SCRAMNet®+ Network

VME/DMA
User's Guide

Document No. D-T-MU-VMEDMA##-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.

SCRAMNet® is a registered trademark of SYSTRAN Corporation.
VMETRO™ is a trademark of VMETRO, Inc.
Universe™ is a trademark of Tundra Semiconductor Corporation

Revised: October 21, 1997

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

TABLE OF CONTENTS

FIGURES

TABLES

1.0 INTRODUCTION

1.1 Overview

This document describes the use and setup for the VMETRO MIDAS-20 board and the **SCRAMNet**+ Network PMC card. The combination of the MIDAS-20 and the **SCRAMNet**+ PMC card enable the user to share data on the **SCRAMNet** real-time network with other **SCRAMNet** Network cards on other host systems and allow DMA transactions across the VMEbus.

1.2 Related Documentation

The user should be familiar with the VMETRO *MIDAS-20 User's Guide* and the Systran *SCRAMNet+ Network PMC Hardware Reference Manual* (D-T-MR-PMC###). These two manuals provide the necessary information to use each of these products. This manual provides a minimal set of information from both manuals for easier configuration. In addition the *Universe User Manual* from Tundra Semiconductor Corporation gives a detailed description of the Universe Register Set.

2.0 ASSUMPTIONS

2.1 Overview

The **SCRAMNet** Network PMC board and the VMETRO MIDAS-20 board provide high DMA transfer rates between the VMEbus and the host computer system memory. The VMETRO MIDAS-20 embeds the Universe VMEbus interface chip which supplies a reliable high performance VMEbus to PCI bus interface in one device. The MIDAS-20 is a PMC carrier for the VMEbus and contains two PMC card slots with slot 1 as the default card slot for the SCRAMNet board (see Figure 2-1).

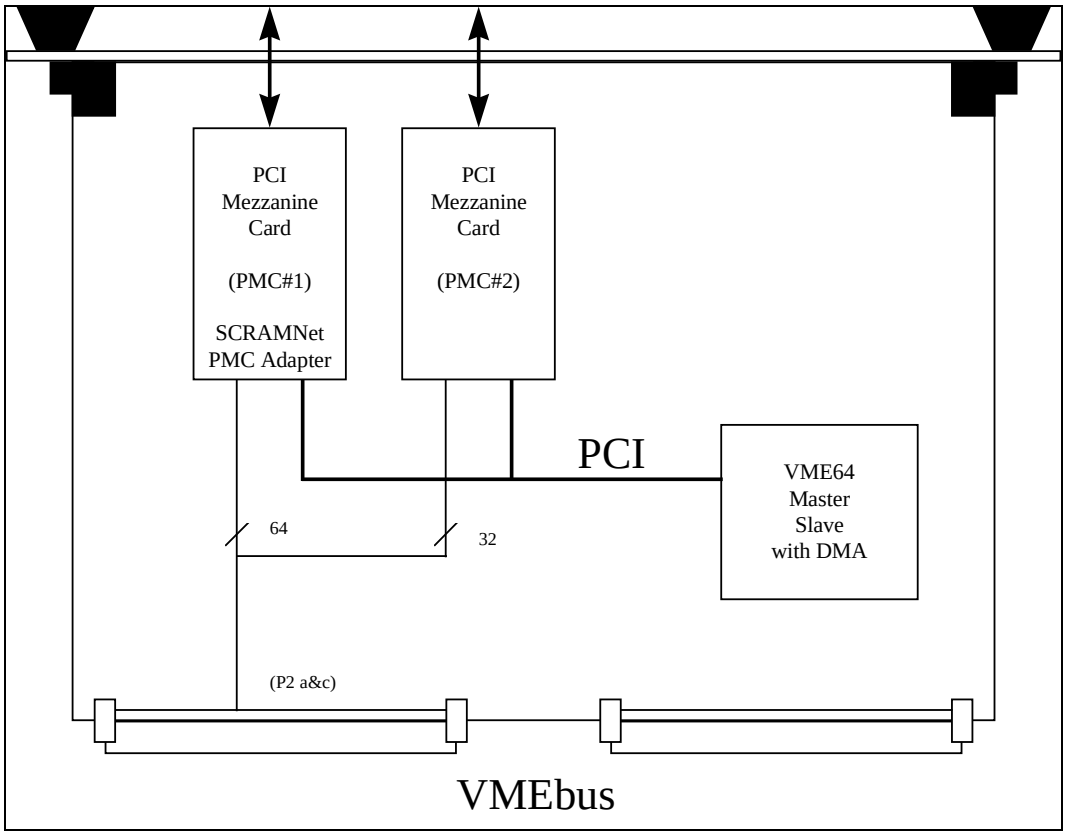

Figure 2-1 Midas-20 Board with SCRAMNet PMC

2.2 MIDAS-20 Board Settings

The Universe chip has a number of configuration registers to be initialized by the host processor. Prior to the host initialization, the switch jumpers on the MIDAS-20 card must be configured to define the operation of the card.

The switch and jumpers on the MIDAS-20 board determine the board options and the memory addresses in the VME space. These settings must be configured correctly so there is no overlap of address space for any other boards on the VMEbus. The VME Register Access Image (VRAI) determines the address of the Universe registers. The Universe registers can then be programmed to provide access to **SCRAMNet** PCI configuration registers. Table 2-1 shows the important VME addresses after the Universe and V3 register sets have been set up (Section 4.0).

Table 2-1 VME Starting Addresses

Device Accessed	VME Starting Address
Universe register set	MIDAS-20 VRAI
V3 register set (PMC slot #1)	MIDAS-20 VRAI + 0x8000 + 0x2800
SCRAMNet CSRs	MIDAS-20 VRAI + 0x810000
SCRAMNet Memory	MIDAS-20 VRAI + 0x10000

Table 2-2 contains the recommended switch and jumper settings on the MIDAS-20 using VME register access image address size A32.

Table 2-2 MIDAS-20 Board Settings

Jumper or DIP Switch	Position
VME Register Access Image (VRAI)	Enabled
VME Register Access Image Address Size	A32
VME64 Auto-Slot ID (AUTO ID)	Disabled
SYSFAIL* Assertion (S FAIL)	Disabled
VME Register Access Base Address (BASE ADDR)	Base address of any free 9 MB block of VME A32 addresses

2.3 Register Configuration

For any transfers to the **SCRAMNet** PMC adapter, the Universe VME interface acts as a VMEbus Slave and the PCI interface acts as a PCI Master. In this example, the Universe register set is mapped to A32 at 14000000 *hex* (14 *hex* is set on the VRAI Base Address DIP switch). The VME Slave Images in the Universe register set are configured to provide access to **SCRAMNet** PCI configuration space (V3 registers). Next, the V3 register set is accessed and configured (VME address 1400a800 *hex* for PMC Slot #1 in this example) to allow access to **SCRAMNet** memory and CSRs.

2.4 SCRAMNet+ Card Settings

All examples assume that the **SCRAMNet+** card's memory is accessed at VME address 14010000 *hex*. The **SCRAMNet+** CSR VME address is at 14810000 *hex*. The **SCRAMNet** PMC card's CSRs are always at an 800000 *hex* (8 MB) offset from the **SCRAMNet** memory base address. Both **SCRAMNet** memory and **SCRAMNet** CSRs generally reside in the same VME address space.

The VME A32 space is used in these examples for both **SCRAMNet** memory and CSRs.

2.5 Unique Node ID

Most applications require each **SCRAMNet** Node in a ring to have a unique node ID. The **SCRAMNet** EEPROM can be programmed (CSR3 and CSR4) to ensure that each node will have a unique node ID upon power-up. In addition, the **SCRAMNet** Utilities Software package allows the node ID to be set in the **scrcfg.dat** file or in **scrhwd.h** if the NO_CONFIG option is used. See the *SCRAMNet Software Installation Manual* for details.

3.0 SOFTWARE

3.1 Universe Register Configuration

The Universe register set must be configured by software for proper **SCRAMNet** access and operation. Section 4.0 describes the steps necessary to configure the Universe register set in software.

3.2 SYSTRAN Supplied Software

SYSTRAN supplies software packages for several popular platforms and operating systems that contain facilities to automatically configure the Universe and V3 register sets for proper operation.

4.0 CONFIGURATION

4.1 Universe VME-PCI Bridge Configuration

The first step in obtaining access to the **SCRAMNet** PMC card is to configure the Universe registers. This section will describe the necessary steps for configuration of the Universe registers using the 'C' language. The byte order of the host must be taken into account and the data written to the Universe registers may have to be byte-swapped.

1. Obtain a pointer variable to the Universe register set. The way in which this is accomplished will be specific to the host system. For example, on most UNIX systems a device driver may have to be written to obtain this access or the *mmap()* call may be used to get user access to the VMEbus. For this example assume that **ucsr** points to the Universe register set. The following definitions are also assumed:

```
#define    VRAI_BASE_ADDRESS       0x14000000
#define    SCR_PCI_BASE_ADDRESS    0x20000000

unsigned long volatile* ucsr;
```

2. Enable Universe Bus Master, Memory and IO access:

```
ucsr[0x1] = 0x7;
```

3. Set Maximum Retries to (8 * 64) and VMEbus Request Level 2:

```
ucsr[0x100] = 0x80800000;
```

4. Set VME slave image 0 to map to SCRAMNet PCI configuration space (V3 registers) where 0x14000000 is the VRAI Base Address:

```
ucsr[0x3C1] = VRAI_BASE_ADDRESS + 0x8000;
ucsr[0x3C2] = VRAI_BASE_ADDRESS + 0xD000;
```

5. Set VME slave image 0 to address 0 on the PCI bus. The translation offset register is added to asserted VME addresses when the slave image is accessed. Any PCI addresses resulting from this translation that are greater than 0xFFFFFFFF roll over to PCI address 0:

```
ucsr[0x3C3] = 0xFFFFFFFF - (VRAI_BASE_ADDRESS + 0x8000) + 1;
ucsr[0x3C0] = 0x80F20002;
```

6. Set VME slave image 1 to map to **SCRAMNet** memory and CSRs

```
ucsr[0x3C6] = VRAI_BASE_ADDRESS + 0x10000;
ucsr[0x3C7] = VRAI_BASE_ADDRESS + 0x900000;
```

7. Set VME slave image 1 to issue correct addresses of **SCRAMNet** on PCI bus:

```
ucsr[0x3C8] = 0xFFFFFFFF - ((VRAI_BASE_ADDRESS + 0x10000) -
SCR_PCI_BASE_ADDRESS) + 1;
ucsr[0x3C5] = 0xE0F20000;
```

8. Set address of Universe registers on PCI bus:

```
ucsr[0x4] = 0x08000000;
```

4.2 V3 Register Configuration

The **SCRAMNet** PCI registers can be configured once the Universe registers are correctly set up. This section contains steps to set up the **SCRAMNet** PCI registers (V3) assuming the following definitions:

```
#define    SCR_PCI_BASE_ADDRESS    0x20000000

unsigned long volatile* v3_ptr;
unsigned short volatile* v3_short_ptr;
```

where **v3_ptr** and **v3_short_ptr** both point to the **SCRAMNet** PCI Configuration register set. These pointers can generally be set as an offset to the **ucsr** pointer above as follows:

```
v3_ptr = (unsigned long volatile*)((unsigned long)ucsr +
0x8000 + 0x2800);
v3_short_ptr = (unsigned short volatile*)v3_ptr;
```

1. Enable SCRAMNet Bus Master, Memory and IO access:

```
v3_ptr[0x1] = 0x7;
```

2. Clear the lock on the SYSTEM register to change aperture 0

```
v3_short_pci[0x3C] = 0xA05F;
```

3. Set aperture 0 to same PCI address that we set in Universe VME slave image 1 above and set prefetch enable:

```
v3_ptr[0x5] = SCR_PCI_BASE_ADDRESS | 0x8;
```

4. Enable the lock on the SYSTEM register:

```
v3_short_ptr[0x3C] = 0xC400;
```

5. Set MAP register for Aperture 0. Eight-bit byte swapping is turned on here for a big-endian host. If the host is little-endian, write 0x00008043:

```
v3_ptr[0x10] = 0x00008243;
```

5.0 INTERRUPTS

5.1 Overview

The **SCRAMNet** card is able to send/receive interrupts across the network on a per-memory location basis as well as signal the end of a DMA transfer. The Universe register set must be properly set up to utilize the **SCRAMNet** interrupt feature. This section contains the necessary steps for configuration of the Universe registers using the C language.

5.2 DMA Controller

To use the DMA controller (with interrupts) on the MIDAS-20, enable the VME Bus Request Level 3 (and the IACK jumper) on the VMEbus prior to installation.

The following assumptions are made for this example:

- The VME interrupt level is set for level 5.
- The following definitions will also be assumed:

```
#define     SCR_INT_LEVEL     5
#define     SCR_VECTOR      0x71

unsigned long volatile* ucsr;
```

1. Map Universe interrupts to indicated vector number

```
ucsr[0xC8] = SCR_VECTOR << 24;
```

2. Map Universe DMA and bus error interrupts to indicated VME IPL

```
ucsr[0xC7] = SCR_INT_LEVEL | (SCR_INT_LEVEL << 4) |
                      (SCR_INT_LEVEL << 8);
```

3. PMC slot 1 INTA# is mapped to LINT#1 as described in MIDAS-20 User's Guide

```
ucsr[0xC6] = SCR_INT_LEVEL << 4;
ucsr[0xC4] = 0x00000702;
```

6.0 PASSIVE CABINET KIT

6.1 Overview

This section discusses the passive cabinet kit. This includes the cabinet kit features, functions, installation, and operation.

6.2 Description

The passive cabinet kit for the SCRAMNet+ Network provides fiber-optic access to the node's connections, and maintains the shielding of the chassis. The passive cabinet kit consists of a bulkhead plate, an auxiliary connector cable and connecting fiber cables. The passive cabinet kit extends the fiber connections from the host interface board to the chassis bulkhead. The passive cabinet kit is shown in Figure 6-1.

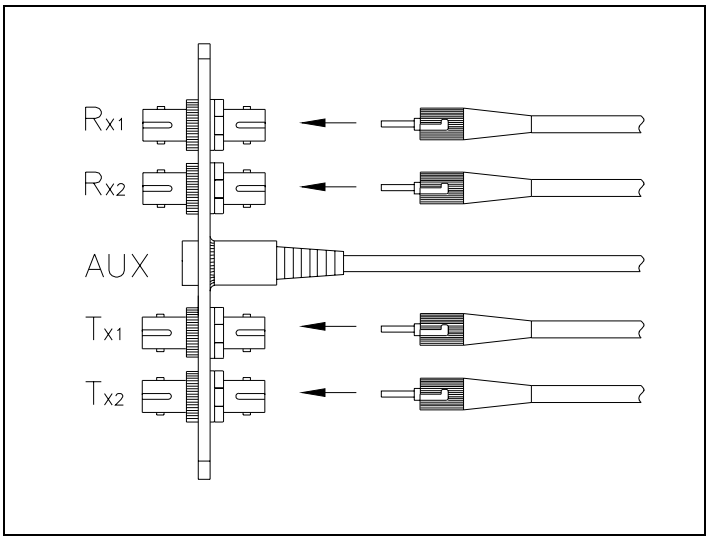

Figure 6-1 Bulkhead Plate Connections

6.3 Faceplate

The passive cabinet kit faceplate is customized for the Silicon Graphics' ONYX computer cabinet.

Table 6-1 Faceplate Abbreviations

Abbreviation	Description
AUX	Auxiliary connector. Allows connection to a Fiber Optic Bypass Switch.
R_X	Receiver connections.
T_X	Transmitter connections.

6.4 Media Card

The passive cabinet kit does not include a media card.

6.5 Auxiliary Connection

The Auxiliary Connection is used for communication with the Fiber Optic Bypass Switch. The 8-pin modular in-line plug connection shown in Figure 6-2 is defined in Table 6-2.

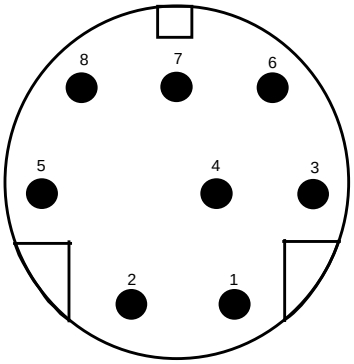

Figure 6-2 Auxiliary Connection

Table 6-2 Auxiliary Connection Pinout

PIN	DESCRIPTION
8	Unused
7	Unused
6	Unused
5	Backup +5 Vdc to External Device
4	Unused
3	Fiber Optic Relay Drive and Sense
2	Unused
1	Logic Ground

6.6 Cabling

Connections between the cabinet kit bulkhead plate and the **SCRAMNet**+ host interface board are shown in Figure 6-3.

Connect cables between the VME/DMA RECEIVER (R_{x1} and R_{x2}) and the cabinet kit receiver (R_{x1} and R_{x2}) connections; and the VME/DMA transmitter (T_{x1} and T_{x2}) with the cabinet kit transmitter (T_{x1} and T_{x2}) connections.

NOTE: It does not matter if R_{x1} or R_{x2} is connected to the bulkhead plate's R_{x1} or R_{x2} as long as both cables are connected to both of the R_x connectors. Likewise, T_{x1} and T_{x2} may be connected to the bulkhead plate's T_{x1} or T_{x2}.

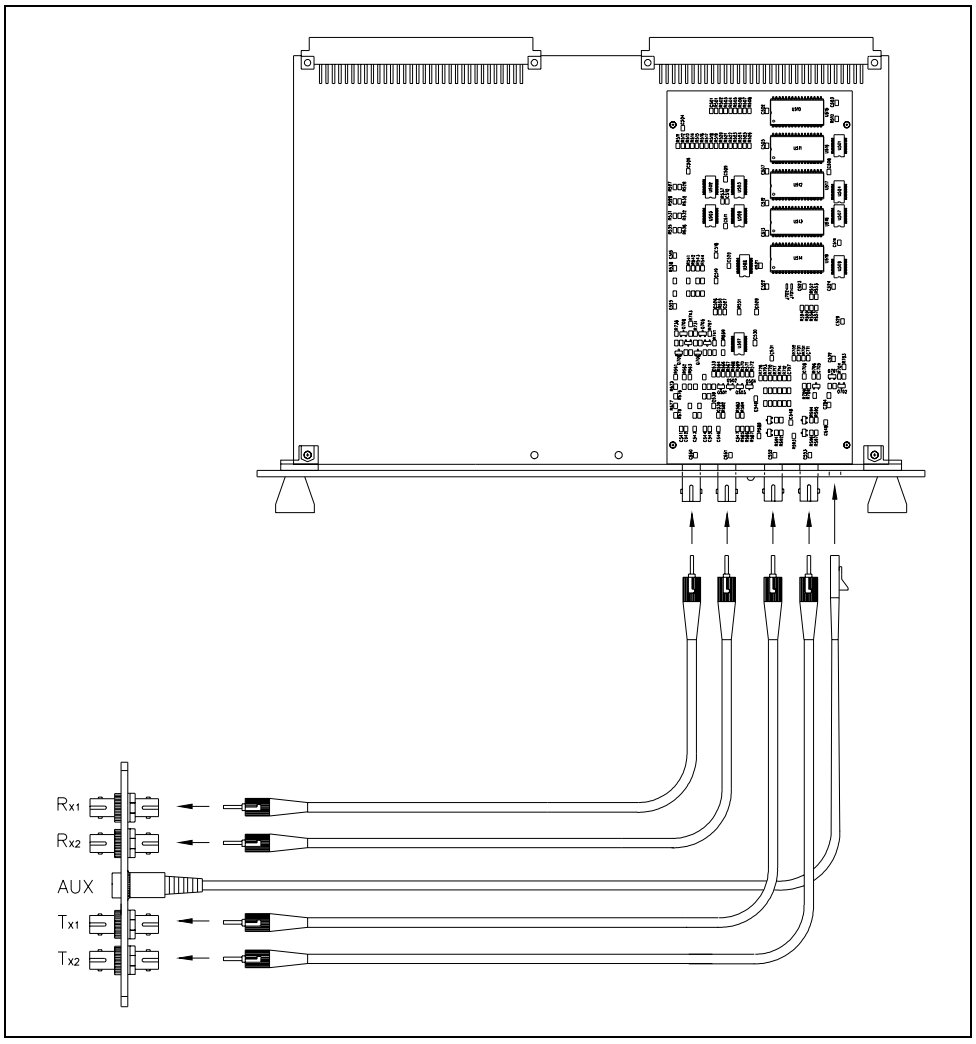

Figure 6-3 Passive Cabinet Kit

This page intentionally left blank