

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*

CORELIS

NetICE-R

NetICE-R
LAN-Based
ROM Emulator

User's Manual

CORELIS

NetICE-R

NetICE-R
LAN-Based
ROM Emulator

User's Manual

CORELIS Inc.

Copyright © 1996-1999 Corelis Inc.
12607 Hiddencreek Way
Cerritos, CA 90703-2146
Tel.: (562) 926-6727 • Fax.: (562) 404-6196

Please send all inquiries or comments to:

Tech Support: support@corelis.com
Sales Info: sales@corelis.com
Web: www.corelis.com

Table of Contents

1.0 PREFACE 1
- 1.1 Printing History 1
- 1.2 General Notice 1
- 1.3 Product Warranty 2
- 1.4 Exclusive Remedies 3
- 1.5 Return Policy 3
- 1.6 Product and Safety Warnings 3

2.0 OVERVIEW 5
- 2.1 System Features 6
- 2.2 Specifications 7

3.0 INSTALLATION 9
- 3.1 Unpacking & Inspection 9
- 3.2 Power Supply 10
- 3.3 LAN Hookup 11
- 3.4 Installation of Software 12
- 3.5 Setting the NetICE-R IP Address 13
- 3.6 Configuration DIP-Switch Settings 15
- 3.7 Target Hookup and Interface Cable 16

4.0 OPERATION 17
- 4.1 Serial Port 17
- 4.2 Parallel Port Host Interface 19
- 4.3 ROM Emulator Pod 21
- 4.4 Access LEDs 23
- 4.5 Option 4/xx 23

5.0 SOFTWARE 25
- 5.1 What's on the diskette 25
- 5.2 README File 25
- 5.3 Network Installation Notes 26
- 5.4 ROMLOAD Utility 27
- 5.5 FLASH Utility 31
- 5.6 ROM Function Library 33

INDEX 54

1.0 PREFACE

1.1 Printing History

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition is published.

A software code may be printed before the date; this indicates the version of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual updates.

Edition 1	March 1997
Edition 1.1	May 1997
Edition 1.2	February 1998
Edition 1.3	July 2000

1.2 General Notice

Information contained in this document is subject to change without notice. Corelis shall not be liable for errors contained herein for incidental or consequential damages in connection with the furnishing, performance, or use of material contained in this manual. This document contains proprietary information which is protected by copyright. All rights reserved. No part of this document may be reproduced without the prior written consent of Corelis.

Corelis assumes no responsibility for the use of or reliability of its software on equipment that is not furnished by Corelis.

1.3 Product Warranty

This Corelis product has a warranty against defects in material and workmanship for a period of 90 days from date of shipment. During the warranty period, Corelis will, at its option, either repair or replace products that prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by Corelis. Outside Corelis service travel areas, warranty service will be performed at the Buyer's facility only upon Corelis' prior agreement and Buyer shall pay Corelis' round trip travel expenses.

For products returned to Corelis for warranty service, the Buyer shall prepay shipping charges to Corelis and Corelis shall pay shipping charges to return the product to the Buyer. However, the Buyer shall pay all shipping charges, duties, and taxes for products returned to Corelis from another country.

Corelis warrants that its software and firmware designated by Corelis for use with an instrument will execute its programming instructions when properly installed on that instrument. Corelis does not warrant that the operation of the instrument, software, or firmware will be uninterrupted or error-free.

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by the Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. CORELIS SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

1.4 Exclusive Remedies

THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. CORELIS SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

Product maintenance agreements and other customer assistance agreements are available for Corelis products. For assistance, contact your nearest Corelis Sales and Service Office.

1.5 Return Policy

No items returned to Corelis for warranty, service, or any other reason shall be accepted unless first authorized by Corelis, either direct or through its authorized sales representatives. All returned items must be shipped pre-paid and clearly display a Return Merchandise Authorization (RMA) number on the shipping container. Freight collect items will ***not*** be accepted. Customers or authorized sales representatives must first contact Corelis with notice of request for return of merchandise. RMA's can only originate from Corelis. If authorization is granted, an RMA number will be forwarded to the customer either directly or through its authorized sales representative.

1.6 Product and Safety Warnings

The following general precautions with respect to product handling and safety must be observed during all phases of handling, operation, service, and repair of this product. Failure to comply with these precautions or with specific warnings in this manual violates the product warranty as well as safety standards of design, manufacture, and intended use of this product. Corelis assumes no liability for the failure to adhere with these requirements.

None of the following warnings and notices are to be in any way construed as authorization to effect repairs or modifications to this product. In the event a service issue arises, Corelis or one of its authorized agents should be promptly consulted.

GROUND EQUIPMENT
For safety Class 1 equipment (i.e., equipment having an earth terminal), an uninterruptible safety earth ground must be provided to the equipment into which this product is to be installed from the main power source to its ground terminals or AC connector via its power cable.

DO NOT operate equipment that is damaged. If any of the safety protection features built into this product are suspected or known to be compromised, either through physical damage, moisture, or any other reason, **remove power** and do not use the product until safe operation can be verified by Corelis authorized personnel. If necessary, return the product to Corelis or an authorized service organization for repair to ensure proper operation.

DO NOT modify equipment, perform an unauthorized repair, or replace parts. Any unauthorized repair or modifications will void the product warranty. Also because of the possibility of violating design specifications and danger of introducing potential hazards, do not install replacement components or effect any such unauthorized repair or modifications. To ensure correct operation, return the product to a Corelis authorized service organization for service and repair.

2.0 OVERVIEW

The NetICE-R LAN-based ROM emulator is a member of the Corelis family of boundary-scan based emulators and diagnostic tools. The NetICE-R integrates the power of a local area network with ROM emulation capabilities for improved software and hardware development, debug and test. It provides an intelligent interface between any PC or workstation that has networking support for Ethernet with the TCP/IP protocol and any EPROM/FLASH based Unit Under Test (UUT).

The NetICE-R was designed to maximize the speed of loading firmware code from a host computer to the unit under test. The NetICE-R contains up to four ROM emulation pods and comes with various memory configurations and data bus widths. The NetICE-R connects to the UUT via cable(s) that plug instead of the on-target EPROM/FLASH device(s) and is processor independent. The UUT firmware code is downloaded from the host computer onto the NetICE-R fast SRAM type memory and eliminates the need to program EPROM/FLASH devices every time that new code is compiled and is ready for testing. The target code is stored to the on-board 1M, 2M or 4M word deep memory per pod.

The NetICE-R is provided with host utility software that enables the user to configure and to download Motorola S-Record formatted files directly to the NetICE-R memory. The S-Record hex file is typically generated by the software development tools and would otherwise be used to program EPROM/FLASH devices on the standard PROM programmer. NetICE-R eliminates having to program FLASH devices until such time that software changes are no longer required.

The NetICE-R is a general purpose tool that can be used along with **any** other development tools such as logic analyzers, emulators, etc. The NetICE-R is compatible with all of the Corelis (and others) JTAG emulators, including the PowerEM-603, PowerEM-604 and EMDT-960RP emulators and provides a quick method of downloading code to the target system.

2.1 System Features

The following are the key features of NetICE-R.

- Fast SRAM based memory, 1M or 2M words each pod
- Up to four (4) pods emulate 8, 16, 32 or 64 bit wide memories
- 45 nSec emulation memory access time
- 3.3V and 5V compatible target interface
- Flash EPROM for firmware updates.
- Ethernet 10BASE-T or 10BASE-2 TCP/IP Protocol network operation.
- Front Panel activity indicator LED's:
  - Self-test
  - Power On
  - Pod 1 Target Access
  - Pod 2 Target Access
  - Pod 3 Target Access (optional)
  - Pod 4 Target Access (optional)
- Back Panel activity indicator LED's:
  - Ethernet communication activity
- Standard EPROM/FLASH cables support DIP, PLCC and custom packages and connectors

2.2 Specifications

Host Interface

Ethernet:	10Base-T or 10Base-2
Protocol:	TCP/IP
Serial (optional):	RS-232 1200 through 115.2K baud rate
IEEE-1284 (optional):	Compatible mode

Memory Configuration

Option 1/1M	One pod,	1M x 16 or 1M x 8
Option 2/1M	Two pods,	1M x 16 or 1M x 8 each
Option 4/1M (*)	Four pods,	1M x 16 or 1M x 8 each
Option 1/2M	One pod,	2M x 16 or 2M x 8
Option 2/2M	Two pods,	2M x 16 or 2M x 8 each
Option 4/2M (*)	Four pods,	2M x 16 or 2M x 8 each

Memory Access

Access Time	45 nSec max
Access Enable	Programmable Write Protect bit

Data Bus Width

Option 1/xx	8 bit	or	16 bit
Option 2/xx	16 (2x8) bit	or	32 (2x16) bit
Option 4/xx (*)	32 (4x8) bit	or	64 (4x16) bit

Target Interface

Driver (Data bus)	TTL, 3.3V compatible, +/-24mA
Input Voltage:	TTL, 3.3V compatible, 5V tolerant
Input Current (max.):	IIL = -5 to +5 µA

Physical Size

Option 1/xx	5.57"W x 7.45"L x 1.52"H
Option 2/xx	5.57"W x 7.45"L x 1.52"H
Option 4/xx (*)	5.57"W x 7.45"L x 2.48"H

(*) Consult factory for availability

Connectors

ROM Emulator Pods:	50 pin SCSI type II male connector each
Parallel Port connector:	DB-25 female
Serial Port connectors:	DB-9 male
10Base-T connector:	RJ-45
10Base-2 connector:	BNC

Front Panel Indicators

POWER:	Green LED indicates that power is supplied to the card
READY:	Green LED indicates that power-up self-test passed
ACCESS (pod 1):	Green LED indicates that pod 1 is “chip selected” by the target EPROM/FLASH interface
ACCESS (pod 2):	Green LED indicates that pod 2 is “chip selected” by the target EPROM/FLASH interface

Back panel indicators

TX/RX:	Red LED indicates LAN data transmit / receive
LINK:	Red LED identifies that the likn is functional (10 Base-T)

Power Consumption

+ 5VDC @ 3.0 A	External AC/DC supply is 110/220V, 50/60 Hz compatible

Operating Environment

Temperature:	0°C to 45°C
Relative Humidity:	10% to 90% non-condensing

Storage Environment

Temperature:	0°C to 85°C
Relative Humidity:	0% to 90% non-condensing

3.0 INSTALLATION

This chapter covers the installation and configuration of the NetICE-R

3.1 Unpacking & Inspection

The NetICE-R comes supplied with the following:

NetICE-R Main Control Unit
AC to DC External +5V Power Supply
NetICE-R ROM Emulator Loader program on a 3.5" Diskette
NetICE-R ROM Emulator User's Guide (P/N UM00910100-A0)
RS-232 interface cable for setting the IP address (see section 3.4)
Optional target ROM Interface Cable

Ensure all materials listed are present and free from visible damage or defects before proceeding. If anything appears to be missing or damaged, contact Corelis at the number listed on the front cover immediately.

It is suggested that you preserve the original packing material in the event the product must be shipped back to Corelis for service.

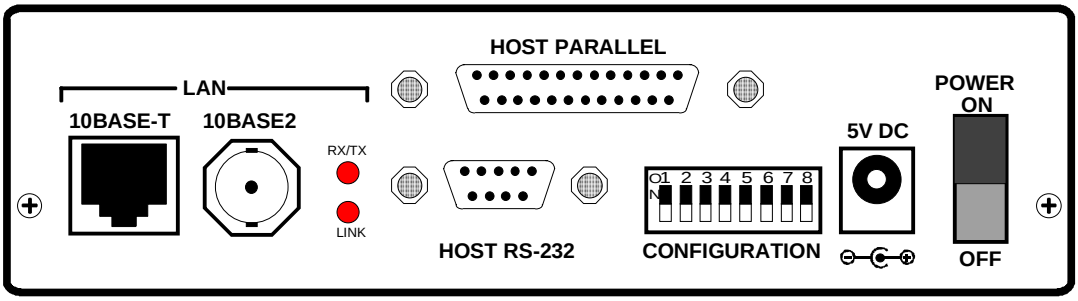


Figure 3.1. REAR PANEL

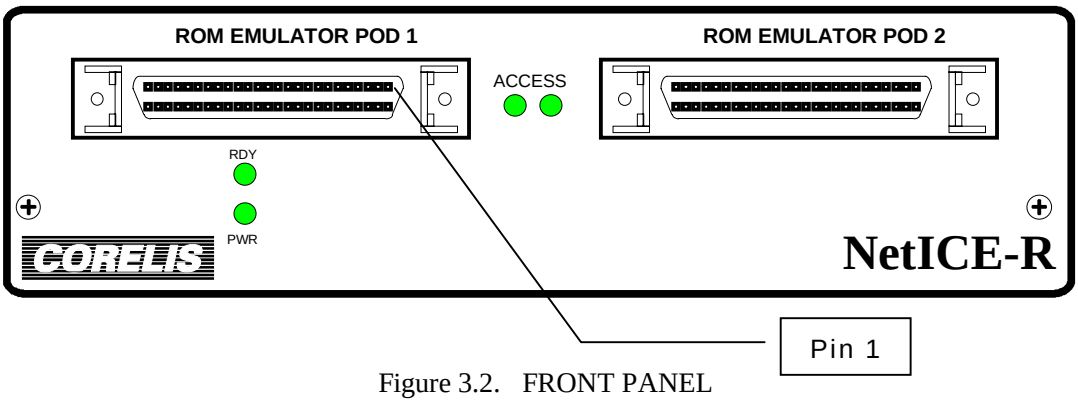


Figure 3.2. FRONT PANEL

3.2 Power Supply

The NetICE-R Main Control Unit requires a +5VDC @ 3.0A regulated external power supply with positive center terminal. The NetICE-R comes supplied with a compliant power supply. The supply requires a standard 110-220VAC 50-60Hz input and provides the required +5VDC and mating connector.

Refer to Fig. 3.1 to locate the NetICE-R's power switch on the rear panel. Ensure it is in the OFF position then insert the mating connector into the socket provided.

Insert the input 110VAC plug into a suitable outlet.

3.3 LAN Hookup

The NetICE-R requires an operating Local Area Network (LAN) to interface the Main Control Unit to the host computer controlling the scanning operations. The NetICE-R is compatible with LANs incorporating the 10Base-T cable format or the 10Base-2 cable format. A compliant connector socket is provided on the rear panel for both formats (see Fig. 3.1). The actual Network Topology (bus, ring, star, etc.) is not relevant to the operation or setup of the NetICE-R.

Provide the appropriate connection into the LAN for the NetICE-R. If using the 10Base-2 connection, ensure a "T" connector is used to allow LAN operation while the NetICE-R Main Control Unit is un-powered. For the 10Base-2 connection, ensure the RJ45 connector is locked into place (listen for the 'snap' of the locking tab).

You will next need to configure the NetICE-R for the selected network format using the Configuration DIP-Switch. Place switch position 2 in the ON position for the 10Base-2 format or in the OFF position for the 10Base-T format. Refer to Section 3.5 for details on Configuration DIP-Switch settings.

Following the network physical connection and Configuration DIP-Switch setting, you will need to set an IP address for the NetICE-R on your network. Refer to the following Section 3.4 for instructions on setting the IP address via the RS-232 serial port interface.

Alternately, you can use the optional Boot Protocol (BOOTP) mode to allow the net server to establish the NetICE-R's IP address automatically. See Section 3.5 for DIP-switch details.

3.4 Installation of Software

No special installation procedure for the NetICE-R's software is required. Just copy the contents of the supplied diskette to a location of choice on your system, and they are ready to run. For more information on the supplied software, see sections 5.0 through 5.6.

3.5 Setting the NetICE-R IP Address

The IP Address identifies the NetICE-R on the Local Area Network. Under normal operation, the default IP Address (192.168.1.1) for the NetICE-R should be sufficient for proper operation. If the IP Address needs to be changed (i.e. another NetICE-R on the LAN), this can be facilitated by a built-in utility accessed using the **Host RS-232** serial port and a computer terminal capable of communicating via RS-232 and the ANSI character set (known as a "dumb terminal").

Ensure power to the NetICE-R is off. Connect a "dumb terminal" to the Host RS-232 serial connector on the rear of the NetICE-R main Control Unit. When using a PC with terminal emulation software (Procomm, Kermit, HyperTerm, etc.), use the provided RS-232 interface cable. For Description of required cable is shown in Section 4.1. The serial communications setting is **8 data bits**, **no parity**, **1 stop bit** (8N1) and a baud rate matching the setting of the NetICE-R's Configuration DIP-switch (see Section 3.5).

Connect the dumb terminal as described and place Configuration DIP-switch position 1 to the OFF position. Turn on the power to the NetICE-R.

The dumb terminal will display various self-test and configuration status as the NetICE-R powers up. Assuming there are no problems during the power up, you will be presented with the command prompt:

```
Net-1149.1>
```

Verify the current version information by typing the **version** command at the prompt:

```
Net-1149.1> version
```

The NetICE-R will respond with its current version information. For example:

```
Serial Number : 127
Core O/S v1.01 - 3/26/97
SFL Module: SFL v1.01 - 3/26/97
```

Examine the current network settings and determine if these are suitable or require modification by typing **network** at the command prompt:

Net-1149.1> network

The NetICE-R will respond with its current network setup information. For example:

IP Address: 192.168.1.1
Subnet Mask: 255.255.255.0
Default Gateway: 0.0.0.0
SFL Services Port: 1149
Ethernet Hardware Address = 00:60:B8:00:00:01

To modify the services port name, IP address, Gateway or Mask, enter the following command string at the command prompt:

NETWORK [-PORT #] [-IP #.#.#.#] [-GATEWAY #.#.#.#] [-MASK #.#.#.#]

The parameters enclosed in brackets are optional but if used require the completed setting. For example:

NETWORK -IP 192.168.1.24 -GATEWAY 192.168.1.254

Once the desired network configuration has been set, turn off power to the NetICE-R and place the Configuration DIP-switch position 1 to the ON position. Re-power on the NetICE-R for the new settings to take effect.

Note: The default IP address is 192.168.1.1

3.6 Configuration DIP-Switch Settings

The rear panel of the NetICE-R Main Control Unit contains an 8-position DIP switch used to configure several functions. Refer to Fig. 3.1 for DIP-switch location. The individual switches can be placed in either the ON position as indicated by the label on the switch or the OFF position.

Position	ON	OFF
1	Access via the LAN	Access via Host RS-232
2	Use 10-Base2 connection	Use 10-BaseT connection
3	BOOTP Enabled at power-up	BOOTP Disabled
4	Reserved	Reserved
5	Reserved	Reserved
6	RS-232 Baud Setting	RS-232 Baud Setting
7	RS-232 Baud Setting	RS-232 Baud Setting
8	RS-232 Baud Setting	RS-232 Baud Setting

Table 3.1 Configuration DIP-Switch settings

6	7	8	Baud
OFF (0)	OFF (0)	OFF (0)	1200
OFF (0)	OFF (0)	ON (1)	2400
OFF (0)	ON (1)	OFF (0)	4800
OFF (0)	ON (1)	ON (1)	9600
ON (1)	OFF (0)	OFF (0)	19200
ON (1)	OFF (0)	ON (1)	38400
ON (1)	ON (1)	OFF (0)	57600
ON (1)	ON (1)	ON (1)	115200

Table 3.2 Configuration DIP-Switch Baud setting

Position 1 enables communication from a host to the LAN port or through the Host RS-232 serial port.

Position 2 selects the LAN cable format between 10-Base2 and 10-BaseT.

Position 3 enables the Boot Protocol function that if enabled will allow the NetICE-R on power up to request from a host on the LAN self-configuration information including the unit's IP address.

Positions 4 and 5 are reserved and should be left in the OFF position.

Positions 6,7 and 8 are used to set the RS-232 serial port baud rate. See Table 3.2.

3.7 Target Hookup and Interface Cable

The NetICE-R interfaces to the target ROM socket (or connector) via an optional cable. The interface cable either plugs instead of the socketed EPROM/FLASH device(s) or connects to the target via a custom designed cable connector. Contact Corelis for the list of available interface cables.

The front panel connectors that are used for each configuration option are shown in the following table:

Configuration	# of Pods	Front Panel Connector(s)
Option 1/1M	1	Pod 2
Option 2/1M	2	Pod 2, Pod 1
Option 4/1M (*)	4	Pod 2, Pod 1, Pod 4, Pod 3
Option 1/2M	1	Pod 2
Option 2/2M	2	Pod 2, Pod 1
Option 4/2M (*)	4	Pod 2, Pod 1, Pod 4, Pod 3

(*) Consult factory for availability

4.0 OPERATION

This chapter contains information that is required when connecting the NetICE-R to the outside world.

4.1 Serial Port

The NetICE-R provides a host RS-232 serial port for interfacing to an external host computer. The **Host RS-232** serial port is located on the rear panel of the NetICE-R (Fig. 3.1) and is used primarily for access to status and embedded function control (i.e. re-programming the internal FLASH PROM).

The serial port of the NetICE-R is configured to emulate the function of a Data Communications Equipment (DCE). This configuration allows direct cable interface to a user Data Terminal Equipment (DTE) device. Should the application require the connection of the serial port to an external DCE device, a crossover or "null-modem" cable must be used. The **Host RS-232** serial port permits full-duplex, bi-directional data transfer using the RS-232 protocol.

Standard 9-pin DB9 type female connector for the serial port is provided on the NetICE-R for easy hook-up to external targets as well as communicating with the host PC. The provided one-to-one DB9M-to-DB9F cable can be used to connect to the standard COM port of the host PC. The following table shows the pinout of the serial port connector.

PIN	SIGNAL NAME	I/O	DESCRIPTION
1	No Connection		
2	TxD	Out	Transmit Data
3	RxD	In	Receive Data
4	DTR*	In	Data Terminal Ready
5	GND		Ground
6	DSR*	Out	Data Set Ready
7	No Connection		
8	No Connection		
9	No Connection		

*Indicates an active low signal

Table 4.1 Serial Ports Connector Pin Assignment

Baud rate for the serial port is set by the Configuration DIP-Switch located on the rear panel
(See Fig. 3.1).

4.2 Parallel Port Host Interface

The NetICE-R has an optional IEEE-1284 peripheral side compliant bi-directional parallel port interface. This interface is optional and is not currently supported. It is designed as an alternate method of host communications.

The parallel port is implemented using a IEEE-1284, type A 25-pin connector (DB25 female). The following table shows the pinout of the parallel port connector.

PIN	SIGNAL NAME	I/O	DESCRIPTION
1	STROBE*	In	Parallel Port Strobe
2	AD0	In/Out	Channel Data
3	AD1	In/Out	Channel Data
4	AD2	In/Out	Channel Data
5	AD3	In/Out	Channel Data
6	AD4	In/Out	Channel Data
7	AD5	In/Out	Channel Data
8	AD6	In/Out	Channel Data
9	AD7	In/Out	Channel Data
10	ACT*	Out	Parallel Port Acknowledge
11	BUSY	Out	Parallel Port Busy
12	PERROR	Out	Peripheral Error
13	SELECT	Out	Peripheral On-Line
14	AUTOFD*	In	Parallel Port Autofeed
15	FAULT*	Out	Peripheral Error
16	INIT*	In	Initialize to Compatible Mode
17	SELECTIN*	In	Peripheral selected
18	GND		Ground
19	GND		Ground
20	GND		Ground

21	GND		Ground
22	GND		Ground
23	GND		Ground
24	GND		Ground
25	GND		Ground

*Indicates an active low signal

Table 4.2 Parallel Port Connector Pin Assignment

NOTE: At the time of this printing, the **Parallel Port Host Interface** is not supported by the firmware. Future firmware upgrades will support this port.

4.3 ROM Emulator Pod

The NetICE-R contains either two (2) or four (4) front panel ROM Emulator pod connectors. The ROM Emulator pods are accessed via the 50-pin SCSI type II connectors on the front panel, one connector for each pod. Refer to Fig. 3.2 for connector location. The NetICE-R interfaces to the target system via optional custom and standard cables. The 50-pin SCSI connector pinout is as follows:

PIN	SIGNAL NAME	I/O	DESCRIPTION
1	GND	Ground	Logic Ground
2	DATA 3	I/O	Data Bus
3	DATA 4	I/O	Data Bus
4	DATA 5	I/O	Data Bus
5	DATA 6	I/O	Data Bus
6	DATA 7	I/O	Data Bus
7	CS*	IN	ROM Chip Select, Active Low
8	ADDRESS 10	IN	Address Bus
9	OE*	IN	ROM Output Enable, Active Low
10	ADDRESS 11	IN	Address Bus
11	ADDRESS 9	IN	Address Bus
12	ADDRESS 8	IN	Address Bus
13	ADDRESS 13	IN	Address Bus
14	ADDRESS 14	IN	Address Bus
15	ADDRESS 17	IN	Address Bus
16	ADDRESS 18	IN	Address Bus
17	DATA 12	I/O	Data Bus
18	DATA 13	I/O	Data Bus
19	DATA 14	I/O	Data Bus

20	DATA 15	I/O	Data Bus
21	3.3V	Power	Power for Active Probe
22	ADDRESS 20	IN	Address Bus (most significant bit)
23	BUFFER_EN*	OUT	Buffer Enable (for active probes)
24	5V	Power	Power for Active Probe
25	GND	Ground	Logic Ground
26	GND	Ground	Logic Ground
27	ADDRESS 19	IN	Address Bus
28	ADDRESS 16	IN	Address Bus
29	ADDRESS 15	IN	Address Bus
30	ADDRESS 12	IN	Address Bus
31	ADDRESS 7	IN	Address Bus
32	ADDRESS 6	IN	Address Bus
33	ADDRESS 5	IN	Address Bus
34	ADDRESS 4	IN	Address Bus
35	ADDRESS 3	IN	Address Bus
36	ADDRESS 2	IN	Address Bus
37	ADDRESS 1	IN	Address Bus
38	ADDRESS 0	IN	Address Bus (least significant bit)
39	DATA 0	I/O	Data Bus
40	DATA 1	I/O	Data Bus
41	DATA 2	I/O	Data Bus
42	GND	Ground	Logic Ground
43	DATA 8	I/O	Data Bus
44	DATA 9	I/O	Data Bus
45	DATA 10	I/O	Data Bus

46	DATA 11	I/O	Data Bus
47	GND	Ground	Logic Ground
48	GND	Ground	Logic Ground
49	WRITE*	IN	ROM Write Enable, Active Low
50	GND	Ground	Logic Ground

*Indicates an active low signal

Table 4.3 (JTAG Connector Pin Assignment

Note: Suggested mating connector is AMP 749111-4 or equivalent.

4.4 Access LEDs

The NetICE-R contains 2 front panel LEDs (4 for Option 4/xx) that with *ACCESS* nomenclature. Each LED corresponds to the pod connector adjacent to it on the front panel. The LED is ON whenever the chip select for the pod is active (low). The LED is powered by a "one-shot" of about 20 msec, so that even a single access to the ROM with a single activation of the chip select signal is clearly visible

4.5 Option 4/xx

Option 4/1M and Option 4/2M of the NetICE-R provide the user with four (4) ROM Emulator pods and require a physically taller enclosure to accommodate the two additional front panel connectors. The height of the enclosure for Option 4/xx is 2.48". The additional height is used for a second pair of ROM Emulator pod connectors with their matching *ACCESS* green LEDs. The pinout for all of the pod connectors is identical, as shown in table 4.3.

This page left intentionally blank

5.0 SOFTWARE

Note: Refer to the README.TXT file on the included diskette for the latest information, changes and/or release notes for the ROMLOAD software or ROM Function Library DLL.

5.1 What's on the diskette

The diskette contains the following programs:



ROMLOAD loader utility for loading S-Record compatible Hex files to the NetICE-R memory. This utility is provided for Windows 95/NT, Solaris and HP-UX host systems.



FLASH update utility for remote re-programming of the firmware inside the NetICE-R box. This utility is provided for Windows 95/NT only.



ROM FUNCTION LIBRARY (NetSFL*.*), included to facilitate host application development. This is provided in DLL format for Windows 95/NT, and .a format for SUN and HP-UX.

5.2 README File

Please read the README.TXT file on the diskette for the installation procedure and for the latest information, changes and errata items.

5.3 Network Installation Notes

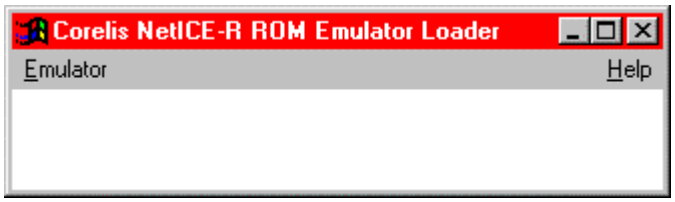
Windows 95/NT, when properly configured, takes care of WinSock and operation of the network card for you. No special setup beyond what Windows 95/NT has done when you installed your network card is required. The same applies to the HP-UX and Solaris versions of the software.

5.4 ROMLOAD Utility

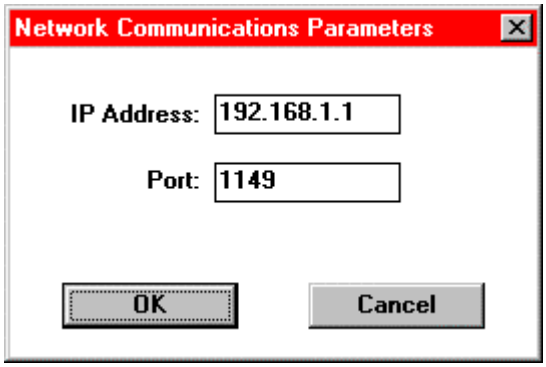
The ROMLOAD utility is used for remotely setting-up the NetICE-R memory configuration and for downloading Hex files from the PC computer to the NetICE-R memory.

S-Record format is a standard file format for hex files and is generated by most software development tools such as C Compilers and Assemblers. The S-record files are used to program EPROM/FLASH devices using a standard PROM programmer. The NetICE-R ROM Emulator eliminates the need for EPROM/FLASH device programming during the development phase of the project and enables the user to quickly and conveniently download new hex files to the emulation memory on board.

The graphical user interface (GUI) of the ROMLOAD software is simple and easy to use. The pull down Emulator menu has 4 options: **Communications**, **Configure**, **Load** and **Exit**.

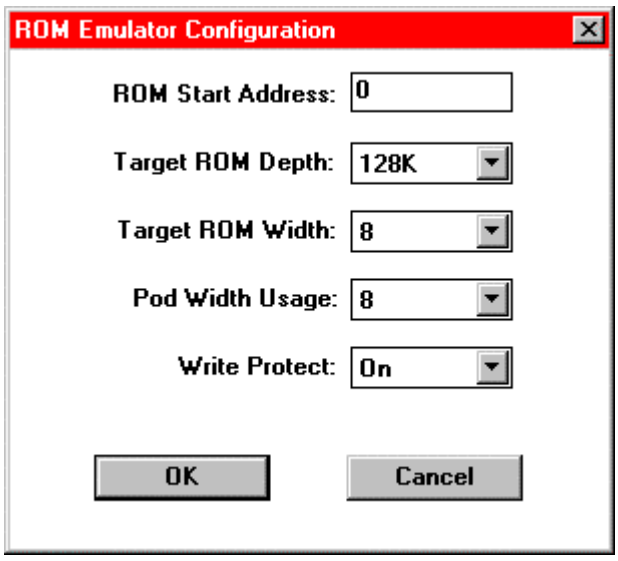


The **Communications** selection enables the user to specify the network communications parameters. First, the user needs to set the Ethernet IP address assigned to the NetICE-R box (see previous chapter for how to configure the IP address). Second, the user must specify the port number to which the software will communicate. The default port number is 1149 and it should not be modified by the user (unless instructed so in the readme.txt file on the disk).



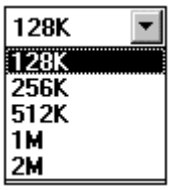
Once the communications parameters are programmed, the software is ready to communicate with the NetICE-R box.

The **Configure** option on the pull-down menu is for configuring the emulation memory to the desired target board interface configuration. The screen options are shown below:



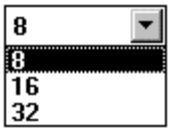
ROM Start Address is the location of the first physical address of the EPROM/FLASH memory on the target system. This parameter is subtracted from each memory address in the S-record file to create the actual offset address to the emulation memory.

Target ROM Depth is the total number of memory words being emulated. For example, if the target memory is 128K x16 or 128Kx8, the Target ROM Depth should be set to 128K. The pull down menu provides different selections as shown below. For selections smaller that 128K the user must specify 128K.



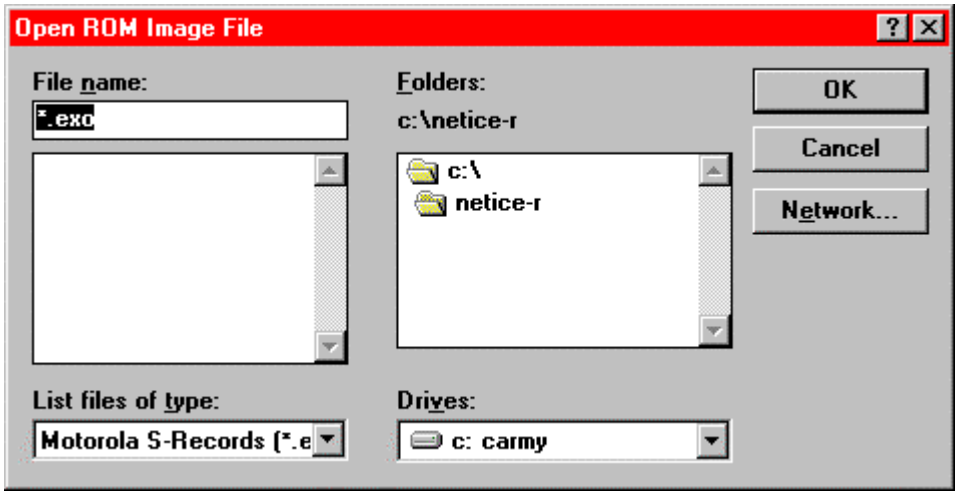
Note that memory size selected must agree with the total amount emulation memory that is in the NetICE-R. For example, specifying a depth of 2M for NetICE-R Option 1/1M would result in an error as the maximum memory in the box is limited to 1M words.

Target ROM Width specifies the width of the target EPROM/FLASH device(s) that is (are) being emulated. The available options are:



Write Protect option is for protecting the emulation memory from target write cycles. This feature, if ON, masks all target write cycles such that the emulation memory content can only be altered by the host computer. This feature is useful when emulating target memories that are read only by the target. It would also protect the emulation memory from spurious write cycles when the target power is cycled.

The **LOAD** menu selection is the final step of sending a hex file to the emulation memory. The user must specify any Motorola S-Record type hex file with a .exo suffix. The file can be anywhere on the host computer hard disk, or on a network drive.



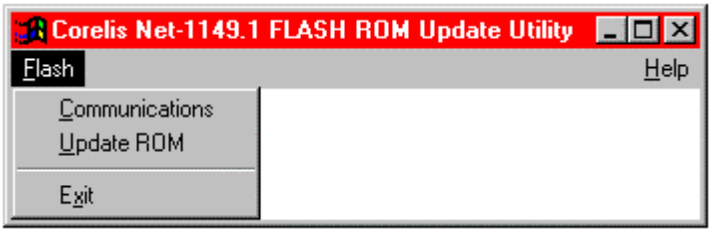
Once selected, loading the file to the ROM emulator is very fast. Downloading of 1M byte of code, in S-record format takes between 5 to 10 seconds, depending on the type of host computer used and the network traffic loading.

The **EXIT** menu selection is used to close the program and exit.

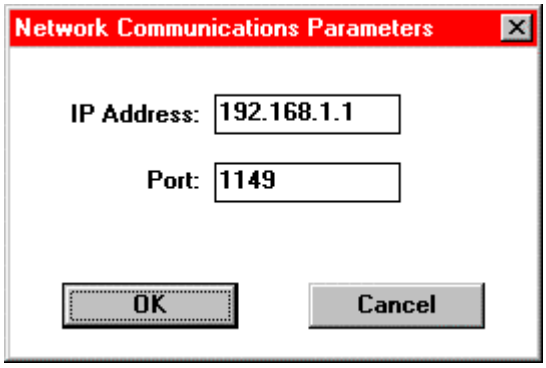
5.5 FLASH Utility

The FLASH utility is used for remotely program the firmware inside the NetICE-R unit. The FLASH software is only used if and when firmware upgrades are made and enable the user to update his unit without having to send it back to the factory.

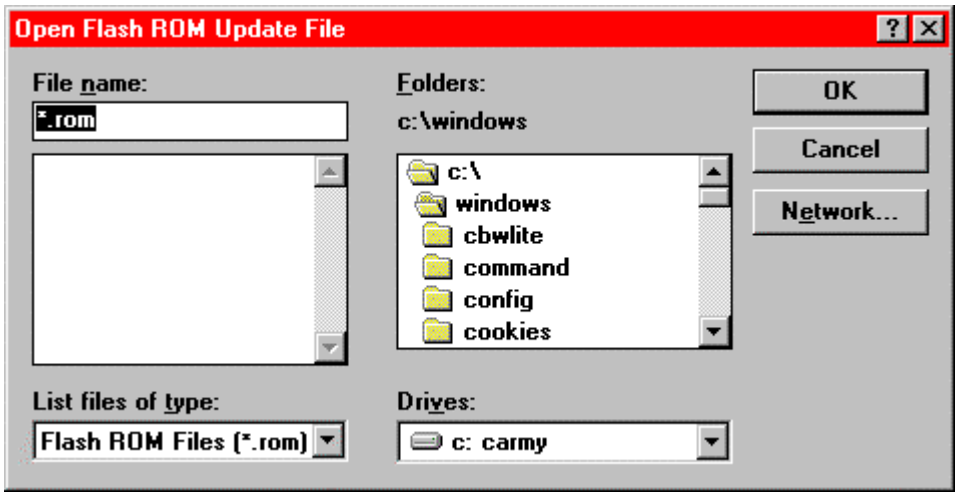
In order to use the FLASH utility, the user must have a PC computer with an Ethernet card and Windows 95 operating system.



The **Communications** menu selection is similar to the same selection of the ROMLOAD utility. This selection enables the user to specify the network communications parameters. First, the user needs to set the Ethernet IP address assigned to the NetICE-R box (see previous chapter for how to configure the IP address). Second, the user must specify the port number to which the software will communicate. The default port number is 1149 and it should not be modified by the user (unless instructed so in the readme.txt file on the disk).



Once the communications parameters are programmed, the software is ready to communicate with the NetICE-R box. The next step is to select the firmware file which would then be downloaded to and get stored in non-volatile FLASH memory in the NetICE-R unit. Select **Update ROM** on the menu to invoke the file browser and select the appropriate .ROM file.



The update .ROM file would be sent to the user if and when a firmware update is required.

5.6 ROM Function Library

To facilitate host application development, the NetICE-R is packaged with a ROM Function Library (RFL) Dynamic Link Library for use with 32-bit Windows program development.

5.6.1 Notes on Usage

When designing your Windows application program, you must create a .LIB import library file from the supplied DLL. Consult the User's Guide for your compiler for more information on how to accomplish this and how to link DLLs. Include the NETSFL32.H header in your application to ensure correct DLL function prototyping.

The NetICE-R communicates over the network using a TCP/IP connection. A connection between the host client program and the NetICE-R must be established first before any other functions can be performed. The application program must successfully execute the JTAG_Connect() DLL call to establish the NetICE-R socket number and connection parameters with the NetICE-R. It is equally important to correctly disconnect from the NetICE-R before exiting the host application program. This is accomplished by the execution of the JTAG_Disconnect() DLL function call.

Furthermore, RFL functions ROM_Control, ROM_Read, ROM_Write, ROM_Fill, and ROM_Download all require a previous call to ROM_Configure() to configure the ROM emulation module.

5.6.2 Summary (in alphabetical order) of RFL functions

JTAG_Connect()

This function will connect the client application with the NetICE-R. The socket number of the connection is returned as an informational item. (**NOTE:** Successful completion of this function must occur before any other function calls may execute.)

JTAG_Disconnect()

This function will disconnect the client application from the NetICE-R. (**NOTE:** Successful completion of this function must occur before termination of the application program.)

ROM_Configure()

This function configures the ROM emulation module.

ROM_Control()

This function controls the enabling of the host and target interfaces.

ROM_Download_File()

This function downloads a file into the ROM emulation memory.

ROM_Fill()

This function takes one item and writes it to a specified number of sequential locations in the ROM emulation memory.

ROM_Pod_Info()

This function returns binary data describing the pod configuration of the detected ROM emulator boards.

ROM_Read()

This function reads data from ROM emulation memory.

ROM_RFL_Version()

This function returns a zero-terminated ASCII string which provides the version number of the ROM Function Library DLL.

ROM_Version()

This function returns a zero-terminated ASCII string which provides the version level of the ROM emulator.

ROM_Write()

This function writes data to ROM emulation memory.

5.6.3 Detailed Descriptions of RFL functions

Function: **JTAG_Connect()**

Description:

This function connects the client application to the NetICE-R.

Prototype:

extern int _far_pascal JTAG_Connect (char FAR *, char FAR *);

Example Call:

```
unsigned int hSock;
char szHost[]="192.168.1.1";
char szService[]="1149";

/* Assign value of IP address to szHost */
/* Assign name of service (should be "1149") to szService */

hSock = JTAG_Connect((char FAR *) szHost, (char FAR *) szService);
```

Input Parameters:

szHost: This null-terminated character string represents either the name entry in a local computer hosts file, or the character representation of a full IP address in "dot format" for the NetICE (i.e. 192.168.1.1). The local host file is an ASCII file named *hosts* with the following format:

```
#
# a demo hosts file:  hosts
#
127.0.0.1       localhost
192.168.1.1     netice
192.168.1.15    hpsys
192.18.1.4      MyPC
```

The IP address on the left is associated with the name on the right. The hosts file needs to be located in the current path of the computer running the application. The IP address for

	the NetICE-R is set by the user prior to operation. It is preferred to use the IP address input format rather than the alias name input format. This avoids having to create and maintain a local host file.
szService:	This null-terminated character string represents the name of the service you wish to connect to in the NetICE-R. The function will accept the name and resolve it to a port number. You can configure the service port number via the RS-232 setup. See section 3.5. It is recommended to use the default JTAG service port “1149”.

Return Values:

0x00	INVALID_SOCKET (Operation failed)
other	Socket number of connected service

Function: **JTAG_Disconnect()**

Description:

This function disconnects the client application from the NetICE-R.

Prototype:

```
extern int _far_pascal JTAG_Disconnect();
```

Example Call:

```
int result;

result = JTAG_Disconnect();
```

Input Parameters:

None.

Return Values:

0x00	Command executed successfully
other	Command failed

Function: **ROM_Configure()**

Description:

This function configures the ROM emulation module.

Prototype:

```
extern int _far_pascal ROM_Configure (unsigned long, unsigned char, unsigned char,
                                      unsigned char, unsigned char);
```

Example Call:

```
unsigned long start_address;
unsigned char romsize;
unsigned char romwidth;
unsigned char podwidth;
unsigned char wp_enable;

/* Assign value of starting address to start_address */
/* Set target ROM size and width to romsize and romwidth, respectively */
/* Set pod width usage value */
/* Set wp_enable to 1 to turn on write_protect, or 0 to turn it off */

result = ROM_Configure(address, romsize, romwidth, podwidth, wp_enable);
```

Input Parameters:

start_address:	32-bit unsigned long which holds starting ROM address
romsize:	8-bit unsigned char which holds the encoded value of the ROM size. 0 = 128KB or smaller, 1 = 256 KB, 2 = 512 KB, 3 = 1 MB, 4 = 2 MB
romwidth:	8-bit unsigned char (assigned to either 8, 16, or 32) which specifies the target ROM width
podwidth:	8-bit unsigned char (assigned to either 8 or 16) which specifies the pod width usage
wp_enable:	8-bit unsigned char (assigned to either 1 or 0) which

indicates if write-protect is enabled

Return Values:

0x00	Success
0x11	Error on command header transmit
0x22	Error on data transmit
0x33	Error on receiving reply header
0x40	Invalid target ROM size
0x41	Invalid target ROM width
0x42	Invalid pod width usage size
0x43	Illegal pod usage for requested configuration
0x44	Error on receiving reply data
0x45	Insufficient pod depth for requested configuration
0x55	Reply Packet header <SOH> not 1
0x66	Reply Packet Sequence # incorrect
0x77	Unknown error from Net-1149.1

Function: **ROM_Control()**

Description:

This function controls the enabling of the host and target interfaces.

Prototype:

extern int _far_pascal ROM_Control (unsigned char);

Example Call:

unsigned char enable_option;

/* Set enable_option to 1 for host enable, 2 for target enable, or 0 for both disable */

result = ROM_Control(enable_option);

Input Parameters:

enable_option:	8-bit unsigned char which holds the encoded value of what to enable and disable. 0 = Both disable, 1 = Host enable/target disable, 2 = Target enable/host disable

Return Values:

0x00	Success
0x11	Error on command header transmit
0x22	Error on data transmit
0x33	Error on receiving reply header
0x44	Error on receiving reply data
0x50	ROM_Configure() not called yet
0x55	Reply Packet header <SOH> not 1
0x66	Reply Packet Sequence # incorrect
0x77	Unknown error from Net-1149.1

Comments:

Host and target cannot be enabled at the same time.

Function: **ROM_Download_File()**

Description:

This function downloads a file into the ROM emulation memory.

Prototype:

```
extern int _far_pascal ROM_File_Download (unsigned char, char FAR *,
                                          long *, unsigned char);
```

Example Call:

```
unsigned char device_number;
char dfilename[MAX_FILENAME_SIZE];
long brecvd;
unsigned char file_type;

/* Set device_number to current device number (should be 0) */
/* Copy the name of the desired HEX file to download from into dfilename */
/* Set the file type in file_type.  This should also be 0
   (which indicates Motorola S-Records) */

result = ROM_File_Download(device_number, (char FAR *)dfilename,
                                    &brecvd, file_type);
```

Input Parameters:

device_number:	8-bit unsigned char indicating the device number of the ROM emulator. This should be always set to 0.
dfilename:	A character array containing the name of the HEX file to download from.
brecvd:	A pointer to the 32-bit long which will store either the number of bytes received by the ROM Emulator, or the line number in the download file where an error occurred.
file_type:	The type of records to download into ROM emulation memory. This should always be set to 0 (to indicate Motorola S-Records).

Return Values:

0x00	Success
0x11	Unable to open file
0x22	Invalid record in file
0x33	Invalid format of record in file
0x40	Invalid device number
0x41	Data block out of ROM device range
0x44	Invalid file type
0x50	ROM_Configure() not called yet
0x55	Unexpected length of record file line
0x66	Checksum mismatch
0x77	Unknown error from Net-1149.1

Comments:

Error codes 0x22, 0x33, 0x41, 0x55, and 0x66 are all associated with a specific line in the data file being downloaded. The line number where any of these errors occurred is returned in the *brecvd* parameter.

Do not open the data file prior to calling this function. All file openings and closings are done within the function itself.

Function: **ROM_Fill()**

Description:

This function takes one item, as specified in "pattern", and writes it to the specified number of sequential locations in the ROM emulation memory.

Prototype:

extern int _far_pascal ROM_Fill (unsigned char, unsigned char, unsigned long, long);

Example Call:

```
unsigned char device_number;
unsigned char pattern;
unsigned long address;
long count;

/* Set device_number to current device number (should be 0) */
/* Put the desired fill pattern in pattern */
/* Set address to the starting address of the location in ROM Emulation
   memory to fill */
/* Set count to the number of times to put the pattern consecutively in memory */

result = ROM_File_Download(device_number, pattern, address, count);
```

Input Parameters:

device_number:	8-bit unsigned char indicating the device number of the ROM emulator. This should be always set to 0.
pattern:	8-bit unsigned char indicating the fill pattern to write to ROM emulation memory.
address:	A 32-bit unsigned long which indicates the starting address in ROM emulation memory where the filling will occur.
count:	A 32-bit long which indicates the number of consecutive times to fill the pattern in memory.

Return Values:

0x00	Success
0x11	Error on command header transmit
0x22	Error on data transmit
0x33	Error on receiving reply header
0x40	Invalid device number
0x41	Data block out of ROM device range
0x44	Error on receiving reply data
0x50	ROM_Configure() not called yet
0x55	Reply Packet header <SOH> not 1
0x66	Reply Packet Sequence # incorrect
0x77	Unknown error from Net-1149.1

Function: **ROM_Pod_Info()**

Description:

This function returns binary data describing the pod configuration of the detected ROM emulator boards.

Prototype:

```
extern int _far_pascal ROM_Pod_Info (unsigned char *, unsigned short *,
                                     unsigned short *, unsigned short *,
                                     unsigned short *, unsigned short *,
                                     unsigned short *, unsigned short *,
                                     unsigned short *);
```

Example Call:

```
unsigned char boardcount;
unsigned short e1p0_size;
unsigned short e1p0_depth;
unsigned short e1p1_size;
unsigned short e1p1_depth;
unsigned short e2p0_size;
unsigned short e2p0_depth;
unsigned short e2p1_size;
unsigned short e2p1_depth;

result = ROM_Pod_Info(&boardcount, &e1p0_size, &e1p0_depth,
                      &e1p1_size, &e1p1_depth, &e2p0_size,
                      &e2p0_depth, &e2p1_size, &e2p2_depth);
```

Input Parameters:

boardcount:	Pointer to a 8-bit unsigned char where the number of emulation boards is stored. (This will always be set to 1 – other values will be for future expansion).
e1p0_size:	Pointer to a 16-bit unsigned short indicating the size of Pod 0 of Emulator 1.
e1p0_depth:	Pointer to a 16-bit unsigned short indicating the

depth of Pod 0 of Emulator 1.

e1p1_size:	Pointer to a 16-bit unsigned short indicating the size of Pod 1 of Emulator 1.
e1p1_depth:	Pointer to a 16-bit unsigned short indicating the depth of Pod 1 of Emulator 1.
e2p0_size:	Pointer to a 16-bit unsigned short indicating the size of Pod 0 of Emulator 2. (Unused – reserved for future expansion)
e2p0_depth:	Pointer to a 16-bit unsigned short indicating the depth of Pod 0 of Emulator 2. (Unused – reserved for future expansion)
e2p1_size:	Pointer to a 16-bit unsigned short indicating the size of Pod 1 of Emulator 2. (Unused – reserved for future expansion)
e2p1_depth:	Pointer to a 16-bit unsigned short indicating the depth of Pod 1 of Emulator 2. (Unused – reserved for future expansion)

Return Values:

0x00	Success
0x11	Error on command header transmit
0x33	Error on receiving reply header
0x44	Error on receiving reply data
0x55	Reply Packet header <SOH> not 1
0x66	Reply Packet Sequence # incorrect
0x77	Unknown error from Net-1149.1

Function: **ROM_Read()**

Description:

This function reads data from ROM emulation memory.

Prototype:

```
extern int _far_pascal ROM_Read (unsigned char, unsigned long,  long,
                                 unsigned char FAR *);
```

Example Call:

```
unsigned char device_number;
unsigned long address;
long count;
unsigned char buf[MAX_READ_SIZE];

/* Set device_number to current device number (should be 0) */
/* Set address to the starting address of the location in ROM Emulation memory
   to read from */
/* Set count to the number of bytes to read from ROM Emulation memory */

result = ROM_Read(device_number, address, count, (unsigned char FAR *) buf);
```

Input Parameters:

device_number:	8-bit unsigned char indicating the device number of the ROM emulator. This should be always set to 0.
address:	A 32-bit unsigned long which indicates the starting address in ROM emulation memory to read from.
count:	A 32-bit long which indicates the number of bytes to read.
buf:	A character array which holds the read-in data.

Return Values:

0x00	Success
0x11	Error on command header transmit
0x22	Error on data transmit

0x33	Error on receiving reply header
0x40	Invalid device number
0x41	Data block out of ROM device range
0x44	Error on receiving reply data
0x50	ROM_Configure() not called yet
0x55	Reply Packet header <SOH> not 1
0x66	Reply Packet Sequence # incorrect
0x77	Unknown error from Net-1149.1

Function: **ROM_RFL_Version()**

Description:

This function returns a zero-terminated ASCII string which provides the version number of the ROM Function Library DLL.

Prototype:

```
extern int _far_pascal ROM_RFL_Version(char FAR *);
```

Example Call:

```
unsigned version[255];

ROM_RFL_Version((char FAR *) version);
```

Input Parameters:

version:	A character array that holds the return version number of the RFL portion of the SFL DLL.

Return Values:

0x00 Success

Function: **ROM_Version()**

Description:

This function returns a zero-terminated ASCII string which provides the version level of the ROM emulator.

Prototype:

extern int _far_pascal ROM_Version(char FAR *);

Example Call:

unsigned version[255];

ROM_Version((char FAR *) version);

Input Parameters:

version:	Pointer to a buffer where the version level will be stored. The buffer should be no shorter than 80 characters.

Return Values:

0x00	Success
0x11	Error on command header transmit
0x33	Error on receiving reply header
0x44	Error on receiving reply data
0x55	Reply Packet header <SOH> not 1
0x66	Reply Packet Sequence # incorrect
0x77	Unknown error from Net-1149.1

Miscellaneous:

An example return string would be “1.00”.

Function: **ROM_Write()**

Description:

This function writes data to ROM emulation memory.

Prototype:

```
extern int _far_pascal ROM_Write (unsigned char, unsigned long,  long,
                                  unsigned char FAR *);
```

Example Call:

```
unsigned char device_number;
unsigned long address;
long count;
unsigned char buf[MAX_READ_SIZE];

/* Set device_number to current device number (should be 0) */
/* Set address to the starting address of the location in ROM Emulation memory to
   write to */
/* Set count to the number of bytes to write to ROM Emulation memory */
/* Copy the data to be written into buf */

result = ROM_Write(device_number, address, count, (unsigned char FAR *) buf);
```

Input Parameters:

device_number:	8-bit unsigned char indicating the device number of the ROM emulator. This should be always set to 0.
address:	A 32-bit unsigned long which indicates the starting address in ROM emulation memory to write to.
count:	A 32-bit long which indicates the number of bytes to write.
buf:	A character array which holds the data to write.

Return Values:

0x00	Success
0x11	Error on command header transmit

0x22	Error on data transmit
0x33	Error on receiving reply header
0x40	Invalid device number
0x41	Data block out of ROM device range
0x44	Error on receiving reply data
0x50	ROM_Configure() not called yet
0x55	Reply Packet header <SOH> not 1
0x66	Reply Packet Sequence # incorrect
0x77	Unknown error from Net-1149.1

Index

10-Base2, 16
10-BaseT, 16
back panel indicators, 8
baud settings, 15
connectors, 8
data bus width, 7
DIP Switches. *See* Installation
diskette, 25
FLASH program 25, 31-32
  Communications selection, 31
  Update ROM selection, 32
front panel indicators, 8
front panel LEDs, 23
host interface, 7
included software, 25
installation
  contents of product, 9
  DIP Switches, 15
  IP Address, 13
  network, 11
  power supply, 10
interface cables, 16
IP Address. *See* Installation
JTAG_Connect, 33, 36
JTAG_Disconnect, 33, 38
JTAG connector pin assignment, 23
memory access, 7
memory configuration, 7
Motorola S-Record, 5, 27
NetICE-R
  features, 6
  host utility software, 5
  overview, 5
network. *See* Installation
operating environment, 8
options 4/1M and 4/2M, 23
parallel port, 19
physical size, 7
pin assignments, 18
power consumption, 8
power supply. *See* Installation
README.TXT, 25
repair, 2
Return Merchandise Authorization, 3
revisions, 1
RFL. *See* ROM Function Library
ROM Emulator pods, 21
ROM Function Library 25, 33-53
  import library file, 33
  JTAG_Connect() requirement, 33
  JTAG_Disconnect() requirement, 33
ROM_Configure, 39
ROM_Control, 41
ROM_Download_File, 42
ROM_Fill, 44
ROM_Pod_Info, 46
ROM_Read, 48
ROM_RFL_Version, 50
ROM_Version, 51
ROM_Write, 52
ROMLOAD program 25, 27-30
  communications selection, 27
  configure option, 28
  Load selection, 29
  ROM Start Address, 28
  Target ROM Depth, 29
  Target ROM Width, 29
  write protect, 29
RS-232 port, 13, 17
safety warnings, 3
serial port, 13, 17
storage environment, 8
target interface, 7
Unit Under Test (UUT), 5
version, 1
warranty service, 2

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP