USER MANUAL

1260-00C
SLOT 0 / Resource Manager

PUBLICATION NO. 980721

RACAL INSTRUMENTS

Racal Instruments, Inc.
4 Goodyear St., Irvine, CA 92618-2002
Tel: (800) 722-3262, FAX: (949) 859-7309

Racal Instruments, Ltd.
480 Bath Road, Slough, Berkshire, SL1 6BE, United Kingdom
Tel: +44 (0) 8706 080134; FAX: +44 (0) 1753 791290

Racal Systems Electronique S.A.
18 Avenue Dutartre, 78150 LeChesnay, France
Tel: +33 (1) 3923 2222; FAX: +33 (1) 3923 2225

Racal Systems Elettronica s.r.l.
Strada 2-Palazzo C4, 20090 Milanofiori Assago, Milan, Italy
Tel: +39 (02) 5750 1796; FAX +39 (02) 5750 1828

Racal Elektronik System GmbH.
Frankenforster Strasse 21, 51427 Bergisch Gladbach, Germany
Tel:+49 2204 92220; FAX: +49 2204 21491

Racal Australia Pty. Ltd.
3 Powells Road, Brookvale, NSW 2100, Australia
Tel: +61 (2) 9936 7000, FAX: +61 (2) 9936 7036

Racal Electronics Pte. Ltd.
26 Ayer Rajah Crescent, 04-06/07 Ayer Rajah Industrial Estate, Singapore 0513.
Tel: +65 7792200, FAX: +65 7785400

Racal Instruments, Ltd.
Unit 5, 25F., Mega Trade Center, No 1, Mei Wan Road, Tsuen Wan, Hong Kong, PRC
Tel: +852 2405 5500, FAX: +852 2416 4335

http://www.racalinst.com

PUBLICATION DATE: June 6, 2000

WARRANTY STATEMENT

All Racal Instruments, Inc. products are designed and manufactured to exacting standards and in full conformance to Racal's ISO 9001 procedures.

For the specific terms of your standard warranty, or optional extended warranty or service agreement, contact your Racal customer service advisor. Please have the following information available to facilitate service.

1. Product serial number
2. Product model number
3. Your company and contact information

You may contact your customer service advisor by:

E-Mail:	Helpdesk@racalinstruments.com	
Telephone:	+1 800 722 3262	(USA)
	+44(0) 8706 080134	(UK)
	+852 2405 5500	(Hong Kong)
Fax:	+1 949 859 7309	(USA)
	+44(0) 1628 662017	(UK)
	+852 2416 4335	(Hong Kong)

RETURN of PRODUCT

Authorization is required from Racal Instruments before you send us your product for service or calibration. Call your nearest Racal Instruments support facility. A list is located on the last page of this manual. If you are unsure where to call, contact Racal Instruments, Inc. Customer Support Department in Irvine, California, USA at 1-800-722-3262 or 1-949-859-8999 or via fax at 1-949-859-7139. We can be reached at: helpdesk@racalinstruments.com.

PROPRIETARY NOTICE

This document and the technical data herein disclosed, are proprietary to Racal Instruments, and shall not, without express written permission of Racal Instruments, be used, in whole or in part to solicit quotations from a competitive source or used for manufacture by anyone other than Racal Instruments. The information herein has been developed at private expense, and may only be used for operation and maintenance reference purposes or for purposes of engineering evaluation and incorporation into technical specifications and other documents which specify procurement of products from Racal Instruments.

FOR YOUR SAFETY

Before undertaking any troubleshooting, maintenance or exploratory procedure, read carefully the **WARNINGS** and **CAUTION** notices.

This equipment contains voltage hazardous to human life and safety, and is capable of inflicting personal injury.

If this instrument is to be powered from the AC line (mains) through an autotransformer, ensure the common connector is connected to the neutral (earth pole) of the power supply.

Before operating the unit, ensure the conductor (green wire) is connected to the ground (earth) conductor of the power outlet. Do not use a two-conductor extension cord or a three-prong/two-prong adapter. This will defeat the protective feature of the third conductor in the power cord.

Maintenance and calibration procedures sometimes call for operation of the unit with power applied and protective covers removed. Read the procedures and heed warnings to avoid “live” circuit points.

Before operating this instrument:

1. Ensure the instrument is configured to operate on the voltage at the power source. See Installation Section.

2. Ensure the proper fuse is in place for the power source to operate.

3. Ensure all other devices connected to or in proximity to this instrument are properly grounded or connected to the protective third-wire earth ground.

If the instrument:

- fails to operate satisfactory
- shows visible damage
- has been stored under unfavorable conditions
- has sustained stress

Do not operate until performance is checked by qualified personnel.

This page was left intentionally blank.

Table of Contents

List of Figures

List of Tables

Chapter 1
GENERAL DESCRIPTION

Introduction

This section provides information about the 1260-00C and its VXIbus/GPIB capabilities, local command set, an introduction to Code Instruments (CIs), and a description of the front panel.

General Information

The 1260-00C is a C-sized VXIbus module that links the IEEE-488 (GPIB) bus and the VXIbus. The 1260-00C performs transparent conversion of the GPIB signals and protocols to VXIbus signals and protocols so a GPIB Controller can control VXIbus instruments in the same way it controls GPIB instruments. Figure 1-1 shows the 1260-00C Interface Module.

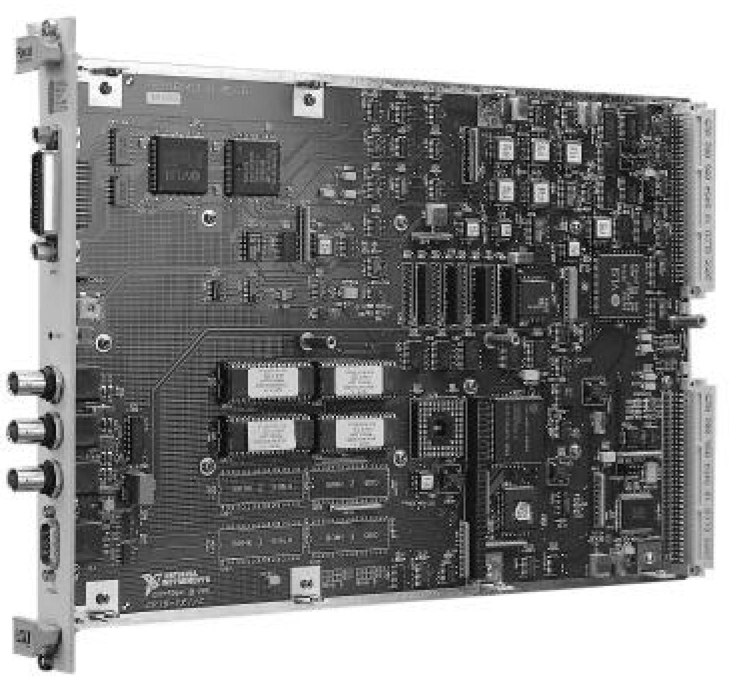

Figure 1-1. The 1260-00C Interface Module

The 1260-00C is factory configured to function as the system Resource Manager (RM). It performs the VXIbus start-up configuration, self-test, initialization functions, and VXIbus Slot 0-related services.

The RM and Slot 0 functions can be defeated individually to allow the 1260-00C to coexist with another RM, and/or be located in any slot.

What Your Kit Should Contain

Your 1260-00C kit should contain the following components:

Kit Component	Part Number
1260-00C Module	921277-XYZ
1260-00C User Manual	980721

The 1260-00C part number and serial number are printed on the label affixed to its shield casing.

If your kit is missing any of the listed items or if you have received the wrong version, contact Racal-Dana.

NOTE

The full part of the 1260-00C is determined by configuration options corresponding to the extension -XYZ in the part number shown in the previous table. The X, Y, and Z options are described below.

X 68881 Co-Processor
- 0 Without Co-Processor
- 1 With Co-Processor

Y ROM Option
- 1 User Firmware
- 2 Development Firmware
- 3 User Firmware and EPROM Expansion

	4	Development Firmware and EPROM Expansion
Z	RAM Option	
	1	512 kilobytes
	2	1 Megabyte
	3	2 Megabytes
	4	4 Megabytes

This page was left intentionally blank.

Chapter 2

INSTALLATION INSTRUCTIONS

Introduction

This section defines unpacking the 1260-00C, VXIbus characteristics, GPIB characteristics, local command set overview, Code Instruments and the Front Panel features.

Unpacking and Inspection

Prior to unpacking the switching module, check the exterior of the shipping carton for any signs of damage. All irregularities should be noted on the shipping bill. Remove the instrument from the carton, preserving the factory packaging as much as possible. Inspect the switching module for any defect or damage. Immediately notify the carrier if any damage is apparent. Before use, have qualified personnel perform a safety check.

The 1260-00C module is shipped packaged in an anti-static plastic bag to prevent electrostatic damage to the module. Several components on the module can be damaged by electrostatic discharge. To avoid such damage while handling the module, touch the plastic bag to a metal part of your VXIbus mainframe chassis before removing the module from the bag.

Use the original packaging when returning the switching module to Racal-Dana for calibration or servicing. The shipping carton and packaging will provide the necessary support for safe reshipment. If the original packaging is unavailable, wraping the switching module in plastic sheeting and use plastic spray foam to surround and protect the instrument. Reship in the original or a new shipping carton.

VXIbus Characteristics

The 1260-00C has the following VXIbus capabilities:

- Fully compatible with VXIbus System Specification
- VXIbus Resource Manager (RM) (defeatable)
- VXIbus Slot 0 support (defeatable)
- VXIbus Message-Based Commander and Message-Based

Servant

- VXIbus Master (A16, A24, D16, D08(EO))
- VXIbus Slave (A16, A24, A32, D16, D08(EO))
- Up to 4 Megabytes of dual-ported (shared) memory
- Three programmable VXIbus interrupt handlers
- IEEE 488.1 and IEEE 488.2 compatible multiple primary or multiple secondary
- 488-VXIbus translator

GPIB Characteristics

The 1260-00C has the following GPIB characteristics:

Communication With VXIbus Message-Based Devices

- VXI logical addresses are mapped to GPIB addresses
- Automatically configured at start-up
- Programmable

Interface

- NAT4882 and Turbo488 ASICs coupled with DMA
- Full, transparent support of individual status bytes for each GPIB address
- Buffered operation decouples GPIB and VXIbus operation
- Controller can address one VXIbus device to talk, and one or more other VXIbus devices to listen

IEEE 488.1 Capabilities

- SH1 (Source Handshake)
- AH1 (Acceptor Handshake)
- T5, TE5 (Talker, Extended Talker): multiple primary or multiple secondary addressing
- L3, LE3 (Listener, Extended Listener): multiple primary or multiple secondary addressing

- SR1 (Service Request)
- DC1 (Device Clear)
- DT1 (Device Trigger)
- RL0 (Remote Local)
- PP0 (Parallel Poll)

IEEE 488.2 Compatible, 488 VXIbus Translation

The IEEE 488.1 capabilities are supported for all VXIbus devices associated with GPIB addresses. The IEEE 488.2 compatibility applies to 488.2 compatible VXIbus devices associated with GPIB addresses through the 1260-00C.

Local Command Set Overview

The 1260-00C local command set supports the following types of operations:

System Configuration and Control

- Help
- General configuration
- RM information extraction
- VXI-defined common ASCII system commands
- Dynamic system configuration and reconfiguration
- GPIB address configuration
- VXIbus interrupt handler configuration
- IEEE 488.2 common commands

Instrument Development and Test

- VXIbus access
- Word Serial communication

CI Use and Development

- CI configuration

The command set can be accessed from the GPIB port, the serial port, and through Word Serial Protocol communication. Separate programmable local command response modes can be used for interactive and control program operation.

Code Instruments

The 1260-00C can run software modules called Code Instruments or CIs that perform special functions in the VXIbus environment. Typical applications of CIs include:

- Translating and interpreting command language
- Creating virtual (hierarchical) instrument
- Implementing Message-Based interface for Register-Based devices and non-VXI devices

CIs can be implemented in three forms:

- As part of the Racal-Dana-supplied firmware (Resident CIs, or RCIs)
- As user-developed downloadable object code (Downloaded CIs, or DCIs)
- As user-add-on firmware (EPROMed CIs, or ECIs)

For more information about CI capabilities and applications, see Appendix A, Code Instrument Overview.

Front Panel Features

The 1260-00C has the following front panel features:

Five Front Panel LEDs

- SYSFAIL LED reflects the status of the backplane SYSFAIL* signal and indicates that a VXIbus device in the system has failed.
- FAILED, TEST, and ON LINE LEDs indicate the current status of the 1260-00C.
- ACCESS LED indicates when the 1260-00C is accessed from GPIB or VXIbus, or when its MODID is asserted.

Five Front Panel Connectors

- GPIB interface
- Serial port

- Trigger input
- Trigger output
- External CLK10 I/O

Configurable Reset Pushbutton

- Pushbutton resets backplane
- Pushbutton resets 1260-00C
- Pushbutton resets both backplane and 1260-00C

This page was left intentionally blank.

Chapter 3

OPERATION

Introduction

This section contains information about the system configuration, 1260-00C configuration, and start-up operation.

System Configuration

The typical system includes the following components:

a. A VXIbus system mainframe containing the 1260-00C and instrument modules

b. A host computer with a GPIB interface module and associated driver software connected to the 1260-00C GPIB port

c. A dumb terminal or host running a terminal emulator connected to the 1260-00C serial port (optional)

The serial port settings are 9600 baud, 8-bit data, no parity, and one stop bit. Refer to Appendix D, Connectors, for descriptions of the RS-232 serial connector and the GPIB interface connector.

1260-00C Configuration

The 1260-00C, factory configuration is shown in Table 3-1. The RAM, firmware and co-processor are configured according to the 1260-00C purchase options.

Table 3-1. 1260-00C Factory Configuration

Function	**Factory Configuration**
Start-Up Mode	**488 VXI Runtime System Mode**
VXIbus Characteristics	
Resource Manager (RM)	Enabled
Logical Address	0
Servant Area Size	0
Shared Memory	0% of Installed Memory
Address Modifiers	Supervisor A16, Supervisor A24 Data
VXIbus Slot 0 Services	
CLK10 Driver	Enabled
CLK10 Source	Onboard Clock
SYSCLK Driver	Enabled
Priority Arbiter	Enabled
Bus Timeout	Enabled (BTO ≥250µsec)
Bus Requester	Level 3
VXI Interrupt Handlers	Unassigned
GPIB Addressing Mode	Multiple Secondary Addressing
1260-00C GPIB Primary Address	1
Serial Port	
System Start-Up Messages	Disabled
Console Local Command Port	Enabled
Discrete Fault Indicator (DFI)	Normally Open
Front Panel BNC Termination	
External Clock Input	Unterminated
External Trigger Input	Unterminated

The 1260-00C factory configuration does not have to be changed to use it as a Slot 0 Resource Manager. The following pages describe the factory configuration settings, and present alternate configurations.

Figure 2-1 shows the location of the 1260-00C configurable components and their physical location relative to some of the major circuit components. The jumpers and switches are represented in their factory default positions.

NOTE

The 1260-00C is housed in a metal enclosure that has cut-outs for access to all switches and jumpers associated with Slot 0/Non-Slot 0 settings, start-up mode, and Shared RAM settings. Under normal circumstances, you do not need to open the enclosure.

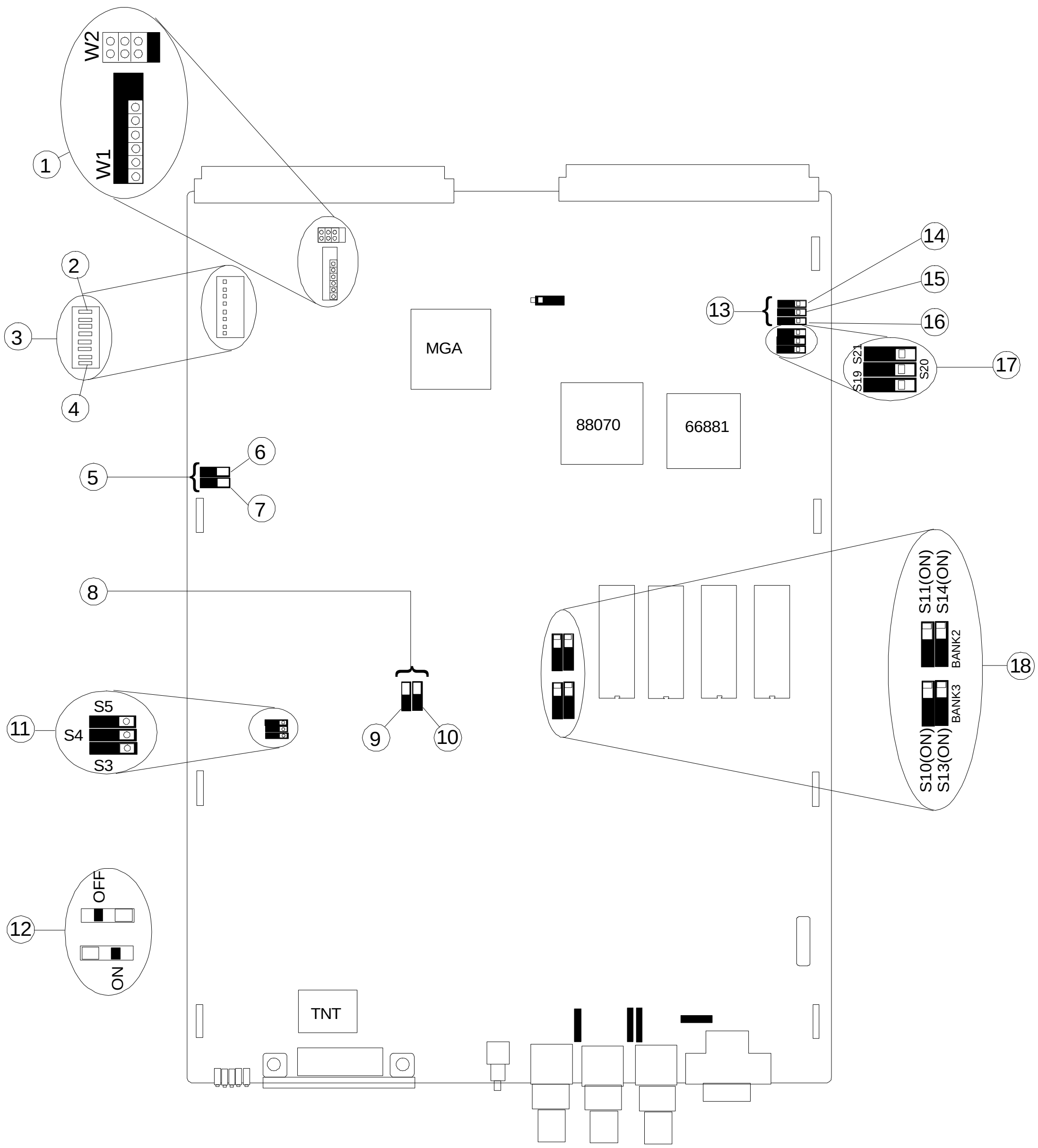

Figure 3-1, 1260-00C Parts Locator Diagram

1. VXIbus Requester Level
2. MSB
3. Logical Address DIP switch (set to FFh)
4. LSB
5. Shared RAM Switches
6. S2 (OFF)
7. S1 (OFF)
8. Installed RAM Switches
9. S6
10. S7
11. Address Modifiers (ON)
12. Detail of Switch Settings
13. Slot 0 Switches
14. S24 (ON)
15. S23 (ON)
16. S22 (ON)
17. Startup Mode Switches (ON)
18. Eprom Expansion Switches

Setting the Logical Address, GPIB Primary Address, and Servant Area Size

To change the logical address, GPIB primary address, and Servant area size, run the non-volatile memory configuration utility described in Chapter 5, Non-volatile Configuration.

The logical address can be changed by setting DIP switch SW1. By default, all the switches are set to the up position (0xFF). At this setting, the 1260-00C reads the logical address from the onboard EEPROM. To change the logical address, set the switches to the hex value of the logical address. Switch position 1 is the MSB; 8 is the LSB. Up is logical 1; down is logical 0.

Verifying the Installed RAM Size

Up to 4 Megabytes of local RAM is factory-installed on the 1260-00C, but is configured to use the minimum amount of memory is 512 kilobytes. Table 3-2 lists the RAM configurations and their associated switch settings. Use this information to change the board configuration.

Table 3-2. Installed RAM Configuration

Installed Memory Size	Switch S6 Setting	Switch S7Setting
512 kilobytes	OFF	OFF
1 Megabyte	OFF	ON
2 Megabytes	ON	OFF
4 Megabytes	ON	ON

Table 3-3 shows the relationship between the amount of installed memory, local address range occupied by the memory, and the range of VXI A24 addresses accessible by the 1260-00C as a bus master.

Table 3-3. 1260-00C CPU Local and A24 Memory Ranges

Installed Memory Size	Installed Memory Local Address Range		Accessible VXI A24 Address Range	
	Start	End	Start	End
512 kilobytes	000000h	07FFFFh	080000h	E7FFFFh
1 Megabyte	000000h	0FFFFFh	100000h	E7FFFFh
2 Megabytes	000000h	1FFFFFh	200000h	E7FFFFh
4 Megabytes	000000h	3FFFFFh	400000h	E7FFFFh

Setting The Shared Memory Size

To set the amount of installed memory shared with the VXIbus, change the settings of switches S1 and S2. Table 3-4 gives the S1 and S2 switch settings for sharing various portions of RAM with the VXIbus for each possible installed memory configuration.

Table 3-4. Shared Memory Switch Settings

Configured **Memory Size**	Amount of Installed Memory Shared With VXIbus			
	S1 ON S2 ON	S1 OFF S2 ON	S1 ON S2 OFF	S1 OFF S2 OFF
512 kilobytes	512 kilobytes	256 kilobytes	128 kilobytes	None
1 Megabyte	1 Megabyte	512 kilobytes	256 kilobytes	None
2 Megabytes	2 Megabytes	1 Megabyte	512 kilobytes	None
4 Megabytes	4 Megabytes	2 Megabytes	1 Megabyte	None

NOTE

The RAM shared with the VXIbus will be the upper portion of the installed memory.

The 1260-00C Offset Register holds the shared memory VXI A24 base address, as described in the VXIbus specification. The RM automatically configures the Offset Register at start-up.

Setting the Reset Operation

The 1260-00C has three configurable reset parameters. They can be enabled or disabled, and are as follows:

- Pushbutton resets backplane (asserts SYSRESET* signal).
- Pushbutton resets 1260-00C (asserts local reset signal).
- Backplane SYSRESET* signal resets 1260-00C (SYSRESET* on backplane asserts local reset).

The reset parameters can be altered by the non-volatile memory configuration as described in Chapter 4, Change Configuration Information.

Setting the VXIbus Requester Level

To change the VXIbus requester level of the 1260-00C, move the jumpers on jumper blocks W1 and W2 as shown in Figure 3-2. The 1260-00C is configured at the factory to be a Level 3 requester.

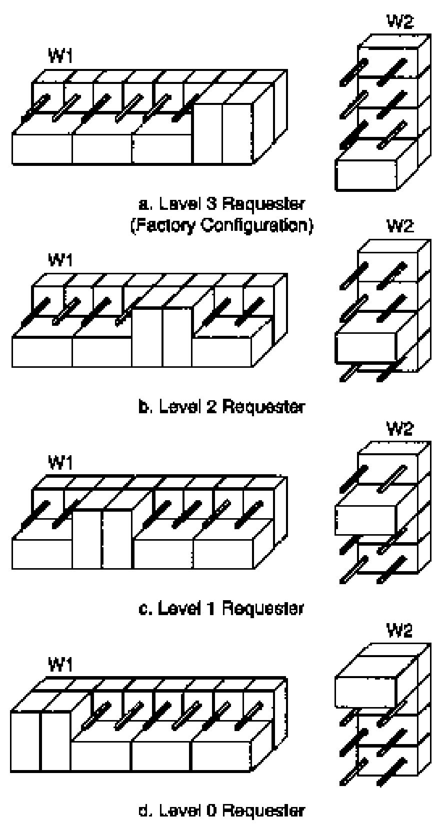

Figure 3-2 VXIbus Requester Jumper Settings

Setting the VXI

As part of the hardware capabilities on the 1260-00C, there are

Interrupt Handler Levels

three VXI programmable interrupt handlers. They are assigned dynamically by the RM, or statically according to the contents of the non-volatile memory, as described in Chapter 5.

External Input Termination

Switches S12 and S16 enable a 50-ohm termination to ground for the external trigger and external clock inputs. The 1260-00C is factory-configured with the termination disabled for both the external trigger and the external clock inputs. Figure 3-3 shows the settings required to enable or disable the termination on the external trigger. Figure 3-4 shows the settings required to enable or disable the termination on the external clock.

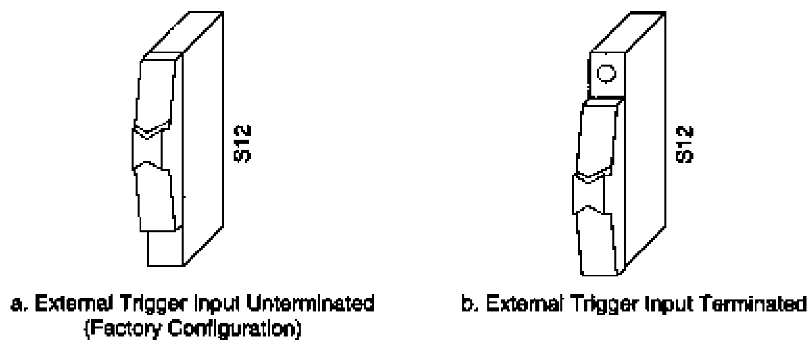

Figure 3-3, External Trigger Input Termination

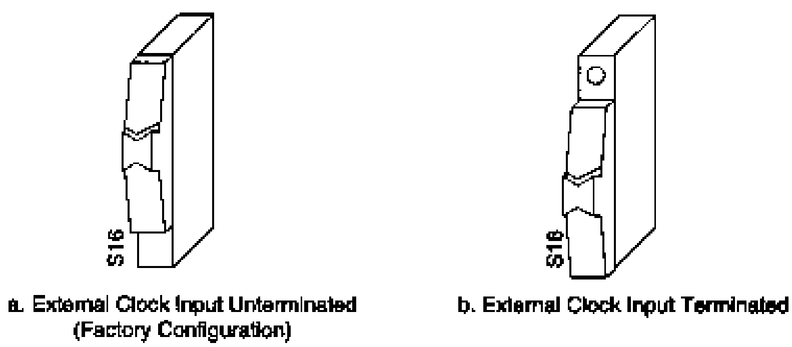

Figure 3-4, External Clock Input Termination

EPROM Configuration

The amount of Read Only Memory (ROM) in the 1260-00C can vary from 512 kilobytes to 1 Megabyte. The standard configuration consists of 512 kilobytes of EPROM used for the operating firmware. An EPROM expansion option can be used to give an additional 512 kilobytes of EPROM space. It contains four sockets and four switches that can be used to install a user-developed code.

The EPROM expansion sockets accommodate combinations of 2764, 27128, 27256, 27512, and 27010 EPROMs. Table 2-5 lists the possible EPROM memory configurations. Bank 2 has a base address of E80000h, and Bank 3 starts at EC0000h. The maximum EPROM expansion memory size is 512 kilobytes.

Table 3-5. Expansion EPROM Configurations

EPROM Size	BANK 2 (U47, U55)	BANK 3 (U53, U59)	S11	S14	S10	S13	End Address
16K	2764	None	OFF	OFF	OFF	OFF	E83FFFh
32K	27128	None	OFF	OFF	OFF	OFF	E87FFFh
64K	27256	None	OFF	ON	OFF	OFF	E8FFFFh
128K	27512	None	ON	ON	OFF	OFF	E9FFFFh
256K	27010	None	ON	ON	OFF	OFF	EBFFFFh
272K	27010	2764	ON	ON	OFF	OFF	EC3FFFh
288K	27010	27128	ON	ON	OFF	OFF	EC7FFFh
320K	27010	27256	ON	ON	OFF	ON	ECFFFFh
384K	27010	27512	ON	ON	ON	ON	EDFFFFh
512K	27010	27010	ON	ON	ON	ON	EFFFFFh

When inserting EPROMs into the expansion EPROM slots, orient them according to the silkscreen printed on the board, as shown in Figure 3-1. The 2764, 27128, 27256 and 27512 EPROMs have fewer pins than the expansion sockets. In these cases, align the **bottom** pins of the EPROM with the **bottom** pins of the socket, leaving the top pins open, as illustrated in Figure 3-5.

WARNING

Improper EPROM installation can result in damage to the EPROM, 1260-00C, or both

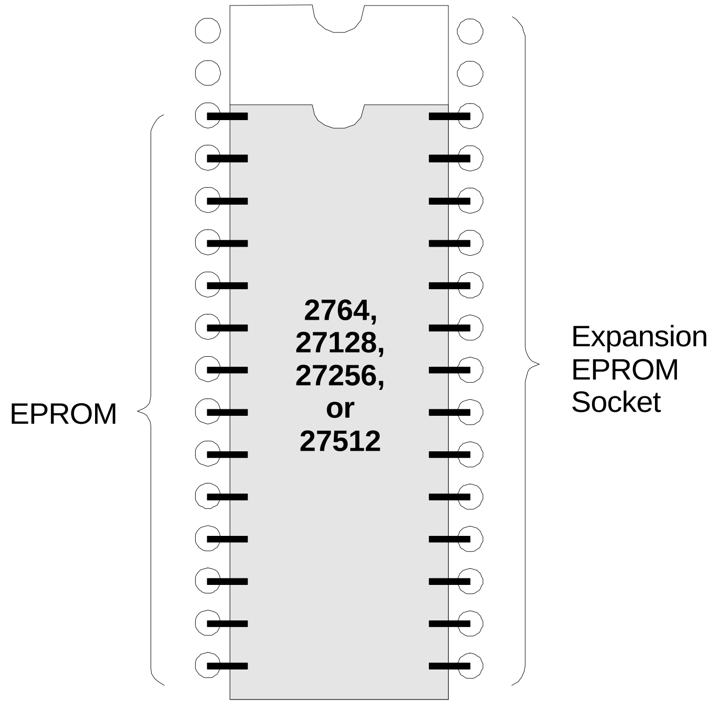

Figure 3-5, EPROM Insertion Position

Discrete Fault Indicator Configuration

The 1260-00C comes with a MATE-compatible Discrete Fault Indicator (DFI). The 1260-00C monitors the status of the VXIbus SYSFAIL* signal and relays the status to pins 1 and 6 of the RS-232 serial port (see Appendix D, Connectors, in the back of this manual.

As shown in Figure 3-6 and Table 3-6, switch S17 determines the relationship between the SYSFAIL* signal and the serial port pins. If S17 is in the OFF position, the 1260-00C DFI is set to the normally open mode. Therefore, if SYSFAIL* is not asserted while the backplane is powered up, pins 1 and 6 will present an electrical open-circuit. In contrast, if the backpane is unpowered or SYSFAIL* is asserted, pins 1 and 6 will present an electrical short-circuit.

If S17 is in the ON position, the 1260-00C DFI is set to the normally closed mode. Therefore, if SYSFAIL* is not asserted while the backplane is powered up, pins 1 and 6 will present an electrical chort-circuit. In contrast, if the backplane is unpowered or SYSFAIL* is asserted, pins 1 and 6 will present an electrical open-circuit.

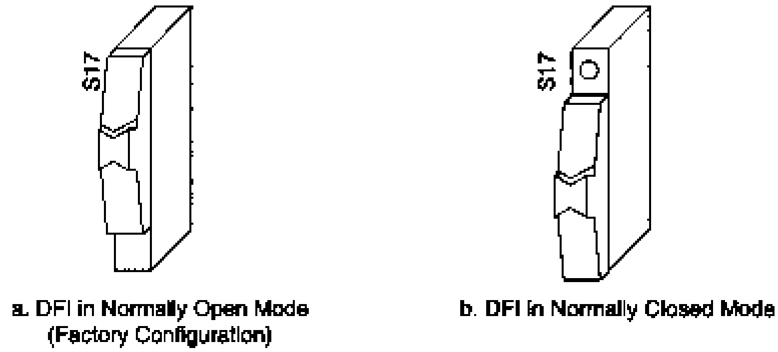

Figure 3-6, Discrete Fault Indicator Configuration

Table 3-6, Discrete Fault Indicator Options

Switch S17	Power	SYSFAIL	Pins 1 & 6
OFF Figure 3-6 (a)	OFF ON ON	N/A Asserted Unasserted	Short-Circuit Short-Circuit Open-Circuit
ON Figure 3-6 (b)	OFF ON ON	N/A Asserted Unasserted	Open-Circuit Open-Circuit Short-Circuit

Address Modifier Configuration

By setting onboard switches, the 1260-00C can specify the state of the VXIbus Address Modifiers during a VXI master access. During A16 accesses, the lines AM5, AM4, and AM3 are needed high, low, and high; and AM1 is needed low. During A24 accesses, the lines AM5, AM4, and AM3 are all needed high. The 1260-00C drives the upper three address modifier lines appropriately for every access. Configure the 1260-00C to drive the lower three address modifier lines as needed.

Switches S3, S4, and S5 control the AM0, AM1, and AM2 signals. Figure 3-7 shows the valid settings of S3, S4, and S5.

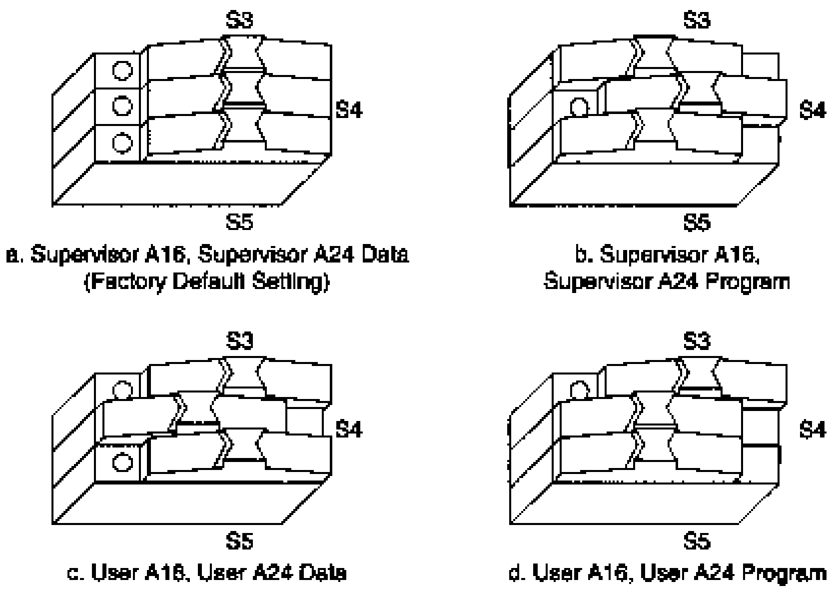

Figure 3-7, Address Modifier Signals Switch Settings

1260-00C Start-Up Mode Configuration

Start-up mode switches S19 and S20 control the 1260-00C operation mode at system start-up. They select one of four modes, as shown in Figure 3-8. The four possible modes of start-up are:

1. 488-VXI Runtime System Mode - The start-up mode for normal operation in a VXI system, and is configured at the factory to start up in this method. The remainder of this section contains a description of this operation.

2. Non-Volatile Configuration Mode - Edits the contents of the non-volatile configuration parameter memory. See Chapter 5, Non-Volatile Configuration for more information.

3. Diagnostics Mode - Performs extensive offline diagnostic tests on the 1260-00C. See Chapter 6, Diagnostic Tests, for a description of the self-tests.

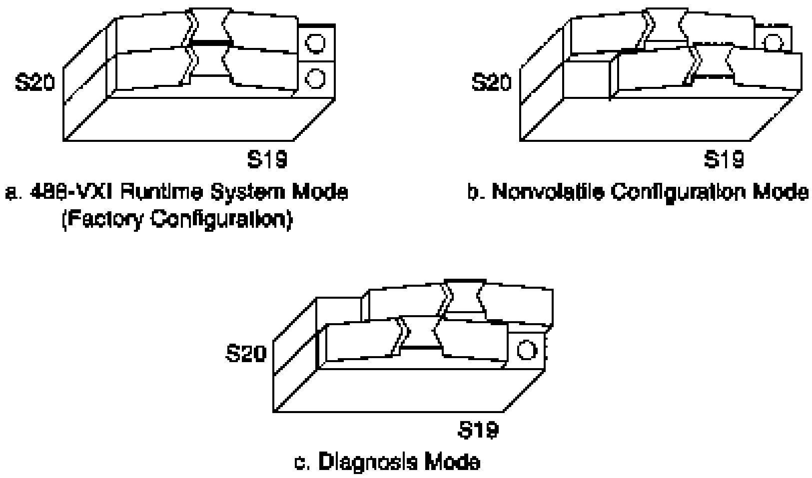

Figure 3-8, Start-Up Mode Switch Settings

488 VXI Runtime System Operation

The 1260-00C is factory configured as a Slot 0 Resource Manager. The Slot 0 and Resource Manager (RM) functions can be independently defeated, resulting in four modes of operation:

1. Slot 0 Resource Manager (factory configuration)
2. Non-Slot 0 Resource Manager
3. Non-Slot 0 Message-Based device (Non-Resource Manager)
4. Slot 0 Message-Based device (Non-Resource Manager)

This section describes the 1260-00C configuration procedures and start-up behavior for each mode of operation.

WARNING

Do not install a 1260-00C configured for Non-Slot 0 operation in Slot 0, or a 1260-00C configured for Slot 0 operation in any slot other than Slot 0. Doing so can damage the 1260-00C, mainframe, or other modules.

System Start-Up Message Printing

The serial port start-up printout enable switch S21 controls whether or not VXI system start-up messages are printed to the serial port, as shown in Figure 3-9. The factory default configuration disables this function.

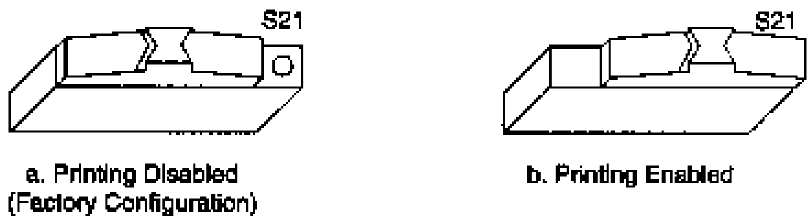

Figure 3-9, VXI System Start-Up Message Switch Settings

Slot 0 Resource Manager Configuration

To configure the 1260-00C for Slot 0 Resource Manager operation, enable the VXIbus Slot 0 functions, and set the logical address to 0, as shown in Table 3-7.

Table 3-7, Slot 0 Resource Manager Operation Switch and Jumper Settings

Jumper/Switch	Position	Function
Switches S9 and S15	See Table 3-8.	CLK10 routing options.
Switch S22	ON	VXI BTO enabled.
Switch S23	ON	Bus arbiter and SYSCLK enabled. CLK10 sourcing for backplane is enabled.
Switch S24	ON	MODID pulled up
Logical Address	See Chapter 5.	Logical address is 0. Set in non-volatile configuration or use the DIP switch.
Slot 0 Model Code	See Chapter 5.	Model code is set to the Slot 0 value. Set in non-volatile configuration.

Table 3-8, CLK10 Routing Options

Switch S15	Switch S59	Function
OFF	OFF	CLK10 sourced from onboard clock.
ON	OFF	CLK10 and EXT CLK connector sourced from onboard clock.
OFF	ON	CLK10 sourced from an external clock via the EXT CLK connector.
ON	ON	Invalid. Do not use this setting.

Slot 0 Resource Manager Operation

At start-up, a 1260-00C configured as a Slot 0 Resource Manager performs its self-tests, executes the RM functions, and enters its normal mode of operation.

Front Panel LED Indications for RM Operation

The five front panel LEDs are SYSFAIL, FAILED, TEST, ON LINE, and ACCESS. The 1260-00C uses the FAILED, TEST, and ON LINE LEDs to indicate the progress of its self-initialization, self-test, and RM functions. The LED indications are shown in Table 3-9. A successful system start-up will sequence through the first five states. The point of failure is indicated for states in which the FAILED LED is lit for an extended period of time. The LED indications are identical for Slot 0/Non-Slot 0 Resource Manager operation.

Table 3-8. Front Panel LED Indications For RM Operation

Sequence	FAILED	TEST	ON LINE	State	Point of Failure
1	OFF	OFF	OFF	No power	Failed before self-test
2	ON	OFF	OFF	In self-initialization	Failed in self-test
3	ON	ON	OFF	In self-test	
4	OFF	ON	ON	Performing RM	
5	OFF	OFF	ON	Online	
	ON	ON	ON	Failed	Failed while in RM
	ON	OFF	ON	Failed	Failed while online
	OFF	ON	OFF	In non-volatile configuration, or diagnostics mode	

The SYSFAIL LED is lit whenever any device in the system is asserting the VXIbus SYSFAIL* signal.

The ACCESS LED flashes whenever the 1260-00C is accessed from the GPIB, or from the VXIbus. It also indicates when its MODID is asserted.

Self-Test Operation

The self-test sequence tests the basic functionality of many 1260-00C components, including EPROM, RAM, I^2C bus, RS-232 port, DMA channels, GPIB port, interrupt logic, timer, and VXIbus registers (MIGA). You can execute full tests of the 1260-00C in diagnostics mode, as described in Chapter 6.

RM Operation

The RM waits until all devices have stopped driving the VXIbus SYSFAIL* signal, or until five seconds have elapsed after the VXIbus SYSRESET* signal is negated. During this period, all VXIbus devices in the system should have completed their self-tests.

NOTE

Configure the 1260-00C to wait for any number of seconds before RM operations begin.

The RM scans Logical Addresses 1 through 254 for Static Configuration Devices (SC Devices). For each SC device found, it reads the device class and manufacturer's ID code from the ID Register, and the model code from the Device Type Register. If the device is an extended device, the RM reads its Subclass Register. The RM then performs slot associations for each static configuration device by reading its Status Register while asserting each MODID line.

The RM looks for Dynamic Configuration Devices (DC Devices) at Logical Address 255 by asserting each MODID line and reading the device's ID Register. DC devices initially have a logical address of 255. The RM subsequently assigns each DC device a different logical address. For each DC device found, it not only reads the device's configuration registers as with SC devices, but also assigns each device the next unused logical address by writing the appropriate value to the device's Logical Address Register. Using the non-volatile configuration mode, set the starting logical address for the RM to begin assigning DC devices. Refer to Chapter 5 for more information on non-volatile configuration.

If any device has not passed its self-test, the RM forces that device offline by setting the Sysfail Inhibit and Reset bits in that device's Control Register.

The RM determines the address space of each device by reading its ID Register. If the device's address space is A16/A24 or A16/A32, the RM allocates a section of A24 or A32 memory space to the device according to the memory requirements indicated by the contents of its Device Type Register, and writes an appropriate value to the device's Offset Register.

The RM configures the initial Commander/Servant hierarchy according to each Commander's Servant area size, using the algorithm described in the VXIbus specification. The RM issues the appropriate *Read Servant Area* and *Device Grant* commands to each SC Commander. The RM retains all devices not assigned to other Commanders as its immediate Servants. Regardless of where DC device logical addresses are assigned, they are never granted to an SC Commander. The DC Commander/Servant hierarchy can be created in one of two ways:

- All DC devices can be automatically assigned as Servants of Logical Address 0 (the Resource Manager).

- A custom hierarchy can be created through the use of the local command set functions, as described in the *DC Commands and Queries* section of Chapter 4, *Local Command Set*.

The RM sends the *Read Protocols* query to all Message-Based devices. The response to the query is saved internally for later use in interrupt handler and GPIB configuration.

The RM configures the VXI interrupter and interrupt handlers using a seven-entry table contained in non-volatile configurations. During the VXI interrupt configuration, the RM assigns interrupt levels to all Programmable Handlers (PH) and Programmable Interrupters (PI). Each entry in the table represents the logical address of the handler that handles the corresponding level (1 through 7). If the handler is static, PI Servants are assigned to the level. If the device is a PH device, the RM assigns both it and any PI Servants to the corresponding level. If the table entry is FFh, the level is free to be assigned to any PH device. If only PH and PI devices are in a system, all entries may contain FFh. See Chapter 4 for more complete details.

The remainder of the RM procedure depends upon whether the RM found any DC devices in the system.

Static Configuration

When all of the previous operations are complete and successful,

Operation

the RM sends the Word Serial command *Identify Commander* to all immediate Message-Based Servants with bus master capability. At this point, the RM is ready to bring the system into the Normal Operation sub-state. This is accomplished by sending the Word Serial query *Begin Normal Operation* to all top-level Commanders, and immediate Message-Based Servants.

Dynamic Configuration Operation

If the system is a DC system (at least one DC device was found), and the non-volatile configuration specifies the RM should create a hierarchy with DC devices assigned to Logical Address 0, the RM follows the same steps as previously described in <u>Static Configuration Operation</u>. DC devices are treated as SC devices from this point on.

In order to customize a DC hierarchy and the non-volatile configuration specifies the RM not finish configuring the hierarchy, the 1260-00C RM does not send *Identify Commander* or *Begin Normal Operation* to any devices, either static or dynamic. The outside controller (or EPROMed CI) can create the DC Commander/Servant hierarchy without having to dynamically reconfigure the system. Use the 1260-00C local command `DCGrantDev` to create the DC hierarchy. When the system is configured and ready to make a transition to the Normal Operation sub-state, send the 1260-00C local command `DCBNOSend`. `DCBNOSend` sends the *Identify Commander* and *Begin Normal Operation* commands to Message-Based devices as previously described in <u>Static Configuration Operation</u>. See the <u>DC Commands and Queries</u> section of Section 4 for further information about dynamic configuration operation.

The 1260-00C performs general configuration operations. The 1260-00C creates GPIB address links for its immediate Message-Based SC Servants. After this, the 1260-00C RM and general configuration operations are complete.

GPIB Address Assignment

The 1260-00C automatically assigns GPIB addresses (primary or secondary) to itself and to each of its immediate Message-Based SC Servants. If the Message-Based device does not support minimal Word Serial[I] or VXIbus 488.2[I4] capabilities, no GPIB address link is created. The 1260-00C assigns a GPIB address to each device according to the top five bits of its logical address. For example, the GPIB address of a device with Logical Address 96 (01100000b) would be 12 (01100b).

If two or more devices have logical addresses with the same top five bits, the 1260-00C assigns GPIB addresses to devices in order of the least significant three bits. Conflicting devices are given the next available GPIB address. For example, if the 1260-00C and its Message-Based Servants have Logical Addresses 0, 24, 27, and 33, the 1260-00C assigns GPIB addresses as shown in Table 3-10.

Table 3-9. Example GPIB Address Assignment

Logical Address		3 LSB (Order of Assignment)	5 MSB	GPIB Address
Decimal	**Binary**	**Binary**	**Binary**	**Decimal**
0	00000000b	000b	00000b	0
24	00011000b	000b	00011b	3
33	00100001b	001b	00100b	4
27	00011011b	011b	00011b	5

In the example shown in Table 3-10, the device at Logical Address 27 was assigned GPIB Address 5 because addresses 3 and 4 were previously assigned. By spacing the 1260-00C Message-Based Servants at intervals of eight logical address locations, situations are avoided by which removing or adding one device changes the GPIB address of another device.

The default configuration for the 1260-00C is to use multiple GPIB secondary addresses (not multiple primary addresses). To change the configuration to use multiple primary addresses through non-volatile memory configuration is described in Chapter 5, Change Configuration Information.

Changing the self-assigned default GPIB address of the 1260-00C through the non-volatile memory configuration is described in Section 5. The default GPIB address of the 1260-00C when configured for multiple secondary addresses is Secondary Address 0 (Primary Address 1). The default GPIB address of the 1260-00C when configured for multiple primary addresses is Primary Address 1 (no secondary address).

At times, when using multiple primary addressing, it may be necessary to avoid particular GPIB addresses to avoid conflicts with GPIB instruments outside of the VXI mainframe. GPIB addresses to avoid can be specified through the non-volatile memory configuration as described in Chapter 5.

System Configuration

During the execution of the RM and general configuration

Table

operations, the 1260-00C builds up a table of system configuration information. Each device has an entry in the table containing the device's logical address, Commander's logical address, GPIB address, slot number, device class, manufacturer ID number, model code, memory space requirement, memory base address, and memory size. The 1260-00C retains this table after the RM and general configuration operations are complete. The information in the table is accessible through the 1260-00C local command set. The GPIB address entry is meaningful only for immediate Message-Based Servants of the 1260-00C.

Non-Slot 0 Resource Manager Configuration

To configure the 1260-00C for Non-Slot 0 Resource Manager operation, use the following procedure:

a. Disable the VXIbus Slot 0 hardware functions.

b. Set the model code of the 1260-00C to be configured for Non-Slot 0 operation using the non-volatile configuration mode.

c. Set the logical address to 0 in non-volatile configuration mode or by using DIP switch SW1.

d. Set the various switches, jumpers and non-volatile configurations as shown in Table 3-11.

Table 3-11, Non-Slot 0 Resource Manager Operation Switch and Jumper Settings

Jumper/Switch	Position	Function
Switch S15	OFF	If S15 is ON, the 1260-00C also routes CLK10 to the EXT CLK connector on the front panel.
Switch S22	OFF	VXI BTO disabled.
Switch S23	OFF	Bus arbiter and SYSCLK disabled. CLK10 receiving from backplane.
Switch S24	OFF	MODID pulled down.
Logical Address	See Chapter 5.	Logical address is 0. Set in non-volatile configuration or use the DIP switch.
Non-Slot 0 Model Code	See Chapter 5.	Model code is set to the Non-Slot 0 value. Set in non-volatile configuration.

Non-Slot 0 Resource Manager Operation

The start-up sequence for a 1260-00C configured for Non-Slot 0 Resource Manager operation is nearly identical to the Slot 0 Resource Manager operation. In Non-Slot 0 RM operation, the 1260-00C controls the Slot 0 resources remotely.

A VXIbus Slot 0 device must be in the system. It must be either a Register-Based device that implements the MODID Register, or a Message-Based device that supports the Word Serial commands *Read* MODID, *Set Lower* MODID, and *Set Upper* MODID. VXIbus Specification Revision 1.2 Message-Based Slot 0 devices are **not** supported.

Non-Slot 0 Message-Based Device Configuration (Non-Resource Manager)

To configure the 1260-00C for Non-Slot 0 Message-Based operation, use the following procedure:

a. Disable the VXIbus Slot 0 functions.

b. Set the model code of the 1260-00C to be configured for Non-Slot 0 operation using the non-volatile configuration mode.

c. Set the logical address to a non-zero value with an appropriate Servant area size using the non-volatile configuration mode. If the logical address is set to FFh, the 1260-00C will participate in dynamic configuration. Otherwise, the 1260-00C is a static configuration device.

d. Set the various switches, jumpers and non-volatile configurations as shown in Table 3-12.

Table 3-12, Non-Slot 0 Message-Based Device Operation Switch and Jumper Settings

Jumper/Switch	Position	Function
Switch S15	OFF	If S15 is ON, the 1260-00C sources CLK10 at the front panel EXT CLK connector.
Switch S22	OFF	VXI BTO disabled.
Switch S23	OFF	Bus arbiter and SYSCLK disabled. CLK10 receiving form backplane.
Switch S24	OFF	MODID pulled down.
Logical Address	See Chapter 5.	Logical address is not equal to 0. Set in non-volatile configuration or by using DIP switch SW1.
Non-Slot 0 Model Code	See Chapter 5.	Model code is set to the Non-Slot 0 value. Set in non-volatile configuration.
Servant Area Size	See Chapter 5.	Set appropriate Servant area size. Set in non-volatile configuration.

Non-Slot 0 Message-Based Device Operation

At start-up, a 1260-00C configured as a Non-Slot 0 Message-Based device performs its self-tests. It then waits until it receives its *Device Grant* and *Begin Normal Operation* Word Serial commands. The RM grants any logical addresses to the 1260-00C that reside within its Servant area. When it responds to the *Begin Normal Operation* command, the 1260-00C enters its normal mode of operation.

Front Panel LED

The 1260-00C indicates the progress of its self-test with the

Indications for Message-Based Device Operation

FAILED, TEST, and ON LINE LEDs. The LED indications are shown in Table 3-13. A successful system start-up sequences through the first five states. The point of failure is indicated for states in which the FAILED LED is lit for an extended period of time. The LED indications are identical for Non-Slot 0 Message-Based device and Slot 0 Message-Based device operation.

Table 3-13, Front Panel LED Indications For Message-Based Device Operation

Sequence	FAILED	TEST	ON LINE	State	Point of Failure
1	OFF	OFF	OFF	No power	
2	ON	OFF	OFF	In self-initialization	Failed before self-test
3	ON	ON	OFF	In self-test	Failed in self-test
4	OFF	ON	ON	Waiting for BNO	
5	OFF	OFF	ON	Online	
	ON	OFF	ON	Failed	Failed while online
	OFF	ON	OFF	In non-volatile configuration, or diagnostics mode	

Slot 0 Message-Based Device Configuration

To configure the 1260-00C for Slot 0 Message-Based operation, use the following procedure:

a. Enable the VXIbus Slot 0 functions.

b. Set the model code of the 1260-00C to be configured for Slot 0 operation using the non-volatile configuration mode.

c. Set the logical address to a non-zero value with an appropriate Servant area size. If the logical address is set to FFh, the 1260-00C will participate in dynamic configuration. Otherwise, the 1260-00C is a static configuration device.

d. Set the various switches, jumpers and non-volatile configurations as shown in Table 3-14.

Table 3-14, Slot 0 Message-Based Device Operation Switch and Jumper Settings

Jumper/Switch	Position	Function
Switches S9 and S15	See Table 3-15.	CLK10 routing options.
Switch S22	ON	VXI BTO enabled.
Switch S23	ON	Bus arbiter and SYSCLK enabled. CLK10 sourcing backplane.
Switch S24	ON	MODID pulled up
Logical Address	See Chapter 5.	Logical address is not equal to 0. Set in non-volatile configuration.
Slot 0 Model Code	See Chapter 5.	Model code is set to the Slot 0 value. Set in non-volatile configuration.
Servant Area Size	See Chapter 5.	Set appropriate Servant area size. Set in non-volatile configuration .

Table 3-15, CLK10 Routing Options

Switch S14	Switch S17	Function
OFF	OFF	CLK10 sourced from onboard clock.
OFF	ON	CLK10 and EXT CLK connector sourced from onboard clock.
ON	OFF	CLK10 sourced from an external clock via the EXT CLK connector.
ON	ON	Invalid. Do not use this setting.

Slot 0 Message-Based Device Operation

At start-up, a 1260-00C configured as a Slot 0 Message-Based device performs its self-tests. It then waits until it receives its *Device Grant* (if any) and *Begin Normal Operation* Word Serial commands. The RM grants any logical addresses to the 1260-00C that reside within its Servant area. When the 1260-00C responds to the *Begin Normal Operation* command, it enters the normal mode of operation.

After the 1260-00C Passed bit is set, the RM can manipulate or read the MODID lines by sending the Word Serial queries *Read MODID, Set Lower MODID,* or *Set Upper MODID* to the 1260-00C.

Chapter 4

LOCAL COMMAND SET

Introduction

This section contains descriptions of the 1260-00C local command set and queries which includes syntax, format and error handling information, and examples of the use of the commands and queries. The local command set supports the following types of operations:

a. System Configuration and Control

1) Help
2) General configuration
3) Resource Manager (RM) information extraction
4) Dynamic system configuration and reconfiguration
5) VXI-defined Common ASCII System Commands
6) GPIB address configuration
7) VXIbus interrupt handler configuration
8) IEEE-488.2 common commands

b. Instrument Development and Test

1) VXIbus access
2) VXI TTL/ECL trigger access
3) Word Serial communication

c. Code Instrument (CI) Use and Development

1) CI configuration

The 1260-00C command set consists of commands and queries. *Commands* cause the 1260-00C to take some action. A *query* may also cause the 1260-00C to take some action, but it also returns a response containing data or other information.

Command Set Access

Execute the local commands from the following ports:

RS-232

GPIB

VXI Word Serial Communication

Individual Code Instruments

All ports are active when the 1260-00C is in the Normal Operation substate, and operate independently of one another. The 1260-00C returns query responses only to the port originating the query. The 1260-00C also maintains a separate status state for each port. Use local commands to disable and re-enable each port's access to the local command set. The RS-232 port prompts you to enter a local command with the `1260-00C>` prompt.

Command Syntax

The local command set parser is syntactically compatible with the IEEE-488.2 standard. It will accept numeric parameters in the 488.2 binary, octal, decimal, or hexadecimal formats. 488.2 binary parameters are prefixed with **#b**. Octal parameters are prefixed with **#q**, and hexadecimal parameters are prefixed with **#h**. Table 4-1 lists the most common numeric parameter types. The ranges given in Table 4-1 apply unless otherwise specified.

Table 4-1. Valid Ranges For Common Numeric Command Parameters

Parameter	488.2 Decimal	488.2 Hexadecimal
`<logical address>`	0 to 254	#h0 to #hFE
`<GPIB address>`	0 to 30	#h0 to #h1E
`<handler>`	1 to 3	#h1 to #h3
`<level>`	0 to 7	#h0 to #h7
`<A16 address>`	0 to 65535	#h0 to #hFFFF
`<A24 address>`	2097152 to 14680062	#h200000 to #hE7FFFE
`<word value>`	0 to 65535	#h0 to #hFFFF
`<byte value>`	0 to 255	#h0 to #hFF
`<Boolean>`	0 to 1	#h0 or #h1

The logical value of a `<Boolean>` parameter is FALSE for the numeric value 0, and TRUE for the numeric value 1.

The first parameter is delimited from the command name by a space ( ). Additional parameters are delimited from one another by a comma (,). The command names are not case-sensitive.

In the command descriptions, parameters are enclosed in angle brackets (< >), and optional parameters are also enclosed in square brackets ([ ]). Do not enter the brackets as part of the command.

Multiple commands may be concatenated in a single command line if they are separated with semicolons (for example, `OBRAM?; DPRAM?`<CR>).

Command Line Termination

The serial port command line termination is a carriage return, shown in the subsequent function descriptions as <CR> (ASCII 0Dh). If the command contains a trailing linefeed, shown in the subsequent function descriptions as <LF> (ASCII 0Ah), it is ignored. The GPIB termination is EOI. Commands issued to the 1260-00C via VXI Word Serial Protocol are terminated by setting the END bit in the last *Byte Available* command. Responses are terminated by setting the END bit in response to the last *Byte Request* query.

Command and Query Responses

The local commands and queries have two response formats: *program mode* and *console mode.* Program mode responses have a terse data-only format intended for a control program to read and parse. Console responses are returned in the form of readable sentences, which are better suited for interactive command entry.

Each mode can be enabled or disabled independently, but one response mode must be enabled at all times. If both modes are simultaneously enabled, the program response is returned first, followed by the console response. The local commands used to control the response modes are `ProgMode` and `ConsMode`.

The response mode configuration is independent for each command source. Table 4-2 lists the default (start-up/reset) response mode configurations.

Table 4-2. Default Response Mode Configurations

Port	Response Mode
RS-232	Console mode enabled, program mode disabled
GPIB	Program mode enabled, console mode disabled
VXI Word Serial	Program mode enabled, console mode disabled
Individual Code Instruments	Program mode enabled, console mode disabled

Command Response Format

Commands do not have program mode responses. They do not return a response to a port configured for console mode response only, unless the 1260-00C detects an error condition.

Console mode command responses are self-explanatory, and are not described in this manual.

Query Response Format

Queries have both program and console mode responses. Program mode query responses are fixed-field formatted with commas delimiting the fields. For example, the list of logical addresses returned by the `Laddrs?` query is returned as groups of three characters (to allow the field to accommodate the valid range of 0 to 254) separated by commas. The values are right-justified and padded with the ASCII space character ( ) (20h). For example, Logical Address 45 would be returned as ( )`45`. Unless otherwise noted, all returned values are decimal.

Console mode query responses are self-explanatory, and are not described in this manual.

The query response line termination sequence, shown in the query descriptions as <CRLF>, indicates an ASCII 0Dh followed by 0Ah.

Error Reporting

Command syntax and execution errors are reported to the port where the command originated. If the program response mode is enabled, the 1260-00C returns an error message in the following format:

```
$ <error code><CRLF>
```

The distinguishing characteristic of a program mode error message is the leading dollar sign character ($). A list of error code descriptions is given in Appendix E, Error Codes.

If the console response mode is enabled, the 1260-00C returns an error message in the following format:

`<error description><CRLF>`

If both response modes are enabled, the program mode error message is returned first, followed by the console mode message.

The Help Query

The `Help?` query is a quick online reference to the syntax and functionality of the 1260-00C local command set.

Help?

Purpose: List syntax and descriptions of local command set.

Query

Syntax: `Help? [<type>[,<type>,....]]`

or

`Help [<type>[,<type>,....]]`

`<type>` is the category of command information requested, as follows:

`he` Help	`ci` Code Instruments
`al` All	`sa` GPIB address configuration
`gc` General configuration	`ih` Interrupt handler configuration
`dc` Dynamic configuration	`ba` VXIbus access
`rc` Dynamic reconfiguration	`ws` Word Serial communication
`rm` Resource Manager	`tr` TTL trigger access
`cc` Common commands	

The default type is All.

Response: The local command set is displayed in the following format:

`1260-00C Local Command Set<CRLF>`

`Command/Query Format` `Description<CRLF>`

`<Command Syntax>` `<Command description><CRLF>`

`<Command Syntax>` `<Command description><CRLF>`

`<Command Syntax>          <Command description>`<CRLF>

Example: List syntax and descriptions of general configuration and GPIB address commands.

```
Help?  gc,sa
```

General Configuration Commands and Queries

The general configuration commands and queries are described on the following pages.

- `CONF`
- `ConsoleEna`
- `ConsMode`
- `DIAG`
- `DPRAM?`
- `NVconf?`
- `OBRAM?`
- `ProgMode`
- `WordSerEna`

The `ConsMode` and `ProgMode` commands enable and disable the console and program response modes for the port originating the command.

The `ConsoleEna` and `WordSerEna` commands control access to the local command set from the RS-232 and VXI Word Serial ports.

The `NVconf?` query returns the contents of the onboard non-volatile memory. `CONF` reboots the 1260-00C and enters the non-volatile configuration editor.

`DIAG` reboots the 1260-00C and enters diagnostic mode.

The `OBRAM?` query can be used to determine the amount of 1260-00C installed RAM, and the `DPRAM?` query returns the amount of the installed RAM that is shared with VXI A24 space.

CONF

Purpose: Reboot into non-volatile configuration mode.

Command

Syntax: `CONF`

Example: Reboot into non-volatile configuration mode.

`CONF`

ConsoleEna

Purpose: Enable or disable the RS-232 port as the console. When the RS-232 port is disabled as the console, a CI can take control of the serial port.

Command Syntax: `ConsoleEna <Boolean>`

Action: If `<Boolean>` is TRUE, `ConsoleEna` sets the RS-232 port to be a local command set input.

If `<Boolean>` is FALSE, `ConsoleEna` disables the RS-232 port connection to the local command set. Note that once the console has been disabled, it must be re-enabled from a command source other than the RS-232 port (such as the GPIB port).

Examples: **Disable console**.

`ConsoleEna 0`

Enable console.

`ConsoleEna 1`

ConsMode

Purpose: Enable or disable the console data mode.

Command Syntax: `ConsMode <Boolean>`

Action: If `<Boolean>` is TRUE, `ConsMode` enables console format responses for the command source issuing the command.

If `<Boolean>` is FALSE, `ConsMode` disables console format responses for the command source issuing the command.

The console response mode applies only to the response path connected to the `ConsMode` command source. For example, disabling the console response mode from the GPIB port does not affect the response mode on the serial port.

Example: Disable console format responses.

`ConsMode 0`

Enable console format responses.

`ConsMode 1`

DIAG

Purpose: Reboot into diagnostics mode.

Command Syntax: `DIAG`

Example: Reboot into diagnostics mode.

`DIAG`

Dpram?

Purpose: Get the A24/A32 starting address and the size of the 1260-00C VXI shared RAM.

Query Syntax: `DPram?`

Response: Program response:

`<A24/A32 starting address>, <shared RAM size>`<CRLF>

Console response:

`This 1260-00C has <shared RAM size>K bytes shared with [A24, A32] Address <A24/A32 hex starting address>`<CRLF>

where `<A24/A32 starting address>` is the shared RAM base address in decimal integer format.

`<shared RAM size>` is in kilobytes.

`<A24/A32 hex starting address>` is in C language hexadecimal format.

Nvconf?

Purpose: Display the contents of the non-volatile (NV) configuration parameter memory.

Query

Syntax: `NVconf?`

Response: The contents of the onboard EEPROM are displayed in the following format:

```
=============== Non-Volatile Configuration Information ===============
Logical Address:     0x00          Device Type:      Message Based
Manufacturer Id:     0xFF6         Model Code:       0x0FF (Slot 0)
Slave Addr Spc :     A24           Protocol Reg:     0x0FF0
RESET Config:        PBtoLocalRESET PBtoSYSRESET SYSRESETtoLocalRESET

Serial Number:       0x00010003    User pROBE Pars:  0x000000 (None)
Region 1 Size:       0x070000      Number Procs:     0x20
Number Exchgs:       0x20          Number Msgs:      0x180
Console:             Enabled
VXI Interrupt Level To Handler Logical Address (0xFF = free to assign):
          1:0xFF, 2:0xFF, 3:0xFF, 4:0xFF, 5:0xFF, 6:0xFF, 7:0xFF
A24 Assign Base:     0x200000           A32 Assign Base:  0x20000000
DC Starting LA:      0x01,BNO=YES       For FAILED Dev:   DO set Reset Bit
Servant Area:        0x00               GPIB Primary:     0x01
GPIB Addr Assgn:     Default            GPIB Flags:       MultSecond NAT4882 DMA
GPIB Addr Avoid:     0x00000000

CI Block Base:       0x080000           CI Num Blocks:    0x00
              ------ Resident Code Instrument Locations ------
              # 0:   0     # 1:   0     # 2:   0
              # 3:   0     # 4:   0     # 5:   0
              # 6:   0     # 7:   0     # 8:   0
              # 9:   0     # a:   0     # b:   0
            ------ CI Non-Volatile User Configuration Variables ------
            # 0:  0   # 1:  0   # 2:  0   # 3:  0
            # 4:  0   # 5:  0   # 6:  0   # 7:  0
            # 8:  0   # 9:  0   #10:  0   #11:  0
            #12:  0   #13:  0   #14:  0   #15:  0
            #16:  0   #17:  0   #18:  0   #19:  0
            #20:  0   #21:  0   #22:  0   #23:  0
            #24:  0   #25:  0   #26:  0   #27:  0
            #28:  0   #29:  0   #30:  0   #31:  0
```

Obram?

Purpose: Get the amount of RAM installed onboard the 1260-00C.

Query Syntax: `OBram?`

Response: Program response:

<memsize><CRLF>

where <memsize> is the amount of installed RAM, in kilobytes.

Console response:

This 1260-00C has <expression> of RAM installed onboard.<CRLF>

where <expression> is the amount of installed RAM.

ProgMode

Purpose: Enable or disable the program data mode.

Command Syntax: `ProgMode <Boolean>`

Action: If <Boolean> is TRUE, ProgMode enables program format responses for the command source issuing the command.

If <Boolean> is FALSE, ProgMode disables program format responses for the command source issuing the command.

The program response mode applies only to the response path connected to the ProgMode command source. For example, disabling the program response mode from the GPIB port does not affect the response mode on the serial port.

Examples: Disable program format responses.

```
ProgMode 0
```

Enable program format responses.

```
ProgMode 1
```

WordSerEna

Purpose: Assign control of the 1260-00C physical Word Serial registers to an onboard logical address (1260-00C command interpreter or code instrument).

Command Syntax: `WordSerEna <logical address>`

Action: Control of the physical Word Serial registers is passed to `<logical address>`. `<logical address>` must be the logical address of the 1260-00C or an onboard code instrument.

The default control of the physical registers is given to the 1260-00C local command set parser.

Examples: Pass control of the physical registers to code instrument at Logical Address 5.

```
WordSerEna 5
```

Pass control of the physical registers back to 1260-00C local command parser at Logical Address 0.

```
WordSerEna 0
```

RM Information Queries

The RM information queries are described on the following pages.

- `A24MemMap?`
- `A32MemMap?`
- `Cmdr?`
- `CmdrTable?`
- `Laddrs?`
- `NumLaddrs?`
- `RmEntry?`
- `Srvnts?`
- `StatusState?`

NOTE

The system information commands (`NumLaddrs?`, `Laddrs?`, `CmdrTable?`, `A24MemMap?`, and

A32MemMap?) return information about the known system. If the 1260-00C is the system RM, it can access information about the entire system. If it is not the RM, it has information only about itself and its immediate Servants.

The `Numladdrs?` query is used to find out how many devices are in the system. The number of devices could be used by a control program to determine the allocation size for an array that is to hold the logical addresses of each device.

The `Laddrs?` query returns a list of logical addresses for devices in the system.

The `RmEntry?, Srvnts?, Cmdr?, and StatusState?` queries return RM information for a particular device.

The `CmdrTable?` query returns the system hierarchy table.

The `A24MemMap?` and `A32MemMap?` queries return the A24 and A32 memory configuration lists.

A24MemMap?

Purpose: Get the A24 address space allocation for the system.

Query Syntax: `A24MemMap?`

Response: Program response:

`<la1>,<A24 memory base>,<A24 memory size><CRLF>`

`<la2>,<A24 memory base>,<A24 memory size><CRLF>`

•

•

`<laN>,<A24 memory base>,<A24 memory size><CRLF>`

where `<la1>` through `<laN>` are logical addresses containing A24 address space.

Console response:

`A24 Memory Map is as follows:<CRLF>`

`Logical Address <la1> has <A24 memory`

`size>K`

`(<A24 memory size> bytes) at A24 Address<A24 memory base><CRLF>`

•

•

`Logical Address <laN> has <A24 memory size>K`

`(<A24 memory size> bytes) at A24 Address<A24 memory base><CRLF>`

Example: Get A24 address map for the system.

`A24MemMap?`

A32MemMap?

Purpose: Get the A32 address space allocation for the system.

Query Syntax: `A32MemMap?`

Response: Program response:

`<la1>,<A32 memory base>,<A32 memory size><CRLF>`

`<la2>,<A32 memory base>,<A32 memory size><CRLF>`

•

•

`<laN>,<A32 memory base>,<A32 memory size><CRLF>`

where `<la1>` through `<laN>` are logical addresses containing A32 address space.

Console response:

`A32 Memory Map is as follows:<CRLF>`

`Logical Address <la1> has <A32 memory size>K`

`(<A32 memory size> bytes) at A32 Address<A32 memory base><CRLF>`

•

- `Logical Address <laN> has <A32 memory size>K`

  `(<A32 memory size> bytes) at A32 Address<A32 memory base><CRLF>`

Example: Get A32 address map for the system.

`A32MemMap?`

Cmdr?

Purpose: Get the logical address of a device's Commander.

Query Syntax: `Cmdr? <logical address>`

where `<logical address>` is the logical address of the device.

Response: Program response:

`<Commander's logical address><CRLF>`

Console response:

`The Commander of Logical Address <logical address> is Logical Address <Commander's logical address><CRLF>`

Example: Get the Commander's logical address for Logical Address 15.

`Cmdr? 15`

CmdrTable?

Purpose: Get the known system hierarchy table.

Query Syntax: `CmdrTable?`

Response: Program response:

`<cla0>,<cla1>,<cla2>,<cla3>,<cla4>, . . .,<cla254><CRLF>`

where `<claN>` is either the Commander's logical address for Logical Address `N`, or `0` for top-level Commanders and unused logical addresses. Note that no value is returned for Logical Address 255.

Console response:

`Known Hierarchy is as follows:<CRLF>`

`Logical address <la1> has Servants: <sa1,1>,...,<sa1,M>  <comment><CRLF>`

`Logical address <la2> has Servants: <sa2,1> ,...,<sa2,M> <comment><CRLF>`

•
•

`Logical address <laN> has Servants: <saN,1> ,...,<saN,M> <comment><CRLF>`

where `<laX>` is a valid logical address with Servant addresses

`<saX,1>` through `<sa1,M>`.

The `<comment>` field indicates any relevant information about the status and/or capabilities of the device at Logical Address `<laX>`.

Laddrs?

Purpose: Get a list of the known logical addresses.

Query Syntax: `Laddrs?`

Response: Program response:

`<la1>,<la2>,..., <laN><CRLF>`

where `<la1>` through `<laN>` are the known logical addresses.

Console response:

`Known logical addresses are <la1>,<la2>,..., <laN><CRLF>`

CI logical addresses are terminated with an asterisk (*) in the console mode response.

NumLaddrs?

Purpose: Get the number of known logical addresses.

Query Syntax: NumLaddrs?

Response: Program response:

<num las><CRLF>

where <num las> is the number of known logical addresses.

Console response:

There are <num las> known Logical Addresses<CRLF>

RmEntry?

Purpose: Return RM information about a device or all devices. RmEntry? does not return the Servant list.

Query Syntax: RmEntry? [<logical address>]

(If <logical address> is omitted, RmEntry? returns the RM information for all devices.)

Response: Program response:

<la>,<cla>,<sa>,<slot>,<devclass>,<sub class>,<manID>,

<modelcode>, <memspace>,<membase>,<memsize>,<state> ,<line status><CRLF>

Console response:

Resource manager entry for Logical Address <logical address>: <CRLF>

<CRLF>

```
Commander's Logical Address      :<cla><CRLF>
GPIB Address                     :<addr><CRLF>
Slot                             :<slot><CRLF>
Device class                     :<devclass> (class)<CRLF>
Extended Sub Class               :<subclass><CRLF>
Manufacturer's ID                :<manID>(manuf's name)<CRLF>
 Model code                      :<modelcode><CRLF>
 Memory space                    :<memspace>(memory space)<CRLF>
Memory Base                      :<membase><CRLF>
Memory Size                      :<memsize>K(<memsize> bytes)<CRLF>
```

```
Status State        :<state> (state)<CRLF>
Forced Offline?           :<line status> (yes/no)<CRLF>
```

The mnemonics have the following meanings:

`la`	Device's logical address
`cla`	Commander's logical address
`addr`	Device's GPIB address (255 if not assigned GPIB address)
`slot`	Slot number (255 if unknown, such as if the device does not have MODID capability)
`devclass`	Device class; the following values may be used: 0 = Memory Class 1 = Extended Class 2 = Message-Based 3 = Register-Based
`subclass`	Extended class device's subclass
`manID`	Manufacturer's ID number
`modelcode`	Device's manufacturer-assigned model code
`memspace`	Memory space requirement: 0 = A16 only 1 = A16/A24 2 = A16/A32
`membase`	Memory base address
`memsize`	Memory size in bytes
`state`	Status state: 0 = Failed and not Ready 1 = Passed and not Ready 2 = Failed and Ready 3 = Passed and Ready
`line status`	Online/offline status:

0 = online
1 = forced offline

The program mode response format is the same for all devices. However, the console mode response returns only the lines that are relevant. For example, the memory base address and memory size lines are not returned for A16-only memory space devices.

Example: Get RM information for a device at Logical Address 78.

`RmEntry? 78`

Srvnts?

Purpose: Get a list of a device's Servants.

Query Syntax: `Srvnts?  <logical address>`

`<logical address>` is the device's logical address.

Response: Program response:

`<sla1>,<sla2>,...,<slaN><CRLF>`

where `<sla1>` through `<slaN>` are the Servant device logical addresses.

Console response:

`Logical Address <logical address> has Servants:`

`<sla1>, <sla2>,..., <slaN> <comment><CRLF>`

if the device has Servants, or

`Logical Address <logical address> has Servants:`

`none <comment><CRLF>`

if the device has no Servants.

The `<comment>` field indicates any relevant information about the status and/or capabilities of the device.

Example: Get a list of Servants for device at Logical Address 15.

`Srvnts? 15`

StatusState?

Purpose: Get a device's current self-test status.

Query Syntax: `StatusState? <logical address>`

`<logical address>` is the logical address for the device.

Response: Program response:

`<val><CRLF>`

The value of `<val>` is equivalent to the value of the field in the device's status register containing the Ready and Passed bits. `<val>` can be interpreted as follows:

0	The device is Failed and not Ready.
1	The device is Passed and not Ready.
2	The device is Failed and Ready.
3	The device is Passed and Ready.

Console response:

```
Device at Logical Address <logical
address> is FAILED and not Ready<CRLF>
```

or

```
Device at Logical Address <logical
address> is PASSED and not Ready<CRLF>
```

or

```
Device at Logical Address <logical
address> is FAILED and Ready<CRLF>
```

or

```
Device at Logical Address <logical
address> is PASSED and Ready<CRLF>
```

Example: Get self-test status for device at Logical Address 48.

`StatusState? 48`

Dynamic Configuration Commands and Queries

The dynamic configuration (DC) commands and queries are described in the following paragraphs.

- `DCBNOSend`
- `DCGrantDev`
- `DCSystem?`

The DC commands are used to configure the VXI system when all of these conditions are present:

- The 1260-00C is the RM.
- At least one DC device is present in the system.
- The non-volatile configuration setup specifies **not** to send *Begin Normal Operation* (user-specified hierarchy).
- The system is still in the start-up Configure substate (*DCBNOSend* has not been sent).

The `DCSystem?` query response indicates whether the system contains a DC device. If the system is found to be a DC system, the `DCGrantDev` command is used to configure the Commander/Servant hierarchy. The `DCBNOSend` command is used to end the DC phase and to cause the system to enter normal operation.

DCBNOSend

Purpose: Cause a DC system to exit the Configure substate and enter the Normal Operation substate.

Command Syntax: `DCBNOSend`

Action: Send the *Begin Normal Operation* command to all top-level Commanders.

DCGrantDev

Purpose: Grant a device to a Message-Based Commander in a DC system. `DCGrantDev` can be used only to configure the initial Commander/Servant hierarchy of a DC system, and before `DCBNOSend` is used to cause the system to enter the Normal Operation substate.

Command Syntax: `DCGrantDev <Commander's logical address>, <Servant's logical address>`

Action: `DCGrantDev` sends the *Device Grant* command to the Commander at `<Commander's logical address>`, granting it the device at `<Servant's logical address>`.

Example: Grant Servant at Logical Address 7 to Commander at Logical Address 5.

```
DCGrantDev 5,7
```

DCSystem?

Purpose: Determine if the system is a DC system. A system is DC if it has at least one DC device.

Query Syntax: `DCSystem?`

Response: Program response:

`1` <CRLF>

if it is a DC system, or

`0` <CRLF>

if it is not a DC system, or if it is no longer dynamically configurable because the *Begin Normal Operation* command has already been sent to the top-level Commanders through the `DCBNOSend` local command.

Console response:

```
This IS a Dynamic Configured
system.<CRLF>
```

if it is a DC system, or

```
This is NOT a Dynamic Configured
system.<CRLF>
```

if it is not a DC system.

Dynamic Reconfiguration Queries

The dynamic reconfiguration queries are described on the following pages.

- `Broadcast?`
- `GrantDev?`
- `RelSrvnt?`

The dynamic reconfiguration commands are used to reconfigure the 1260-00C's Servant subtree after the system has entered the Normal Operation substate. If the 1260-00C is RM, these commands can be used to reconfigure the entire system.

The `Broadcast?` query can be used to make the system or subtree enter the Configure substate by broadcasting the *End Normal Operation* Word Serial query, or the Clear Word Serial command followed by the *Abort Normal Operation* Word Serial query.

The `RelSrvnt?` and `GrantDev?` queries can then be used to restructure the Commander/Servant hierarchy. You could perform dynamic reconfiguration directly by using the `WSCmd` and `WSCmd?` local commands, but the 1260-00C's RM table would not be updated. By using the `RelSrvnt?` and `GrantDev?` queries to reconfigure the system, you ensure the 1260-00C's system hierarchy, and GPIB address link records do not become corrupted.

You can return the system or subtree to the Normal Operation substate by using the `Broadcast?` query to broadcast the *Identify Commander* and *Begin Normal Operation* Word Serial commands.

Broadcast?

Purpose: Broadcast dynamic reconfiguration initialization or termination Word Serial commands to the 1260-00C's Message-Based Servants or to all top-level Commanders in the system.

Query Syntax: `Broadcast? <Boolean>,<ws cmd>`

If `<Boolean>` is 1, the 1260-00C broadcasts `<ws cmd>` to all top-level Commanders. If `<Boolean>`

is 0, it broadcasts <ws

cmd> to its Message-Based Servants. Note that the 1260-00C should only broadcast to top-level Commanders when it is RM.

The `Broadcast?` query can fail due to inability to complete a Word Serial operation, or because an invalid code was returned from a device in response to ANO or ENO.

`<ws cmd>` is a mnemonic as follows:

<ws cmd>	Word Serial Command Name	Type
ANO	*Abort Normal Operation*	Query
BNO	*Begin Normal Operation*	Query
CLR	*Clear*	Command
ENO	*End Normal Operation*	Query
IDN	*Identify Commander*	Command

Response: Program response:

<CRLF>

if the command was successful, or

`<la>,<cmd val>,<ws response>,<ws error code>`

if the command failed.

Console response:

```
Done broadcasting Word Serial command:
<Word Serial command name>.
```

if the command was successful, or

```
Logical address <la> returned <ws response>
from ENO (Unable to halt)
```

or

```
Logical address <la> returned <ws response>
from ANO (Invalid response)
```

or

`Error sending Logical Address <la> Word Serial command <hex cmd val><CRLF><space><space><ws error><CRLF>`

if the command failed.

`<la>` is the logical address of the device to which the broadcast failed.

`<cmd val>` is the value of the Word Serial command, in decimal. `<hex cmd val>` is the value in hexadecimal.

For Word Serial queries, `<ws response>` is the Word Serial response of the device at Logical Address `<la>`. For Word Serial commands `<ws response>` is 0.

`<Word Serial command name>` is the name of the command name as shown in the previous table.

`<ws error code>` is a decimal value that can be interpreted by converting it to a binary bit pattern. A value of 1 in the bit positions shown in the following table indicates that an error occurred during the attempt to broadcast the Word Serial command:

Bit	Word Serial Error
0	Word Serial Command completed successfully (no Word Serial error)
1	Timeout waiting to send Word Serial command to device at <la>
2	Timeout waiting for Word Serial response from device at <la>
3	Device at <la> did not recognize the command
6	Multiple query error
10	Read Protocol error not supported
13	Read Ready (RR) violation
14	Write Ready (WR) violation

None of the other bits has significance in this context.

`<ws error>` is a string explaining the Word Serial error as shown in the previous table.

Example: Broadcast the *Identify Commander* Word Serial command to all top-level Commanders.

`broadcast? 1,IDN`

GrantDev?

Purpose: Grant a Servant to a Commander.

Query Syntax: `GrantDev? <Commander's logical address>, <Servant's logical address>`

Action: Grants the device at `<Servant's logical address>` to device at `<Commander's logical address>`.

The 1260-00C must own the device at `<Servant's logical address>`. The 1260-00C can get ownership of any device with the `RelSrvnt?` command.

Note that before the `GrantDev?` query is used, the Word Serial *End Normal Operation* query, or a *Clear* command followed by the *Abort Normal Operation* query should have been broadcast with the `Broadcast?` query.

Response: Program response:

0<CRLF>

indicates the command was successful.

Console response:

`Logical Address <Commander's logical address> granted device at Logical Address <Servant's logical address>.`

Example: Grant Device 16 to Commander at Logical Address 8.

`Grantdev? 8,16`

RelSrvnt?

Purpose: Recover a Servant from a Commander.

Query Syntax: `RelSrvnt? <Commander's logical address>, <Servant's logical address>`

Action: Commands device at `<Commander's logical address>` to release ownership of the device at `<Servant's logical address>`. The 1260-00C assumes ownership of the device.

Response: Program response:

254<CRLF>

if the Commander released the Servant. Any other response indicates that an error occurred.

Console response:

```
Logical Address <Commander's logical
address> released device at Logical
Address <Servant's logical address>.
```

Example: Recover Servant at Logical Address 16 from Commander at Logical Address 8.

```
Relsrvnt? 8,16
```

VXI-Defined Common ASCII System Commands

The VXI-defined Common ASCII System Commands and Queries are described on the following pages.

- `DCON?`
- `DINF?`
- `DLAD?`
- `DNUM?`
- `DRES?`
- `RREG?`
- `WREG`

These commands and queries are used to retrieve device information/configuration, perform a soft reset, and peek/poke a device's registers.

The `DNUM?` query is used to find out how many devices are in the system. The `DLAD?` query returns a list of logical addresses for devices in the system.

The `DINF?` query returns static information about a device. The

`DCON?` query returns configuration information about a device.

The `DRES?` query is used to perform a soft-reset sequence on a device.

The `RREG?` query and `WREG` command are used to peek (read from) and poke (write to) registers on a VXI device.

DCON?

Purpose: Return system configuration information about a device or all devices.

Query Syntax: `DCON? [<logical address>]`

(If `<logical address>` is omitted, `DCON?` returns the configuration information for all devices.)

Response: Program response:

```
<la1>,<cla>,<IHANS>,<INTS>,<status>,<sstate>,<com><CRLF>
```

Console response:

```
Device configuration at Logical Address <la>:<CRLF>

<CRLF>

Commander's Logical Address:<cla><CRLF>
Interrupt Handlers         :<IHANS><CRLF>
Interrupters               :<INTS><CRLF>
Passed/Failed/Ready        :<status><CRLF>
Device Substate            :<sstate><CRLF>
Manufacturer Specific Comment     :<com><CRLF>
```

The mnemonics have the following meanings:

`la`	Device's logical address
`cla`	Commander's logical address
`IHANS`	Interrupt handler levels used by this device where `IHANS` is a 7-digit binary representing the seven VXI interrupt levels and a 1 in each position, meaning Interrupt Handler present
`INTS`	Interrupter levels used by this device

where `INTS` is a 7-digit binary representing the seven VXI interrupt levels and a 1 in each position, meaning Interrupter present

`status` Status state of the device:

PASS
FAIL
IFAIL
READY

`sstate` Substate of the device

NOP
CONF
NONE

`com` Not used; always returns ""

Example: Get device configuration information for Logical Address 6.

DCON? 6

DINF?

Purpose: Return static system information about a device.

Query Syntax: `DINF? [<logical address>]`

(If `<logical address>` is omitted, `DINF?` returns static information for all devices.)

Response: Program response:

`<la1>,<manID>,<modelcode>,<devclass>,<memspace>,`

`<membase>,<memsize>,<slot>,<slot0>,<ext>,<attr>,`

`<com>`<CRLF>

Console response:

`Device configuration at Logical Address <la>`:<CRLF>

<CRLF>

```
Manufacturer ID Number         :<manid>(manufacturer
                                 name)<CRLF>
Model Code                     :<modelcode><CRLF>
Device Class                   :<devclass><CRLF>
A16/A24/A32 Memory Space       :<memspace><CRLF>
A16/A24/A32 Memory Base        :<membase><CRLF>
A16/A24/A32 Memory Size        :<memsize>1<CRLF>
Slot                           :<slot><CRLF>
Slot 0 Logical Address         :<slot0><CRLF>
Extended Subclass              :<ext><CRLF>
Attribute                      :<attr><CRLF>
Manufacturer Specific Comment  :<com><CRLF>
```

The mnemonics have the following meanings:

`la`	Device's logical address
`manid`	Manufacturer's ID number
`devclass`	Device class; the following values may be used: REG = Register-Based device MSG = Message-Based device EXT = Extended-Class device MEM = Memory-Based device
`memspace`	Memory space requirement A16 A16/A24 A16/A32
`membase`	Memory-based address for A16, A24, A32 "HHHH, HHHHHH, HHHHHHHH"
`memsize`	Memory sizes for A16, A24, A32 "HHHH, HHHHHH, HHHHHHHH"
`slot`	Slot number (-1 if unknown)
`slot 0`	Slot 0 Logical Address (-1 if unknown)
`ext`	Extended device's subclass
`attr`	Memory device's attributes
`com`	Not used, always ""

DLAD?

Purpose: Get a list of the known logical addresses.

Query Syntax: `DLAD?`

Response: Program response:

`<la1>,<la2>,..., <laN><CRLF>`

where `<la1>` through `<laN>` are the known logical addresses.

Console response:

```
Known logical addresses are
<la1>,<la2>,..., <laN><CRLF>
```

CI logical addresses are terminated with an asterisk (*) in the console mode response.

Example: Get a list of the known logical addresses.

`DLAD?`

DNUM?

Purpose: Get the number of the known logical addresses.

Query Syntax: `DNUM?`

Response: Program response:

`<num las><CRLF>`

where `<num las>` is the number of known logical addresses.

Console response:

```
There are <num las> known Logical
Addresses.<CRLF>
```

Example: Get the number of the known logical addresses.

`DNUM?`

DRES?

Purpose: Perform a soft-reset sequence on a device.

Query Syntax: `DRES? <logical address> [, <sysfail flag>]`

NOTE

If the device stays failed for five seconds after the soft-reset sequence, <sysfail flag> determines whether or not the device is kept sysfail-inhibited.

Response: Program response:

`<status>`<CRLF>

Console response:

`Logical Address <logical address> is <status>.  SYSFAIL Inhibit is <state>.`<CRLF>

where `<status>` is one of the following:

PASS
FAIL
IFAIL
READY

and `<state>` is one of the following:

ON
OFF

Example: Soft-reset device at Logical Address 3.

`DRES? 3`

RREG?

Purpose: Read a 16-bit VXI register from a device.

Query Syntax: `RREG? <logical address>, <reg offset>`

where `<logical address>` is the device to read from and `<reg offset>` is the number of bytes to offset from the base of the VXI registers for that device.

Response: Program response:

`<hex word value><CRLF>`

Console response:

```
Value 0x<hex word value> (<word
value>) read from Logical Address
<logical address>, Register offset
0x<reg offset><CRLF>
```

Example: Read Device Type register from Logical Address 12.

```
RREG? 12,2
```

WREG

Purpose: Write a 16-bit VXI register on a particular device.

Query Syntax:

```
WREG <logical address>, <reg offset>,
<value>
```

where `<logical address>` is the device to write, `<reg offset>` is the register offset to write to, and `<value>` is the 16-bit value to write.

Action: Write `<value>` to `<logical address>`, register offset `<reg offset>`.

Example: Write the Data Low register for Logical Address 4 with the value 65535.

```
WREG 4,14,65535
```

GPIB Address Configuration Commands and Queries

The GPIB address configuration commands are described on the following pages.

- `LaSaddr`
- `LaSaddr?`
- `Primary?`
- `SaddrLa?`
- `Saddrs?`
- `SaDisCon`

These commands and queries configure and report the relationships between VXI logical addresses and GPIB addresses.

Determine the 1260-00C's primary address when used for multiple GPIB secondary addressing by using the Primary? query from the serial port. Determine the relationships between GPIB addresses and VXI logical addresses by using the Saddrs? query followed by SaddrLa? queries, or by using the RM information query Laddrs? followed by LaSaddr? queries.

Assign GPIB address links to Message-Based Servants of the 1260-00C with the LaSaddr command. The SaDisCon command deletes all GPIB address links except the link to the 1260-00C local commands.

NOTE

The letters SA or SADDR in this chapter originally stood for GPIB *Secondary Address.* The 1260-00C can be configured to handle multiple primary addresses as well. The terminology has been left the same to maintain backwards compatibility.

LaSaddr

Purpose: Attach or detach a GPIB address to a logical address.

Command Syntax: LaSaddr <logical address>, <GPIB address>

Action: If <GPIB address> is not equal to 255, attach <GPIB address> to <logical address>.

If <GPIB address> is equal to 255, release <GPIB address> currently attached to <logical address>.

Attaching a GPIB address to a logical address that already has a GPIB address will cause the first GPIB address to be replaced by the new GPIB address.

Attempting to release or change a GPIB address will result in a Delete I/O Link error if any of the following conditions are true:

- The GPIB address does not exist.

- The GPIB address is addressed to talk or listen; unable to delete.
- There is still data in the GPIB address input or output queue.

Examples: Attach GPIB Address 6 to Logical Address 4.

```
LaSaddr 4,6
```

Release GPIB address currently attached to Logical Address 8.

```
LaSaddr 8,255
```

LaSaddr?

Purpose: Get the GPIB address attached to a logical address.

Query Syntax:

```
LaSaddr? <logical address>
```

Response: Program response:

```
<GPIB address><CRLF>
```

where `<GPIB address>` is the GPIB address attached to the logical address. A value of 255 indicates that no GPIB address is attached to the logical address.

Console response:

```
Logical Address <logical address> is
attached to GPIB Address <GPIB
address><CRLF>
```

for logical addresses with attached GPIB addresses, or

```
Logical Address <logical address> is
NOT attached to a GPIB <type>
Address<CRLF>
```

for logical addresses without attached GPIB addresses.

Example: Get the GPIB address attached to Logical Address 9.

`LaSaddr? 9`

Primary?

Purpose: Get a GPIB primary address.

Query Syntax: `Primary?`

Response: Program response:

`<primary address><CRLF>`

where `<primary address>` is the GPIB primary address of 1260-00C.

Console response:

`The GPIB primary address of the 1260-00C is <primary address><CRLF>`

SaddrLa?

Purpose: Get the logical address that a GPIB address is attached to.

Query Syntax: `SaddrLa? <GPIB address>`

Response: Program response:

`<logical address><CRLF>`

where `<logical address>` is the logical address that the GPIB address is attached to. A value of 255 indicates that the GPIB address is not attached to a logical address.

Console response:

`GPIB <type> Address <GPIB address> is attached to Logical Address <logical address><CRLF>`

for a GPIB address that is attached to a logical address, or

`GPIB <type> Address <GPIB address> is NOT attached to a Logical Address<CRLF>`

for a GPIB address that is not attached to any logical address.

Example: Get the logical address attached to GPIB Address 9.

`SaddrLa? 9`

Saddrs?

Purpose: Get a list of used GPIB addresses.

Query Syntax: `Saddrs?`

Response: Program response:

`<sa1>,<sa2>, . . .,<saN><CRLF>`

where `<sa1>` through `<saN>` are the GPIB addresses currently attached to logical addresses.

Console response:

```
Current GPIB Addresses used:

Address <sa1>:  connected to Logical
Address <la1>.

Address <sa2>:  connected to Logical
Address <la2>.

     •

     •

<type> Address <saN>:  connected to
Logical Address <laN><CRLF>
```

SaDisCom

Purpose: Detach **all** GPIB address links except the GPIB address link to the 1260-00C command set.

Command Syntax: `SaDisCon`

Action: Detaches all GPIB address links from Servants of the 1260-00C.

VXI Interrupt Handler Configuration Commands and Queries

The interrupt handler configuration commands and queries are described on the following pages.

- `AllHandlers?`
- `AssgnHndlr`
- `HandlerLine?`
- `RdHandlers?`

The interrupt handler commands and queries configure and report the relationships between the 1260-00C interrupt handlers and VXIbus interrupt levels.

The 1260-00C has three programmable interrupter handlers. An application program can confirm this with the `RdHandlers?` query. The `AllHandlers?` and `Handerline?` queries return the current VXI interrupt level assignments for the handlers. The `AssgnHndlr` command can be used to change the level assignments.

AllHandlers?

Purpose: Get the VXIbus interrupt level assigned to all 1260-00C interrupt handlers.

Query Syntax: `AllHandlers?`

Response: Program response:

```
<level1>,<level2>,<level3><CRLF>
```

where `<level1>` is the interrupt level assigned to Handler 1, `<level2>` is the interrupt level assigned to Handler 2, and `<level3>` is the interrupt level assigned to Handler 3.

If `<levelN>` equals 0, Interrupt Handler `<handlerN>` is not assigned to an interrupt level.

Console response:

```
VXI Interrupt Handler 1 assigned to
interrupt level<level1><CRLF>

VXI Interrupt Handler 2 assigned to
interrupt level<level2><CRLF>

VXI Interrupt Handler 3 assigned to
```

`interrupt level<level3><CRLF>`

if all handlers are assigned to levels, or

`VXI Interrupt Handler <handler> NOT assigned to any interrupt level.<CRLF>`

if `<handlerN>` is not assigned to a level.

Example: Get the interrupt level assigned to all interrupt handlers.

`AllHandlers?`

AssgnHndlr

Purpose: Assign a VXIbus interrupt level to a 1260-00C interrupt handler.

Command Syntax: `AssgnHndlr <handler>, <level>`

where `<handler>` is a numeric integer quantity in the range 1 to 3, and `<level>` is a numeric integer quantity in the range 0 to 7.

Action: If `<level>` is in the range 1 to 7, VXIbus Interrupt Line `<level>` is assigned to Interrupt Handler `<handler>`.

If `<level>` is 0, the current VXIbus interrupt line held by Interrupt Handler `<handler>` is released.

Examples: Assign the Interrupt Level 6 to the 1260-00C Interrupt Handler 2.

`AssgnHndlr 2,6`

Release the interrupt level currently held by the 1260-00C Interrupt Handler 1.

`AssgnHndlr 1,0`

HandlerLine?

Purpose: Get the level assigned to a 1260-00C interrupt handler.

Query Syntax: `HandlerLine? <handler>`

Response: Program response:

`<level><CRLF>`

Console response:

```
VXI Interrupt Handler <handler>
assigned to interrupt level
<level><CRLF>
```

`<level>` is the interrupt level assigned to handler `<handler>`. If `<level>` equals 0, Interrupt Handler `<handler>` is not assigned an interrupt level.

Example: Get the interrupt level assigned to Interrupt Handler 3.

```
HandlerLine? 3
```

RdHandlers?

Purpose: Get the number of assignable 1260-00C interrupt handlers.

Query Syntax: `RdHandlers?`

Response: Program response:

3 <CRLF>

Console response:

```
This 1260-00C has 3 configurable VXI
interrupt handlers.<CRLF>
```

Example: Get the number of assignable 1260-00C interrupt handlers.

```
RdHandlers?
```

IEEE-488.2 Common Commands and Queries

The IEEE-488.2 commands and queries are as follows:

- `*CLS`
- `*ESE`
- `*ESE?`
- `*ESR?`

- `*IDN?`
- `*OPC`
- `*OPC?`
- `*RST`
- `*SRE`
- `*SRE?`
- `*STB?`
- `*TRG`
- `*TST?`
- `*WAI`

These commands conform to the minimal 488.2 requirements. Many of these 488.2 commands have limited meaning in the VXI environment, but are included for compatibility. The 1260-00C has no reason to interrupt as a 488.2 instrument. It is only a parser. All other functions of the 1260-00C are considered to be interface functions for other 488.2 VXI devices. It is the responsibility of each VXI device connected via the GPIB to the 1260-00C to implement 488.2 protocols if individual device 488.2 compatibility is required.

*CLS

488.2
Intent: Clear the device status data structures, and force it to the Operation Complete Query Idle state.

Command Syntax: `*CLS`

Action: None

*ESE

488.2
Intent: Set the 1260-00C's Standard Event Status Enable (ESE) register bits.

Command Syntax: `*ESE <byte value>`

where <byte value> is the new value of the ESE register.

Action: Sets ESE to <byte value>.

Example: Set the ESE register to 45.

`*ESE 45`

*ESE?

488.2
Intent: Get the contents of the ESE register.

Query Syntax: `*ESE?`

Response: <ESE val><CRLF>

where <ESE val> is the current value of the ESE register. The default value is FFh.

*ESR?

488.2
Intent: Read and clear the Standard Event Status register (ESR).

Query Syntax: `*ESR?`

Response: <ESR val><CRLF>

<ESR val> is the current value of the ESR.

*IDN?

488.2
Intent: Get the 1260-00C's manufacturer, model, serial number, and firmware level.

Query Syntax: `*IDN?`

Response: "Racal-Dana","1260-00C",<serial number>,<firmware version><CRLF>

*OPC

488.2

Intent: Cause the 1260-00C to generate the operation complete message in the ESR when all pending selected device operations have been finished.

Command Syntax: `*OPC`

Action: None.

Note that because the 1260-00C only parses and routes commands, there are never any pending commands on the 1260-00C

*OPC?

488.2

Intent: Cause the 1260-00C to place an ASCII 1 in its output queue when all pending operations have completed.

Query Syntax: `*OPC?`

Response: 1 <CRLF>

*RST

488.2

Intent: Return a device to a known initial state.

Command Syntax: `*RST`

Action: None.

Other than the response mode configuration, the 1260-00C does not depart from its initial state.

*SRE

488.2

Intent: Set the device's Service Request Enable (SRE) register bits.

Command Syntax: `*SRE <byte value>`

where `<byte value>` is the new value of the SRE register.

Action: Sets the SRE to `<byte value>`.

Example: Set the SRE register to 120.

`*SRE 120`

*SRE?

488.2
Intent: Get the contents of the SRE register.

Query
Syntax: `*SRE?`

Response: `<SRE val><CRLF>`

`<SRE val>` is the current value of the SRE register. The default value is FFh.

*STB?

488.2
Intent: Get the contents of a device's Status Byte.

Query
Syntax: `*STB?`

Response: `<STB value><CRLF>`

where `<STB value>` is the current status of the path to the 1260-00C local command parser.

*TRG

488.2
Intent: Cause a device to execute a stored trigger sequence.

Command
Syntax: `*TRG`

Action: None

*TST

488.2
Intent: Perform self-test and return passed or failed status.

Query
Syntax: `*TST?`

Response: `0` <CRLF>

Failure to complete the self-test is indicated by a failure to respond to this query. If the response is received, the self-test was successful.

*WAI

488.2
Intent: Prevent device from executing any further commands until the No-Operation Pending flag is TRUE.

Command Syntax: `*WAI`

Action: None.

VXIbus Access Commands and Queries

The VXIbus access commands and queries are described on the following pages.

- `A16`
- `A16?`
- `A24`
- `A24?`
- `SYSRESET`

The `A16` and `A24` commands can be used to **poke,** or write, locations in VXI A16 and A24 memory space. The `A16?` and `A24?` queries can be used to **peek,** or read, locations in VXI A16 and A24 memory space.

NOTE

If your application requires block moving or higher speed accesses to or from the VXIbus address spaces, refer to Appendix B, using the DMAmove and CDS-852 Adapter Code Instruments.

The `SYSRESET` command can be used to remotely reset the system.

A16

Purpose: Write a 16-bit value into VXI A16 space.

Command Syntax: `A16 <A16 address>, <word value>`

Action: Write `<word value>` to `<A16 address>`.

Example: Write A502h to VXI A16 address 4305h.

`A16 #h4305, #hA502`

A16?

Purpose: Read word value from VXI A16 address space.

Query Syntax: `A16? <A16 address>`

Response: Program response:

`<word value><CRLF>`

Console response:

`Value <hex word value> (<word value>) read from A16 address <A16 hex address>` (<A16 address>)<CRLF>

where `<word value>` is in decimal integer format, `<hex word value>` is in C language hexadecimal format, `<A16 hex address>` is in C language hexadecimal format, and `<A16 address>` is in decimal integer format.

Example: Read the ID register of Logical Address 16.

`A16? #hc400`

A24

Purpose: Write a 16-bit value into VXI A24 space.

Command Syntax: `A24 <A24 address>, <word value>`

Note that the `<A24 address>` has a valid range of 2097152 to 14680062 (`#h200000 to #hE7FFFE`).

Action: Write `<word value>` to `<A24 address>`.

Example: Write the value A502h to VXI A24 address 504305h.

```
A24  #h504305, #hA502
```

A24?

Purpose: Read a word value from VXI A24 address space.

Query Syntax: `A24?  <A24 address>`

Response: Program response:

```
<word value><CRLF>
```

Console response:

```
Value <hex word value> (<word value>)
read from A24 address <A42 hex address
(<A24 address>)<CRLF>
```

where `<word value>` is in decimal integer format, `<hex word value>` is in C language hexadecimal format, `<A24 hex address>` is in C language hexadecimal format, and `<A24 address>` is in decimal integer format.

Example: Read the word at A24 address 205634h.

```
A24?  #h205634
```

SYSRESET

Purpose: Remotely reset system.

Command Syntax: `SYSRESET`

Action: Asserts the VXI backplane signal SYSRESET*.

TTL/ECL Trigger Access Commands

The TTL/ECL Trigger Access commands are described on the following pages.

- `AckTrig`
- `DisTrigSense`
- `EnaTrigSense`
- `GetTrigHndlr`

- `MapTrigTrig`
- `SetTrigHndlr`
- `SrcTrig`
- `TrigAsstConf`
- `TrigCntrConf`
- `TrigExtConf`
- `TrigTickConf`
- `TrigToREQT`
- `UMapTrigTrig`
- `WaitForTrig`

These commands can be used to directly manipulate the VXI TTL/ECL trigger lines and the front panel trigger connectors of the 1260-00C. The trigger functions are grouped into the following four categories.

1. *Source trigger* commands act as a standard interface for asserting (sourcing) TTL and ECL triggers, and for detecting acknowledgements from accepting devices. These commands can source any of the VXI-defined trigger protocols from the 1260-00C. The source trigger commands are `SrcTrig, SetTrigHndlr, GetTrigHndlr.`

2. *Acceptor trigger commands* act as a standard interface for sensing (accepting) TTL and ECL triggers, and for sending acknowledgements back to the sourcing device. These functions can sense any of the VXI-defined trigger protocols on the 1260-00C. The acceptor trigger commands are `EnaTrigSense, DisTrigSense, SetTrigHndlr,` and `GetTrigHndlr`.

3. *Map trigger commands* are configuration commands for routing and signal conditioning. The `MapTrigTrig` and `UMapTrigTrig` commands can be used to configure the 1260-00C hardware to route specified source trigger locations to destination trigger locations. These commands are also used as a cross-point switch/signal conditioning configurator.

4. *Trigger configuration commands* are configuration tools for

configuring not only the general settings of the trigger inputs and outputs, but also the Racal-Dana Trigger Interface Chip (TIC) counter and tick timers. The trigger configuration commands are `TrigAsstConf`, `TrigExtConf`, `TrigCntrConf`, `TrigTickConf`, and `TrigToREQT`. `TrigToREQT` is a special function for the 1260-00C to map trigger interrupt sources to SRQ on the GPIB so VXI trigger protocols can be completely controlled from an external GPIB controller.

AckTrig

Purpose: Acknowledge the specified TTL/ECL or external (GPIO) trigger.

Query Syntax: `AckTrig <line>`

where the value of `<line>` corresponds to the trigger line to acknowledge:

Value	Trigger Line
0 to 7	TTL trigger lines 0 to 7
8 to 9	ECL trigger lines 0 to 1
40 to 49	External source/destination (GPIO 0 to 9)

Response: Program response: 0

Console response:

`Trigger acknowledged (line = <line text>).<CRLF>`

where the meaning of `<line text>` corresponds to the value of `<line>` as follows:

Value of `<line>`	Value of `<line text>`
0 to 7	TTL `<line>`
8 to 9	ECL `(<line> - 8)`
40 to 49	GPIO `(<line> - 40)`

Example: Acknowledge a trigger interrupt for TTL line 4.

```
AckTrig 4
```

DisTrigSense

Purpose: Disable the sensing of the specified TTL/ECL trigger line, counter, or tick timer that was enabled by `EnaTrigSense`.

Query Syntax: `DisTrigSense <line>`

where the value of `<line>` corresponds to the trigger line on which to disable sensing:

Value	Trigger Line
0 to 7	TTL trigger lines 0 to 7
8 to 9	ECL trigger lines 0 to 1
50	TIC counter
60	TIC TICK timers

Response: Program response: 0

Console response:

```
Trigger sense disabling (line = <line
text>) complete.<CRLF>
```

where the meaning of `<line text>` corresponds to the value of `<line>` as follows:

Value of `<line>`	Value of `<line text>`
0 to 7	`TTL <line>`
8 to 9	`ECL (<line> - 8)`
50	`TCNTR`
60	`TICK 1`

Example: Disable sensing of TTL line 4.

```
DisTrigSense 4
```

EnaTrigSense

Purpose: Enable the sensing of the specified TTL/ECL trigger line or starts up the counter or tick timer for the specified protocol.

Query Syntax: `EnaTrigSense <line>, <protocol>`

where the value of `<line>` corresponds to the trigger line on which to disable sensing:

Value	Trigger Line
0 to 7	TTL trigger lines 0 to 7
8 to 9	ECL trigger lines 0 to 1
50	TIC counter
60	TIC TICK timers

and the value of <protocol> specifies the protocol to use:

Value	Protocol
2	START
3	STOP
4	SYNC
5	SEMI-SYNC
6	ASYNC

Response: Program response: 0

Console response:

```
Trigger sense enabling (line = <line
text>, protocol = <protocol text>)
complete.<CRLF>
```

where the meaning of <line text> corresponds to the value of <line> as follows:

Value of <line>	Value of <line text>
0 to 7	TTL <line>
8 to 9	ECL (<line> - 8)
50	TCNTR
60	TICK 1

and the meaning of <protocol text> corresponds to the value of <protocol> as follows:

Value of <protocol>	Value of <protocol text>
2	START
3	STOP
4	SYNC
5	SEMI-SYNC
6	ASYNC

Example: Enable sensing of TTL line 4 for SEMI-SYNC protocol.

```
EnaTrigSense 4, 5
```

GetTrigHndlr

Purpose: Get the address of the current TTL/ECL trigger, counter, or tick timer interrupt handler for a specified trigger source.

Query Syntax:

```
GetTrigHndlr <line>
```

where the value of `<line>` corresponds to the trigger line or counter/tick source:

Value	Trigger Line
0 to 7	TTL trigger lines 0 to 7
8 to 9	ECL trigger lines 0 to 1
50	TIC counter
60	TIC TICK1 tick timer

Response: Program response: 0

Console response:

```
Trigger handler (line = <line text>):
DefaultTrigHandler().<CRLF>
```

where the meaning of `<line text>` corresponds to the value of `<line>` as follows:

Value of `<line>`	Value of `<line text>`
0 to 7	`TTL <line>`
8 to 9	`ECL (<line> - 8)`
50	`TCNTR`
60	`TICK 1`

Example: Get the address of the trigger interrupt handler for TTL trigger line 4.

```
GetTrigHndlr 4
```

MapTrigTrig

Purpose: Map a specified TTL, ECL, Star X, Star Y, external connection (GPIO), or miscellaneous signal line to another.

Query Syntax:

```
MapTrigTrig <srcTrig>, <destTrig>,
<mode>
```

where `<srcTrig>` is the source line to map to a destination, the value of `<destTrig>` corresponds to the destination line to map from the source:

Value	Trigger Line
0 to 7	TTL trigger lines 0 to 7
8 to 9	ECL trigger lines 0 to 1
40 to 49	External source/destination (GPIO 0 to 9)
40	Front panel In (connector 1)
41	Front panel Out (connector 2)
42	Front panel In (unbuffered)

43	Connection to EXTCLK input pin
44 to 49	Hardware-dependent GPIOs 4 to 9
50	TIC counter pulse output (TCNTR)
51	TIC counter finished output (GCNTR)
60	TIC TICK1 tick timer output
61	TIC TICK2 tick timer output

and the value of `<mode>` specifies the signal conditioning mode (where 0 = no conditioning). The conditioning effects for bits 0 to 3 are as follows:

Bit	Conditioning Effect
0	Synchronize with next CLK edge
1	Invert signal polarity
2	Pulse stretch to one CLK minimum
3	Use EXTCLK (not CLK10) for conditioning

Response: Program response: 0

Console response:

```
Mapping complete (line = <line text
source> mapped to line = <line text
destination>,  mode = <mode>).<CRLF>
```

where the meaning of `<line text source>` and `<line text destination>` correspond to the value of `<srcTrig>` and `<destTrig>` as follows:

Value of `<srcTrig>` or <destTrig>	Value of `<line text source>` or `<line text destination>`
0 to 7	`TTL <line>`
8 to 9	`ECL (<line> - 8)`
40 to 49	`GPIO (<line> - 40)`
50	`TCNTR`
51	`GCNTR`
60	`TICK1`
61	`TICK2`

Example: Map TTL line 4 to go out of the front panel with no signal conditioning.

```
MapTrigTrig 4, 41, 0
```

SetTrigHndlr

Purpose: Replace the current TTL/ECL trigger, counter, or tick timer interrupt handler with a specified trigger source with a specified function.

Query Syntax: `SetTrigHndlr <lines>, <function>`

where `<lines>` is a bit vector of trigger lines (1 = set; 0 = do not set), where the value corresponds to the trigger line(s) to set:

Value	Trigger Line
0 to 7	TTL trigger lines 0 to 7
8 to 9	ECL trigger lines 0 to 1
14	TIC counter
15	TIC TICK timers

and `<function>` is a pointer to the new trigger interrupt handler, where the value is defined as follows:

Value	Interrupt Handler
0	Specify `DefaultTrigHandler`
1	Specify `DefaultTrigHandler2`
other	User-installed trigger interrupt handler

Response: Program response: 0

Console response:

```
Trigger handler(s) installed (lines = <lines text>):
DefaultTrigHandler().<CRLF>
```

where the meaning of `<lines text> = x, y, z...`, where `x, y,` and `z` are bits that are set in `<lines>`.

Example: Set a trigger interrupt handler for TTL trigger line 4.

```
SetTrigHndlr 16
```

Note:

`DefaultTrigHandler` automatically acknowledges acceptor protocols that require acknowledgement, while `DefaultTrigHandler2` does not. If `DefaultTrigHandler2` is used, send `AckTrig` to acknowledge the trigger.

SrcTrig

Purpose: Source a specified protocol on a specified TTL, ECL, or external trigger line.

Query Syntax: `SrcTrig <line>, <protocol>, <timeout>`

where the value of `<line>` corresponds to the trigger line to source:

Value	Trigger Line
0 to 7	TTL trigger lines 0 to 7
8 to 9	ECL trigger lines 0 to 1
40 to 49	External source/destination (GPIO 0 to 9) (supports ON, OFF, START, STOP, and SYNC protocols only)
50	TIC counter (supports SYNC and SEMI-SYNC protocols only)
60	TIC TICK timers (supports SYNC and SEMI-SYNC protocols only)

the value of `<protocol>` specifies the protocol to use:

Value	Protocol
0	ON
1	OFF
2	START
3	STOP
4	SYNC
5	SEMI-SYNC
6	ASYNC
7	SEMI-SYNC and wait for Acknowledge
8	ASTNC and wait for Acknowledge
ffffh	Abort previous Acknowledge pending (5 and 6)

and `<timeout>` is the timeout value in milliseconds

Response: Program response: 0

Console response:

```
Trigger sourcing (line = <line text>,
protocol = <protocol text>)
complete.<CRLF>
```

where the meaning of `<line text>` corresponds to the value of `<line>` as follows:

Value of `<line>`	Value of `<line text>`
0 to 7	`TTL <line>`
8 to 9	`ECL (<line> - 8)`
40 to 49	`GPIO (<line> - 40)`
50	`TCNTR`
60	`TICK1`

and the meaning of `<protocol text>` corresponds to the value of `<protocol>` as follows:

Value of `<protocol>`	Value of `<protocol text>`
0	ON
1	OFF
2	START
3	STOP
4	SYNC
5	SEMI-SYNC
6	ASYNC
7	SEMI-SYNC wait ACK
8	ASYNC wait ACK
ffffh	wait ACK ABORT

Example: Source TTL line 4 for SEMI-SYNC protocol.

```
SrcTrig 4, 5, 0L
```

TrigAsstConf

Purpose: Configure a specified TTL/ECL trigger line assertion method.

Query Syntax:

```
TrigAsstConf <line>, <mode>
```

where the value of `<line>` corresponds to the trigger line to configure:

Value	Trigger Line
0 to 7	TTL trigger lines 0 to 7
8 to 9	ECL trigger lines 0 to 1
ffffh	General assertion configuration (all lines)

and `<mode>` specifies the configuration mode, where

Bit	Specific Line Configuration Modes
0	1 = Synchronize falling edge of CLK10 0 = Synchronize rising edge of CLK10

Bit	General Configuration Modes
1	1 = Pass trigger through asynchronously 0 = Synchronize with next CLK10 edge
2	1=Participate in SEMI-SYNC with external trigger acknowledge protocol 0 = Do not participate

All other values are reserved for future expansion.

Response: Program response: 0

Console response:

```
Trigger assertion configuration (line
= <line text>, mode = <mode>)
complete.<CRLF>
```

where the meaning of `<line text>` corresponds to the value of `<line>`:

Value of `<line>`	Value of `<line text>`
0 to 7	`TTL <line>`
8 to 9	`ECL (<line> - 8)`
ffffh	`GENERAL CONFIG`

Example 1: Configure all TTL/ECL trigger lines generally to synchronize to the falling edge of CLK10 (as opposed to the rising edge).

```
TrigAsstConf -1, 1
```

Example 2: Configure TTL trigger line 4 to synchronize to CLK10 for any assertion method and do not participate in SEMI-SYNC.

```
TrigAsstConf 4, 0
```

TrigCntrConf

Purpose: Configure the TIC chip internal 16-bit counter.

Query Syntax:

```
TrigCntrConf <mode>, <source>, <count>
```

where `<mode>` specifies the configuration mode:

Value	Mode
0	Initialize the counter
2	Reload the counter leaving enabled
3	Disable/abort any count in progress

`<source>` specifies the trigger line to configure as input to counter:

Value	Trigger Line
0 to 7	TTL trigger lines 0 to 7
8 to 9	ECL trigger lines 0 to 1
70	CLK10
71	EXTCLK

and `<count>` specifies the number of input pulses to count before terminating.

Response: Program response: 0

Console response:

```
CNTR configured (mode = <mode text>,
source = <source text>, count =
<count>).<CRLF>
```

where the meaning of `<mode text>` corresponds to the value of `<mode>`:

Value of <mode>	Value of <mode text>
0	INIT
2	RELOAD
3	DISABLE

and the meaning of `<source text>` corresponds to the value of `<source>`:

Value of <source>	Value of <source text>
0 to 7	TTL <line>
8 to 9	ECL (<line> - 8)
70	CLK10
71	EXTCLK

Example : Configure the counter count 25 assertions on TTL trigger line 5 (the `<protocol>` parameter when calling `EnaTrigSense` will determine whether the counter accepts SYNC or SEMI-SYNC assertions).

```
TrigCntrConf 0, 5, 25
```

TrigExtConf

Purpose: Configure the external trigger (GPIO) lines.

Query Syntax:

```
TrigExtConf <extline>, <mode>
```

where the value of `<extline>` is the trigger line to configure:

Value	Trigger Line
40 to 49	External source/destination (GPIO 0 to 9)
40	Front panel In (connector 1)
41	Front panel Out (connector 2)
42	ECL bypass from front panel
43	EXTCLK
44 to 49	Hardware-dependent GPIOs 4 to 9

and `<mode>` specifies the configuration mode, where

Bit	Configuration Modes
0	1 = Feed back any line mapped as input into the cross-point switch
	0 = Drive input to external (GPIO) pin
1	1 = Assert input (regardless of feedback)
	0 = Leave input unconfigured
2	1 = If assertion selected, assert low
	0 = If assertion selected, assert high
3	1 = Invert external input (not feedback)
	0 = Pass external input unchanged

All other values are reserved for future expansion.

Response: Program response: 0

Console response:

```
External connection (GPIO)
configuration complete (extline =
<extline text>, mode = <mode>).<CRLF>
```

where the meaning of `<extline text>` corresponds to the value of `<extline>` as follows:

Value of `<extline>`	Value of `<extline text>`
40 to 49	`GPIO (<extline> - 40)`

Example 1 : Configure external line 41 (front panel Out) to not be used as feedback and left tri-stated for use as a mapped output via `MapTrigTrig`.

```
TrigExtConf 41, 0
```

Example 2 : Configure external line 40 (front panel In) to not be used as feedback and left tri-stated for use as a mapped input via `MapTrigTrig`.

```
TrigExtConf 40, 16
```

Example 3 : Configure external line 48 (GPIO 8) to be used as feedback for use as a cross-point switch input and output via `MapTrigTrig`.

```
TrigExtConf 48, 1
```

TrigTickConf

Purpose: Configure the TIC chip internal dual 5-bit tick timers.

Query Syntax:

```
TrigTickConf <mode>, <source>, <tcount1>, <tcount2>
```

where `<mode>` specifies the configuration mode:

Value	Mode
0	Initialize the tick timers (rollover mode)
1	Initialize the tick timers (non-rollover mode)
2	Reload the tick timers leaving enabled
3	Disable/abort any count in progress

and the value of `<source>` is the trigger line to configure as input to counter:

Value	Trigger Line
40 to 49	External source/destination (GPIO 0 to 9)
70	CLK10
71	EXTCLK

and the values of `<tcount1>` and `<tcount2>` are the number of input pulses (as a power of two) to count before asserting TICK1 output or TICK2 output (and terminating the tick timer if configured for non-rollover mode).

Response: Program response: 0

Console response:

```
TICKs configured (mode = <mode text>, source = <source text>, t1 = tcount1, t2 = tcount2).<CRLF>
```

where the meaning of `<mode text>` corresponds to the value of `<mode>`:

Value of <mode>	Value of <mode text>
0	`INIT w/ROLL`
1	`INIT w/NOROLL`
2	`RELOAD`
3	`DISABLE`

and the meaning of `<source text>` corresponds to the value of `<source>`:

Value of <source>	Value of <source text>
40 to 49	`GPIO (<source> - 40)`
70	`CLK10`
71	`EXTCLK`

Example 1 : Configure the tick timers to interrupt every 6.55 milliseconds by dividing down CLK10 as an input. Call `EnaTrigSense` to start the tick timers and enable interrupts.

```
TrigTickConf 0, 70, 16, 0
```

Example 2 : Configure the tick timers to output a continuous 9.765-kHz square wave on TICK1 output and a 1.25 MHz clock on TICK2 output by dividing down CLK10 as an input. Call `SrcTrig` to start the tick timers.

```
TrigTickConf 0, 70, 10, 3
```

TrigToREQT

Purpose: Map trigger interrupt to GPIB SRQ condition (REQT generation for a particular GPIB address).

Command Syntax:

```
TrigToREQT <la>, <line>
```

where `<la>` identifies the device for which to assert SRQ, and `<line>` is the trigger line for which to map the interrupt, where the value of `<line>` corresponds to the trigger line or counter/tick:

Value	Trigger Line
0 to 7	TTL trigger lines 0 to 7
8 to 9	ECL trigger lines 0 to 1
50	TIC counter
60	TIC TICK1 tick timer

Action: The 1260-00C is set up to assert SRQ for a device attached to a GPIB address for a given trigger line's interrupt, as configured using either the `SrcTrig` or `EnableSense` function.

Response: Program response: 0

Console response:

```
Line: <line text>, configured to
generate an REQT for Logical Address
<la>.<CRLF>
```

where the meaning of `<line text>` corresponds to the value of `<line>` as follows:

Value of `<line>`	Value of `<line text>`
0 to 7	`TTL <line>`
8 to 9	`ECL (<line> - 8)`
50	`TCNTR`
60	`TICK1`

Example: Set up Logical Address 4 to assert SRQ when a trigger interrupt occurs on TTL trigger line 2.

```
TrigToREQT 4, 2
```

UMapTrigTrig

Purpose: Unmap a specified TTL, ECL, Star X, Star Y, external connection (GPIO), or miscellaneous signal line that was mapped to another line using the `MapTrigTrig` function.

Query Syntax:

```
UMapTrigTrig <srcTrig>, <destTrig>
```

where `<srcTrig>` is the source line to unmap from a destination, and the value of `<destTrig>` corresponds to the destination line mapped from the source:

Value	Trigger Line
0 to 7	TTL trigger lines 0 to 7
8 to 9	ECL trigger lines 0 to 1
40 to 49	External source/destination (GPIO 0 to 9)
40	Front panel In (connector 1)
41	Front panel Out (connector 2)
42	Front panel In (unbuffered)
43	Connection to EXTCLK input pin
44 to 49	Hardware-dependent GPIOs 4 to 9
50	TIC counter pulse output (TCNTR)

51	TIC counter finished output (GCNTR)
60	TIC TICK1 tick timer output
61	TIC TICK2 tick timer output

Response: Program response: 0

Console response:

`Unmapping complete (line <line text source> unmapped from line <line text destination>).<CRLF>`

where the meaning of `<line text source>` and `<line text destination>` correspond to the value of `<srcTrig>` and `<destTrig>`:

Value of `<srcTrig>` or `<destTrig>`	Value of `<line text source>` or `<line text destination>`
0 to 7	`TTL <line>`
8 to 9	`ECL (<line> - 8)`
40 to 49	`GPIO (<line> - 40)`
50	`TCNTR`
51	`GCNTR`
60	`TICK1`
61	`TICK2`

Example: Unmap route of TTL line 4 to go out of the front panel as mapped by `MapTrigTrig`.

```
UMapTrigTrig 4, 49
```

WaitForTrig

Purpose: Wait for the specified trigger line to be sensed for the specified time. `EnaTrigSense` must be called to sensitize the hardware to the particular trigger protocol to be sensed.

Query Syntax:

```
WaitForTrig <line>, <timeout>
```

where the value of `<line>` corresponds to the trigger line to wait for:

Value	Trigger Line
0 to 7	TTL trigger lines 0 to 7
8 to 9	ECL trigger lines 0 to 1
50	TIC counter
60	TIC TICK1 tick timer

Response: Program response: 0

Console response:

`Trigger received (line = <line text>),`
`wait complete.<CRLF>`

where the meaning of `<line text>` corresponds to the value of `<line>` as follows:

Value of `<line>`	Value of `<line text>`
0 to 7	`TTL <line>`
8 to 9	`ECL (<line> - 8)`
50	`TCNTR`
60	`TICK1`

Example: Wait for TTL line 4 to be encountered.

`WaitForTrig 4, 10000L`

Word Serial Communication Command and Queries

The Word Serial communication commands and queries are described on the following pages.

- `ProtErr?`
- `RespReg?`
- `WScmd`
- `WScmd?`
- `WSresp?`
- `WSstr`
- `WSstr?`

These commands are used to directly generate Word Serial communication operations with any Message-Based device, including the 1260-00C itself, regardless of whether or not it is the 1260-00C's Servant.

NOTE

The Word Serial communication commands and queries are intended for debugging purposes. Racal-Dana does not guarantee these commands will work when other Word Serial paths are open, such as the GPIB address

link.

Some of the Word Serial commands defined in the VXIbus specification require a response from the Message-Based device, while other commands do not. To distinguish between the two types of Word Serial commands, and to avoid confusion between Word Serial commands and 1260-00C local commands and queries, the following terminology will be used in this section:

a. *Word Serial Command* - A VXI-defined Word Serial command that does not require a response from the Message-Based device.

b. *Word Serial Query* - A VXI-defined Word Serial command that requires a response from the Message-Based device.

c. *Command* - A 1260-00C command, as defined in this chapter.

d. *Query* - A 1260-00C query, as defined in this chapter.

The `WScmd` command can be used to send a Word Serial command to a Message-Based device. The `WScmd?` query is used to send a Word Serial query to the Message-Based device, and to automatically read and return the device's response.

The `WScmd` can be used to send a Word Serial query to a Message-Based device. Because `WScmd` does not read the query response, the intermediate state of the device can be examined using the `RespReg?` query, after which the response can be read using the `WSresp?` query.

The `WSstr` command can be used to send device-dependent commands and queries to a device. If the string sent to the device was a device-dependent query, use the `WSstr?` query to read the device's response.

The `ProtErr?` query sends a *Read Protocol Error* Word Serial query to a device and reports the error response. The `RespReg?` query returns the value of a device's response register.

ProtErr?

Purpose: Send a *Read Protocol Error* Word Serial query to a Message-Based device.

Query Syntax: `ProtErr? <log addr>`

Action: *Read Protocol Error* query is sent to a Message-Based device. Response is read and reported.

Response: Program response:

<hex value><CRLF>

where <hex value> is the hexadecimal value of the Data Low register response.

Console response:

```
Read Protocol Error for Logical Address <log
addr> returned 0x<hex value>:
           <description>
```

where <description> is text explaining the error response.

Example: `ProtErr? 3`

RespReg?

Purpose: Get the Response register contents of a Message-Based device.

Query Syntax: `RespReg? <log addr>`

Action: Returns the contents of the device's Response register at Logical Address <log addr>.

Response: Program response:

<hex value><CRLF>

where <hex value> is the hexadecimal value of the Response register contents.

Console response:

```
Logical Address  <log addr>'s Response
register:<CRLF>

[0x<hex value>]: <dor> <dir> <err> <rr> <wr>
<fhs> <locked><CRLF>
```

where <dor>, <dir>, <err>, <rr>, <wr>, <fhs>, and <locked> are text flags that interpret the state of the Response register bit flags. Capitalized text in a text flag indicates that the

corresponding bit flag is in the logic TRUE state.

Lowercase text indicates that the corresponding bit flag is in the logic FALSE state.

WScmd

Purpose: Send a 16-bit Word Serial command or query to a Message-Based device.

Command Syntax: `WScmd  <log addr>, <WS cmd>`

Action: Sends the Word Serial command `<WS cmd>` to the device at `<log addr>`.

Example: Write the *Begin Normal Operation* Word Serial query (FCFFh) to a device at Logical Address 3.

```
WScmd 3, #hFCFF
```

WScmd?

Purpose: Send a 16-bit Word Serial query to a Message-Based device.

Query Syntax: `WScmd? <log addr>, <WS cmd>`

Action: Sends the Word Serial query `<WS cmd>` to the device at `<log addr>`. Reads and returns the device's response.

Response: Program response:

`<hex value>`<CRLF>

where `<hex value>` is the hexadecimal value of the Data Low register response.

Console response:

```
Logical Address <log addr> Word
Serial Query 0xceff returned 0x<hex
value>.<CRLF>
```

Example: Write the *Read Servant Area* Word Serial query (CEFFh) to a device at Logical Address 4.

```
WScmd? 4, #hCEFF
```

Wsresp?

Purpose: Read a 16-bit Word Serial response to a previously sent query.

Query Syntax: `WSresp?  <log addr>`

Action: Reads and returns the response of the device at `<log addr>`.

Response: Program response:

`<hex value><CRLF>`

where `<hex value>` is the hexadecimal value of the Data Low register response.

Console response:

`Logical Address <log addr> returned response 0x<hex value><CRLF>`

Example: Read the 16-bit response to a previously sent Word Serial query from Logical Address 3.

`WSresp? 3`

WSstr

Purpose: Send a device-dependent command string to a Message-Based device.

Command Syntax: `WSstr  <log addr>, <string>`

where `<string>` is an ASCII character sequence enclosed by double quotation marks (").

The following sequences of characters within the `<string>` parameter are special cases and will be interpreted as follows:

`\n`	linefeed (LF)
`\r`	carriage return (CR)
`\\`	backslash (\)
`\xHH`	any binary 8-bit value where `HH` is the ASCII hexadecimal representation of that value

Action: Writes the string <string> to the device at <log addr> as a series of *Byte Available* commands.

Example: Write the string "start" to a device at Logical Address 8.

```
WSstr  8, "start"
```

WSstr?

Purpose: Read a device-dependent response string from a Message-Based device.

Query Syntax: `WSstr?  <log addr>, <max cnt>`

Action: Reads and returns a string, up to a maximum character count of <max cnt>, using a series of *Byte Request* commands.

Response: Program response:

```
<resp string>
```

where <resp string> is the response string returned by the device.

Console response:

```
Logical Address <log addr> read <#
bytes> (<hex # bytes>) through Word
Serial: <CRLF><CRLF> <resp string>
```

where <# bytes> and <hex # bytes> are the number of bytes in <resp string>, in decimal and hexadecimal.

Example: Read a device-dependent response up to 20 characters long from a device at Logical Address 10.

```
WSstr? 10, 20
```

CI Configuration Commands and Queries

The CI configuration commands and queries are described on the following pages.

- CIAddr?
- CIArea

- `CIArea?`
- `CIBlocks?`
- `CIDelete?`
- `CIList?`
- `DCIDownLdPI`
- `DCIDownLoad`
- `DCISetup?`
- `DCISetupPI?`
- `ECIboot?`

These commands and queries manipulate CIs and their related resources, and extract information about the CI configuration.

The query `CIList?` returns the list of code instrument logical addresses. The RM information queries that access information for physical devices (`Cmdr?, RmEntry?, Srvnts?, StatusState?`) can be used to retrieve the equivalent information for a CI. The `CIDelete` query deletes a CI.

The amount of RAM reserved for all CIs set by the 1260-00C depends upon its non-volatile configuration, the amount of RAM installed, and the use of the command `CIArea`. The CI RAM area is partitioned into blocks of 4 kilobytes in size. The `CIArea?` query can determine the current location and size of the CI RAM area. The `CIBlocks?` query returns the allocation state of each block in the CI RAM area. The `CIAddr?` query can determine the base address of a particular CI's RAM area.

Static Downloaded CIs (DCIs) are downloaded to the 1260-00C and initialized with the local commands `DCISetUp?` and `DCIDownLoad`.

Position Independent DCIs are downloaded to the 1260-00C and initialized with the local commands `DCISetupPI?` and `DCIDownLdPI`.

`ECIboot?` may be used for debug or runtime purposes to start up an EPROMed Code Instrument that is already installed on the 1260-00C. This is an alternative to non-volatile configuration.

CIAddr?

Purpose: Get the local base RAM address of a CI.

Query Syntax: `CIAddr? <logical address>`

Response: Program response:

`<base address><CRLF>`

where `<base address>` is the CI's base address in decimal.

Console response:

`CI at Logical Address <logical address> base Local Address is <hex base address><CRLF>`

where `<hex base address>` is the CI's base address in C language hexadecimal notation.

Example: Get the local address of the CI at Logical Address 9.

`CIAddr? 9`

CIArea

Purpose: Change the location and size of the CI RAM area.

Command Syntax: `CIArea <Base Address>, <Number of blocks>`

Action: Sets the CI global RAM to start at `<Base Address>`, and span `<Number of blocks>` blocks of 4096 bytes each.

The default base address and size of the CI RAM area are set by the non-volatile configuration parameters `CI Block Base` and `CI Num Blocks`.

`<Base Address>` is the new base address of the CI RAM area. It must be a multiple of 4096 decimal (1000h), and must be in the region above the top of pSOS Region 1 and below the top of memory. pSOS Region 1 starts at 10000h, and its size is determined by the non-volatile configuration parameter `Region 1 Size.` For example, if `Region 1 Size` = 60000h, the lowest allowed value for `<Base Address>` is as follows:

10000h + 60000h = 70000h

`<Number of blocks>` is the number of 4096 (1000h) byte blocks in the CI RAM area. The size of the CI RAM area is limited by the amount of physical RAM on the 1260-00C, so the maximum allowed value for `<Number of blocks>` is as follows:

```
(<RAM size> - <Base Address>) / 1000h
```

For example, if the 1260-00C is configured with 512 kilobytes (80000h) of RAM, and `<New Base Address>` is 70000h, the maximum allowed value for `<Number of blocks>` is given by the following formula:

```
(80000h - 70000h)/ 1000h = 10h = 16
```

If `<Number of blocks>` is set to 0, CI's are disabled.

Example: Set the base of CI RAM area to 80000h, and the size to 128 blocks of 4 kilobytes.

```
CIArea  #h80000, 128
```

CIArea?

Purpose: Return the base address and size of CI global memory area.

Query Syntax: `CIArea?`

Response: Program response:

```
<base address>, <number of blocks><CRLF>
```

where `<base address>` and `<number of blocks>` are the current base address and size of the CI RAM area in blocks of 4 kilobytes.

Console response:

```
CI Global Base is local Address <hex base address> with <number of blocks> 4K blocks<CRLF>
```

`<hex base address>` is the base address of the CI RAM area in C language hexadecimal notation. `<number of blocks>` is the size of the CI RAM area (in blocks of 4 kilobytes) in decimal.

CIBlocks?

Purpose: Return a listing of used and unused CI memory area blocks.

Query Syntax: `CIBlocks?`

Response: Program response:

`<b0>, <b1>, . . ., <bL-1>`<CRLF>

where `<bJ>` is a Boolean value that indicates whether the Jth block is unused (0) or used (1). `L` is the number of blocks of 4 kilobytes in the CI global memory area.

Console response:

`Blocks Used of the <L> Blocks of CI Global Memory:` <CRLF>

`<r0start> - <r0stop>, <r1start> - <r1stop>, . . .,<rN-1start> - <rN-1stop>`<CRLF>

where `<rMstart>` and `<rMstop>` are the start and stop block numbers for the Mth occupied memory region.

CIDelete?

Purpose: Delete a CI.

Query Syntax: `CIDelete? <code instrument logical address>`

`<code instrument logical address>` is the logical address of the CI to be deleted.

Response: Program response:

`<error code>`<CRLF>

where `<error code>` is a decimal value that indicates the result of the attempt to delete the CI. If `<error code>` is equal to 0, the attempt was successful.

Console response:

```
Code Instrument at Logical Address
<code instrument logical address>
successfully deleted<CRLF>
```

if the attempt was successful, or

```
Error Deleting Code Instrument (Error
code = <hex error code>)
```

if the attempt was unsuccessful.

`<hex error code>` is a value in C language hexadecimal format that indicates the result of the attempt to delete the CI.

`<error code>` and `<hex error code>` can be interpreted by converting them to a binary bit pattern. A value of 1 in any bit position indicates the error shown in the following table occurred during the attempt to delete the CI.

Bit	Error Condition
0	The 1260-00C was unable to delete the CI's GPIB address link.
1	The 1260-00C was unable to delete the CI's message exchange.
2	The 1260-00C was unable to delete the CI's Async process.
3	The 1260-00C was unable to delete the CI's Worker process.
4	The 1260-00C was unable to delete the CI's Word Serial I/O structures.
5	The 1260-00C was unable to free the PI CI's dynamic memory.

Any error encountered is unrecoverable because the CI is not restored. Any further attempts to communicate with it will have undetermined results, and can adversely affect the behavior of the 1260-00C.

CIList?

Purpose: Get a list of logical addresses for CIs running on the 1260-00C.

Query Syntax: `CIList?`

Response: Program response:

`<ci la1>,<ci la2>, . . .,<ci laN><CRLF>`

where `<ci laJ>` is the logical address of the Jth local CI. `N` is the total number of local CIs.

Console response:

`Local CI Logical Addresses are: <ci la1>,<ci la2>, . . .,`

`<ci laN><CRLF>`

DCIDownLdPI

Purpose: Download a Position Independent (PI) CI to RAM and start running it.

Command Syntax: `DCIDownLdPI [<Boolean>]`

Action: Bytes of code and data (up to the number requested in the `DCISetupPI?` command) are downloaded from the command source. When the download is complete, the pSOS processes associated with the PI DCI are initiated.

DCIs can only be downloaded from the GPIB port or via Word Serial Protocol, because the download is terminated on GPIB EOI or the Word Serial `END` command. Because there is no analogy for EOI or END for the serial port (that is, a carriage return is a valid binary number), it cannot be used to download PI DCIs.

If `<Boolean>` is 1, debug statements are printed to the serial port during the various stages of PI DCI initialization. If `<Boolean>` is 0 or if it is omitted, the debug statements are not output. The debug printing mode is only available with the development firmware option.

The `DCIDownLdPI` command should always be immediately preceded by a `DCISetupPI?` command that configures the download parameters. Executing intermediate 1260-00C commands between DCISetupPI? and DCIDownLdPI may invalidate the download setup.

Example: Download and initialize a PI DCI, generating debug statements.

```
DCIDownLdPI 1
```

DCIDownLoad

Purpose: Download a CI to RAM and start running it.

Command Syntax: `DCIDownLoad [<Boolean>]`

Action: Blocks of code and data (up to the number requested in the `DCISetup?` command) are downloaded from the command source. When the download is complete, the pSOS processes associated with the DCI are initiated.

DCIs can only be downloaded from the GPIB port or via Word Serial protocol, because the download is terminated on GPIB EOI or the Word Serial `END` command. Because there is no analogy for EOI or END for the serial port, it cannot be used to download DCIs.

If `<Boolean>` is 1, debug statements are printed to the serial port during the various stages of DCI initialization. If `<Boolean>` is 0 or if it is omitted, the debug statements are not output. The debug printing mode is only available with the development firmware option.

The `DCIDownLoad` command should always be immediately preceded by a `DCISetup?` command that configures the download parameters. Executing intermediate 1260-00C commands between `DCISetup?` and `DCIDownLoad` may invalidate the download setup.

Example: Download and initialize a DCI, generating debug statements.

```
DCIDownLoad 1
```

DCISetup?

Purpose: Set up parameters for a DCI download.

Query

Syntax: `DCISetup? <logical address>, <Commander's logical address>, <start block>, <number of blocks>, <stack size>, [, <Servant1>, [<Servant2>,..., <ServantN>]]`

The `DCISetup?` query provides the 1260-00C with the information it needs to prepare for executing a `DCIDownLoad` command. This command is performed separately from the `DCIDownLoad` command so that the download parameters can be validated before the object code download is initiated.

The 1260-00C interprets the `DCISetup?` query parameters as follows:

- The DCI is to be assigned Logical Address `<logical address>`, and granted to the Commander at `<Commander's logical address>` as a Servant.
- Up to `<number of blocks>` of DCI code and data are to be loaded into CI RAM area starting at `<start block>`.
- A stack of size `<stack size>` words is to be allocated for the CI worker process. If `<stack size>` is less than 1024, a stack size of 1024 words (2 kilobytes) is to be allocated for the CI.
- `<Servant1>` through `<ServantN>` are to be granted to the CI as Servants.

Any GPIB address links to `<Servant1>`, `<Servant2>` through `<ServantN>` must be disconnected before the DCI is downloaded. You can delete the links by using the `SaDisCon` or `LaSaddr` commands.

Response: Program response:

`0` <CRLF>

`0` is returned to report the successful completion of the `DCISetup?` command. Any errors are reported in the form of an error code.

Console response:

`DCI parameters OK, Ready to`

`download.`<CRLF>

Example: Set up to download a DCI at Logical Address C0h, to be a Servant of the device at Logical Address 2 and Commander of device at Logical Address 50. Set up to download to the first 20 blocks of DCI memory area, and allocate a 2048-word stack.

```
DCISetup?  #hC0,2,0,20,#h800,50
```

DCISetupPI?

Purpose: Set up parameters for a PI DCI download.

Query

Syntax:

```
DCISetupPI? <logical address>,
<Commander's logical address>,
<dynamic RAM size>, <stack size>, [,
<Servant1>, [<Servant2>,...,
<ServantN>]]
```

The `DCISetupPI?` query provides the 1260-00C with the information it needs to prepare for executing a `DCIDownLdPI` command. This command is performed separately from the `DCIDownLdPI` command so that the download parameters can be validated before the object code download is initiated.

The 1260-00C interprets the `DCISetupPI?` query parameters as follows:

- The PI DCI is to be assigned Logical Address `<logical address>,` and granted to the Commander at `<Commander's logical address>` as a Servant.

- Up to `<dynamic RAM size>` bytes of PI DCI code and data are to be loaded into a pSOS dynamic RAM segment allocated when the `DCIDownLoadPI` command is sent.

- A stack of size `<stack size>` words is to be allocated for the CI worker process. If `<stack size>` is less than 1024, a stack size of 1024 words (2K) is to be allocated for the CI.

- `<Servant1>` through `<ServantN>` are to be granted to the CI as Servants.

Any GPIB address links to <Servant1>, <Servant2> through <ServantN> must be disconnected before the PI DCI is downloaded. You can delete the links by using the `SaDisCon` or `LaSaddr` commands.

Response: Program response:

`0` <CRLF>

`0` is returned to report the successful completion of the `DCISetupPI?` command. Any errors are reported in the form of an error code.

Console response:

`PI DCI parameters OK, Ready to download.`<CRLF>

Example: Set up to download a PI DCI at Logical Address C0h, to be a Servant of the device at Logical Address 2 and Commander of device at Logical Address 50. Set up to download up to 10000 bytes of code and data to a pSOS dynamic memory segment, and allocate a 2048-word stack.

```
DCISetupPI?  #hC0,0,10000,#h800,50
```

ECIboot?

Purpose: Start up an EPROMed Code Instrument (static or position independent) that already resides in memory of the 1260-00C.

Query Syntax: `ECIboot? <address>, <debug>`

Response: Program response:

none

Console response:

`EPROMed Code Instrument at address <address> booting complete.`<CRLF>

where `<address>` is the local 1260-00C memory-mapped address of the EPROMed Code Instrument.

Example: Boot EPROMed Code Instrument at address f7C000h with debug off.

```
ECIboot?  #hf7C000, 0
```

This page was left intentionally blank.

Chapter 5

NON-VOLATILE CONFIGURATION

Non-Volatile Overview

This Chapter describes the method for editing and reviewing the contents of the non-volatile memory used for storing configuration information on the 1260-00C.

The 1260-00C non-volatile (NV) memory is a 256-byte EEPROM accessible as 64 longword locations. The first half of the NV memory (32 longwords) is reserved for use by Racal-Dana. The second half of NV memory is allocated for storing Code Instrument (CI) configuration variables.

The configuration parameters include the following:

- Local register configuration
- pSOS configuration
- VXI interrupt line assignment
- Resource Manager (RM) A24 and A32 address assignment base
- Servant area size
- DC starting logical address and hierarchy configuration
- Device failure mode
- GPIB configuration
- Default CI configuration
- CI RAM area configuration
- Resident CI locations
- CI user configuration variables

The NV configuration mode can be entered through any of the following methods.

a. Set the start-up mode switches to the non-volatile configuration mode as described in Chapter 3 Start-up Mode Configuration. Set switch S19 to the ON position, and set switch S20 to the OFF position. Restart the system.

b. In 488-VXI runtime system mode, enter NV configuration mode through the `CONF` command.

c. In VXI pROBE mode (on development modules only), enter NV configuration mode through the CONF command.

The non-volatile configuration commands must be executed from the RS-232 port.

The EEPROM is connected to the microprocessor via a I^2C serial bus. It takes five to ten seconds to write the contents of the memory. The 1260-00C creates a copy of the contents of the EEPROM in RAM, which can be quickly edited. When the editing is complete, the entire contents of the RAM copy is written back to the EEPROM.

Note that some changes (such as the pSOS parameters) do not take effect until the system is restarted. This is accomplished by the pROBE commands `IN` or `BO`, by resetting the system, or by cycling the system power.

The 1260-00C Non-Volatile Configuration Main Menu

When entering the NV configuration mode, the 1260-00C displays the menu shown in Figure 5-1.

1260 VXI Non-Volatile Configuration Main Menu

==

1. Read In Non-Volatile Configuration
2. Print Configuration Information
3. Change Configuration Information
4. Set Configuration to Factory Settings
5. Write Back (Save) Changes
6. Quit Configuration

Choice (1-6):

Figure 5-1, The 1260-00C Non-Volatile Configuration Main Menu

From the main menu, select the NV memory editing function and enter its number at the prompt. The effect of selecting each item is described below.

Read In Non-Volatile

`Read In Non-Volatile` Configuration reads the contents of

Configuration

the EEPROM into RAM.

`Print Configuration Information` displays the Non-Volatile Configuration Information from the RAM copy. Figure 5-2 shows an example of this display.

```
========Non-Volatile Configuration Information=========
Logical Address:     0x00   Device Type:   Message Based
Manufacturer ID:     0xFF6  Model Code:    0x0FF (Slot)
Slave Addr Spc:      A24    Protocol Reg:  0x0FF0
RESET Config:  PBtoLocalRESET pBtoSYSRESET SYSRESETtoLocalRESET

Serial Number:    0x00010003    User pROBE Pars:    0x000000(None)
Region 1 Size:           0x070000       Number Proc:         0x20
Number Exchgs:  0x20            Number Msgs:        0x180
Console:        Enabled
VXI Interrupt Level To Handler Logical Address (0xFF = free to assign):
  1:0xFF,   2:0xFF,   3:0xFF,   4:0xFF,   5:0xFF,   6:0xFF,   7:0xFF
A24 Assign Base:0x200000        A32 Assign Base:0x20000000
DC Starting LA:  0x01,BNO=YES  For FAILED Dev:  DO set Reset Bit
Servant Area:            0x00           GPIB Primary:        0x01
GPIB Addr Assgn:  Default   GPIB Flags:     MultSecond NAT4882 DMA
GPIB Addr Avoid  0x00000000
CI Block Base:   0x080000       CI Num Blocks:   0x00

--------Resident Code Instruments Locations--------

0x00:  00000000    0x01:  00000000    0x02:  00000000
0x03:  00000000    0x04:  00000000    0x05:  00000000
0x06:  00000000    0x07:  00000000    0x08:  00000000
0x09   00000000    0x0A:  00000000    0x0B:  00000000
--------CI Non-Volatile User Configuration Variables--------

 0x00:00000000          0x01:00000000  0x02:00000000    0x03:00000000
 0x04:00000000  0x05:00000000  0x06:00000000  0x07:00000000
 0x08:00000000  0x09:00000000  0x0A:00000000  0x0B:00000000
 0x0C:00000000  0x0D:00000000  0x0E:00000000  0x0F:00000000
 0x10:00000000  0x11;00000000  0x12:00000000  0x13:00000000
 0x14:00000000  0x15:00000000  0x16:00000000  0x17:00000000
 0x18:00000000  0x19:00000000  0x1A:00000000  0x1B:00000000
 0x1C:00000000  0x1D:00000000  0x1E:00000000  0x1F:00000000
```

Figure 5-2. The Non-Volatile Configuration Information Display

The first four Chapters display the Racal-Dana-reserved variables. The last Chapter displays hexadecimal values representing the contents of the user-defined variables. In this example, no user-defined variables have been initializd.

The following paragraphs contain descriptions of the fields in the Non-Volatile Configuration Information Display.

Logical Address

This field contains the VXI logical address of the 1260-00C. It specifies the location of the registers in VXI A16 space. The formula is as follows:

```
C000h + (40h * Logical Address)
```

If the Logical Address is set to 0, the 1260-00C will attempt to be the VXI Resource Manager. If the Logical Address is set in the range of 01 to FEh (1 through 254), the 1260-00C is set up to be a Static Configuration (SC) Message-Based (or possibly Register-Based) device. If the Logical Address is set to FFh (255), the 1260-00C is set up to be a Dynamic Configuration (DC) Message-Based (or possibly Register-Based) device. The factory setting is Logical Address 0.

Device Type

The 1260-00C can be set up either as a Message-Based or a Register-Based VXI device. Normally, the 1260-00C should be a Message-Based device. In Register-Based mode, the 1260-00C can reside virtually transparently in the VXI system for use as a debugging tool. It can still access the VXIbus directly as a bus master, perform Word Serial operations and GPIB transactions, and use Code Instruments. None of the functionality is removed. The Resource Manager does not grant any Servants to a Register-Based 1260-00C.

Manufacturer ID

The Manufacturer ID is set at the factory and cannot be changed. Manufacturer IDs are assigned by the VXI Consortium. The Manufacturer ID for Racal-Dana is 4091.

Model Code, Slot 0/Non-Slot 0

The 1260-00C can be configured for either Slot 0 or Non-Slot 0 operation. According to the VXIbus specification, a device configured to be in Slot 0 must have a Model Code between 000h and 0FFh. A device configured to be in a slot other than Slot 0 must have a Model Code greater than 0FFh. The 1260-00C Model Codes are assigned by Racal-Dana. This field is used only to configure which one of the two Codes to use. The factory setting is for Slot 0.

Slave Address Space

The 1260-00C can be configured to share 0%, 25%, 50%, or 100% of its onboard RAM with the VXIbus in either A24 or A32 address spaces. The percentage shared with the VXIbus is set via switches S6 and S8. See Table 3-4 in Chapter 3 Setting The Shared Memory for further information. The VXI address space to be shared with the local RAM is set with this field in non-volatile configuration mode. The factory setting is 0% dual-ported RAM.

Protocol Register

The 1260-00C can be configured to have a user-defined Protocol register. Only the FHS* and INT bits are not permitted to be active.

RESET Configuration

The 1260-00C has three configurable reset parameters. They can be enabled or disabled, and are as follows:

a. Pushbutton resets backplane (asserts SYSRESET* signal).

b. Pushbutton resets 1260-00C (asserts local reset signal).

c. Backplane SYSRESET* signal resets 1260-00C (SYSRESET* on backplane asserts local reset).

Serial Number

The serial number is a 32-bit quantity used to identify a particular 1260-00C. This value is set at the factory and cannot be altered.

User pROBE Parser

For developmental 1260-00Cs, install a parser to implement any commands needed for a custom test/debug environment within the pROBE environment. If the specified address is not 0 (no parser), use the following code to call the specified address:

```
long UserParser (inputline)

   char *inputline;
```

where `inputline` is a pointer to the typed line on the pROBE command line. The return value should be 1 if the command on the `inputline` was valid, and 0 if it was not valid.

PSOS Region 1 Size

pSOS Region 1 is the Dynamic Memory pool used for the majority of memory requirements of the 1260-00C. All process control blocks (PCBs), process stacks, queues, messages, GPIB buffers, etc., are allocated from this region. The first 64 kilobytes (0 to FFFFh) of any size onboard RAM configuration are reserved for use by Racal-Dana. Allocation of the rest can be made for Region 1, Code Instruments, or a device-dependent use. Region 1 always starts at local address 10000h. The minimum size is 60000h. The maximum size is the amount of RAM minus 10000h.

Number of pSOS Processes

This parameter is used to configure the maximum allowable number of pSOS processes. The 1260-00C requires a minimum of 16 processes. The factory setting is 32.

Number of pSOS Message Exchanges

This parameter is used to configure the maximum allowable number of pSOS message exchanges. The 1260-00C requires a minimum of 16 message exchanges. The factory setting is 32.

Number of pSOS Message Buffers

This parameter is used to configure the maximum allowable number of pSOS message buffers. The 1260-00C requires a minimum of 100h message buffers. The factory setting is 180h.

Console

This parameter is used to set the 1260-00C RS-232 local command console to default to enabled or disabled. The local command `ConsoleEna` is used to change the setting at runtime.

VXI Interrupt Level to Handler Logical Address

This is a table of logical addresses the Resource Manager can use during resource management of the VXI interrupt lines. If an interrupter is hard-configured (not a VXI programmable interrupter), place the logical address of the interrupt handler device in the corresponding level. If an interrupt handler is hard-configured (not a VXI programmable handler), place its logical address in the corresponding level to avoid conflicts with other programmable handlers, and also to permit the Resource Manager to assign programmable interrupters to the correct levels. If all interrupters and interrupt handlers are programmable, keep the value of FFh for all entries in the table.

As part of the hardware capabilities on the 1260-00C, there are three VXI programmable interrupt handlers. They can be assigned dynamically by the RAM or statically according to the contents of the non-volatile memory.

A24 Assign Base

This entry determines the A24 address where the Resource Manager will begin allocating A24 address space for VXI devices. This field can be used to avoid conflicts with VME devices using A24 address space. A bus master can access the range of address space a particular device is configured to occupy. The VXIbus specification requires A24 bus masters to **see** addresses from 200000h to DFFFFFh.

A32 Assign Base

This entry determines the A32 address where the Resource Manager will begin allocating A32 address space for VXI devices. This field is used to avoid conflicts with VME devices using A32 address space. A bus master can access the range of address space that a particular device is configured to occupy. The VXIbus specification requires A32 bus masters to **see** addresses from 20000000h to DFFFFFFFh.

DC Starting Logical Address

This parameter specifies the first logical address the Resource Manager uses to begin assigning Dynamic Configuration (DC) devices. DC devices will be assigned the next higher unassigned logical address.

BNO

This parameter specifies whether the Resource Manager should send *Identify Commander* and *Begin Normal Operation* in a DC system. DC systems cannot specify an intended hierarchy, and must be configured externally (normally through the local commands `DCGrantDev` and `DCBNOsend`). The most common configuration is to assign all DC devices to Logical Address 0 (the RM). If BNO is specified to be sent, all DC devices are assigned to Logical Address 0, and the *Identify Commander* and *Begin Normal Operation* commands are sent. If BNO is specified **not** to be sent, no devices (either SC or DC) will be sent the *Begin Normal Operation* command. `DCBNOsend` must be sent to the local command set to initiate normal operation after the hierarchy is established.

For FAILED Device

The VXIbus specification requires Commanders to Sysfail-Inhibit a Servant device that has failed (asserted the SYSFAIL* line and the Passed bit in its Status register). The specification permits Commanders to also set the reset bit of a failed device. This parameter specifies which method to use.

Servant Area This parameter specifies what Servant area value to return to the Resource Manager during a Word Serial *Read Servant Area* query. This applies only when the 1260-00C is **not** Resource Manager.

GPIB Primary This parameter specifies the GPIB primary address of the 1260-00C to be used when in multiple secondary addressing mode.

GPIB Address Assignment Method This parameter specifies what method to use to configure the GPIB address of the 1260-00C. The choices are as follows:

a. Default:

0 for multiple secondary addressing

1 for multiple primary addressing

b. Always a particular GPIB address

c. No GPIB address

GPIB Flags

MultPrimary: Multiple primary addressing mode is set.

MultSecond: Multiple secondary addressing mode is set.

Others: These flags are for information purposes only; do not modify.

GPIB Address to Avoid This is a 32-bit bit mask of GPIB addresses to avoid during address assignment for either multiple primary or multiple secondary addressing modes.

Code Instrument Block Base This parameter specifies the local 1260-00C address base for Static Code Instruments.

Code Instrument Number of RAM Blocks This parameter specifies the number of 4-kilobyte RAM blocks allocated from the block base for use by Static Code Instruments.

Resident Code Instrument Locations These parameters specify the base addresses of EPROMed Code Instruments. These Code Instruments are automatically started up after the Resource Manager operations complete.

Code Instrument Non-Volatile User Configuration Variables

These parameters are completely user-defined, and can be used for any purpose.

Change Configuration Information

`Change Configuration Information` displays the 1260-00C Non-Volatile Configuration Changer as shown in Figure 5-3.

```
1260-00C Non-Volatile Configuration Changer

===========================================

0). Edit Local Register Configuration
1). Edit pSOS Configuration
2). Edit VXI Interrupt Handler Logical Address
3). Edit Resource Manager A24/A32 Assign Bases
4). Edit Servant Area and DC Configuration
5). Edit FAILED Device Handling Mode
6). Edit GPIB Configuration
7). Edit Default CI Configuration
8). Edit Resident CI Base Locations
9). Edit CI User Configuration Variables
Q). Quit Editor

    Choice (0-9,Q):
```

Figure 5-3. The 1260-00C Non-Volatile Configuration Changer

Edit the Racal-Dana-reserved configuration parameters and CI user configuration variables by selecting the corresponding menu item. In each case, you are prompted to enter constants for the new values, with default values supplied where appropriate. For the pSOS configuration parameters, the 1260-00C prints a formula for calculating an appropriate value for each parameter when typing in 0 in response to the prompt requesting the value.

The Default CI Configuration and Resident CI Base Locations options are only important when installing a Resident CI. Refer to Appendix B, <u>Using The DMAmove and CDS-852 Adapter Code Instruments</u>, for instructions on installing the Resident CIs.

NOTE

The `Change Configuration Information` editor modifies only the RAM copy of the NV memory contents. The NV memory must be updated with the `Write Back`

`(Save)` Changes command in the main menu to retain the changes after the 1260-00C has been reset or powered-down.

Select `Quit Editor` to return the display to the main menu.

Set Configuration to Factory Settings

`Set Configuration To Factory Settings` sets the contents of the RAM copy of the NV memory to the default (original) factory settings. Note that only the RAM copy is affected. You must use the `Write Back (Save) Changes` command in the main menu to write back the NV memory, and retain the changes after the 1260-00C has been reset or powered-down.

Write Back (Save) Changes

`Write Back (Save) Changes` writes the modified copy of the NV memory back to the EEPROM. The write-back procedure takes five to ten seconds.

Quit Configuration

`Quit Configuration` prompts selection of a different start-up configuration, or re-enters pROBE depending upon how the non-volatile configuration mode was entered.

Chapter 6

DIAGNOSTIC TESTS

Introduction

This section contains information for executing the 1260-00C diagnostic self-tests. The diagnostics test each 1260-00C subcircuit, and are useful in detecting and isolating problems.

The diagnostics mode can be entered through any of the following methods.

a. Set the start-up mode switches to the diagnostics mode as described in Section 3 Start-Up Mode Configuration. Set switch S19 to the OFF position, and set switch S20 to the ON position. Restart the system.

b. In VXI pROBE mode (on development modules only), enter the diagnostic mode through the pROBE `DIAG` command.

c. In 488-VXI runtime system mode, enter the diagnostic mode through the `DIAG` command.

The diagnostic commands must be executed from the RS-232 port.

Configuration for Diagnostic Testing

The diagnostic tests require the 1260-00C to be disconnected from all other GPIB devices to prevent interference with the GPIB tests.

Diagnostic Test Structure

A total of 263 diagnostic routines, or tests, are organized in groups as shown in Table 6-1. Each test is composed of one or more subroutines called *commands*.

Each test is designed to functionally test a specific part of the 1260-00C circuitry. The diagnostics can be executed by test groups or by individual tests.

Table 6-1. Diagnostic Tests

		Test Numbers	
Test Name	**Group Number**	**From**	**To**
RAM	1	1	4
68070 CPU	2	5	21
MIGA	3	22	44
GPIB	4	45	132
TIC	5	133	236
DMA	6	237	247
68881 Co-processor	7	248	248
RAM (exhaustive)	8	249	250
Interrupts	9	251	253
Miscellaneous Tests	10	254	263

Diagnostics Mode Selection

Two hierarchical levels of menus control execution of the diagnostic tests. The highest level menu is the Diagnostics Mode menu used to select whether to execute a test group or tests, and the mode in which to run them. The Diagnostics Mode menu is shown in Figure 6-1 and described in Table 6-2.

```
1260-00C DIAGNOSTICS:    < XXX DRAM Reported >

            Default Diags (all)         ===> d
            Tests                       ===> t
            Test Groups                 ===> g
            Over Night Loop             ===> o
            Quit                        ===> q
      PRINT TOGGLE                      ===> p
      SINGLE STEP TOGGLE                ===> s
      LOOPING TOGGLE                    ===> l
      ERROR REPORT TOGGLE               ===> e
      REPORT ERROR LOG                  ===> r
      CLEAR ERROR LOG                   ===> c
      (Current settings:  PRINT(OFF), ERROR(ON), SINGLE(OFF), LOOP(OFF)

            Enter Selection:
```

Figure 6-1. The Diagnostics Mode Menu

Table 6-2. Diagnostics Mode Menu Option Descriptions

Selection	Description
Default Diags (all)	Runs all the tests.
Tests	Presents a menu of tests to be selected.
Test Groups	Presents a menu of test groups to be selected.
Over Night Loop	Runs selected tests continuously. Only stops when an error occurs or when the system is reset.
Print Toggle	Turn printing of test groups/tests on or off.
Single Step Toggle	Turns single-stepping of test groups/test on or off.
Looping Toggle	Turns looping of selected test groups/tests on or off.
Error Report Toggle	Turns printing of error statements on or off.
Report Error Log	Prints the first 11 errors that occurred.
Clear Error Log	Erases the buffer of errors.

By default, single-stepping, looping, and test message printing are turned off while error reporting is turned on. The selected diagnostics run uninterrupted until they complete or until an error occurs. If an error has occurred, an error message is printed to the screen. The message displays the test number, group number, value expected, and value received. Please contact Racal-Dana for further interpretation of diagnostic error messages.

Suppress the error reporting with the e command. With error reporting turned off, selected tests run to completion without being interrupted by error messages. The 1260-00C indicates if any errors have occurred after all tests selected have completed. To view the errors, select r to display the error log and c to clear the error log.

Single Step Toggle is used to pinpoint problems. Select s to toggle this command on or off. With this feature, each access to memory or to a register is reported on the screen. The 1260-00C waits for a key to be pressed before performing the displayed step.

Diagnostic Test Selection

If the Default Diags (all) option or the Over Night Loop option is selected, the appropriate tests begin immediately. The default option performs all of the tests, while the Over Night Loop option performs all except the interactive tests and tests that drive signals on the VXIbus. When either the Tests or the Test Groups option is selected, a new menu appears from which you can select any or all tests or test groups. Figure 6-2 shows the Test Selection menu. The Test Group Selection menu is very similar.

Group Name	Group NUM	Test Numbers From - To
RAM	1	1 - 4
68070 - I2C	2	5 - 7
68070 - UART	2	8 - 20
68070 - TIMER	2	19 - 21
MIGA	3	22 - 24
GPIB - NAT4882	4	45 - 99
GPIB - TURBO 488	4	100 - 132
TIC	5	133 - 236
DMA	6	237 - 247
MC68881	7	248 - 248
RAM(exhaustive)	8	249 - 250
INTERRUPTS	9	251 - 253
MISC TESTS	10	254 - 263

ENTER TEST NUMBERS

e.g.: 15,30,31,40-80,99,*(all)
Hit "q" to quit

Enter:

Figure 6-2. The Test Selection Menu

When all diagnostic tests have been completed, select q to quit and exit diagnostics mode. If the diagnostics had been entered from pROBE, quitting would return you to pROBE. Otherwise, the 1260-00C gives a prompt to reboot the system in a different start-up mode.

Diagnostic Test Groups

Group 1-RAM

This group tests the RAM to ensure the CPU can correctly read and write from RAM addresses. Table 6-3 gives the test numbers and names of the RAM tests.

Table 6-3. RAM Tests

Test Number	Test Description
1	Data Path Test
2	Self-Test Cell Test (not exhaustive)
3	Cell Test
4	Read and Modify Write Test

Group 2-68070 CPU

This group tests the 68070 I^2C interface, UART interface, and timers. Tests 5 through 7 test the 68070 I^2C interface; Tests 8 through 18 test the UART interface; and Tests 19 through 21 test the timers. Table 6-4 gives the test numbers and names of the 68070 CPU tests.

Table 6-4. 68070 CPU Tests

Test Number	Test Description
5	I^2C Interface Test - Maximum Clock Frequency
6	I^2C Interface Test - Minimum Clock Frequency
7	I^2C Interface Test - Interrupt Trigger
8	Test Baud Rate 75
9	Test Baud Rate 150
10	Test Baud Rate 300
11	Test Baud Rate 1200
12	Test Baud Rate 2400
13	Test Baud Rate 4800
14	Test Baud Rate 9600
15	Test Baud Rate 19200
16	Baud = 9600; test odd parity with two stop bits
17	Baud = 9600: test even parity with two stop bits
18	Baud = 9600; test interrupts
19	Test Timer 0 interrupt capability
20	Test Timer 1 matched mode
21	Test Timer 2 matched mode

Group 3-MIGA

This group tests the MIGA (gate array) registers. The MIGA contains the VXI registers as defined for Message-Based devices. Table 6-5 gives the test numbers and names of the MIGA tests.

Table 6-5. MIGA Tests

Test Number	Test Description
22	Logical Address Test
23	ID Test
24	Device Type Test
25	Offset Test
26	Protocol Test
27	A24 Pointer High Test
28	A24 Pointer Low Test
29	A32 Pointer High Test
30	A32 Pointer Low Test
31	Data Extended Test
32	Data High (Device) Test
33	Data Low (Device) Test
34	Data High (Local) Test
35	Data Low (Local) Test
36	Status Test
37	Control Test
38	Response Test
39	ICR & ISR Test
40	I/O Test
41	Signal Test
42	Interrupts Test
43	Word Serial Protocol Test
44	SYSFAIL Circuitry Test

Group 4-GPIB

This group tests the NAT4882 and Turbo488 GPIB interface IC's. Tests 45 through 99 test the NAT4882, and Tests 100 through 132 test the Turbo488. Table 6-6 gives the test numbers and names of the GPIB tests.

Table 6-6. GPIB Tests

Test Number	Test Description
45	INIT
46	Presence test using ADSR
47	Check SPMR and SPSR
48	Check address register bits
49	Check can be listener
50	Check can be talker
51	Check can listen to all 32 listen addresses
52	Check can be unaddressed as listener
53	Check can talk to all 32 talk addresses
54	Check can be unaddressed as talker
55	Check can listen to all 960 external addresses
56	Check can be unaddressed as external listener
57	Check can talk to all 960 external addresses
58	Check can be unaddressed as external talker
59	Check can recognize DI and HLDA bits
60	Check can recognize ERR
61	Check can recognize DCL command
62	Check can recognize SDC
63	Check can set END bit on EOI
64	Check can set EOI bit on EOI

Table 6-6. GPIB Tests (continued)

Test Number	Test Description
65	Check can set END on 8-bit EOS
66	Check can set END on 7-bit EOS
67	Check can recognize GET command
68	Check set APT on unrecognized command
69	Check can recognize undefined command
70	Check can set REM, REMC, LOK, LOKC bits
71	Check can clear REM and LOK bits
72	Check can set SRQI
73	Check can do serial poll
74	Check can do parallel poll
75	Check DHADT
76	Check DHADC
77	Check DHATA
78	Check DHALA
79	Check DHUNTL
80	Check NTNL
81	Check NTNL with ATN asserted
82	Check RPP
83	Check CHES
84	Check PP2
85	Check SDB
86	Check NL
87	Check EOS

Table 6-6. GPIB Tests (continued)

Test Number	Test Description
88	Check 9914 and 7210 mode switch
89	Check able to untalk
90	Check able to unlisten
91	Check NBAF and NTNL
92	Check REQTC
93	Check REQFC
94	Check holdoff now command
95	Check DHALL
96	Check effect of REQT during serial poll
97	Check for spurious interrupts
98	Check INT on SYNC
99	Check global interrupt
100	Set and clear SC
101	Set and clear DUALADD
102	Trigger INTSCR1 and INTSRC2
103	Set STOP DONE HALT and DAV in read mode
104	Verify set of STS1, ISR3 bits with 8-bit read
105	Verify write mode and TLCINT set by error
106	Read/write CNTL, CNTH registers
107	Verify bits in IMR3 register
108	Reset ISR3
109	Reset ISR3 and STS1
110	Reset STS2

Table 6-6. GPIB Tests (continued)

Test Number	Test Description
111	Reset TIMER
112	Check flags on 16-bit independent FIFO
113	Fill and empty 16-bit independent FIFO
114	Fill and empty 16-bit FIFO
115	Reset non-full FIFO
116	Reset full FIFO
117	16-bit FIFO read
118	Fill and empty 16-bit FIFO
119	STOP and HALT in 16-bit A-1st mode
120	Holdoff when FIFO and DIR are full
121	HALT and EOI when last byte in FIFO
122	Enable carry cycle
123	Disable carry cycle
124	16-bit FIFO write
125	Fill and empty 16-bit FIFO
126	Carry cycle with EOI
127	Halt on ERROR
128	Interrupt on DONE
129	GPIB-> MEMORY DMA
130	GPIB-> MEMORY DMA with EOI
131	MEMORY-> GPIB DMA
132	MEMORY-> GPIB DMA, commands

Group 5-TIC

This group tests the TIC ASIC. The TIC, an ASIC designed by Racal-Dana, handles the TTL/ECL trigger interface and CLK10 conversion. Table 6-7 gives the test numbers and names of the TIC tests.

Table 6-7. TIC Tests

Test Number	Test Description
133	Initialization
134	Register initialization
135	CNTH Register
136	CNTL Register
137	TTCR and TTSR Registers
138	ETCR and ETSR Registers
139	MODIDH and MODIDL Registers
140	PGP0 and PGP1 Registers
141	TSR0 and TOR0 Registers
142	TSR1 and TOR1 Registers
143	TSR2 and TOR2 Registers
144	TSR3 and TOR3 Registers
145	TSR4 and TOR4 Registers
146	TSR5 and TOR5 Registers
147	TSR6 and TOR6 Registers
148	TSR7 and TOR7 Registers
149	TSR8 and TOR8 Registers
150	TSR9 and TOR9 Registers
151	GPIN0 connection
152	GPIN1 connection

Table 6-7. TIC Tests (continued)

Test Number	Test Description
153	GPIN2 connection
154	GPIN3 connection
155	GPIN4 connection
156	GPIN5 connection
157	GPIN6 connection
158	GPIN7 connection
159	GPIN8 connection
160	GPIN9 connection
161	Trig0 connection
162	Trig1 connection
163	Trig2 connection
164	Trig3 connection
165	Trig4 connection
166	Trig5 connection
167	Trig6 connection
168	Trig7 connection
169	Trig8 connection
170	Trig9 connection
171	Counter using CLK10
172	Counter using Trig0
173	Counter using Trig1
174	Counter using Trig2
175	Counter using Trig3

Table 6-7. TIC Tests (continued)

Test Number	Test Description
176	Counter using Trig4
177	Counter using Trig5
178	Counter using Trig6
179	Counter using Trig7
180	Counter using Trig8
181	Counter using Trig9
182	Counter using EXT CLK
183	Interrupt on Trig0 by ASTS and USTS
184	Interrupt on Trig1 by ASTS and USTS
185	Interrupt on Trig2 by ASTS and USTS
186	Interrupt on Trig3 by ASTS and USTS
187	Interrupt on Trig4 by ASTS and USTS
188	Interrupt on Trig5 by ASTS and USTS
189	Interrupt on Trig6 by ASTS and USTS
190	Interrupt on Trig7 by ASTS and USTS
191	Interrupt on Trig8 by ASTS and USTS
192	Interrupt on Trig9 by ASTS and USTS
193	Interrupt on counter count down on CLK10
194	Interrupt on counter count down on EXTCLK
195	Interrupt AOVER, UOVER, and PSOVER on Trig0
196	Interrupt AOVER, UOVER, and PSOVER on Trig1
197	Interrupt AOVER, UOVER, and PSOVER on Trig 2
198	Interrupt AOVER, UOVER, and PSOVER on Trig3

Table 6-7. TIC Tests (continued)

Test Number	Test Description
199	Interrupt AOVER, UOVER, and PSOVER on Trig4
200	Interrupt AOVER, UOVER, and PSOVER on Trig5
201	Interrupt AOVER, UOVER, and PSOVER on Trig6
202	Interrupt AOVER, UOVER, and PSOVER on Trig7
203	Interrupt AOVER, UOVER, and PSOVER on Trig8
204	Interrupt AOVER, UOVER, and PSOVER on Trig9
205	Interrupt from scalar
206	Software Semi-Sync accept on Trig0
207	Software Semi-Sync accept on Trig1
208	Software Semi-Sync accept on Trig2
209	Software Semi-Sync accept on Trig3
210	Software Semi-Sync accept on Trig4
211	Software Semi-Sync accept on Trig5
212	Software Semi-Sync accept on Trig6
213	Software Semi-Sync accept on Trig7
214	Software Semi-Sync accept on Trig8
215	Software Semi-Sync accept on Trig9
216	Hardware Semi-Sync and Automatic ACK on Trig0
217	Hardware Semi-Sync and Automatic ACK on Trig1
218	Hardware Semi-Sync and Automatic ACK on Trig2
219	Hardware Semi-Sync and Automatic ACK on Trig3
220	Hardware Semi-Sync and Automatic ACK on Trig4
221	Hardware Semi-Sync and Automatic ACK on Trig5

Table 6-7. TIC Tests (continued)

Test Number	Test Description
222	Hardware Semi-Sync and Automatic ACK on Trig6
223	Hardware Semi-Sync and Automatic ACK on Trig7
224	Hardware Semi-Sync and Automatic ACK on Trig8
225	Hardware Semi-Sync and Automatic ACK on Trig9
226	Automatic Semi-Sync source
227	Sync triggers with no conditioning
228	Sync triggers with synchronously
229	Pulse stretch synchronous with EXT CLK
230	Pulse stretch with 1CLK synchronous
231	Pulse stretch asynchronously
232	Scalar values 1 through 32 with EXT CLK
233	Scalar value 2 with all GPIO lines
234	Scalar value 0x0f using CLK 10, NOROLL1, and INT
235	TRIGIN and TRIGOUT on front panel
236	TRIGIN and TRIGOUT by zig zag of all triggers

Group 6-DMA

This group tests the DMA Channel 2 and memory-to-memory DMA transfers. Table 6-8 gives the test numbers and names of the DMA tests.

Table 6-8. DMA Tests

Test Number	Test Description
237	Poll test (burst, bytes, mem-to-dev, from even addresses)
238	Poll test (cycle steal, bytes, mem-to-dev, even addresses)
239	Poll test (burst, words, mem-to-dev, from even addresses)
240	Poll test (cycle steal, words, mem-to-dev, even addresses)
241	Poll test (burst, bytes, mem-to-dev, from odd addresses)
242	Poll test (cycle steal, bytes, mem-to-dev, odd addresses)
243	Poll test (burst, bytes, dev-to-mem, from even addresses)
244	Poll test (burst, words, dev-to-men, from even addresses)
245	DMA interrupt test
246	Minimum functionality test
247	Maximum transfer test

Group 7-68881 Co-processor

This is a test of the numeric co-processor operation. If the 68881 is not installed, the 1260-00C skips this test. Table 6-9 gives the test number and name of the 68881 Co-processor test.

Table 6-9. 68881 Co-Processor Test

Test Number	Test Description
248	Test floating point co-processor

Group 8-RAM (Exhaustive)

This exhaustive RAM test checks the entire onboard RAM and the address and data paths. Table 6-10 gives the test numbers and names of the exhaustive RAM tests.

Table 6-10. RAM (Exhaustive) Tests

Test Number	Test Description
249	Data path test (exhaustive)
250	Address path and cell test (exhaustive)

Group 9-Interrupts

This group tests the GPIB and MIGA local interrupts. Table 6-11 gives the test numbers and names of the Interrupt tests.

Table 6-11. Interrupt Tests

Test Number	Test Description
251	Verify interrupt on INT2N using SYSFAIL
252	Verify SYSFAIL disable interrupt
253	Verify interrupt on INT1N using done int

Group 10-Miscellaneous Tests

This group tests the EPROM, EEPROM, Sanity Timer, LEDs, MODID register, Local Bus timeouts, and VXI bus timeout. Table 6-12 gives the test numbers and names of the Miscellaneous tests.

Table 6-12. Miscellaneous Tests

Test Number	Test Description
254	LED test: SYSFAIL and FAILED LED
255	LED test: ACCESS LED
256	LED test: TEST LED
257	LED test: ONLINE LED
258	Switch test: Start-up switches (S10,S11,S12)
259	Local BTO test: Test local bus timeout unit
260	VXI BTO test: Test VXI bus timeout unit
261	Sanity timer test: Test enabled/disabled
262	EPROM checksum test
263	EEPROM stamp and checksum test

This page was left intentionally blank.

Chapter 7

PRODUCT SUPPORT

Product Support

Racal Instruments has a complete Service and Parts Department. If you need technical assistance or should it be necessary to return your product for repair or calibration, call 1-800-722-3262. If parts are required to repair the product at your facility, call 1-949-859-8999 and ask for the Parts Department.

When sending your instrument in for repair, complete the form in the back of this manual.

For worldwide support and the office closes to your facility, refer to the Support Offices section on the following page.

Reshipment Instructions

Use the original packing material when returning the 1260-00C to Racal Instruments for calibration or servicing. The original shipping crate and associated packaging material will provide the necessary protection for safe reshipment.

If the original packing material is unavailable, contact Racal Instruments Customer Service for information.

Support Offices

Racal Instruments, Inc.
4 Goodyear St., Irvine, CA 92618-2002
Tel: (800) 722-3262, FAX: (949) 859-7309

Racal Instruments, Ltd.
480 Bath Road, Slough, Berkshire, SL1 6BE, United Kingdom
Tel: +44 (0) 8706 080134; FAX: +44 (0) 1753 791290

Racal Systems Electronique S.A.
18 Avenue Dutartre, 78150 LeChesnay, France
Tel: +33 (1) 3923 2222; FAX: +33 (1) 3923 2225

Racal Systems Elettronica s.r.l.
Strada 2-Palazzo C4, 20090 Milanofiori Assago, Milan, Italy
Tel: +39 (02) 5750 1796; FAX +39 (02) 5750 1828

Racal Elektronik System GmbH.
Frankenforster Strasse 21, 51427 Bergisch Gladbach, Germany
Tel:+49 2204 92220; FAX: +49 2204 21491

Racal Australia Pty. Ltd.
3 Powells Road, Brookvale, NSW 2100, Australia
Tel: +61 (2) 9936 7000, FAX: +61 (2) 9936 7036

Racal Electronics Pte. Ltd.
26 Ayer Rajah Crescent, 04-06/07 Ayer Rajah Industrial Estate, Singapore 0513.
Tel: +65 7792200, FAX: +65 7785400

Racal Instruments, Ltd.
Unit 5, 25F., Mega Trade Center, No 1, Mei Wan Road, Tsuen Wan, Hong Kong, PRC
Tel: +852 2405 5500, FAX: +852 2416 4335