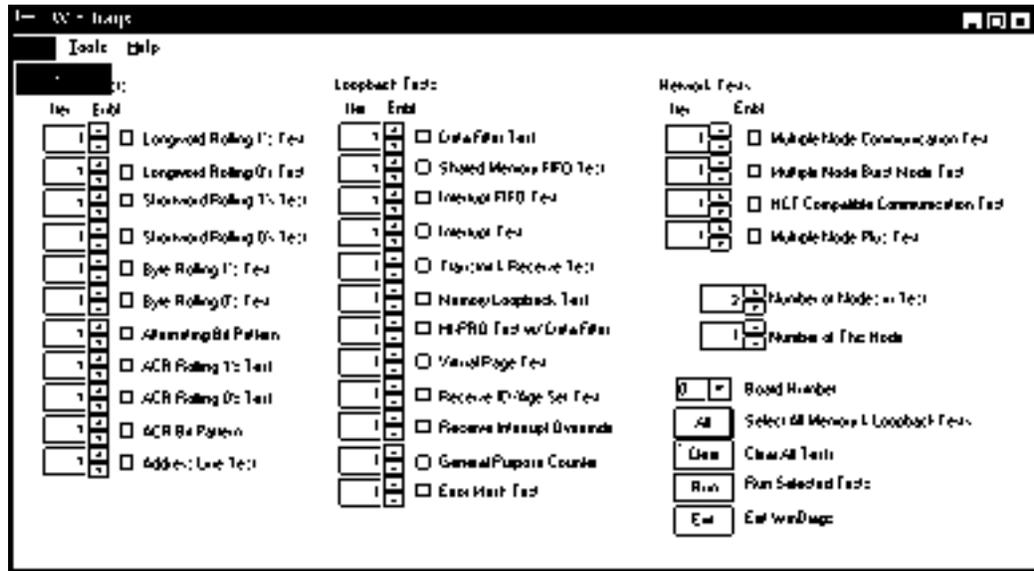


Figure 5-3 File Menu

Functions include:

**Exit**..... Exits WinDiags application.

### 5.4.2 Tools Menu

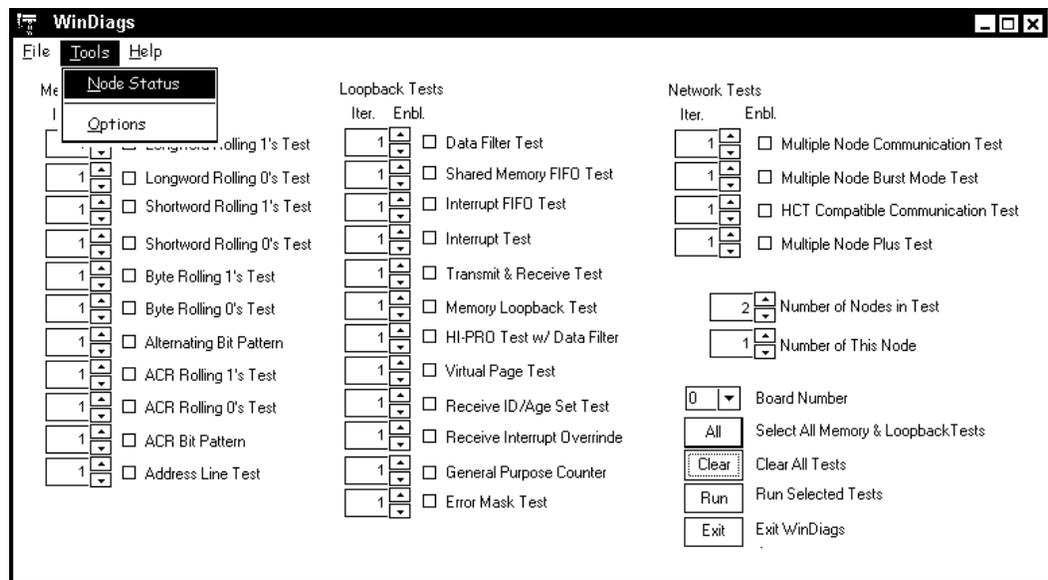


Figure 5-4 Tools Menu

Functions include:

- Node Status** ..... Displays the node status dialog for the currently selected SCRAMNet device.
- Options** ..... Displays options dialog for setting maximum error count, disabling the screen saver and setting the test application priority.

### 5.4.3 Help

#### ABOUT

Click/select About from the Help menu to display a message dialog with information about the SCRAMNet Diagnostics Utility.

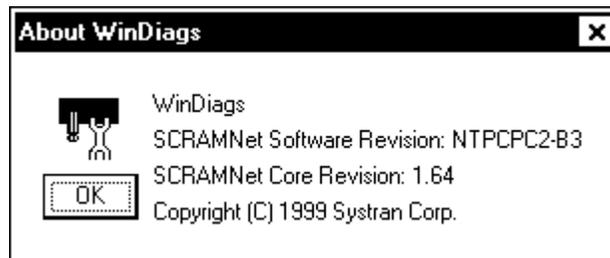


Figure 5-5 About Diagnostics

## 5.5 Running Memory and Loopback Tests

The main application window permits the selection of the desired diagnostic tests to be run on the current SCRAMNet board. Select the check boxes next to the desired tests and set the number of iterations in the corresponding edit box. Press the “All” button to select all of the Memory and Loopback tests and prompts for the number of iterations desired. Similarly, press the “Clear” button to remove the checks of any currently selected tests.

Press the “Run” button to run the tests. Initially, a file-open dialog box will display. Enter a suitable name for the output file in the filename box. If any loopback tests are selected to be run, an additional dialog box will appear as shown in Figure 5-6.

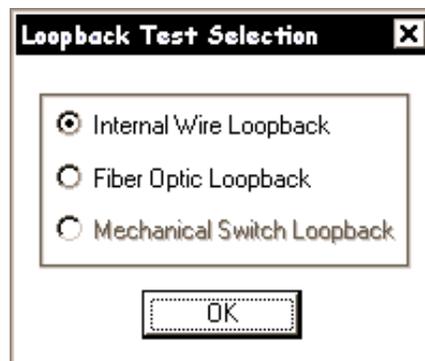


Figure 5-6 Loopback Options

From this dialog, select the type of loopback used on the SCRAMNet card. If Fiber-optic Loopback is used, a single loopback cable must be installed on the SCRAMNet card.

Refer to Appendix B for a detailed description of the actual test algorithms.

### 5.5.1 The Testing Screen

The testing screen provides a real time status of the tests being run as shown in Figure 5-7.

| MEMORY TESTS         | Iteration | Passed | Failed | Errors |
|----------------------|-----------|--------|--------|--------|
| Longword Rolling 1s  | 2         | 2      | 0      | 0      |
| Longword Rolling 0s  | 2         | 2      | 0      | 0      |
| Shortword Rolling 1s | 2         | 2      | 0      | 0      |
| Shortword Rolling 0s | 1         | 1      | 0      | 0      |

| Current Test: Shortword Rolling 0s

Cancel Tests    View Output

Figure 5-7 Test Status Screen

The test status screen shows the tests results as they occur. Press the “Cancel” button to stop the tests currently in progress. Once the tests are completed, the “View Output” button may be pressed to view the results as shown in the output file. Press the “Exit” button when completed to exit the test status dialog.

## 5.6 Running Network Tests

The network tests are selected in a similar manner to the memory and loopback tests. These tests, however must be run individually and the current SCRAMNet card must be connected to at least one other node. Once a network test is selected, set the number of nodes in the network loop via the appropriate edit box. In addition, assign a unique node number to this node via the Node Number edit box (Nodes are to be numbered sequentially 0, 1, 2, ..., etc.).



**CAUTION:** The number of nodes in this test **MUST** be consistent for all nodes running the network tests and **MUST** always be at least one greater than the number of the last node running the tests; that is  $N + 1$ .



**NOTE:** The iteration count must be the same for all tests that are running the multiple node communication test.

To run the selected test, press the “Run” button. Once again, a file dialog will appear to prompt for an output filename. Enter the filename and press “Open.” Now start the same

network test on the remaining nodes in the loop. A test status screen similar to Figure 5-7 will appear, once again providing a real-time report of the test status.



**NOTE:** Node number 0 is the master node. Start network tests on all other nodes in the loop before the master node. When node 0 is started, the network test begins running on all of the machines in the loop.

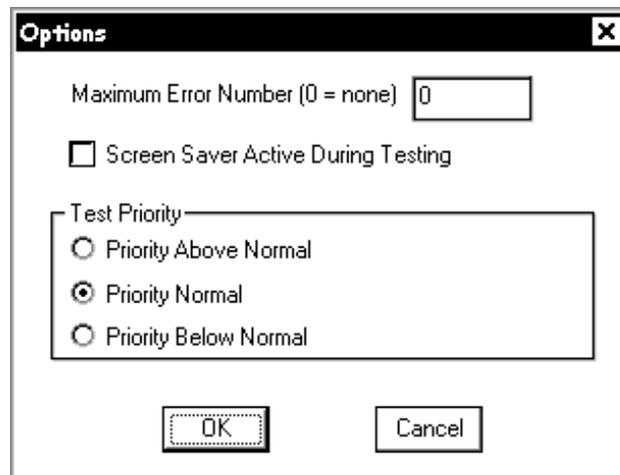
Refer to Appendix B for a detailed description of the actual test algorithms.

## 5.7 Board Selection

The Main Application window has a board selection option in order to test alternate SCRAMNet devices on the PCI bus. Upon selecting another board, all tests run will occur on the selected board. By running multiple instances of **WinDiags**, and setting each to a different SCRAMNet device, multiple SCRAMNet boards can be tested simultaneously.

## 5.8 Diagnostic Options

Selecting **Options** from the tools menu produces a dialog box as shown in Figure 5-8.



**Figure 5-8 Options Screen**

The Maximum-Error-Number value is used by the diagnostic tests to determine when to terminate in case of failure. This essentially acts as a maximum-retry count.

Check the 'Screen Saver Active During Testing' box to activate the Windows NT screen saver while the diagnostic tests are running. Leave this box unchecked if running a long test (many iterations) to prevent the screen saver from wasting processor time.

The 'Test Priority' panel selection assigns a higher or lower priority to the main test thread. The default value is 'Priority Normal.'

# 6. WINEPI OPERATION

---

## 6.1 Overview

The EEPROM Initialization (**WINEPI**) SCRAMNet+ utility simplifies configuration of the network node, although the SCRAMNet Network can be used without running this utility. Node configuration was previously done as the first step in the application software. The **WINEPI** program stores the configuration in the serial EEPROM, which initializes the node upon power-up. This is only a start-up configuration. Therefore, application programs may still WRITE to the CSR registers and change these values at run-time as desired. Consult the appropriate *SCRAMNet Hardware Reference Manual* for a detailed description of the SCRAMNet CSRs. This section describes the basic operations of the **WINEPI** program for windows operating systems.

## 6.2 Executing the Program

Start the program by double-clicking on the icon or running the program from the **File Manager** or **Start Menu** and the SCRAMNet EPI screen displays. This window allows read and write access to the SCRAMNet CSR EEPROM values.

## 6.3 Menus

The main application screen (Figure 6-1) groups the programs functionality into two menu items: **EPI Options** and **Help**. The File menu provides read and write options for programming the EEPROM. The **About** menu displays basic program version information.

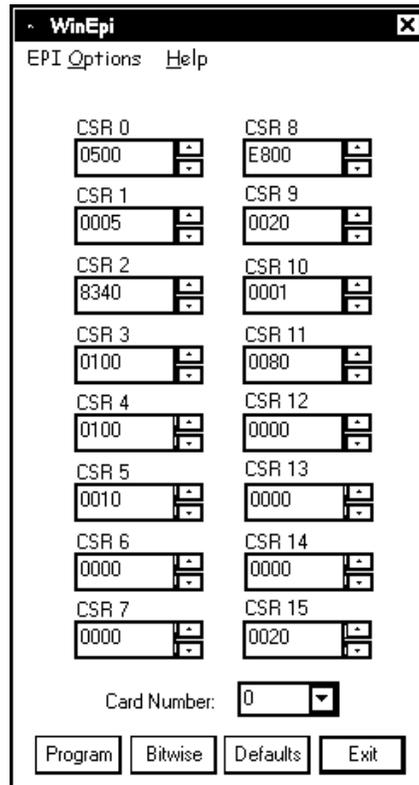


Figure 6-1 Main Application Screen

### 6.3.1 File Menu

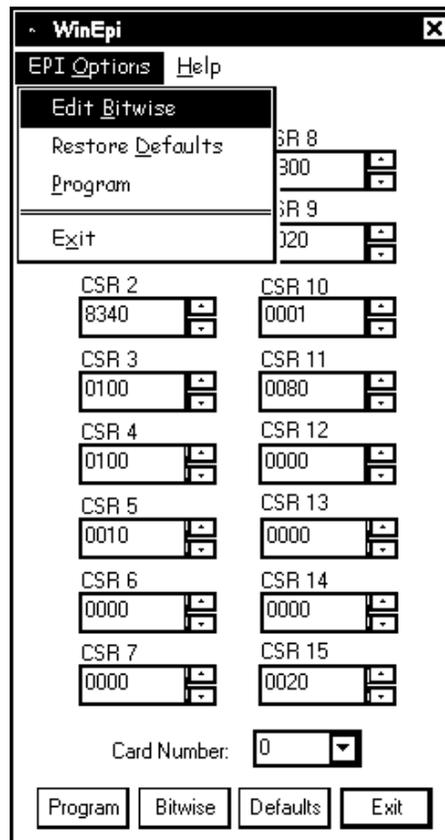


Figure 6-2 File Sub-menu

Functions include:

- Edit Bitwise**..... Produces a dialog with descriptions of individual CSR bits.
- Restore Defaults** ... Restores the CSR's to factory defaults.
- Program** ..... Writes the currently displayed values to the SCRAMNet EEPROM.

### 6.3.2 Help

#### ABOUT

Click/select **About** from the **Help** menu to display a message dialog with information about the SCRAMNet Memory Monitor Utility.

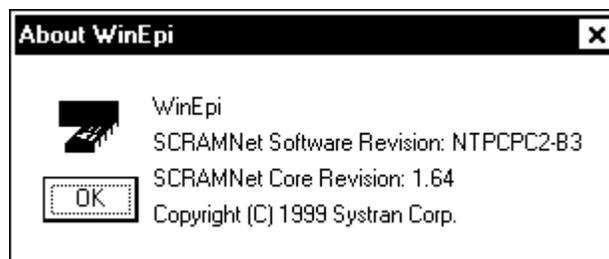


Figure 6-3 About EPI

## 6.4 Editing CSR Values

The default CSR values are changed by simply entering the desired value in the corresponding CSR edit box. Press the Program button or select Program from the file menu to write the displayed values to the SCRAMNet EEPROM.

## 6.5 Bitwise Editing

Press the “Bitwise” button for a more intuitive method of viewing the CSRs. This produces a tabbed dialog box with a tab corresponding to each of the CSRs. Within the individual tab spaces is an individual listing of the CSR bit representations along with a check box for each as shown in Figure 6-4.

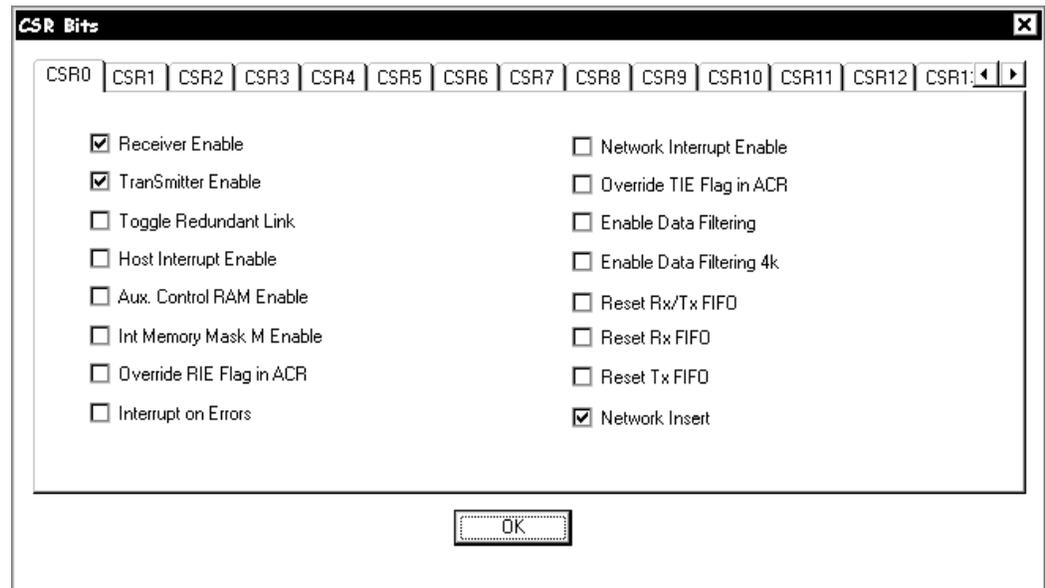


Figure 6-4 Bitwise Edit Screen

Select the appropriate check to flag the corresponding bit. Once the bits are set as desired, select “OK” to return to the main application screen. Notice that the hexadecimal values are changed to reflect those bits checked. Press the “Program” button or select “Program” from the file menu to write the displayed values to the SCRAMNet EEPROM.

## 6.6 Restoring Defaults

Press the “Defaults” button or select “Restore Defaults” from the EPI Options menu to load the CSR edit boxes with the SCRAMNet factory default values. Press the “Program” button or select “Program” from the **File** menu to write the displayed values to the SCRAMNet EEPROM.

# **APPENDIX A**

## **SAMPLE DIAGNOSTIC TEST FILE**

### **TABLE OF CONTENTS**

|                         |     |
|-------------------------|-----|
| A.1 MEMORY TESTS.....   | A-1 |
| A.2 LOOP TESTS.....     | A-2 |
| A.3 NETWORK TESTS ..... | A-3 |



## A.1 MEMORY TESTS

-----  
---- Begin Longword rolling 1's test ----  
No Errors detected in 5 iterations  
----- End Longword rolling 1's test -----

---- Begin Longword rolling 0's test ----  
No Errors detected in 5 iterations  
----- End Longword rolling 0's test -----

---- Begin Shortword rolling 1's test ----  
No Errors detected in 5 iterations  
----- End Shortword rolling 1's test -----

---- Begin Shortword rolling 0's test ----  
No Errors detected in 5 iterations  
----- End Shortword rolling 0's test -----

---- Begin Byte rolling 1's test ----  
No Errors detected in 5 iterations  
----- End Byte rolling 1's test -----

---- Begin Byte rolling 0's test ----  
No Errors detected in 5 iterations  
----- End Byte rolling 0's test -----

---- Begin Alternating bit pattern test ----  
No Errors detected in 5 iterations  
----- End Alternating bit pattern test -----

---- Begin ACR rolling 1's test ----  
No Errors detected in 5 iterations  
----- End ACR rolling 1's test -----

---- Begin ACR rolling 0's test ----  
No Errors detected in 5 iterations  
----- End ACR rolling 0's test -----

---- Begin ACR bit pattern test ----  
No Errors detected in 5 iterations  
----- End ACR bit pattern test -----

---- Begin Address line Test ----  
No Errors detected in 5 iterations  
----- End Address line Test -----

## A.2 LOOP TESTS

----- Begin Data filter Test -----  
No Errors detected in 5 iterations  
----- End Data filter Test -----

----- Begin Shared memory fifo Test -----  
Shared Memory Fifo almost full after 895 writes.  
No Errors detected in 5 iterations  
----- End Shared memory fifo Test -----

----- Begin Interrupt fifo Test -----  
No Errors detected in 5 iterations  
----- End Interrupt fifo Test -----

----- Begin Interrupt Test -----  
No Errors detected in 5 iterations  
----- End Interrupt Test -----

----- Begin Transmit and Receive Test -----  
No Errors detected in 5 iterations  
----- End Transmit and Receive Test -----

----- Begin Memory loopback Test -----  
No Errors detected in 5 iterations  
----- End Memory loopback Test -----

----- Begin HI-PRO Test with Data Filter -----  
No Errors detected in 5 iterations  
----- End HI-PRO Test with Data Filter -----

----- Begin Virtual Page Test -----  
FATAL ERROR: Virtual Paging is not viable for 8 MB memory size  
----- End Virtual Page Test -----

----- Begin Receive ID/Age Set Test -----  
No Errors detected in 5 iterations  
----- End Receive ID/Age Set Test -----

----- Begin Receive Interrupt Override Test -----  
No Errors detected in 5 iterations  
----- End Receive Interrupt Override Test -----

----- Begin General Purpose Counter Test -----  
Message Transit Time = 45 ticks (26.66ns/tick)  
Message Transit Time = 45 ticks (26.66ns/tick)  
Message Transit Time = 45 ticks (26.66ns/tick)  
Message Transit Time = 46 ticks (26.66ns/tick)  
Free Run Timer = 321 (26.66ns/tick)  
Free Run Timer = 3650 (26.66ns/tick)

Free Run Timer = 5269 (26.66ns/tick)  
Free Run Timer = 6879 (26.66ns/tick)  
Free Run Timer = 0 (1.706us/tick) read & cleared  
Free Run Timer = 1 (1.706us/tick) read & cleared  
Free Run Timer = 0 (1.706us/tick) read & cleared  
Free Run Timer = 0 (1.706us/tick) read & cleared

The following counter values should be incrementing at a steady pace

Free Run Timer = 40 (1.706us/tick) read w/o clear  
Free Run Timer = 80 (1.706us/tick) read w/o clear  
Free Run Timer = 106 (1.706us/tick) read w/o clear  
Free Run Timer = 131 (1.706us/tick) read w/o clear  
No Errors detected in 5 iterations  
----- End General Purpose Counter Test -----

----- Begin Error Mask Test -----  
No Errors detected in 5 iterations  
----- End Error Mask Test -----

## A.3 NETWORK TESTS

-----  
----- Begin Multiple node communication test -----  
No Errors detected in 5 iterations  
----- End Multiple node communication test -----

----- Begin Multiple node Burst mode test -----  
No Errors detected in 5 iterations  
----- End Multiple node Burst mode test -----

----- Begin HCT Compatible Communication test -----  
No Errors detected in 5 iterations  
----- End HCT Compatible Communication test -----

*This page intentionally left blank*

**APPENDIX B**

**DIAGNOSTIC  
TEST ALGORITHMS**

# TABLE OF CONTENTS

|                                                                 |      |
|-----------------------------------------------------------------|------|
| B.1 Overview .....                                              | B-1  |
| B.2 Memory Tests.....                                           | B-1  |
| B.2.1 Longword Rolling 1's Test .....                           | B-1  |
| B.2.2 Longword Rolling 0's Test .....                           | B-2  |
| B.2.3 Shortword Rolling 1's Test .....                          | B-3  |
| B.2.4 Shortword Rolling 0's Test .....                          | B-3  |
| B.2.5 Byte Rolling 1's Test .....                               | B-4  |
| B.2.6 Byte Rolling 0's Test .....                               | B-4  |
| B.2.7 Alternating Bit Pattern Test .....                        | B-5  |
| B.2.8 ACR Byte Rolling 1's Test .....                           | B-5  |
| B.2.9 ACR Byte Rolling 0's Test .....                           | B-6  |
| B.2.10 ACR Bit Pattern Test .....                               | B-7  |
| B.2.11 Address Line Test .....                                  | B-7  |
| B.3 Loop Tests.....                                             | B-8  |
| B.3.1 Data Filter Test .....                                    | B-8  |
| B.3.2 Shared Memory (Transmit) FIFO Test .....                  | B-10 |
| B.3.3 Byte Swapping Test (Used only with SCRAMNet Classic)..... | B-11 |
| B.3.4 Interrupt FIFO Test .....                                 | B-12 |
| B.3.5 Interrupt Test.....                                       | B-13 |
| B.3.6 General Purpose Counter Test .....                        | B-13 |
| B.3.7 Error Mask Test .....                                     | B-15 |
| B.3.8 Transmit and Receive Test.....                            | B-17 |
| B.3.9 Memory Loopback Test .....                                | B-18 |
| B.3.10 HIPRO Test with Data Filter.....                         | B-19 |
| B.3.11 Virtual Page Test .....                                  | B-20 |
| B.3.12 Receive ID/Age Set Test.....                             | B-21 |
| B.3.13 Receive Interrupt Override Test.....                     | B-22 |
| B.4 Network Tests .....                                         | B-23 |
| B.4.1 Multiple Node Communication Test.....                     | B-24 |
| B.4.2 Multiple Node Burst Buffer Test .....                     | B-27 |
| B.4.3 Network Communication Test .....                          | B-28 |
| B.4.4 HCT Compatible Communication Test.....                    | B-29 |

# FIGURES

|                                                               |      |
|---------------------------------------------------------------|------|
| Figure B-1 Longword (32-bit) Rolling 1's Pattern.....         | B-2  |
| Figure B-2 Longword (32-bit) Rolling 0's Pattern.....         | B-2  |
| Figure B-3 Shortword Rolling 1's Pattern .....                | B-3  |
| Figure B-4 Shortword Rolling 0's Pattern .....                | B-3  |
| Figure B-5 Byte Rolling 1's Pattern.....                      | B-4  |
| Figure B-6 Byte Rolling 0's Pattern.....                      | B-4  |
| Figure B-7 Longword Alternating Bit Pattern.....              | B-5  |
| Figure B-8 ACR Byte Rolling 1's Pattern.....                  | B-5  |
| Figure B-9 ACR Byte Rolling 0's Pattern.....                  | B-6  |
| Figure B-10 ACR Bit Pattern Test.....                         | B-7  |
| Figure B-11 Data Filtering Test Algorithm.....                | B-9  |
| Figure B-12 Transmit FIFO Test Algorithm.....                 | B-10 |
| Figure B-13 Byte Swapping Algorithm .....                     | B-11 |
| Figure B-14 Interrupt FIFO Test Algorithm .....               | B-12 |
| Figure B-15 GPC Iteration Algorithm .....                     | B-13 |
| Figure B-16 GPC Mode Test Algorithm.....                      | B-14 |
| Figure B-17 Error Mask Test Iteration Algorithm .....         | B-15 |
| Figure B-18 Error Mask Setup and Verification Algorithm ..... | B-16 |
| Figure B-19 Transmit and Receive Test Algorithm .....         | B-17 |
| Figure B-20 Memory Loopback Test Algorithm .....              | B-18 |
| Figure B-21 HIPRO With Data Filter Test .....                 | B-19 |
| Figure B-22 Virtual Paging Test.....                          | B-20 |
| Figure B-23 Breakdown of Test Results .....                   | B-21 |
| Figure B-24 Receive ID/Age Set Test Algorithm .....           | B-21 |
| Figure B-25 Receive Interrupt Override Test Algorithm.....    | B-22 |
| Figure B-26 Read/Write Pattern .....                          | B-24 |
| Figure B-27 Multiple Node Communication Test Algorithm .....  | B-26 |
| Figure B-28 Multiple Node Burst Buffer Test Algorithm.....    | B-27 |
| Figure B-29 Network Communication Test Algorithm.....         | B-28 |
| Figure B-30 Read/Write Pattern .....                          | B-29 |
| Figure B-31 HCT Compatible Communication Test Algorithm ..... | B-31 |

*This page intentionally left blank*

## B.1 Overview

The SCRAMNet Diagnostics are a collection of various error detection procedures designed to debug hardware setup, detect possible hardware errors and evaluate performance of the SCRAMNet network boards.

The tests cover three major areas.

- Memory Tests (Page B-1)
- Loop Tests (Page B-8)
- Network Tests (Page B-23)

Each of these three areas contain a series of test procedures that are discussed in detail.

## B.2 Memory Tests

The following Memory Tests are available:

- Longword rolling 1's test
- Longword rolling 0's test
- Shortword rolling 1's test
- Shortword rolling 0's test
- Byte rolling 1's test
- Byte rolling 0's test
- Alternating bit pattern test
- Auxiliary Control RAM (ACR) rolling 1's test
- Auxiliary Control RAM (ACR) rolling 0's test
- Auxiliary Control RAM (ACR) bit pattern test
- Address line test

SCRAMNet supports most systems with three types of transactions to its shared memory region. They are Longword (32-bit), Shortword (16-bit), and Byte (8-bit) addressing modes. The memory tests can perform an operating verification check on each of the addressing modes as well as on each of the address lines themselves.

All network communications are disabled when memory tests are run. This way no other node may interfere with the test on the node being tested, and no extraneous network traffic is generated.

### B.2.1 Longword Rolling 1's Test

This longword addressing mode option tests for data and address line integrity. A rolling one pattern is written to the low address in memory and then verified. This bit is shifted one place from the LSB toward the MSB and written to the next consecutive longword address in memory. The remaining bits are '0'. Each of the memory-write operations is verified to detect any write errors.

Following the last write to memory, a comparison is made with the rolling pattern of the entire memory block. This verifies data integrity of the address line. The starting rolling pattern is shifted by one for the next iteration. This procedure is repeated for as many iterations as specified. The rolling pattern is described in Figure B-1.

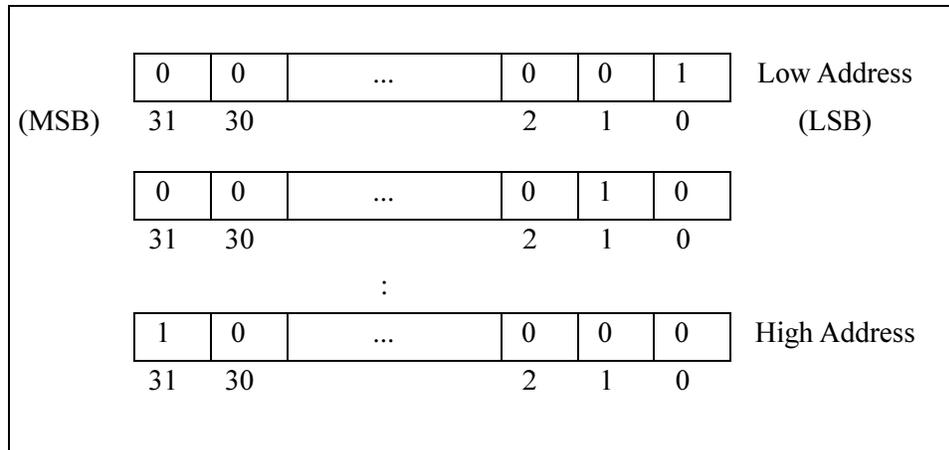


Figure B-1 Longword (32-bit) Rolling 1's Pattern

## B.2.2 Longword Rolling 0's Test

This option tests the memory and address line data integrity. The only difference between the Longword Rolling one and this test is that instead of a '1' being shifted through the entire address range, a '0' is used, with the remaining bits being '1'. The range of iterations is between 0-99999 (decimal). Figure B-2 is an illustration of the pattern used for this test.



**NOTE:** Some VME systems may be set up to operate in 16-bit mode. Therefore, true 32-bit operation does not take place. Longword tests are not available on these nodes.

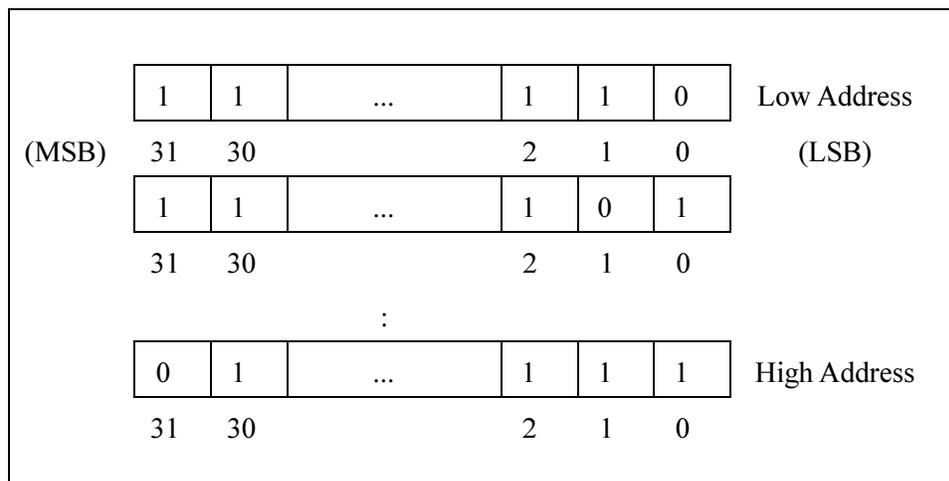


Figure B-2 Longword (32-bit) Rolling 0's Pattern

### B.2.3 Shortword Rolling 1's Test

This is the Shortword (16-bit) addressing mode version of the rolling one test pattern. A rolling one pattern is written to the low address in memory and then verified. This bit is shifted one place from the LSB toward the MSB and written to the next address in memory. Each of these memory write operations is immediately verified to detect any write errors.

Following the last write to memory, a comparison is made with the rolling pattern of the entire memory block. This verifies data integrity of the address line. The beginning rolling pattern is shifted by one for the next time around. This procedure is repeated for as many iterations as specified. The rolling pattern is described in Figure B-3.

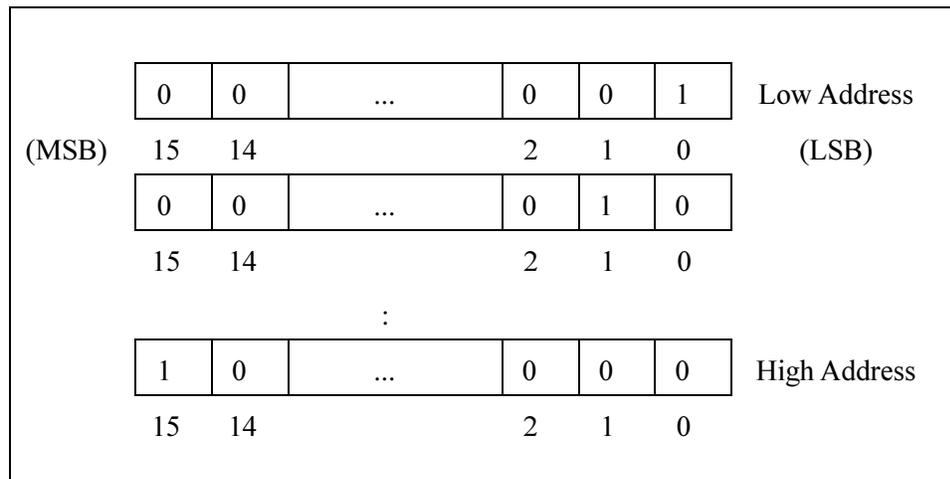


Figure B-3 Shortword Rolling 1's Pattern

### B.2.4 Shortword Rolling 0's Test

This option is identical to the shortword rolling one pattern, only a '0' is shifted within the address range instead of a '1'. Figure B-4 illustrates this pattern.

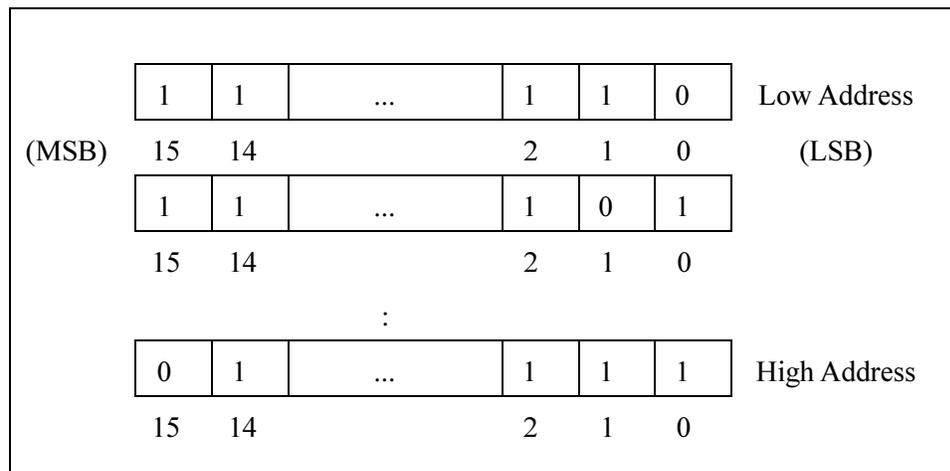


Figure B-4 Shortword Rolling 0's Pattern

## B.2.5 Byte Rolling 1's Test

This is a Byte version of the rolling 1's pattern error detection technique. Instead of the Longword (32-bit) or the Shortword (16-bit), this is the case for a Byte (8-bit) operation. The same principle applies as described for previous techniques. A '1' is shifted through every bit of every byte (8-bit) of the address range. The pattern is checked after every write for address line errors or extraneous memory writes, if any. This procedure is repeated for as many iterations as specified at the Memory Test menu. Figure B-5 illustrates this test pattern.

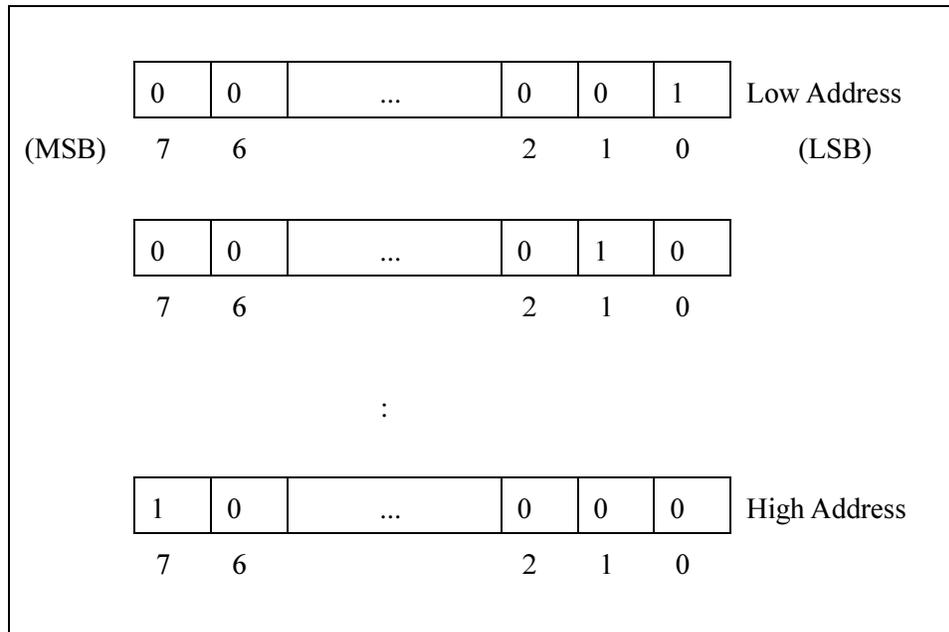


Figure B-5 Byte Rolling 1's Pattern

## B.2.6 Byte Rolling 0's Test

This option is identical to the byte rolling 1's pattern, only that a '0' is shifted within the address range instead of a '1'. Figure B-6 illustrates the byte size rolling '0' pattern.

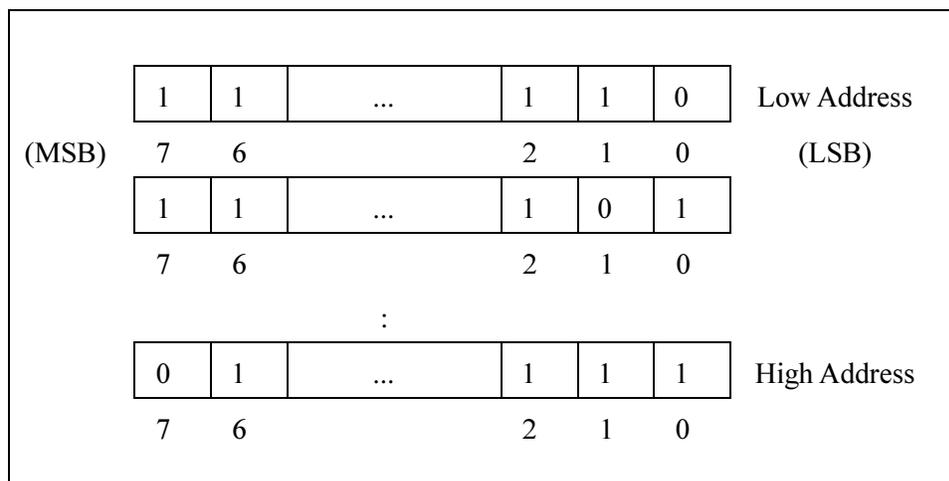


Figure B-6 Byte Rolling 0's Pattern

## B.2.7 Alternating Bit Pattern Test

The purpose of this test is to test memory and address line data integrity. This selection is different from previous tests in terms of the data format applied while testing. An alternating 0-1 pattern is written to each memory address. The data is checked after each write operation. A second alternating pattern is then written to every other memory address. Finally, the entire memory block is checked for any errors in matching the pattern. This test is repeated for the specified number of iterations. Figure B-7 describes the data format.

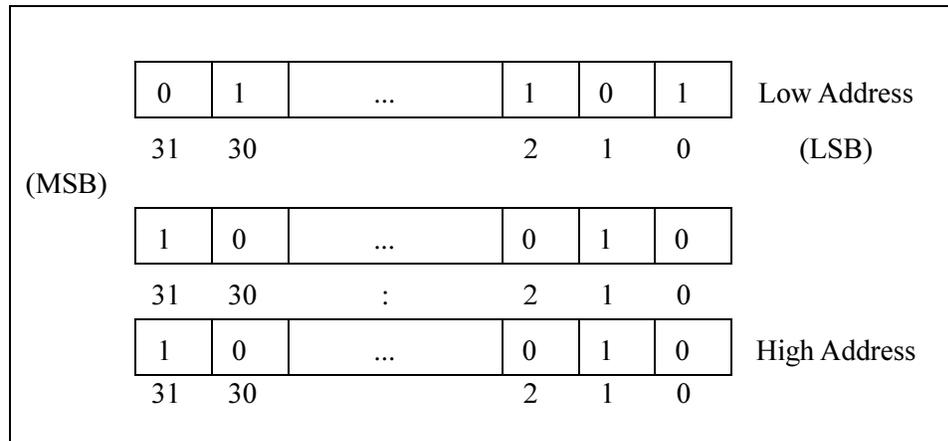


Figure B-7 Longword Alternating Bit Pattern

## B.2.8 ACR Byte Rolling 1's Test

This selection performs a bit-by-bit ACR Memory Test. The ACR is an 8-bit-wide memory, which occupies the same address space as regular SCRAMNet memory and controls certain events for the 32-bit memory location it occupies.

A rolling 1's pattern is written to an ACR low memory address. The write operation is followed by a read to verify the pattern. This procedure can uncover an error in a memory write operation. The bit is then shifted one place from the LSB toward the MSB and written to the next consecutive longword address in the ACR memory. This procedure is repeated for the entire ACR address range. Figure B-8 describes the rolling 1's pattern.

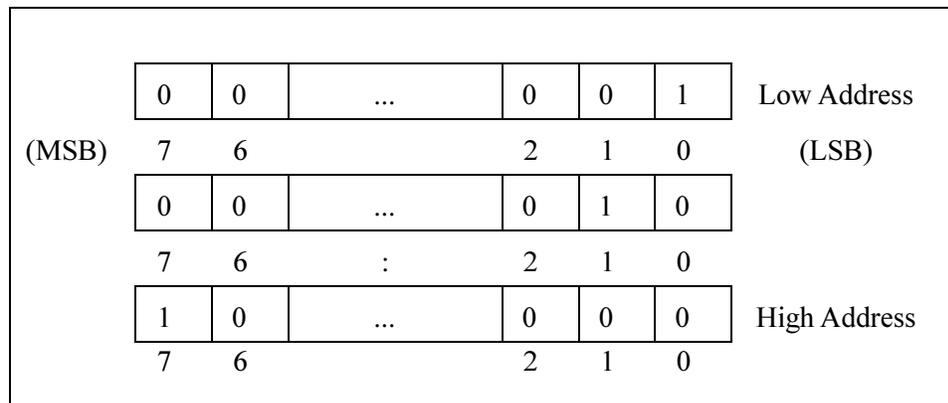


Figure B-8 ACR Byte Rolling 1's Pattern

After writing into the last address, the entire ACR Memory block is compared to a rolling 1's pattern to detect any address line error. This procedure is repeated for the number of iterations specified.

### B.2.9 ACR Byte Rolling 0's Test

This is identical to the ACR byte rolling 1's option. A rolling 0's pattern is shifted, bit by bit, through the address range to detect errors. Figure B-9 describes this pattern.



**NOTE:** The ACR Rolling 1's and Rolling 0's tests only verify the accessibility of the ACR for read and write transactions. Neither of these tests actually verify the functionality of the ACR bits themselves. Other tests which do address the functionality of the ACR bits are discussed later.

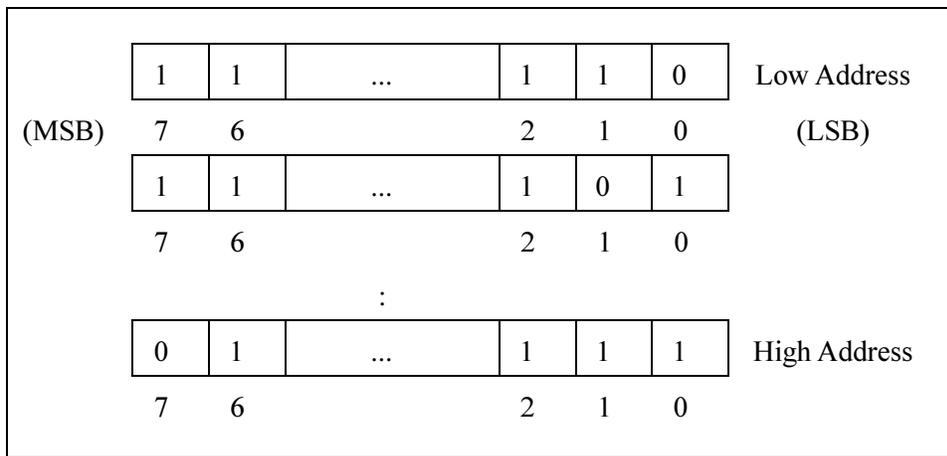


Figure B-9 ACR Byte Rolling 0's Pattern

### B.2.10 ACR Bit Pattern Test

The purpose of this test is the same as the ACR rolling 1's and rolling 0's tests: to test the accessibility of the ACR for reads and writes. It follows the same pattern switching technique performed on regular SCRAMNet memory as described in the previous test. The patterns used are illustrated in Figure B-10.

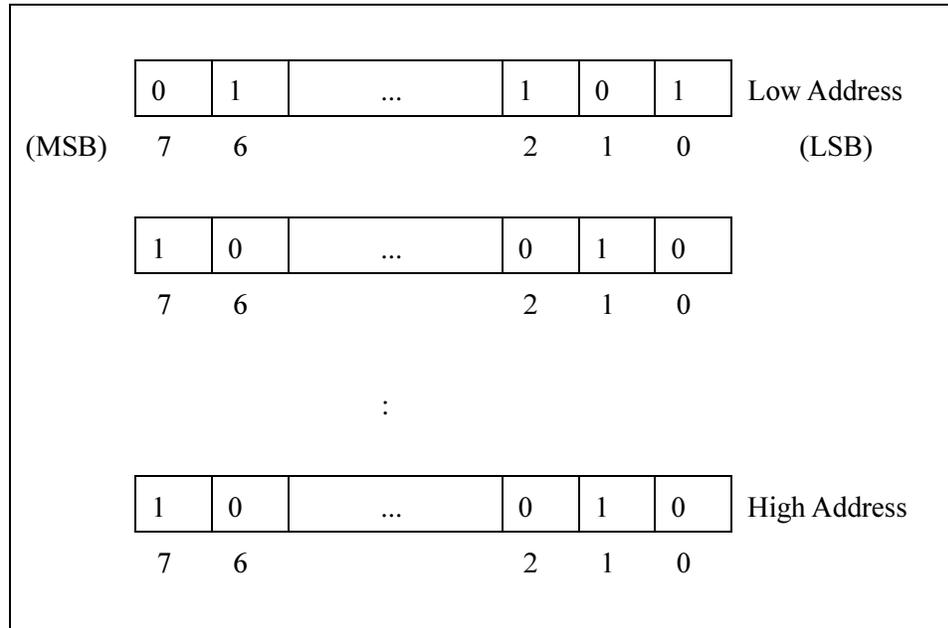


Figure B-10 ACR Bit Pattern Test

### B.2.11 Address Line Test

This test exercises all of the data lines and address lines. The entire SCRAMNet memory is set to zeros at the local node. A hexadecimal value of 'AA' is written to byte 0 of the SCRAMNet memory. The remainder of the SCRAMNet memory is then scanned to see if the 'AA' pattern shows up anywhere else in memory. SCRAMNet memory is zeroed again and the 'AA' pattern is written to byte 1. All of SCRAMNet memory is then scanned again to see if the 'AA' pattern shows up anywhere else. This procedure continues with the SCRAMNet memory byte address changing to 2, then 4, 8, 16, and so on, until the upper address boundary is reached for the particular memory size.

## B.3 Loop Tests

The following tests are found in the Loop Tests menu:

### On-board Tests:

- Data filter Test (Page B-8)
- Shared Memory FIFO (or Transmit FIFO) Test (Page B-10)
- Byte swapping Test\* (Page B-11)
- Interrupt FIFO (or Receive FIFO) Test (Page B-12)
- Interrupt Test (Page B-13)
- General Purpose Counter Test (Page B-13)
- Error Mask Test (Page B-15)

\* This test is only used with SCRAMNet Classic.

### Network Tests:

- Transmit and Receive Test (Page B-17)
- Memory Loopback Test (Page B-18)
- HIPRO Test with Data Filter (Page B-19)
- Virtual Page Test (Page B-20)
- Receive ID/Age Set Test (Page B-21)
- Receive Interrupt Override Test (Page B-22)

The Loop tests require some method of “looping” data out of the node and back via the Internal Wire loop, Fiber Optic Loopback (possibly using the Fiber Optic Bypass Switch), or the Mechanical Switch loop). Most of these tests require data to be transmitted and received via one of the data-looping mechanisms described to test the SCRAMNet features.

The method of looping is menu-selectable unless a Fiber Optic Bypass Switch is detected by the node, in which case the switch is used as the loop.



**NOTE:** The tests are designed for a single node as a stand-alone system. No other node may be communicating on the ring if the fiber optic loop is selected.

Wire Loopback mode and Mechanical Switch Loopback mode require no hardware manipulations for the Loop tests to function properly. The Fiber Optic Loopback mode requires the installation of either a Fiber Optic Bypass Switch, or a pair of fiber optic cables running from transmit (Tx) to receive (Rx) of the node being tested. If a Fiber Optic Bypass Switch is installed correctly, it is detected by the software and automatically used for the Loop tests.

### B.3.1 Data Filter Test

This test determines whether the proper filtering function has been performed using the Interrupt FIFO (previously known as Receive FIFO). Although this test is grouped with the loopback tests, the test is run internal to the node being tested.

Data Filtering is a feature of the hardware that compares data being written to a shared memory address with the contents of that memory location to avoid sending unchanged

data out on the network. Three levels of data filtering exist—OFF, HI, and ALL—and each is covered by this test.

**OFF:** All data filtering is DISABLED. All data written to the SCRAMNet shared memory is transmitted on the ring, provided the node is properly inserted.

**HI:** Data filtering is ENABLED on the portion of SCRAMNet memory above the lowest 4 KB. This means that anything written to the lowest 4 KB of shared memory is always transmitted on the ring, but anything above that is checked against the current contents of memory, and filtered accordingly.

**ALL:** Data filtering is ENABLED on all of SCRAMNet shared memory.

The control bits for the data-filter modes are located in CSR0[10] and CSR0[11]. A write to both the high-memory and low-memory segments is performed for each of the above filter modes. The value found in CSR1 is used to determine the status of the Transmit FIFO.

Figure B-11 is a representation of the algorithm used to test the Data Filtering functionality.

```

Disable Tx (Transmitter)
Do for all cases( OFF, HI, ALL )
  Do for each Longword in SCRAMNet memory
    Clear Shared Memory FIFO flags
    Write Test pattern to current location
    Clear Shared Memory FIFO flags
    Write Test pattern to current location
    If case = OFF then
      If Shared Memory FIFO Empty then ERROR
    Else If case = HI then
      If word is in low 4K of memory
        If Shared Memory FIFO Empty then ERROR
      Else
        If Shared Memory FIFO NOT Empty then ERROR
      End If
    Else If case = ALL then
      If Shared Memory FIFO NOT Empty then ERROR
    End If
  End Do
End Do

```

**Figure B-11 Data Filtering Test Algorithm**

### B.3.2 Shared Memory (Transmit) FIFO Test

The Transmit FIFO controls the following three status bits in CSR1.

Bit 0: Transmit FIFO Full (1024 32-bit Words)

Bit 1: Transmit FIFO Not Empty

Bit 2: Transmit FIFO Almost (7/8) Full



**NOTE:** On SCRAMNet Classic boards, CSR1[2] reflects 1/2 full, not 7/8.

The Transmit FIFO Test uses CSR1, bits 0, 1 and 2 to determine the status of the Transmit FIFO and verify its functionality.

Both Transmit CSR0[1] and Host to Shared Memory Write CSR2[8] are disabled for the first part of this test. This forces all shared memory writes to be queued in the Transmit FIFO.

A series of shared-memory writes is then performed while monitoring the status bits. When the bits are set, the number of writes that caused each bit to be set is checked against the expected value (within a very small tolerance range). Once the Transmit FIFO is full, the Transmit bit is enabled to allow the data in the Transmit FIFO to be written to memory. Memory is then checked to verify the data is correct and at the proper address.

Figure B-12 is the algorithm used to test the Transmit FIFO.

```

Choose Internal wire loopback or set up fiber optic
Reset Tx FIFO
Disable Transmitter and Host to shared memory write
Do For Pass = 0 to Pass = FIFO Size
    Write test pattern to shared memory location
    If Pass = 0 and Tx FIFO Not Empty flag not set then ERROR
    If Pass = Almost (1/2 or 7/8) Full +/- 3 words then
        If Tx FIFO Almost Full flag not set then ERROR
        If Pass = FIFO Size-2 and Tx FIFO Full not set then ERROR
    End Do
Enable transmit and receive bits in CSR #0
Enable Write own slot
Do For Pass = 0 to Pass = FIFO Size-1
    Delay to allow data to transmit
    If value at memory location <> test pattern then ERROR
    Go to next memory location
End Do

```

**Figure B-12 Transmit FIFO Test Algorithm**

### B.3.3 Byte Swapping Test (Used only with SCRAMNet Classic)

The Programmable Byte Swapper is tested with this selection.

This test is grouped under the loopback tests even though the test is run internal to the node tested and no data is looped.

The Byte Swapper is controlled through six bits in CSR2. They are:

- Bit 0: Byte Transaction Byte Swap
- Bit 1: Byte Transaction Word Swap
- Bit 2: Word Transaction Byte Swap
- Bit 3: Word Transaction Word Swap
- Bit 4: Long Transaction Byte Swap
- Bit 5: Long Transaction Word Swap

The Byte Swapper has the ability to control the integer data reordering or reformatting. The swapping is based on the size of the transaction performed on the SCRAMNet shared memory. This is very useful for sharing data between systems that employ different byte ordering schemes; that is, Big Endian versus Little Endian.

This test checks each of the available bus transaction sizes, which are longword, shortword, and byte, with every possible combination of swapping. In each of these cases a pattern is written to a memory location. The pattern is then manipulated by the three possible swapping options. The pattern is then read back and compared to the value expected for the current swapping mode. For example in the case of a longword transaction:

|                      |             |
|----------------------|-------------|
| Initial Pattern:     | 1A 2B 3C 4D |
| Byte Swap:           | 2B 1A 4D 3C |
| Shortword Swap:      | 3C 4D 1A 2B |
| Byte/Shortword Swap: | 4D 3C 2B 1A |

Figure B-13 is the algorithm used to test the byte swapping functionality.

```

For each transaction type (Byte, Shortword, Longword)
  For each swap operation (Byte, Shortword, Byte/Shortword)
    Write test pattern to a memory location
    Perform swap utilizing CSR2 bits
    Verify the swap took place only for the requested transaction size.
    Test data integrity with rolling 1's pattern
  End For
End For

```

**Figure B-13 Byte Swapping Algorithm**

### B.3.4 Interrupt FIFO Test

This test is similar to the Transmit FIFO test in that both use CSR bits to determine the status of the FIFO in question and both verify the data within the FIFO itself. The CSR bits used to determine the status of the Interrupt FIFO are as follows:

CSR1[4]: Interrupt FIFO Full (1024 32-bit Network Address Offsets)  
 CSR5[15]: Interrupt FIFO not Empty

All of CSR4[6:0] and CSR5[6:0], hold the top value in the FIFO, which is updated as each is read by the host processor.

The SCRAMNet board must be initialized correctly for this test to accumulate values in the Interrupt FIFO. This set-up requires several key steps.

- The ACR locations that will generate the interrupt messages must be set.
- The node must be set up to Disable Host to Memory write to SCRAMNet memory directly (CSR2[8], but also allow the node to receive its own network messages by enabling Write Own Slot CSR2[9].
- Network interrupts must be enabled.

Figure B-14 is the algorithm for the Interrupt FIFO test.

```

Choose Internal wire loopback or set up fiber optic
Disable Host Shared Memory write          CSR2[8]
Enable Write own slot                      CSR2[9]
Enable Interrupt own slot                  CSR2[10]
Enable Host interrupt                     CSR0[3]
Enable Interrupt Mem Mask Match           CSR0[5]
Enable Network Interrupt                  CSR0[8]
Choose a series of 1024 random addresses and data values
Set ACR bits to generate and receive interrupt messages at these addresses
Reset the interrupt FIFO and verify
Enable transmit and receive (and insert if fiber loop)
Do While number of data writes < FIFO length
  Write data value to next random address as selected above
  If INT FIFO goes full and count < FIFO length then ERROR
End Do
If INT FIFO not FULL then ERROR
Do for length of Interrupt FIFO
  If address in FIFO does not match locations written in order
  then ERROR
  Check data value written to location for integrity
End Do

```

**Figure B-14 Interrupt FIFO Test Algorithm**

### B.3.5 Interrupt Test

The Interrupt test is identical to the Interrupt FIFO test with the exception that the number of interrupts is random between 1 and 1024. This means that each iteration of the test may or may not fill the Interrupt FIFO.

### B.3.6 General Purpose Counter Test

The purpose of this test is to verify correct operation of the General Purpose Counter and all of its programmable modes. The programmable counter modes tested are:

- Count Errors
- Count Triggers
- Count Transit Time
- Count Network Events
- Free Run Counter (26.66 ns)
- Free Run W/Trigger 2 Clear (1.706  $\mu$ s)

Figure B-15 is the algorithm for each iteration of the General Purpose Counter test.

```

Reset FIFOs
For each counter mode in list above
  Set up counter mode
  Clear the general purpose counter
  Run the test for counter mode
  Verify the value expected with the general purpose counter
End For

```

**Figure B-15 GPC Iteration Algorithm**

The Free Run Counter tests are somewhat unique in that they do not give a pass or fail answer, rather they print out their values to the **scr\_diag.txt** file for the user to evaluate. If the counter seems to be incrementing during the Free Run Counter test, then the counter is probably okay.

There is no way to say how much the counter should be incrementing on each read, because the time elapsed between reads can be vastly different on different machines. If the counter is working, those values will be incrementing at approximately the same magnitude for each read.

Similarly, the time displayed between reads during the Free Run W/Trigger 2 Clear tests should be relatively close because the counter is cleared after each read.

The algorithm to set up and verify correct operation of each counter modes is dependent on the specific mode being tested. Figure B-16 is the algorithm for setting up, testing, and verifying each mode.

*Count Errors*

*Set Counter Mode to Count Errors*  
*Set Error Mask so only Transmit FIFO not empty generates errors*  
*Send a number of messages to the network*  
*Verify the counter value is the same number of messages sent*

*Count Triggers*

*Set Counter Mode to Count Triggers*  
*Set ACR for 2 memory locations (1 - Trigger 1, 2 - Trigger 2)*  
*Write to locations 1 and 2 and a third location with no ACR bits set*  
*Verify counter = num writes to location 1 + num writes to location 2*

*Count Transit Time*

*Set Counter Mode to Count Message Transit Time*  
*For several iterations (currently 3)*  
     *Send a Message to the network*  
     *Read the counter and print this value to the log file*

*End For**Count Network Events (network writes to shared memory)*

*Set Counter Mode to Count Network Events*  
*Set Write-Me-Last Mode*  
*Send a number of messages to the network*  
*Verify that the counter value = number of messages sent*

*Counter Free Run (26.66ns)*

*Set Counter Mode to Free Run @ 26.66ns*  
*For several iterations (currently 3)*  
     *Read the counter and print this value to the log file*

*End For**Counter Free Run W/Trigger 2 clear (1.706us)*

*Set Counter Mode to Free Run @ 1.706us W/Trigger 2 clear*  
*Set a memory location's ACR for Trigger 2*  
*Set Write Me Last Mode*  
*For several iterations (currently 3)*  
     *Write to the memory location to generate a Trigger 2 clear*  
     *Read the counter and print this value to the log file*

*End For**For several iterations (currently 3)*

*Write to non Trig 2 memory location (counter should not clear)*  
     *Read the counter and print this value to the log file*

*End For***Figure B-16 GPC Mode Test Algorithm**

### B.3.7 Error Mask Test

This test will verify correct operation of the error masking feature. The testable error conditions are:

- Transmit FIFO Full
- Transmit FIFO Not Empty
- Transmit FIFO 7/8th Full
- Interrupt FIFO Full
- Redundant Transmitter/Receiver Fault (this test is only valid if a redundant link is detected)

Each of these error conditions is tested by using the General Purpose Counter to count errors. This method of testing for errors excludes the General Purpose Counter/Timer Overflow condition from being tested. Message Transmit and Receive Enable are turned off in order to fill the Transmit FIFO. Figure B-17 is the algorithm for each iteration of the Error Mask Test.

```

Turn off Message Transmit and Receive Enable
For each testable error mask above
  Reset FIFOs
  Set up error mask test
  Clear the General Purpose Counter
  Run test and verify correct operation with counter value
End For

```

**Figure B-17 Error Mask Test Iteration Algorithm**

The algorithm to set up and verify correct operation of each error mask is dependent on the specific mask being tested. Figure B-18 is the algorithm for setting up, testing, and verifying each error mask.

```
Transmit FIFO Full Mask Test
Set Transmit FIFO Full Mask
For some number of iterations (currently 15)
    Send enough messages to fill the Transmit FIFO
    Reset the FIFOs
End For
Verify that the counter has the value expected (15)

Transmit FIFO Not Empty Mask Test
Set Transmit FIFO Not Empty Mask
For some number of iterations (currently 15)
    Write a value to shared memory
    Reset the FIFOs
End For
Verify that the counter has the value expected (15)

Transmit FIFO 7/8 Full Mask Test
Set Transmit FIFO 7/8 Full Mask
For some number of iterations (currently 15)
    Write messages to memory until FIFO almost full (1024 - 1)
    Reset the FIFOs
End For
Verify that the counter has the value expected (15)

Interrupt FIFO Full Mask Test
Set Interrupt FIFO Full Mask
Set write-me-last mode
Set interrupt on receipt of own message
Set all received messages to generate interrupts
For some number of iterations (currently 15)
    Reset FIFOs
    Enable Transmit and Receive Messages and Receive Interrupts
    Send enough messages to fill Receiver FIFO (1024)
End For
Verify that the counter has the value expected (15)

Redundant TXRX Fault Mask Test
Set Redundant TXRX Fault Mask
For some number of iterations (currently 15)
    Toggle Redundant Link
End For
Verify that the counter has the value expected (15)
```

**Figure B-18 Error Mask Setup and Verification Algorithm**

### B.3.8 Transmit and Receive Test

This test is designed to test the functionality of the SCRAMNet Board transmitter/receiver. Disable Host to Memory Write CSR2[8] and Write Own Slot Enable CSR2[9] must be set to ensure that the data written to shared memory was looped through the transmit/receive circuitry. The Transmit FIFO and shared memory will be used to verify transmitter/receiver operation.

Figure B-19 is the algorithm for the test.

```
In CSR0 set receive ON and transmit OFF
Disable Host shared memory write
Enable network write own slot
Write a test pattern to shared memory
If Transmit FIFO empty then ERROR
In CSR0 set receive OFF and transmit ON
If shared memory FIFO not empty then ERROR
If memory location = test pattern then ERROR
In CSR0 set receive ON and transmit ON
Write a test pattern to shared memory
If memory location not = test pattern then ERROR
Set up selected loopback mode
For all memory locations
    Write test pattern to shared memory location
    Delay wait for data to loop back
    If memory location value not = test pattern then ERROR
    Go to next shared memory location
End For
```

**Figure B-19 Transmit and Receive Test Algorithm**

### B.3.9 Memory Loopback Test

The Memory Loopback test is designed to ensure the accuracy of data which has traversed the network. This is done the same way as in the Transmitter/Receiver test; by setting Disable Host to Memory Write CSR2[8] and Write Own Slot Enable CSR2[9]. In this “Write-Me-Last” configuration, data must be transmitted to and received by all other nodes on the network before it is written to the host-node shared memory.

Once the test data is received by the originating node it is compared with the original data value for errors. The rolling 1’s method of generating test data is used for this test in exactly the same way as for the Memory tests.

Figure B-20 is the algorithm for the Memory Loopback test.

```
Disable Host to Shared Memory Write
Enable Write own Slot
Enable selected loopback mode
Do For each shared memory location
    Write test pattern to location
    Delay to allow data to cycle network
    If location not = test pattern then ERROR
    Go to next shared memory location
End Do
Do For each shared memory location
    If location not = expected test pattern then ERROR
    Go to next shared memory location
End Do
```

**Figure B-20 Memory Loopback Test Algorithm**

### B.3.10 HIPRO Test with Data Filter

HIPRO communications mode was intended to support devices which are limited to 16-bit accesses to SCRAMNet shared memory, such as ISA Bus or VME3U. Remember that SCRAMNet shared memory is set up in 32-bit words.

Figure B-21 is the algorithm for the HIPRO With Data Filter Test.

```

Set ACR HIPRO bit for every Shared Memory location except the last
Enable HIPRO (CSR2[13])
Set all memory locations to known value
Write to last location to sync HIPRO flags
Disable host to shared memory write
Disable transmit
Enable Data Filtering on all of shared memory
For each location in shared memory except last
    Write test pattern to 1st byte (8 bits) of location.
    If Transmit FIFO NOT empty then HIPRO ERROR
    Write test pattern to 1st and 2nd byte of location.
    If Transmit FIFO NOT empty then HIPRO ERROR
    Write test pattern to 1st, 2nd, and 3rd byte of location.
    If Transmit FIFO NOT empty then HIPRO ERROR
    Write test pattern to 1st, 2nd, 3rd, and 4th byte of location.
    If Transmit FIFO EMPTY then HIPRO ERROR
    Enable transmit to release data in Transmit FIFO
    Disable Transmit
    Write test pattern to the first 16 bits of a 32-bit location
    If Transmit FIFO NOT empty then HIPRO ERROR
    Write test pattern to first and second 16 bits of the 32-bit location
    If Transmit FIFO EMPTY then HIPRO ERROR
    Enable transmit to release data in Transmit FIFO
    Disable Transmit
    Start with opposite half of 32-bit location and write 16 bits using a new test pattern
    If Transmit FIFO NOT empty then HIPRO ERROR
    Write both 16-bit words of the 32-bit location
    If Transmit FIFO EMPTY then HIPRO ERROR
    Enable transmit to release data in Transmit FIFO
    Disable Transmit
    Write same pattern as last write and write first 16 again
    If Transmit FIFO NOT empty then HIPRO ERROR
    Write first and second half of 32-bit location with same pattern
    If Transmit FIFO NOT empty then DATA FILTER ERROR
    Go to next location
End For
Verify data against the previous pattern except last location.
Verify that last (non-HIPRO) location transmits regardless which portion written
  
```

**Figure B-21 HIPRO With Data Filter Test**

Normally, any 8- or 16-bit write to SCRAMNet memory will result in the entire 32-bit word containing the data to be sent out on the network. HIPRO mode requires the full 32-bit word to be written before it is transferred. This translates to a single network write for two host writes which greatly reduces network traffic and increases throughput.

While testing the HIPRO mode it is necessary to verify that HIPRO is not overridden by data filtering.

### B.3.11 Virtual Page Test

This test is not available for SCRAMNet Classic nodes. This feature is also not available for those nodes with the maximum 8 MB memory.

Virtual paging allows nodes with less than the maximum amount of memory to occupy any portion of the 8 MB possible SCRAMNet network address space provided the address chosen is a multiple of its own memory size. For example: A 4 KB board may occupy the SCRAMNet address space from 0 to 4 KB, 4 KB to 8 KB, 8 KB to 12 KB, etc. up to 8 MB. A 128 KB board may occupy the SCRAMNet address space 0 to 128 KB, 128 KB to 256 KB, etc. up to 8 MB.

This option only translates the SCRAMNet address, not the actual physical address of the board. The Interrupt FIFO will receive all messages sent by the node in Wire Loopback mode and store the addresses which are checked against the expected translation, based on the current virtual page. The number of virtual pages checked will depend on the size of the memory of the board being tested.

Figure B-22 is the algorithm to test the virtual page feature.

```

Make sure board is not more than 4 MB
Reset all FIFOs
Set the Interrupt receive override in CSR8
Set CSR0 for transmit, receive
Enable memory mask and override receive interrupt flag
Enable Write own slot and interrupt own slot
For each virtual page available to the node.
    Set virtual page to next available
    For each location in memory
        Write a test pattern to location
        Wait for receiver FIFO to be not empty
        Check the translation of the address in the FIFO against the current virtual page
        If translation not = (address + virtual page offset) then ERROR
    End For
End For

```

**Figure B-22 Virtual Paging Test**

### B.3.12 Receive ID/Age Set Test

This test checks the operation of a separate transmitter and receiver ID. This test will include all combinations of the following:

- Transmit ID equal to Receiver ID - YES, NO
- Write Own Slot Enable bit - 0, 1
- Disable Host to Memory Write - 0, 1

The behavior of each combination will be checked by the memory pattern (if any) that was written and the number of times the message was received (determined by the value of MSG\_LIFE). MSG\_LIFE is defined as 0xFF - initial message age. The node will be set up to receive an interrupt on receipt of its own message and the Interrupt FIFO is read until empty to determine the number of times the message is received. The Burst mode protocol is used in case the message is never received by the node. Figure B-23 is a breakdown of each test case and the expected state upon completion. Figure B-24 is the algorithm for each iteration of the Receive ID/Age Set Test.

| Test Combination |           |       |      | Determination of Success |                |
|------------------|-----------|-------|------|--------------------------|----------------|
| Case             | TXID=RXID | WOSEN | DHMW | Written to Memory        | Times Received |
| 0                | NO        | 0     | 0    | YES                      | MSG_LIFE       |
| 1                | NO        | 0     | 1    | YES                      | MSG_LIFE       |
| 2                | NO        | 1     | 0    | YES                      | MSG_LIFE       |
| 3                | NO        | 1     | 1    | YES                      | MSG_LIFE       |
| 4                | YES       | 0     | 0    | YES                      | 0              |
| 5                | YES       | 0     | 1    | NO                       | 0              |
| 6                | YES       | 1     | 0    | YES                      | 1              |
| 7                | YES       | 1     | 1    | YES                      | 1              |

**Figure B-23 Breakdown of Test Results**

The setup of each test case is exactly as it appears in Figure B-23. There are two steps in the verification of each test case:

- Read the memory location and determine if it is the value expected.
- Read the Interrupt FIFO to determine if the message was received the expected number of times.

*For each test case above*  
*Reset FIFOs*  
*Set up test case*  
*Write a pattern to memory*  
*Delay until message is removed from the network*  
*Verify that the test case completed successfully*  
*End For*

**Figure B-24 Receive ID/Age Set Test Algorithm**

### B.3.13 Receive Interrupt Override Test

This test will exercise the Receive Interrupt Override CSR8[10]. This bit makes every network message received appear to the node to be an interrupt regardless of whether the interrupt bit in the message is set. It is the equivalent of setting the ACR Receive Interrupt Enable for all memory locations.

The test sets the Override Receive Interrupt Enable CSR0[6] and Interrupt on Memory Mask Match Enable CSR0[5] to allow the node to receive the interrupt message through the Interrupt FIFO. It also sets the Write Own Slot Enable CSR2[9] and Enable Interrupt on Own Slot CSR2[10] to allow the node to receive its own message. The test is always run in wire loopback.

Figure B-25 is the algorithm used to test the Receive Interrupt Override bit. This test is similar to the Virtual Page test.

```
Reset all FIFOs
Set the Interrupt receive override in CSR8
Set CSR0 for transmit, receive
Enable memory mask and override receive interrupt flag
Enable Write own slot and interrupt own slot
For each location in memory
    Write a test pattern to location
    Wait for receiver FIFO to be not empty
    Check the address in the FIFO against the expected address
    If address not = address of location written
        or FIFO remains empty
        then ERROR
End For
```

**Figure B-25 Receive Interrupt Override Test Algorithm**

## B.4 Network Tests

The Network Test is designed to test the SCRAMNet Network system in a network ring configuration. This section will explain how each test verifies the functionality of the nodes installed on the network ring. The tests in the Network test menu are:

- Multiple Node Communication test (Page B-24)
- Multiple Node Burst mode test (Page B-27)
- Network Communication test (Page B-28)
- HCT Compatible Communication test (Page B-29)

Once the physical networking hardware installation is completed, run the network tests to verify the proper configuration of the installed SCRAMNet network boards and fiber optic cables. These tests may also be run by a single node. In this case the tests become loopback tests.

There are two prerequisites for running the network tests.

- Network fiber optic or coaxial cable connections must be in place before these tests are run. One way to verify these connections are correct is to observe the Carrier Detect LED. If it is on, there is a good connection around the ring.
- Each node must also have a unique Node ID.

There may be other problems if the nodes have different memory sizes around the ring.



**NOTE:** If node shared-memory size varies among the nodes being tested, the end results will usually be test failure.

The problems arising from varying memory sizes vary themselves. For the communication test, the nodes will most likely time out on a location. The node with the least amount of memory will most likely time out on location zero. The others will time out on the location beyond the end of the node with the least amount of memory. The other network tests will fail with “bad data” errors because the memory will not be divided evenly between them. This may create conflicts because there may be more than one node writing to a portion of memory at one time.

The following paragraphs describe in detail each of the tests found in the Network Test menu.

### B.4.1 Multiple Node Communication Test

The Multiple Node Communication Test verifies the communication between all of the nodes running the test (as well as the pass-through capability of those which are not).

Communication is tested for each SCRAMNet shared memory location based on the smallest memory size board in the test. Each node reads each location waiting for a pattern written by the previous node. Once the expected value is read, the node will write over the value with its own, which in turn is expected by the next node.

The nodes are synchronized on the first location in memory which is set to '-1' until the "Master" node (node 0), which is started last, can read the memory sizes (as reported by each of the nodes in a designated memory location) and determine the smallest. The "Master" puts this value in the upper half of the first 32-bit word location in memory. It then sets the lower half of the same word to '1' to signify the beginning of communication testing.

If all nodes are not started before the "Master", the "Master" will transmit an error message to start the errant node before restarting the "Master". The entire memory area, except for the first word, is set to '1' by the "Master". A rolling 1's pattern is utilized to verify communication from one node to the next.

The number of nodes and node number, and the number of times to run the test are entered through the menu system. (See the SCRAMNet *Hardware Diagnostics User's Manual*). The Node Number is unique for each node in the test and numbered sequentially from '0' to N, where '0' is the "Master" and N is the highest Node Number.

The value for "Number of nodes in this test" is entered for each node and must be the same for all nodes. It is equal to N + 1.

Figure B-26 shows the read and write pattern for each node in a three node test at each pass of the test. Node '0', the "Master", starts the process. Each pass reads and writes all memory locations. There are 32 passes per iteration. The following data is represented in hexadecimal.

| pass | Node 0    |           | Node 1    |           | Node 2    |           |
|------|-----------|-----------|-----------|-----------|-----------|-----------|
|      | Read      | Write     | Read      | Write     | Read      | Write     |
| 1    | 0000 0001 | 0000 0002 | 0000 0002 | 0000 0004 | 0000 0004 | 0000 0008 |
| 2    | 0000 0008 | 0000 0010 | 0000 0010 | 0000 0020 | 0000 0020 | 0000 0040 |
| 3    | 0000 0040 | 0000 0080 | 0000 0080 | 0000 0100 | 0000 0100 | 0000 0200 |
| 4    | 0000 0200 | 0000 0400 | 0000 0400 | 0000 0800 | 0000 0800 | 0000 1000 |
| 5    | 0000 1000 | 0000 2000 | 0000 2000 | 0000 4000 | 0000 4000 | 0000 8000 |
| 6    | 0000 8000 | 0001 0000 | 0001 0000 | 0002 0000 | 0002 0000 | 0004 0000 |
| 7    | 0004 0000 | 0008 0000 | 0008 0000 | 0010 0000 | 0010 0000 | 0020 0000 |
| :    | :         | :         | :         | :         | :         | :         |
| 32   | 1000 0000 | 2000 0000 | 2000 0000 | 4000 0000 | 4000 0000 | 8000 0000 |

Figure B-26 Read/Write Pattern

Snapshots of the shared memory from the starting point would show that the nodes are waiting for the data to change to the read pattern they are currently holding.

| MEMORY DATA | 0    | 4    | 8    | 10   | N    |
|-------------|------|------|------|------|------|
|             | 0000 | 0002 | 0000 | 0001 | 0000 |
|             |      | 0001 | 0000 | 0001 | 0001 |

Node 0 is writing 2 over 1 and is currently at memory location 4.

Node 1 is waiting for 2 at memory location 0.

Node 2 is waiting for 4 at memory location 0.

| MEMORY DATA | 0    | 4    | 8    | 10   | N    |
|-------------|------|------|------|------|------|
|             | 0000 | 0004 | 0000 | 0001 | 0000 |
|             |      | 0002 | 0000 | 0001 | 0001 |

Node 0 is writing 2 over 1 and is currently at memory location 8.

Node 1 is writing 4 over 2 and is currently at memory location 4.

Node 2 is waiting for 4 at memory location 0.

| MEMORY DATA | 0    | 4    | 8    | 10   | N    |
|-------------|------|------|------|------|------|
|             | 0000 | 0008 | 0000 | 0001 | 0000 |
|             |      | 0004 | 0000 | 0002 | 0001 |

Node 0 is writing 2 over 1 and is currently at memory location 10.

Node 1 is writing 4 over 2 and is currently at memory location 8.

Node 2 is writing 8 over 4 and is currently at memory location 4.

As soon as a node writes its new data, another node has been waiting for that value to appear and then writes its data. When all 32 passes have been accomplished, the following message is written to **scr\_diag.txt**:

```
No errors detected in # iterations
```

where # is the number of iterations selected.

Figure B-27 is the algorithm for the Multiple Node Communication Test:

```

Disable Fiber Optic Loopback
Disable Host to SM Write
IF initial run
    Determine starting read and write pattern for buffer number
    Reset FIFOs and Verify
    Set first memory location to -1 for this node's memory
    Enable RX, TX and INSRT
    Enable WOKEN and DHMW
    IF this is the master node (buffer number = 0)
        IF this is not the first pass
            Wait 70 seconds before continuing for synchronization
        END IF
        Set all memory locations to 1 except first few for sync
        Read memory for all other nodes started and smallest memory size
        IF some node not started then ERROR and exit
        Write test memory size to memory
        Write start flag to first memory location
    ELSE
        Wait until first memory location is not -1
        Read test memory size boundaries from memory
    END IF
END IF
END IF
FOR 32 iterations
    FOR all 4 byte memory locations within memory boundaries except first
        Check CSRI for any errors
        Wait for expected data (read pattern) at current address
        IF data pattern is not found after DELAY time
            Write error message
            Clear initial run flag
            Return
        END IF
        Write new data (write pattern) to current address
        Increment address pointer
    END FOR
    Calculate next iterations read and write patterns
END FOR

```

**Figure B-27 Multiple Node Communication Test Algorithm**

## B.4.2 Multiple Node Burst Buffer Test

The Multiple Node Burst Buffer test validates the host node's ability to write and verify its own data in a network ring environment while using the Burst mode feature. Burst mode allows data to be written from the originating node one right after another without waiting for previous messages to return for error checking. RUN this test before the multiple node communications test to check for network ring integrity.

Each node on the network ring, and participating in the test, will read and write to a selected partition in the shared memory that is unique to that node. Each node will run independent of the other nodes. Each node will disable its transmitter and store up 1000 messages. When the "Master" node, test number '0', enables its transmitter and starts emptying its FIFO, the other nodes will enable their transmitters and release their data. This results in all nodes releasing data onto the network at the same time, saturating the network, and exercising the sharing of the network.

Figure B-28 is the algorithm for the Multiple Node Burst Buffer Test.

```

Disable Fiber Optic Loopback
Disable Host to SM Write
Enable Burst Mode
IF initial run
    Reset FIFOs and Verify
    Calculate buffer size by dividing 4KB memory size by the number of nodes
    Calculate memory boundaries for the node based on its assigned number
END IF

FOR 32 iterations
    FOR all 4-byte memory locations within memory boundaries
        Disable the transmitter
        FOR 1000 times
            Write rolling 1's to Shared Memory FIFO
            IF end of memory
                Change pointer to beginning
            END IF
        END FOR
        Enable the transmitter
        Write last pattern to Shared Memory FIFO
        Save last pattern and address
        Check for errors during emptying of Shared Memory FIFO
        Increment memory counter and check boundaries
        Wait until last write pattern saved is in memory
        Verify Shared Memory FIFO writes to memory, report errors
    END FOR
END FOR

```

**Figure B-28 Multiple Node Burst Buffer Test Algorithm**

### B.4.3 Network Communication Test

This test verifies correct communication between multiple nodes using BURST and BURST+ protocols. The primary logic for this program is taken directly from the Multiple Node Communication test. The primary difference is that only 4 KB of shared memory is used, and communications alternate using BURST and BURST+ mode protocols.

Reads and writes of the patterns are done in blocks instead of single memory locations to test the Plus (+) protocol feature. Figure B-29 is the algorithm for each iteration of the test.

```

If this is the first iteration or if we need to re-synchronize nodes
  Calculate pattern to check for
  Calculate pattern to write
  Set Write-Me-Last Mode
  If this is the Master Node
    Reset Memory including synchronization area
    Check Existence of all nodes then Set Master location
  Else
    Wait for Master Node to write to his location
  End If
End If
For Each Protocol (Burst and Burst+)
  For a number of iterations (currently 32)
    For Each Block of Shared Memory (currently 8 bytes)
      Check for Errors
      If max errors
        Drop out of this iteration and resync
      End If
      Read Block of memory for expected pattern
      If expected pattern not read
        Drop out of this iteration and resync
      End If
      Write a Block length message (1024) to memory
    End For
    Calculate the next pattern to check for
    Calculate the next pattern to write
  End For
End For

```

**Figure B-29 Network Communication Test Algorithm**

### B.4.4 HCT Compatible Communication Test

The HCT Compatible Communication Test verifies the communication between all of the nodes running the test (as well as the pass-through capability of those which are not).

Communication is tested for each SCRAMNet memory location based on the smallest memory size board in the test. Each node reads each location waiting for a pattern written by the previous node. Once the expected value is read, the node will write over the value with its own, which in turn is expected by the next node.

The nodes are synchronized on the first location in memory which is set to '-1' until the "Master" node (node 0), which is started last, sets the same word to '1' to signify the beginning of communication testing. All memory sizes must be the same.

If all nodes are not started before the "Master", the "Master" will transmit an error message to start the errant node before restarting the "Master". The entire memory area, except for the first word, is set to '1' by the "Master". A rolling 1's pattern is utilized to verify communication from one node to the next.

The number of nodes and node number, and the number of times to run the test are entered through the menu system. (See the SCRAMNet Diagnostics User's manual). The Node Number is unique for each node in the test and numbered sequentially from '0' to N, where '0' is the "Master" and N is the highest Node Number. The value for "Number of nodes in this test" is entered for each node and must be the same for all nodes. It is equal to N + 1.

| pass | Node 0    |           | Node 1    |           | Node 2    |           |
|------|-----------|-----------|-----------|-----------|-----------|-----------|
|      | Read      | Write     | Read      | Write     | Read      | Write     |
| 1    | 0000 0001 | 0000 0002 | 0000 0002 | 0000 0004 | 0000 0004 | 0000 0008 |
| 2    | 0000 0008 | 0000 0010 | 0000 0010 | 0000 0020 | 0000 0020 | 0000 0040 |
| 3    | 0000 0040 | 0000 0080 | 0000 0080 | 0000 0100 | 0000 0100 | 0000 0200 |
| 4    | 0000 0200 | 0000 0400 | 0000 0400 | 0000 0800 | 0000 0800 | 0000 1000 |
| 5    | 0000 1000 | 0000 2000 | 0000 2000 | 0000 4000 | 0000 4000 | 0000 8000 |
| 6    | 0000 8000 | 0001 0000 | 0001 0000 | 0002 0000 | 0002 0000 | 0004 0000 |
| 7    | 0004 0000 | 0008 0000 | 0008 0000 | 0010 0000 | 0010 0000 | 0020 0000 |
| :    | :         | :         | :         | :         | :         | :         |
| 32   | 1000 0000 | 2000 0000 | 2000 0000 | 4000 0000 | 4000 0000 | 8000 0000 |

Figure B-30 Read/Write Pattern

Figure B-30 shows the read and write pattern for each node in a three node test at each pass of the test. Node '0', the "Master", starts the process. Each pass reads and writes all memory locations. There are 32 passes per iteration. The following data is represented in hexadecimal.

Snapshots of the shared memory from the starting point would show that the nodes are waiting for the data to change to the read pattern they are currently holding.

|             |      |      |      |      |      |
|-------------|------|------|------|------|------|
| MEMORY DATA | 0    | 4    | 8    | 10   | N    |
|             | 0000 | 0002 | 0000 | 0001 | 0000 |
|             |      |      |      |      | 0001 |

Node 0 is writing 2 over 1 and is currently at memory location 4.  
 Node 1 is waiting for 2 at memory location 0.  
 Node 2 is waiting for 4 at memory location 0.

|             |      |      |      |      |      |
|-------------|------|------|------|------|------|
| MEMORY DATA | 0    | 4    | 8    | 10   | N    |
|             | 0000 | 0004 | 0000 | 0002 | 0000 |
|             |      |      |      |      | 0001 |

Node 0 is writing 2 over 1 and is currently at memory location 8.  
 Node 1 is writing 4 over 2 and is currently at memory location 4.  
 Node 2 is waiting for 4 at memory location 0.

|             |      |      |      |      |      |
|-------------|------|------|------|------|------|
| MEMORY DATA | 0    | 4    | 8    | 10   | N    |
|             | 0000 | 0008 | 0000 | 0004 | 0000 |
|             |      |      |      |      | 0001 |

Node 0 is writing 2 over 1 and is currently at memory location 10.  
 Node 1 is writing 4 over 2 and is currently at memory location 8.  
 Node 2 is writing 8 over 4 and is currently at memory location 4.

As soon as a node writes its new data, another node has been waiting for that value to appear and then writes its data. When all 32 passes have been accomplished, the following message is written to **scr\_diag.txt**:

|                                    |
|------------------------------------|
| No errors detected in # iterations |
|------------------------------------|

where # is the number of iterations selected.

Figure B-31 is the algorithm for the HCT Compatible Communication Test:

```

Disable Fiber Optic Loopback
Disable Host to SM Write
IF initial run
    Determine starting read and write pattern for buffer number
    Reset FIFOs and Verify
    Set first memory location to -1 for this node's memory
    Enable RX, TX and INSRT
    Enable WOSEN and DHMW

    IF this is the master node (buffer number = 0)
        IF this is not the first pass
            Wait 70 seconds before continuing for synchronization
        END IF
        Set all memory locations to 1
        IF some node not started then ERROR and exit
            Write test memory size to memory
            Write start flag to first memory location
        ELSE
            Wait until first memory location is not -1
        END IF
    END IF
END IF
FOR 32 iterations
    FOR all 4 byte memory locations within memory boundaries except first
        Check CSR1 for any errors
        Wait for expected data (read pattern) at current address
        IF data pattern is not found after DELAY time
            Write error message
            Clear initial run flag
            Return
        END IF
        Write new data (write pattern) to current address
        Increment address pointer
    END FOR
    Calculate next iterations read and write patterns
END FOR

```

**Figure B-31 HCT Compatible Communication Test Algorithm**

*This page intentionally left blank*

# **GLOSSARY**



---

|                          |                                                                                                                                                                                                                                             |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ACR-----                 | Auxiliary Control RAM.                                                                                                                                                                                                                      |
| BURST Mode Protocol----- | This mode allows each node to continuously transmit messages onto the network ring without waiting for any message to return for removal by the originating node. Error detection is in force but there is no error correction implemented. |
| byte-----                | 8 bits.                                                                                                                                                                                                                                     |
| CSR-----                 | Control/Status Register(s).                                                                                                                                                                                                                 |
| dialog window/box-----   | A title and frame that can contain various controls for accepting user input.                                                                                                                                                               |
| editing field-----       | An area of a dialog box that accepts a string of user input. All editing fields are used to gather numeric data.                                                                                                                            |
| EPI-----                 | EEPROM Program Initialization.                                                                                                                                                                                                              |
| GOLD RING Protocol-----  | This protocol allows as many messages on the network as there are nodes; but only one message from each node on the ring at any given time. Error detection and correction are implemented.                                                 |
| LED-----                 | Light-emitting diode.                                                                                                                                                                                                                       |
| longword-----            | 32-bit word.                                                                                                                                                                                                                                |
| LSB-----                 | Least significant bit.                                                                                                                                                                                                                      |
| menu-----                | A list of choices typically depicted in a long rectangular box located under the main windows title bar.                                                                                                                                    |
| MSB-----                 | Most significant bit.                                                                                                                                                                                                                       |
| network node-----        | A computers with an installed SCRAMNet boards. Each computer, or node, solves a portion of the same real-time problem.                                                                                                                      |
| PLATINUM Protocol-----   | This mode combines the advantages of GOLD and BURST modes. There may be multiple messages on the network at one time with error correction implemented.                                                                                     |
| RAM-----                 | Random Access Memory                                                                                                                                                                                                                        |
| SCRAMNet Classic-----    | The first network node produced by Systran. It is a two-board configuration and has GOLD and BURST protocol. This board has byte-swapping capability.                                                                                       |
| SCRAMNet LX-----         | The first single slot adaptation of SCRAMNet It has BURST and BURST PLUS (variable length messages), and no error correction. This board does not have on-board byte-swapping capability.                                                   |
| SCRAMNet+-----           | An enhancement of SCRAMNet LX. It has PLATINUM and PLATINUM PLUS protocol as well as BURST and BURST PLUS. The “SCRAMNet+” board does not have byte-swapping capability.                                                                    |
| shortword-----           | 16-bit word.                                                                                                                                                                                                                                |

*This page intentionally left blank*

# INDEX



- B**
- board-selection ..... 4-11
- D**
- diagnostic test
- loopback test ..... 5-2, 5-4, 5-5
  - memory test ..... 5-2
  - network test ..... 5-2, 5-5, 5-6
- Driver Options ..... 3-1, 3-2
- E**
- Edit ..... 2-2, 2-3, 3-2
- execute diagnostics ..... 5-1
- Execute program ..... 6-1
- F**
- function
- Addressing ..... 4-4, 4-8
  - Clear Memory ..... 4-3, 4-6
  - Data Size ..... 4-4, 4-9
  - Driver Options ..... 3-2
  - Edit ..... 3-2
  - Edit Bitwise ..... 6-3
  - Exit ..... 3-2, 5-3, 5-5
  - Fill Memory ..... 4-3, 4-6, 4-7
  - Flash LED ..... 4-3
  - Goto Address ..... 4-3, 4-6
  - Install ..... 3-2
  - Node Status ..... 5-1, 5-4
  - Options ..... 5-4, 5-6, 6-2
  - Program ..... 6-3, 6-4
- Refresh ..... 4-2, 4-5, 4-6
  - Remove ..... 3-2
  - Restore Defaults ..... 6-3, 6-4
  - Search ..... 4-3, 4-7
  - Update ..... 4-2, 4-5, 4-6
  - View CSRs ..... 4-2, 4-5
- functions ..... 5-3, 5-4, 6-3
- Options ..... 6-4
- I**
- Install ..... 1-1, 3-2
- N**
- network node
- Classic ..... 2-1
  - SCRAMNet+ ..... 2-1
  - SCRAMNet+<sup>2X</sup> ..... 2-1
  - SCRAMNet-LX ..... 2-1
- network protocol
- BURST ..... 2-1, 2-2
  - GOLD RING ..... 2-1, 2-2
  - PLATINUM ..... 2-1, 2-2
  - PLUS mode ..... 2-2
- S**
- SCRAMNet Network
- definition ..... 1-1
- W**
- WINEPI Check Boxes ..... 2-3

*This page intentionally left blank*