EPConnect/VXI

for DOS & Windows

User's Guide

VOL 1 of 3

RadiSys® Corporation

15025 S.W. Koll Parkway

Beaverton, OR 97006

Phone: (503) 646-1800

FAX: (503) 646-1850

http://www.radisys.com

RadiSys® CORPORATION

07-0156-02

December 1994

January 1993

Software License and Warranty

YOU SHOULD CAREFULLY READ THE FOLLOWING TERMS AND CONDITIONS BEFORE OPENING THE DISKETTE OR DISK UNIT PACKAGE. BY OPENING THE PACKAGE, YOU INDICATE THAT YOU ACCEPT THESE TERMS AND CONDITIONS. IF YOU DO NOT AGREE WITH THESE TERMS AND CONDITIONS, YOU SHOULD PROMPTLY RETURN THE UNOPENED PACKAGE, AND YOU WILL BE REFUNDED.

LICENSE

You may:

1. Use the product on a single computer;
2. Copy the product into any machine-readable or printed form for backup or modification purposes in support of your use of the product on a single computer;
3. Modify the product or merge it into another program for your use on the single computer—any portion of this product merged into another program will continue to be subject to the terms and conditions of this agreement;
4. Transfer the product and license to another party if the other party agrees to accept the terms and conditions of this agreement—if you transfer the product, you must at the same time either transfer all copies whether in printed or machine-readable form to the same party or destroy any copy not transferred, including all modified versions and portions of the product contained in or merged into other programs.

You must reproduce and include the copyright notice on any copy, modification, or portion merged into another program.

YOU MAY NOT USE, COPY, MODIFY, OR TRANSFER THE PRODUCT OR ANY COPY, MODIFICATION, OR MERGED PORTION, IN WHOLE OR IN PART, EXCEPT AS EXPRESSLY PROVIDED FOR IN THIS LICENSE.

IF YOU TRANSFER POSSESSION OF ANY COPY, MODIFICATION, OR MERGED PORTION OF THE PRODUCT TO ANOTHER PARTY, YOUR LICENSE IS AUTOMATICALLY TERMINATED.

TERM

The license is effective until terminated. You may terminate it at any time by destroying the product and all copies, modifications, and merged portions in any form. The license will also terminate upon conditions set forth elsewhere in this agreement or if you fail to comply with any of the terms or conditions of this agreement. You agree upon such termination to destroy the product and all copies, modifications, and merged portions in any form.

LIMITED WARRANTY

RadiSys Corporation ("RadiSys") warrants that the product will perform in substantial compliance with the documentation provided. However, RadiSys does not warrant that the functions contained in the product will meet your requirements or that the operation of the product will be uninterrupted or error-free.

RadiSys warrants the diskette(s) on which the product is furnished to be free of defects in materials and workmanship under normal use for a period of ninety (90) days from the date of shipment to you.

LIMITATIONS OF REMEDIES

RadiSys' entire liability shall be the replacement of any diskette that does not meet RadiSys' limited warranty (above) and that is returned to RadiSys.

IN NO EVENT WILL RADISYS BE LIABLE FOR ANY DAMAGES, INCLUDING LOST PROFITS OR SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF OR INABILITY TO USE THE PRODUCT EVEN IF RADISYS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

GENERAL

You may not sublicense the product or assign or transfer the license, except as expressly provided for in this agreement. Any attempt to otherwise sublicense, assign, or transfer any of the rights, duties, or obligations hereunder is void.

This agreement will be governed by the laws of the state of Oregon.

If you have any questions regarding this agreement, please contact RadiSys by writing to RadiSys Corporation, 15025 SW Koll Parkway, Beaverton, Oregon 97006.

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT, AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU FURTHER AGREE THAT IT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN US WHICH SUPERSEDES ANY PROPOSAL OR PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATION BETWEEN US RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

NOTES

Table of Contents

1. Introducing EPConnect/VXI for DOS and Windows

This manual introduces you to EPConnect™/VXI for DOS and Windows software (referred to as EPConnect in this manual). Please read this manual first before continuing to the related programmer's reference manuals. The following manuals comprise the entire EPConnect manual set:

EPConnect/VXI for DOS and Windows User's Guide (this manual)

Bus Management for DOS Programmer's Reference Manual

SICL for DOS Programmer's Reference Manual

Bus Management for Windows Programmer's Reference Manual

SICL for Windows Programmer's Reference Manual

This manual is intended for programmers using the EPConnect programming interface to develop DOS and Windows programs that control VXI I/O modules via the VXI expansion interface on an EPC®.

The Bus Management Library and SICL are the application programming interfaces (APIs) that are part of EPConnect. You are expected to read this manual for an understanding of what is in EPConnect, to learn the terms and conventions used in this manual set, and to learn how to install and configure the Bus Management library for use on your system. You are not expected to have in-depth knowledge of DOS or Windows.

EPConnect provides a powerful set of tools for interacting with the VXIbus. RadiSys offers considerable flexibility to users by supplying interfaces for several high-level languages. By observing the MS Pascal binding conventions, you can use EPConnect with these languages.

Chapter 1 introduces you to the RadiSys EPConnect environments. In it you will find the following:

- What is in this manual and how to use it
- Notational conventions
- Terms and definitions
- What is EPConnect?
- Related documentation

1.1 How This Manual is Organized

This manual has six chapters and three appendices:

Chapter 1, *Introduction*, introduces EPConnect/VXI and this manual.

Chapter 2, *Installation and Configuration*, describes how to install and configure EPConnect software for both DOS and Windows users and for both Bus Management and SICL APIs.

Chapter 3, *Using the Start-up Resource Manager*, describes the operation of the Start-up Resource Manager (SURM) program. It provides resource manager capabilities that configure the devices in the system at power-up.

Chapter 4, *Using the VXI Configurator*, describes the operation of the VXI Configurator program. The VXI Configurator is a Windows program that establishes system device names, addresses, and other device parameters.

Chapter 5, *Using the BusMonitor*, describes the operation of the BusMonitor program. BusMonitor is a Windows program that displays the state of various VXIbus backplane signals.

Chapter 6, *Using BusProbe*, describes the operation of the BusProbe program. BusProbe is a Windows program that allows you to directly control the VXIbus and EPC hardware and to interact with system devices.

Appendix A, *SURM Error Messages*, contains a listing of error messages generated by the SURM.

Appendix B, ***DEVICES** File*, contains information about the **DEVICES** file records and gives sample **DEVICES** file entries.

Appendix C, *Configuring Multi-Mainframe Systems*, contains information about configuring multi-mainframe hardware and assigning ULAs to devices in multi-mainframe systems.

1.2 Notational Conventions

The EPConnect manuals in this set use the following typographic conventions:

Example of Convention	Description
iopen, int, char	Bold type indicates EPConnect function names, operators, and keywords. Within syntax, bold type indicates that the text must be entered exactly as shown.
id, format, timeout	Lower case words in italics indicate parameter or variable names.
...\	Three dots in a path tell you that the full path is abbreviated.
`#include "sicl.h"` `void main(void)` `INST instance;` `char FAR *vxiregisters;`	This font is used for C code examples.

[ ]	Square brackets in text or function command lines enclose an optional variable.
[ ]*	Square brackets in text or function command lines followed by an asterisk represents zero or more occurrances of the enclosed text.
`if (inst == NULL) {...}`	A column or row of three dots tells you that some of the code is intentionally omitted.
left | right	Vertical bar indicates a choice between the text on the left or the right.
EPCVXI.386	Upper case bold letters indicate file names, segment names, and terms used at the operating system command level.
I_ERR_NOERROR	Upper case bold words indicate constants when they appear in text.
"string"	Quotation marks required by the language.

1.3 Terms and Definitions

The Bus Management and SICL manuals in this set use the following terms and definitions:

Address String	A character string that uniquely identifies a particular interface (VXIbus or GPIB) or a device on an interface.
ANSI	American National Standards Institute.

Bus	A VXIbus mainframe or interconnect bus.
Bus Error	The inability to access a given VXIbus address either because no register or memory exists at the given address or the register or memory at that address does not respond.
Bus Error Handler	Software that executes when a bus error occurs.
Commander Session	A session between a device running SICL and its commander.
Controller	The computer that controls data and/or command communications. When communications occur between a controller and a device, the controller directs the flow of the communications.
Controller role	A computer that acts as a controller communicating with a device.
Device	An instrument that receives and executes commands.
Device Driver	Software that executes a protocol for communicating with a device or interface. This software may communicate directly with a device by reading and writing registers or it may communicate with an interface driver that reads and writes registers.
Device Session	An instance of communications between a controller and a device.
EOI	GPIB signal that indicates either the last byte of a data message (END) or the parallel poll Identify (IDY) message.
EOS byte	A 7- or 8-bit end-of-string character that is sent as the last byte of a data message.
EPC	The RadiSys family of Embedded Personal Computers for VXIbus and VMEbus systems.

GPIB	General Purpose Instrumentation Bus; the bus described in IEEE Standard 488. Also known as HPIB.
Handler	Software that executes when an asynchronous event occurs (bus error, interrupt, or SRQ).
Interface	The connection between controllers and devices. It includes a communications protocol and the hardware required to support the communications protocol. Examples of interfaces are GPIB and VXI (VXI can also be considered a bus, but in this document it is considered an interface).
Interface Driver	Software that communicates with an interface.
Interface Session	An instance of communications that affects an interface.
Interrupt	An asynchronous event caused by a VXIbus hardware interrupt that requires attention out of the normal program flow.
Local	On the bus under consideration.
LAN	Local Area Network; a group of computers within a limited geographic area, interconnected for communication without aid of a central computer.
Lock	A state that prohibits process access (other than the process that performed the lock) to a device or interface.
Mainframe extender	A device that provides transparent communications between devices in separate VXI mainframes.
Mapping	An operation that returns a pointer to a specified range of an address space and makes that range of addresses accessible to a process.

Message-based	A VXIbus device that contains communication registers that are accessible to other modules in the system, as well as, the configuration registers defined for register based devices. Each device in the system can then use specific communication protocols such as VXIbus Word Serial Protocol to communicate with other devices. See the VXIbus Specification for details of the configuration registers, VXIbus Word Serial Protocol, and Message Based Devices.
Node	VXIbus mainframe or stand-alone device.
Non-Controller Role	A computer acting as a device communicating with a controller.
OLRM	On-Line Resource Manager, part of the RadiSys implementation of the Resource Manager functions described in the VXIbus Specification.
Parallel poll	The process of polling all configured GPIB devices on the interface and reading a composite response.
Process	An operating system object containing one or more threads of execution that share a data space.
Register	A memory location that contains a value representing the hardware state. Also, a memory location that can be written into to change the hardware state.
Register-based	A VXIbus device that contains a set of configuration registers as defined by the VXIbus specification. Communication with a register based device is usually accomplished via read and writes of its device dependent registers. See the VXIbus specification for details of the configuration registers and Register Based Devices.
Remote	On a bus under consideration.
Root bus	The bus to which the device at Logical Address 0 is connected.

Serial poll	The process of polling one device at a time and reading back its status byte.
Session	An instance of communication with a device or interface.
SRQ	A service request. An asynchronous request from a remote device indicating that the device requires service; essentially an interrupt from a non-controller device.
Status Byte	A byte of information that shows the current state and status of the device.
SURM	Start-Up Resource Manager; part of the RadiSys implementation of the Resource Manager functions described in the VXIbus Specification.
Thread	A unit of execution within a program or process.
VXI	VMEbus eXtension for Instrumentation. A test and measurement industry standard for high-performance, modular instrument systems.
VXIbus commander	A message-based device that is a bus master and can control one or more servants.
VXIbus servant	A device that is controlled by exactly one commander.

1.4 What is EPConnect/VXI for DOS & Windows?

EPConnect is a software environment that reduces the time and effort required to develop and integrate VXIbus systems based on a RadiSys Embedded PC (EPC®). EPConnect is a layered architecture (see Figures 1-1 and Figure 1-2) that integrates a number of software packages with the underlying DOS and Windows operating systems.

The primary goal of EPConnect is to provide the software developer with an easy to use software interface to the EPC hardware. The interface is provided via the Bus Management library and the Standard Instrument Control Library (SICL). These two libraries handle the low level details of the underlying hardware.

Supporting the runtime library are two interactive debugging tools (BusProbe and BusMonitor), a Startup Resource Manager (SURM), and a system configuration manager (VXI Configurator). BusManager is the EPConnect device driver that provides a low level programming interface to the VXIbus hardware and coordinates all bus activity for user applications.

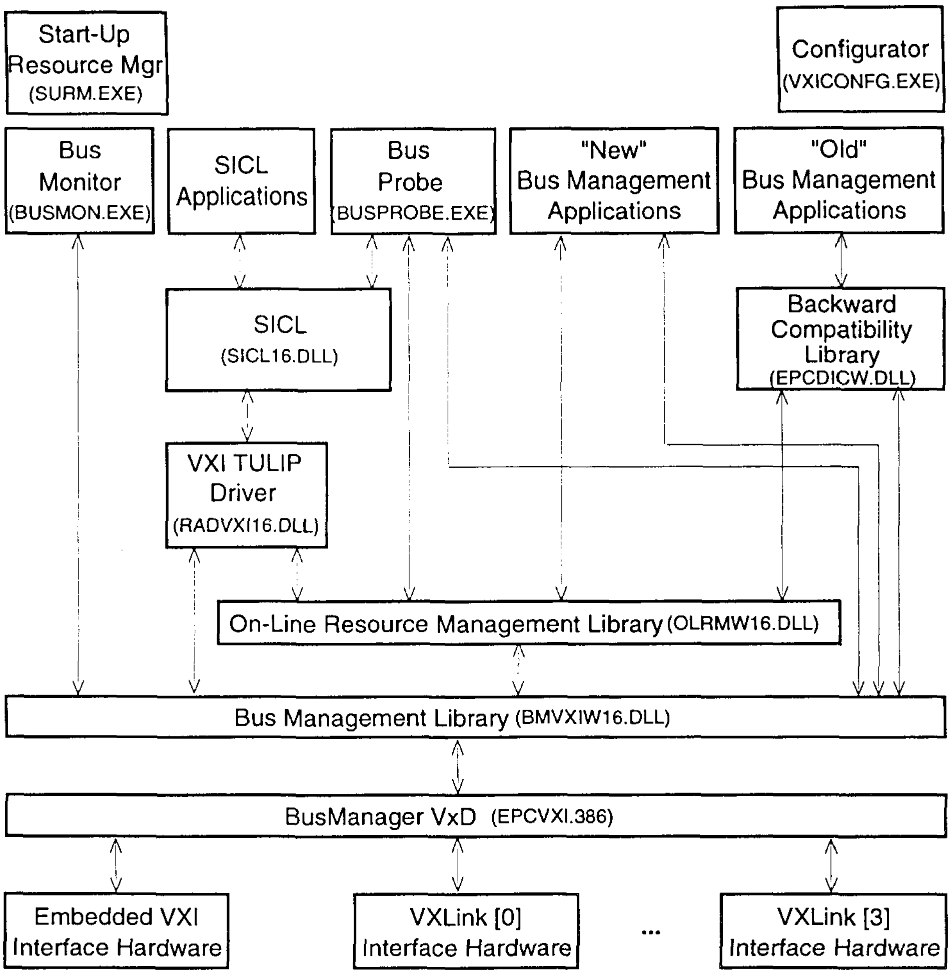


Figure 1-1. EPConnect/VXI for Windows Software Block Diagram.

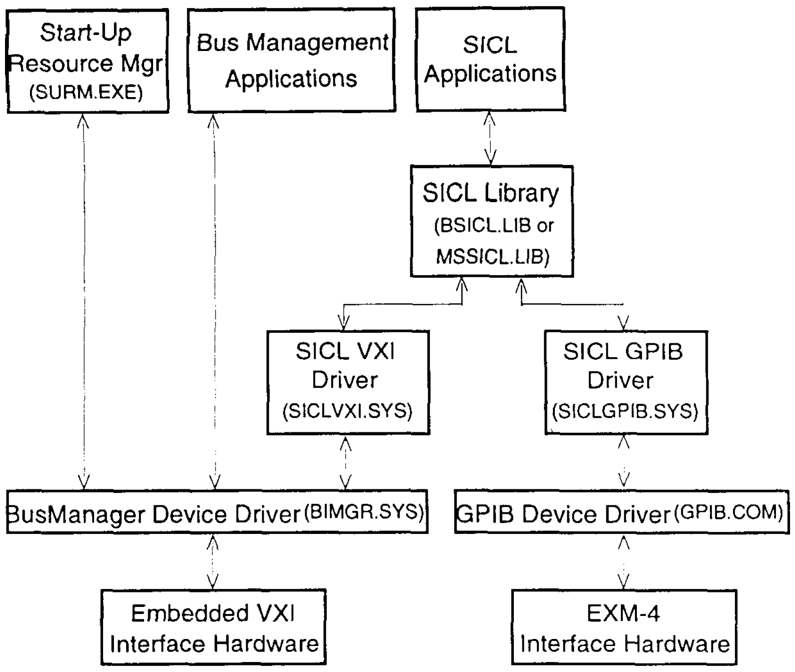

Start-Up Resource Mgr (SURM.EXE)
Bus Management Applications
SICL Applications
SICL Library (BSICL.LIB or MSSICL.LIB)
SICL VXI Driver (SICLVXI.SYS)
SICL GPIB Driver (SICLGPIB.SYS)
BusManager Device Driver (BIMGR.SYS)
GPIB Device Driver (GPIB.COM)
Embedded VXI Interface Hardware
EXM-4 Interface Hardware

Figure 1-2. EPConnect/VXI for DOS Software Block Diagram.

1.5 Related Documentation

The following documents contain additional information you may find helpful.

ANSI IEEE 488-2. published by The Institute of Electrical and Electronic Engineers, Inc., 345 East 47th Street, New York, NY 10017

Microsoft Windows User's Guide, published by Microsoft Corporation, One Microsoft Way, Redmond, WA 98052.

Standard Instrument Control Library for C Programming, published by the VXI Systems Division of Hewlett Packard, 815 SW 14th, Loveland, Colorado 80537

VMEbus Specification, published by VITA, 10229 N. Scottsdale Road, Scottsdale, AZ, 85253

VMEbus Extensions for Instrumentation, System Specification, published by the VXIbus Consortium, Inc., C/O VITA, 10229 N. Scottsdale Road, Scottsdale, AZ, 85253

VMEbus Extensions for Instrumentation, Mainframe Extender Specification, published by the VXIbus Consortium, Inc., C/O VITA, 10229 N. Scottsdale Road, Scottsdale, AZ, 85253

NI-488 MS-DOS Software Reference Manual, published by National Instruments Corporation, 6504 Bridge Point Parkway, Austin, TX, 78730.

The following documents contain additional information about related EPConnect products:

Bus Management for DOS Programmer's Reference Guide, published by RadiSys Corporation.

Bus Management for Windows Programmer's Reference Guide, published by RadiSys Corporation.

SICL for DOS Programmer's Reference Guide, published by RadiSys Corporation.

SICL for Windows Programmer's Reference Guide, published by RadiSys Corporation.

NOTES

2. Installation and Configuration

2.1 Introduction

This chapter contains the information necessary to install and configure any of the four EPConnect APIs:

Bus Management for DOS
Bus Management for Windows
SICL for DOS
SICL for Windows

Refer to the installation information if the software is not pre-installed on your system. Refer to the configuration information to configure the API you wish to use.

2.2 Choosing between DOS and Windows

The first decision users face is whether to use the DOS or Windows operating environment. Since each user environment and application is different, there is no "right" answer. Refer to Figure 2-1.

For users who are comfortable with the DOS operating environment and have already implemented software applications that rely on DOS, EPConnect provides DOS support.

For users who are comfortable with the Windows operating environment, or are interested in converting from DOS to Windows, EPConnect software enables them to do so.

Once you have made the choice to use either DOS or Windows, you can install EPConnect. RadiSys provides different methods for installing EPConnect software in DOS-only and DOS/Windows environments. Whether Windows is present on your EPC determines the methodology to use.

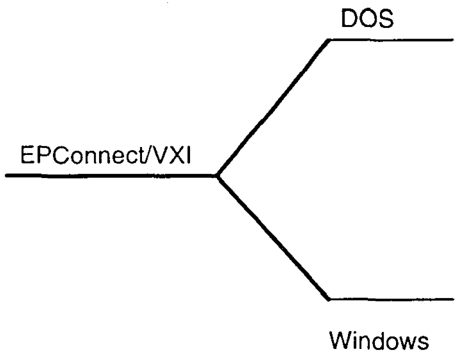


Figure 2-1. EPConnect/VXI Decision Tree: DOS vs. Windows.

2.3 Installing EPConnect on a DOS-only system

Before installing EPConnect for a DOS-only system, verify that:

- Your system is an EPC-7 or greater.
- Your system is running DOS version 5.0 or greater.

Note that the DOS-only installation is programmed to automatically install EPConnect files into default directories and subdirectories. The installation script modifies the system's **AUTOEXEC.BAT** and **CONFIG.SYS** files assuming the default directories were used. If you need to change the location of the EPConnect files at some later date, you can copy the files manually. Be sure to make the appropriate changes to the **AUTOEXEC.BAT** and **CONFIG.SYS** files if you change the location of the EPConnect files.

To begin the installation, follow these procedures:

1. Insert the disk labeled "EPConnect/VXI Disk 1" into drive A:
2. At the command line, type the following:

   A:INSTALL

   then press <ENTER>.
3. Follow the instructions printed on the screen. When finished, the system automatically updates the current **CONFIG.SYS** and **AUTOEXEC.BAT** files.
4. Reboot your system so the changes made to **CONFIG.SYS** and **AUTOEXEC.BAT** take effect.

2.4 Installing EPConnect on a DOS/Windows system

Before installing EPConnect on a DOS/Windows system, verify that:

- Your system is an EPC-7 or greater.
- Your system is running DOS version 5.0 or greater.
- Microsoft Windows version 3.1 or greater is installed.

To begin the installation, follow these procedures:

1. Insert the disk labeled "EPConnect/VXI Disk 1" into drive A:
2. Start Windows from the **C:\** prompt by typing

   WIN
3. Select RUN from the Program Manager's File Menu. Enter

   A:SETUP

4. Follow the instructions printed on the screen. When finished, the system prompts you whether to automatically update the current **CONFIG.SYS**, **AUTOEXEC.BAT**, **SYSTEM.INI** and **WIN.INI** files. If you select "Modify" you do not have to manually change the files for any API except SICL for DOS. Refer to section 2.7, *Manually Configuring the SICL for DOS API.*

5. Exit Windows, reboot your system, and restart Windows so the changes made to **CONFIG.SYS, AUTOEXEC.BAT, SYSTEM.INI** and **WIN.INI** take effect. The setup program allows you to reboot your system from within Windows if you prefer.

Note that it is possible to start Windows and begin the installation simultaneously from the A: drive by typing **SETUP** from the **A:\>** prompt. Windows starts automatically and begins the EPConnect setup routine.

2.5 Choosing the Bus Management or SICL Application Interface

Once you have installed EPConnect, you must choose which variation of the programming interface to configure. Both the Bus Management Library and SICL are available for use with either DOS or Windows. Refer to Figure 2-2. Whether you have determined that DOS or Windows best meets your requirements, you must further choose between using the Bus Management or SICL application interface (API).

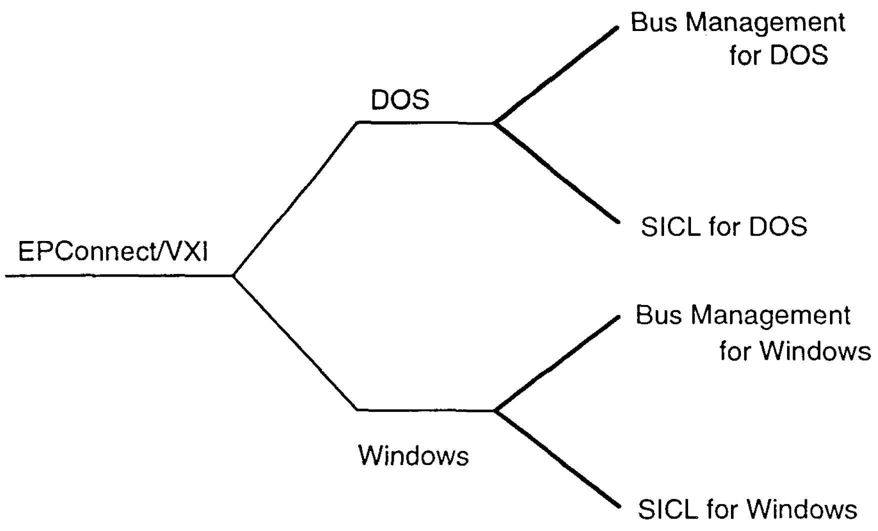


Figure 2-2. EPConnect/VXI Decision Tree: Bus Management vs. SICL.

In choosing between the Bus Management API and the SICL API, consider the following:

Portability

SICL is a standard, portable interface supported across a wide variety of hardware and software environments. Application code programmed under one environment is correct and functionally equivalent under many other environments. The Bus Management API provides portability across several PC operating systems, but is limited to supported EPC hardware.

Interface-independence

SICL source code executes equally well and transparently to both VXI and GPIB devices, whether the receiving device is a VXI message-based device, a VXI register-based device or a GPIB device. SICL handles interface-specific communications details. The Bus Management API supports VXI only.

Naming Abstraction

Devices can be addressed by name using SICL, rather than by an interface-dependent addressing mechanism. Naming abstraction is not supported in the Bus Management API. When using the Bus Management API, programmers must use unique logical addresses (ULAs) to address VXI devices.

Formatted I/O Support

SICL provides easy formatting of standard instrumentation data types. The Bus Management API does not support data type formatting.

The Bus Management API is VXI-specific and lower-level, thus less abstract. It provides greater coverage of VXI-specific and hardware-specific functionality and a generally higher level of performance than SICL. The VXI portion of SICL is built upon the Bus Management API.

2.5.1 What To Do Next

For the majority of users, the installation is now complete. If you allowed the installation program to automatically change configuration files, only the SICL for DOS API requires additional manual changes. All other APIs are ready to be used.

If you selected to manually modify your chosen API, or if you choose to use the SICL for DOS API, refer now to the appropriate section:

- If you will be using the Bus Management for DOS API, refer to section 2.6.
- If you will be using the SICL for DOS API, refer to section 2.7.
- If you will be using the Bus Management for Windows API, refer to section 2.8.
- If you will be using the SICL for Windows API, refer to section 2.9.

2.6 Manually Configuring the Bus Management for DOS Environment

2.6.1 Overview

This section contains the information necessary to complete the steps required to configure the Bus Manager API running in a DOS environment.

2.6.2 Configuration

Make the changes suggested below, assuming that the installation directory is named C:\EPCONNEC:

CONFIG.SYS

The Bus Management for DOS API requires the presence of the BusManager device driver (**BIMGR.SYS**). Assuming that EPConnect is installed in the default **C:\EPCONNEC** directory, add the following line to your **CONFIG.SYS** file to invoke the **BIMGR.SYS** device driver:

```
DEVICE=C:\EPCONNEC\BIMGR.SYS
```

Refer to the information below regarding switches that can be invoked with the **BIMGR.SYS** driver.

BIMGR.SYS

Description	The BusManager device driver.
Syntax	**DEVICE=BIMGR.SYS** [*options*]
Options	
/B=	The number of Bus Manager E/R queue elements. Legal values are 1 through FF, inclusive. The default value is 10 (hexadecimal).
/S=	The number of BusManager device entries. Each device requires one entry. Legal values range are 1 to FF. The default is F.

/T=	The watchdog timer speed. Legal values are:
	F = fast
	S = slow (default)
	Refer to your hardware reference manual for additional information. This option is not part of the BIOS SETUP program.

/U=	The EPC unique logical address (ULA). It also determines the base address of the EPC's registers in A16 space. Legal values are 0 to FF. Refer to your hardware reference for additional information.

WINMEM.SYS

Previous releases of EPConnect contained a Windows Memory Management device driver (WINMEM.SYS). The **WINMEM.SYS** device driver is obsolete and is no longer used. If you are upgrading from a previous release of EPConnect, remove the **WINMEM.SYS** invocation from your **CONFIG.SYS** file to conserve low memory.

EMM386.EXE

You must exclude the upper quarter of the D-page and the E-page from the memory manager. For example,

DEVICE=C:\DOS\EMM386.EXE NOEMS X=DC00-EFFF.

AUTOEXEC.BAT

OLD EPCONNECT VERSIONS

Previous releases of EPConnect contained an application interface called the Message Delivery System (MDS). MDS is obsolete and no longer supported. If you are upgrading from a previous release of EPConnect, remove the **CBRIDGE.EXE**, **VXIBRDG.EXE**, **REGBRDG.EXE**, and **GPIBRDG.EXE** invocations from your **AUTOEXEC.BAT** file to conserve low memory.

PATH

Append the **C:\EPCONNEC** directory to the PATH statement.

INCLUDE

Append the **C:\EPCONNEC\INCLUDE** directory to the INCLUDE variable.

LIBRARY

Append the **C:\EPCONNEC\LIB** directory to the Library statement.

EPCONNECT

EPConnect applications use the EPCONNECT environment variable to locate the EPConnect installation directory and their configuration and database files. Assuming that EPConnect is installed in the default **C:\EPCONNEC** directory, add the following line to your **AUTOEXEC.BAT** file to define the EPCONNECT environment variable:

```
EPCONNECT=C:\EPCONNEC
```

SURM

You must assign a name to a device and execute the SURM before you can communicate with the device using the Bus Management for DOS API. If you do not assign a name, the SURM assigns a temporary name that is used until you reset the system. SURM-assigned temporary names may change if the system configuration changes.

The Bus Management for DOS API requires that the Start-Up Resource Manager (**SURM.EXE**) execute each time the system boots to dynamically configure the devices present in the system. Assuming that EPConnect is installed in the default **C:\EPCONNEC** directory, add the following line to your **AUTOEXEC.BAT** file and reboot the EPC to execute the SURM:

```
C:\EPCONNEC\SURM
```

The Start-Up Resource Manager (SURM) performs the functions of the VXI resource manager. After performing its resource manager functions, SURM outputs its decisions, actions, and any detected errors to the controller display and to ASCII text files.

The Bus Management for DOS API requires that the **AUTOEXEC.BAT** file execute the Start-Up Resource Manager (SURM) in order to configure devices. The EPConnect installation program adds the appropriate line to the **AUTOEXEC.BAT** file. The SURM is described in detail in Chapter 3. SURM error messages are described in Appendix A.

DEVICES

The **DEVICES** file is key to the EPConnect environment. It defines the names and other attributes of devices that EPConnect programs can access. The **DEVICES** file is usually edited using the Configurator application, although it can be edited manually.

The **DEVICES** file is described in detail in Appendix B.

2.7 Manually Configuring the SICL for DOS Environment

2.7.1 Overview

This section contains the information necessary to complete the steps required to configure the SICL API running in a DOS environment.

If you chose to let the installation program automatically update your system, you are nearly finished. If you are using SICL calls to VXI instruments or SICL calls to GPIB instruments, there are two files to "unremark" in the **CONFIG.SYS** file by editing and deleting REM from the beginning of these lines:

```
REM DEVICE=SICLVXI.SYS
REM DEVICE=SICLGPIB.SYS
```

If you chose not to allow the installation program to automatically update your system during installation, you must manually make the changes below. The changes suggested below assume that the installation directory is named **C:\EPCONNEC**.

CONFIG.SYS

Check the **CONFIG.SYS** file to determine if the device driver **BIMGR.SYS** is present. If it is not present, add the following line to the **CONFIG.SYS** file:

DEVICE=C:\EPCONNEC\BIMGR.SYS to the **CONFIG.SYS** file.

BIMGR.SYS

Description The BusManager device driver.

Syntax **DEVICE=BIMGR.SYS** [*options*]

Options

/B=	The number of Bus Manager E/R queue elements. Legal values are 1 through FF, inclusive. The default value is 10 (hexadecimal).
/S=	The number of BusManager router device entries. Each device requires one entry. Legal values range are 1 to FF. The default is F.

/T=	The watchdog timer speed. Legal values are: F = fast; S = slow (default). Refer to your hardware reference manual for additional information. This option is not part of the BIOS SETUP program.
/U=	The EPC unique logical address (ULA). It also determines the base address of the EPC's registers in A16 space. Legal values are 0 to FF. Refer to your hardware reference for additional information.

WINMEM.SYS

Previous releases of EPConnect contained a Windows Memory Management device driver (**WINMEM.SYS**). The **WINMEM.SYS** device driver is obsolete and is no longer used. If you are upgrading from a previous release of EPConnect, remove the **WINMEM.SYS** invocation from your **CONFIG.SYS** file to conserve low memory.

EMM386.EXE

You must also exclude the upper quarter of the D-page and the E-page from the memory manager. For example, add the following line to the CONFIG.SYS file:

```
DEVICE=C:\DOS\EMM386.EXE NOEMS X=DC00-EFFF.
```

SICLGPIB.SYS depends on the presence of **GPIB.COM**. Be sure to invoke **GPIB.COM** from the **CONFIG.SYS** file before invoking **SICLGPIB.SYS**.

GPIB.COM

Description	National Instruments GPIB device driver.
Syntax	**DEVICE=GPIB.COM**
Options	None

To use SICL for DOS with EXM-4 interface hardware, you must install GPIB.COM and execute the National Instruments program **IBCONF.EXE** to configure the board GPIB0 as follows:

1. Change the name of one of the named GPIB0 devices to EPCDEV1.

2. Enable REN line assertion when GPIB0 is the system controller.

3. Set GPIB0's IRQ, I/O address, and DMA channel to match the system's EXM-4 configuration.

SICLGPIB.SYS

Description Defines the GPIB device driver.

Syntax **DEVICE=<path>\SICLGPIB.SYS** [*Options*]

Options

/DE=[*GPIB device driver name*] Specifies the GPIB device driver name as defined by National Instrument's GPIB configuration utility **IBCONFIG**. Contains up to eight printable ASCII characters, not including:

" " blank
"." period
""" double quote
"/" forward-slash
"\" back-slash
"[" left bracket
"]" right bracket
":" colon
"<"less than
">"more than
"+"plus sign
"="equal sign
"," comma

Names longer than eight characters and/or containing invalid characters will be truncated at the first illegal character. A GPIB device name must not conflict with a file or directory name. The default device driver name is EPCDEV1. Device names are not case-sensitive.

To configure the GPIB driver **GPIB.COM,** refer to *GPIB.COM* above.

The device driver name is different from the file name because DOS places the device driver name in the file name space, preventing a file name from being identical to a device driver name.

/DR=[*SICL device driver name*]	Specifies the device driver name. Contains up to eight printable ASCII characters, not including: " " blank "." period """ double quote "/" forward-slash "\" back-slash "[" left bracket "]" right bracket ":" colon "<"less than ">"more than "+"plus sign "="equal sign "," comma Names longer than eight characters and/or containing invalid characters will be truncated at the first illegal character. A device driver's name must not conflict with a file or directory name. The default device driver name is GPIB$1. DOS device driver names are not case-sensitive.
/P=[*process-cnt*]	Specifies the number of processes supported by the device driver. The minimum number of processes is 1. The maximum number is 16. The default value is 4. This SICL implementation does not use this parameter.
/SE=[*session-cnt*]	Specifies the number of sessions the device driver supports. The minimum number of sessions is 1. The maximum number is 256. The default value is 16.

/ST=[*stack-cnt*],[*stack-size*] Specifies the number and size (in bytes) of the device driver interrupt stacks.

The minimum number of stacks is 1. The maximum number is 16. The default value is 4.

The minimum stack size is 256 bytes. The maximum size is 64K bytes. The default stack size is 1K bytes.

SICLVXI.SYS

Description The SICL VXIbus device driver.

Syntax **DEVICE=<path>\SICLVXI.SYS** [*options*]

Options

/**DR**=[*SICL device driver name*] Specifies the device driver name. Contains up to eight printable ASCII characters, not including:

" " blank
"." period
""" double quote
"/" forward-slash
"\" back-slash
"[" left bracket
"]" right bracket
":" colon
"<"less than
">"more than
"+"plus sign
"="equal sign
"," comma

Names longer than eight characters and/or containing invalid characters will be truncated at the first illegal character. A device driver's name must not conflict with a file or directory name.

	The default device driver name is VXI$1. DOS device driver names are not case-sensitive.
	The device driver name is different from the file name because DOS places the device driver name in the file name space, preventing a file name from being identical to a device driver name.
/P=[*process-cnt*]	Specifies the number of processes supported by the device driver. The minimum number of processes is 1. The maximum number is 16. The default is 4.
/SE=[session-cnt]	Specifies the number of sessions the device driver supports. The minimum number of sessions is 1. The maximum number is 256. The default value is 16.
/ST=[*stack-cnt*],[*stack-size*]	Specifies the number and size (in bytes) of the device driver interrupt stacks.
	The minimum number of stacks is 1. The maximum number is 16. The default number of stacks is 4.
	The minimum stack size is 256 bytes. The maximum size is 64 Kbytes. The default stack size is 1 Kbyte.

AUTOEXEC.BAT

OLD EPCONNECT VERSIONS

Previous releases of EPConnect contained an application interface called the Message Delivery System (MDS). MDS is obsolete and no longer supported. If you are upgrading from a previous release of EPConnect, remove the **CBRIDGE.EXE**, **VXIBRDG.EXE**, **REGBRDG.EXE**, and **GPIBRDG.EXE** invocations from your **AUTOEXEC.BAT** file to conserve low memory.

PATH

Append the **C:\EPCONNEC** directory to the PATH statement.

INCLUDE

Append the **C:\EPCONNEC\INCLUDE** directory to the INCLUDE variable.

LIBRARY

Append the **C:\EPCONNEC\LIB** directory to the Library statement.

SET

EPConnect applications use the EPCONNECT environment variable to locate the EPConnect installation directory and their configuration and database files. Assuming that EPConnect is installed in the default **C:\EPCONNEC** directory, add the following line to your **AUTOEXEC.BAT** file to define the EPCONNECT environment variable:

```
EPCONNECT=C:\EPCONNEC
```

SURM

The Start-Up Resource Manager (SURM) performs the functions of the VXI resource manager. After performing its resource manager functions, SURM outputs its decisions, actions, and any detected errors to the controller display and to ASCII text files. The SICL for DOS API requires that the **AUTOEXEC.BAT** file execute the Start-Up Resource Manager (SURM) in order to configure devices. The EPConnect installation program adds the appropriate line to the **AUTOEXEC.BAT** file. The SURM is described in detail in Chapter 3. SURM error messages are described in Appendix A.

Assuming that EPConnect is installed in the default **C:\EPCONNEC** directory, add the following line to your **AUTOEXEC.BAT** file and reboot the EPC to execute the SURM:

```
C:\EPCONNEC\SURM
```

You must assign a name to a device or interface and then execute the SURM before you can communicate with the interface or a device using the SICL for DOS API. If you do not assign a name to a device, the SURM assigns a temporary name that is used until you reset the system. SURM-assigned temporary names may change if the system configuration changes.

DEVICES

The **DEVICES** file is key to the EPConnect environment. It defines the names and other attributes of devices that EPConnect programs can access. The **DEVICES** file is usually edited using the Configurator application, although it can be edited manually.

The **DEVICES** file is described in detail in Appendix B.

SICLIF

The **SICLIF** file defines SICL for DOS interface attributes and SICL for DOS queue sizes. The figure below shows the default **SICLIF** file. The **SICLIF** file is edited manually.

The **SICLIF** file in Figure 2-2 meets the needs of the majority of users. Only users with multiple GPIB boards or with different queue size requirements need to modify this file.

```
IFNAME=VXI,LOGICALUNIT=2,DRIVER=VXI$1
IFNAME=vxi,LOGICALUNIT=2,DRIVER=VXI$1
IFNAME=MXI,LOGICALUNIT=2,DRIVER=VXI$1
IFNAME=mxi,LOGICALUNIT=2,DRIVER=VXI$1
IFNAME=GPIB,LOGICALUNIT=1,DRIVER=GPIB$1
IFNAME=gpib,LOGICALUNIT=1,DRIVER=GPIB$1
IFNAME=HPIB,LOGICALUNIT=1,DRIVER=GPIB$1
IFNAME=hpib,LOGICALUNIT=1,DRIVER=GPIB$1
SICLLIB,ERRORQUEUESIZE=4,EVENTQUEUESIZE=4
```

Figure 2-3. Sample SICLIF file for SICL for DOS.

The default **SICLIF** file defines a typical set of VXI and GPIB interface names to device driver name mappings. The mappings are for use when opening SICL sessions. For example, when using the default **SICLIF** file, all communication directed to sessions opened using the **VXI, vxi, MXI,** or **mxi** interface names, or logical unit **2**, are routed to device driver **VXI$1**. Likewise, communication directed to sessions opened using the **GPIB, gpib, HPIB,** or **hpib** interface names, or logical unit **1** are routed to device driver **GPIB$1**. Device driver names are defined using the SICLVXI.SYS and SICLGPIB.SYS /DR option.

To improve the user interface and because an interface name (IFNAME) can also be a symbolic name, which is case sensitive, the default **SICLIF** file defines an interface device driver mapping for interface names with both all upper-case and all lower-case letters. Mixed case interface names are allowed, but must be defined manually by editing the **SICLIF** file.

The names you assign SICL interfaces must have this form:

logical unit | *symbolic name*

where *logical unit* is an integer greater than zero and less than 32,767 and *symbolic name* is any sequence of letters, digits, underscores, and dashes that begins with a letter. The following are valid interface addresses:

7	An interface at *logical unit* 7.
vxi	A *symbolic name* for the VXIbus interface.

2.8 Manually Configuring the Bus Management for Windows Environment

2.8.1 Overview

This section contains the information necessary to complete the steps required to configure the Bus Management API for the Windows environment.

2.8.2 Configuration

CONFIG.SYS

The Bus Management for Windows API requires no DOS device driver support. Remove all EPConnect device driver invocations from your **CONFIG.SYS** file to conserve low memory.

EMM386.EXE

You must exclude the upper quarter of the D-page and the E-page from the memory manager. For example, **DEVICE=C:\DOS\EMM386.EXE NOEMS X=DC00-EFFF.**

AUTOEXEC.BAT

EPConnect applications use the EPCONNECT environment variable to locate the EPConnect installation directory and their configuration and database files. Assuming that EPConnect is installed in the default **C:\EPCONNEC** directory, add the following line to your **AUTOEXEC.BAT** file to define the EPCONNECT environment variable:

```
EPCONNECT=C:\EPCONNEC
```

The Bus Management for Windows API requires that the Start-Up Resource Manager (**SURM.EXE**) execute each time the system boots to dynamically configure the devices present in the system. Assuming that EPConnect is installed in the default **C:\EPCONNEC** directory, add the following line to your **AUTOEXEC.BAT** file and reboot the EPC to execute the SURM:

```
C:\EPCONNEC\SURM
```

Previous releases of EPConnect contained an application interface called the Message Delivery System (MDS). MDS is obsolete and no longer supported. If you are upgrading from a previous release of EPConnect, remove the **CBRIDGE.EXE**, **VXIBRDG.EXE**, **REGBRDG.EXE**, and **GPIBRDG.EXE** invocations from your **AUTOEXEC.BAT** file to conserve low memory.

SYSTEM.INI

The Bus Management for Windows API requires the presence of the BusManager VxD (**EPCVXI.386**). Assuming that EPConnect is installed in the default **C:\EPCONNEC** directory, add the following line to the [386Enh] section of the **SYSTEM.INI** file to invoke the BusManager VxD:

```
DEVICES=C:\EPCONNEC\EPCVXI.386
```

RadiSys EPCs use the physical address range E0000-EFFFF to access the VXIbus and use DC000-DFFFF for network accesses. These address ranges must be excluded from Windows control in the [386Enh] section of the **SYSTEM.INI** file, as follows:

```
EMMExclude=DC00-EFFF
```

The Windows **SYSTEM.INI** file can contain a section to control **EPCVXI.386** VxD configuration at Windows boot time:

```
[EPCVXI]
SHMEMBufferSize=64
MaxSessions=128
WSPFastPoll=20
WSPOpTime=200
```

SHMEMBufferSize

Is the number of Kbytes of contiguous shared memory. The number is rounded up to the next multiple of 4KBytes. This memory resides on the EPC and is accessible by EPConnect applications and VMEbus masters. The minimum value is zero. The maximum value is system dependent. If your machine is set up for A32 slave state, only the first 16 Mbytes of DRAM are accessible from the VMEbus, and the buffer resides somewhere in this area. For the more restrictive A24 slave space, a buffer size of 32 Kbytes is a practical maximum value.

MaxSessions

Is the maximum number of simultaneously open VxD sessions supported. The minimum value is 1. The maximum value is 256.

WSPFastPoll

Is the number of milliseconds that the VxD will poll a device's VXI Response register before yielding the processor. The minimum value is 20. The maximum value is 2^{32}.

WSPOpTime

Is the number of milliseconds that the VxD will perform a buffered word serial I/O before yielding the processor. The minimum value is 200. The maximum value is 2^{32}.

You may modify the default configuration by changing values associated with one or more of the above parameters in **SYSTEM.INI**. Restart Windows for the changes to take effect.

2.8.3 Bus Management for Windows Configuration Files

You must assign a name to a device and execute the SURM before you can communicate with the device using the Bus Management for Windows API. If you do not assign a name, the SURM assigns a temporary name that is used until you reset the system. SURM-assigned temporary names may change if the system configuration changes.

SURM.EXE

The Start-Up Resource Manager (SURM) performs the functions of the VXI resource manager. After performing its resource manager functions, SURM outputs its decisions, actions, and any detected errors to the controller display and to ASCII text files.

The Bus Management for Windows API requires that the **AUTOEXEC.BAT** file execute the Start-Up Resource Manager (SURM) in order to configure devices. The EPConnect installation program adds the appropriate line to the **AUTOEXEC.BAT** file. The SURM is described in detail in Chapter 3. SURM error messages are described in Appendix A.

DEVICES

The **DEVICES** file is key to the EPConnect environment. It defines the names and other attributes of devices that EPConnect programs can access. The **DEVICES** file is usually edited using the Configurator application, although it can be edited manually.

The **DEVICES** file is described in detail in Appendix B.

2.9 Manually Configuring the SICL for Windows Environment

2.9.1 Overview

This section contains the information necessary to complete the steps required to configure the SICL API for the Windows environment.

2.9.2 Configuration

Make the changes suggested below, assuming that the installation directory is named **C:\EPCONNEC**:

CONFIG.SYS

BIMGR.SYS

The SICL for Windows API requires no DOS device driver support. Remove all EPConnect device driver invocations from your **CONFIG.SYS** file to conserve low memory. Check the **CONFIG.SYS** file to determine if the device driver **BIMGR.SYS** is present. If it is present, delete the line **DEVICE=C:\EPCONNEC\BIMGR.SYS** in the **CONFIG.SYS** file.

You must also exclude the upper quarter of the D-page and the E-page from the memory manager. For example, add the following line:

```
DEVICE=C:\DOS\EMM386.EXE NOEMS X=DC00-EFFF.
```

WINMEM.SYS

If present, remove **DEVICE=C:\...\WINMEM.SYS** from the **CONFIG.SYS** file.

SICLVXI.SYS

Add the SICL VXI device driver by installing the following line in the CONFIG.SYS file:

```
DEVICE=C:\EPCONNEC\SICLVXI.SYS
```

For a list of option parameters, refer to section 2.7.1.

SICLGPIB.SYS

If using the SICL GPIB drivers, add the device driver **SICLGPIB.SYS** to the **CONFIG.SYS** file using the following example:

```
DEVICE=C:\EPCONNEC\SICLGPIB.SYS
```

Note that if you are also using the National Instruments GPIB device driver **GPIB.COM**, the driver **SICLGPIB.SYS** must be called after **GPIB.COM**. For a list of option parameters, refer to section 2.7.1.

AUTOEXEC.BAT

EPCONNECT

EPConnect applications use the EPCONNECT environment variable to locate the EPConnect installation directory and their configuration and database files. Assuming that EPConnect is installed in the default **C:\EPCONNEC** directory, add the following line to your **AUTOEXEC.BAT** file to define the EPCONNECT environment variable:

```
EPCONNECT=C:\EPCONNEC
```

SURM

The SICL for Windows API requires that the Start-Up Resource Manager (**SURM.EXE**) execute each time the system boots to dynamically configure the devices present in the system. Assuming that EPConnect is installed in the default **C:\EPCONNEC** directory, add the following line to your **AUTOEXEC.BAT** file and reboot the EPC to execute the SURM:

```
C:\EPCONNEC\SURM
```

OLD EPCONNECT VERSIONS

Previous releases of EPConnect contained an application interface called the Message Delivery System (MDS). MDS is obsolete and no longer supported. If you are upgrading from a previous release of EPConnect, remove the **CBRIDGE.EXE**, **VXIBRDG.EXE**, **REGBRDG.EXE**, and **GPIBRDG.EXE** invocations from your **AUTOEXEC.BAT** file to conserve low memory.

PATH

Append the **C:\EPCONNEC** directory to the PATH statement.

INCLUDE

Append the **C:\EPCONNEC\INCLUDE** directory to the INCLUDE variable.

LIBRARY

Append the **C:\EPCONNEC\LIB** directory to the Library statement.

EPCONNECT

Add the statement **SET EPCONNECT=C:\EPCONNEC** directory.

WIN.INI

The SICL for Windows API requires an entry in the **WIN.INI** file to locate the SICL for Windows installation directory. Assuming that the SICL for Windows portion of EPConnect is installed in the default **C:\SICL** directory, add the following lines to the **WIN.INI** file:

```
[SICL]
BASEDIR=C:\SICL
```

SYSTEM.INI

VxD

The Windows **SYSTEM.INI** file can contain a section to control **EPCVXI.386** VxD configuration at Windows boot time:

```
[EPCVXI]
SHMEMBufferSize=64
MaxSessions=128
WSPFastPoll=20
WSPOpTime=200
```

SHMEMBufferSize

Is the number of Kbytes of contiguous shared memory. The number is rounded up to the next multiple of 4KBytes. This memory resides on the EPC and is accessible by EPConnect applications and VMEbus masters. The minimum value is zero. The maximum value is system dependent. If your machine is set up for A32 slave state, only the first 16 Mbytes of DRAM are accessible from the VMEbus, and the buffer resides somewhere in this area. For the more restrictive A24 slave space, a buffer size of 32 Kbytes is a practical maximum value.

MaxSessions

Is the maximum number of simultaneously open VxD sessions supported. The minimum value is 1. The maximum value is 256.

WSPFastPoll

Is the number of milliseconds that the VxD will poll a device's VXI Response register before yielding the processor. The minimum value is 20. The maximum value is 2^{32}.

WSPOpTime

Is the number of milliseconds that the VxD will perform a buffered word serial I/O before yielding the processor. The minimum value is 200. The maximum value is 2^{32}.

The SICL for Windows API requires the presence of the BusManager VxD (**EPCVXI.386**). Assuming that EPConnect is installed in the default **C:\EPCONNEC** directory, add the following line to the [386Enh] section of the **SYSTEM.INI** file to invoke the BusManager VxD:

```
DEVICES=C:\EPCONNEC\EPCVXI.386
```

EXCLUDE

RadiSys EPCs use the physical address range E0000-EFFFF to access the VXIbus and use DC000-EFFFF for network accesses. This address range must be excluded from Windows control in the [386Enh] section of the **SYSTEM.INI** file, as follows:

```
EMMExclude=DC000-EFFFF
```

2.9.3 Configuration Files

You must assign a name to a device or interface and then execute the SURM before you can communicate with the interface or a device using the SICL for Windows API. If you do not assign a name to a device, the SURM assigns a temporary name that is used until you reset the system. SURM-assigned temporary names may change if the system configuration changes.

SURM.EXE

The Start-Up Resource Manager (SURM) performs the functions of the VXI resource manager. After performing its resource manager functions, SURM outputs its decisions, actions, and any detected errors to the controller display and to ASCII text files.

The SICL for Windows API requires that the **AUTOEXEC.BAT** file execute the Start-Up Resource Manager (SURM) in order to configure devices. The EPConnect installation program adds the appropriate line to the **AUTOEXEC.BAT** file. The SURM is described in detail in Chapter 3. SURM error messages are described in Appendix A.

SURM.RC

SURM.RC configures SURM to properly update **SICL.INI** with the names of devices. The line must be uncommented to take effect. Also, the specified path should be modified to match where SICL is installed:

```
# Uncomment the following line if you are using SICL under Windows
# SICL_INI_FILE=c:\sicl\sicl.ini
```

DEVICES

The **DEVICES** file is key to the EPConnect environment. It defines the names and other attributes of devices that EPConnect programs can access. The **DEVICES** file is usually edited using the Configurator application, although it can be edited manually.

The **DEVICES** file is described in detail in Appendix B.

SICL.INI

The **SICL.INI** file contains SICL for Windows device and interface names and attributes. Interfaces supported by SICL are named in the **SICL.INI** file. The default **SICL.INI** file defines a typical set of VXI and GPIB interface names and their associated interface drivers:

```
[Aliases]
GPIB=hp341i
gpib=hp341i
HPIB=hp341i
hpib=hp341i
VXI=radvxi
vxi=radvxi
MXI=radvxi
mxi=radvxi
```

```
[PARAMS]
hp341i=LU,Name,Interface,Slot,BusAddr,Switches,SysCtl,IRQ
radvxi=LU,Name,Interface

[INTF0]
LU=7
Name=gpib
Interface=gpib
Slot=0
BusAddr=0
Switches=0b1100
SysCtl=1
IRQ=5

[INTF1]
LU=16
Name=vxi
Interface=vxi
```

The GPIB interface "Switches" parameter must match the EXM-22 BIOS settings. The following table specifies the relationship between an EXM-22's I/O address range, it's BIOS settings, and the GPIB interface "Switches" parameter in the **SICL.INI** file:

I/O Address Range	EXM-22 ID	EXM-22 OB1	EXM-22 OB2	**SICL.INI** Switches
250-257	D9	F9	00	Switches=0b0000
270-277	D9	FB	00	Switches=0b1000
350-357	D9	FD	00	Switches=0b0100
370-377	D9	FF	00	Switches=0b1100

NOTES

3. Using the Start-Up Resource Manager (SURM)

3.1 Introduction

The Start-up Resource Manager (SURM) performs the functions of the VXI resource manager. These functions include:

- Initializing the EPC and processing system configuration files.
- Identifying all devices in the system.
- Configuring the A24 and A32 address maps.
- Managing the system self-test and diagnostic sequences.
- Symbolic naming of devices.
- Configuring the system's commander/servant hierarchies.
- Allocating VXIbus IRQ lines.
- Configuring mainframe extenders.
- Initiating normal system operation.

After performing its resource manager functions, SURM outputs it decisions, actions, and any detected errors to the controller display and to ASCII text files. When the SURM screen is displayed, keyboard function keys move you between reports, select a menu that allows you to rename devices, and obtain help.

3.2 Required Environment and Related Files

By default SURM files are located in the directory **C:\EPCONNEC**. If the SURM files are installed at another location, specify their location using the EPCONNECT environment variable. The following lists the required SURM files at their default locations:

C:\EPCONNEC\DB\VXIMANUF	The manufacturers name database.
C:\EPCONNEC\DB\VXIMODEL	The VXI devices model database.
C:\EPCONNEC\SURM.EXE	The SURM executable file.
C:\EPCONNEC\SURM.RC	The runtime configuration file.
C:\EPCONNEC\SURMHELP.TXT	Text displayed when you press F1, when the SURM is running.

These file are created each time the SURM runs and are not included on the release disk:

C:\EPCONNEC\DB\RESRCMGR	A file output by SURM that contains system resource information.
C:\EPCONNEC\SURM.ERR	Errors and warning detected by the SURM the last time it ran. Also included are progress messages if the /v option was specified.
C:\EPCONNEC\SURM.LOG	Current system configuration.

3.3 Operation

The SURM is one of the first application programs to run after the operating system boots. It should be executed from the **AUTOEXEC.BAT** file and should not be executed more than once after power is applied. Executing SURM more than once may invalidate Online Resource Manager (OLRM) data. However, the system is configured correctly. Table 3-1 lists the main steps taken by the SURM when it executes.

Step	Action	Description
1.	EPC initialization	SURM initializes the EPC's bus interface and sets the EPC's unique logical address (ULA) to 0. A ULA of 0 designates the EPC as the resource manager.
2.	Read configuration files	SURM reads the **SURM.RC** file, the command line, and the **DEVICES** file for system information and how to report its actions and decisions.
3.	Static device identification	SURM searches the 256 VXIbus ULAs for statically configured VXIbus devices.
4.	Non-VXIbus device configuration	SURM examines the file **DEVICES** (maintained by the VXI Configurator) for non-VXIbus devices in the system. SURM uses this information to avoid resource assignment conflicts.
5.	Dynamic ULA configuration	SURM assigns ULAs to all dynamically configurable devices.
6.	Slot search	SURM determines the device slot and the state of each slot. Valid slot states are: empty/nonVXI, operating, non-operating, and indeterminate. In addition, the VXI specification level to which each device conforms is identified and reported..

Table 3-1. SURM Actions.

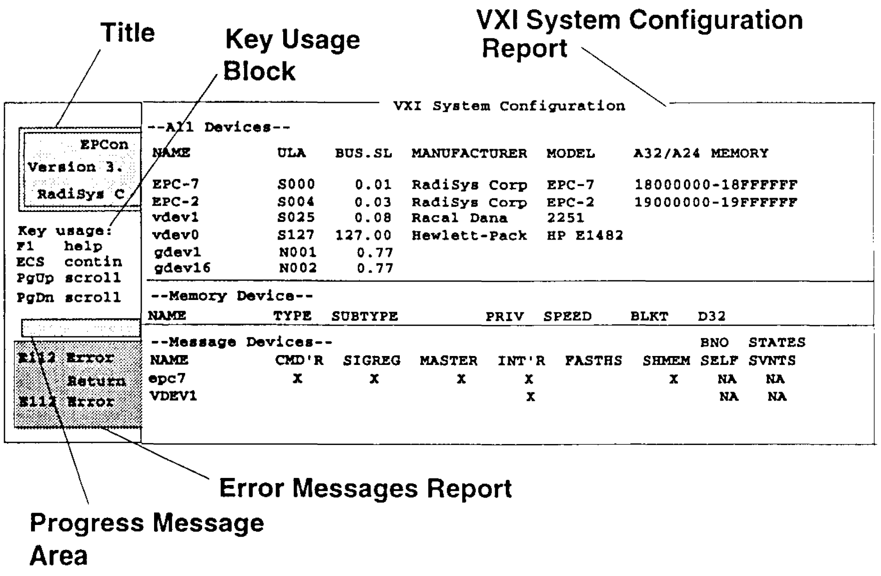
Step	Action	Description
7.	Address map configuration	SURM assigns address ranges to devices that have memory in A24 or A32 space.
8.	Self-test management	SURM examines each device's VXI status register to determine if the device's self test successfully completed. If the self test did not successfully complete before the self test timer (see the **SURM.RC** file) expires, the SURM writes an error message and sets the SYSFAIL INHIBIT and RESET flags in the device's control register.
9.	Symbolic naming	SURM assigns symbolic names to all devices. User supplied symbolic names are taken from the **DEVICES** file (use the Configurator to add symbolic names to the **DEVICES** file). If a name has not been supplied for a device, SURM assigns default names of the form vdevx, where x is a decimal number.
10.	Commander/servant initialization	SURM assigns a hierarchical relationship of commander and servant devices, using the hierarchy information taken from the **DEVICES** file. Use the Configurator to specify the Commander/Servant hierarchy.
11.	Assign interrupts and handlers.	SURM assigns all interrupters and handlers to the system devices.
12.	Get manufacturer name and model numbers	SURM reads manufacturer and model information from the files **VXIMANUF** and **VXIMODEL** and adds that information to the configuration file.
13.	Display SURM screen	Displays the SURM screen for user review and device name verification or change. You can eliminate this step by using a command line option or runtime configuration file switch.

14.	Exit	Writes system configuration information to **SURM.LOG**, writes errors to **SURM.ERR**, and removes SURM screen.

Table 3-1. SURM Actions - (*continued*).

3.4 SURM Startup Screen

Figure 3-1 shows a typical SURM startup screen. The screen consists of five parts: title, VXI System Configuration report, error messages report, progress message area, and key usage block.



```
                                   VXI System Configuration
   EPCon       --All Devices--
Version 3.     NAME          ULA   BUS.SL  MANUFACTURER   MODEL     A32/A24 MEMORY
 RadiSys C
               EPC-7         S000    0.01  RadiSys Corp   EPC-7     18000000-18FFFFFF
               EPC-2         S004    0.03  RadiSys Corp   EPC-2     19000000-19FFFFFF
               vdev1         S025    0.08  Racal Dana     2251
Key usage:     vdev0         S127  127.00  Hewlett-Pack   HP E1482
F1   help      gdev1         N001    0.77
ECS  contin    gdev16        N002    0.77
PgUp scroll
PgDn scroll    --Memory Device--
               NAME          TYPE  SUBTYPE          PRIV  SPEED    BLKT   D32

               --Message Devices--                                          BNO   STATES
E112 Error     NAME          CMD'R   SIGREG   MASTER   INT'R   FASTHS  SHMEM SELF  SVNTS
      Return   epc7            X        X        X        X              X    NA    NA
E112 Error     VDEV1                                      X                   NA    NA
```

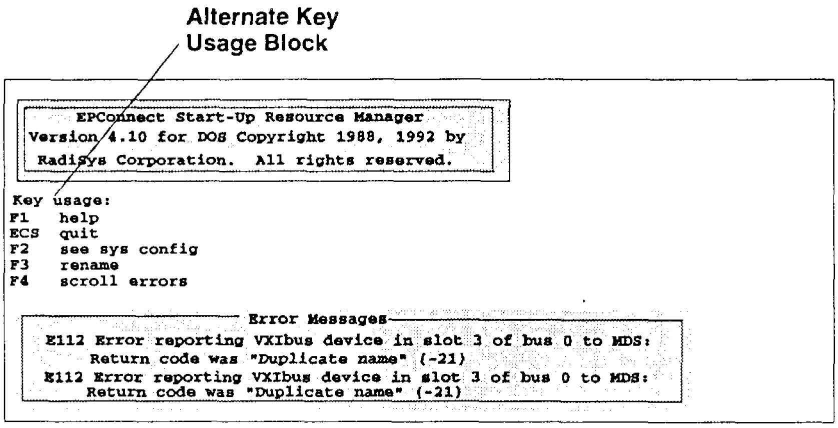
Figure 3-1. SURM Startup Screen.

3.4.1 Title

The title identifies the screen and gives the SURM version number. The title also includes a copyright statement.

3.4.2 Key Usage Block

The Key usage block lists the keys that are valid when the SURM is executing. There are two possible Key usage blocks. The default, shown in Figure 3-1, and the alternate, shown in Figure 3-2. The default key usage block appears when the VXI System Configuration or Error Messages report has the focus. The alternate key usage block has more selections and appears when neither report has the focus. Table 3-2 identifies the actions that the keys perform.



```
   EPConnect Start-Up Resource Manager
Version 4.10 for DOS Copyright 1988, 1992 by
 RadiSys Corporation.  All rights reserved.

Key usage:
F1   help
ECS  quit
F2   see sys config
F3   rename
F4   scroll errors

                    Error Messages
 E112 Error reporting VXIbus device in slot 3 of bus 0 to MDS:
      Return code was "Duplicate name" (-21)
 E112 Error reporting VXIbus device in slot 3 of bus 0 to MDS:
      Return code was "Duplicate name" (-21)
```

Figure 3-2. Alternate Key Usage Block.

Key	Action Performed
F1	Displays the contents of the **SURMHLP.TXT** file.
ESC continue	If the VXI System Configuration report has the focus, removes the VXI System Configuration report and changes the focus to the Error Messages report, if one exists. If the Error Messages report has the focus, changes focus to the alternate key usage block.
ESC quit	Ends SURM execution. SURM writes system configuration information to **SURM.LOG** and error messages to **SURM.ERR**.
F2	Displays the VXI System Configuration report and changes focus to the report.
F3	Displays the Rename dialog box so you can change any system device name.
F4	Changes the focus to Error Messages report.
PgUp	Scrolls the report with the focus up.
PgDn	Scrolls the report with the focus down.

Table 3-2. Valid SURM Keys and Their Actions.

3.4.3 VXI System Configuration Report

The VXI System Configuration report shows information about all devices and slots in the system. It also includes information about decisions made by the SURM. There are nine parts to the report (see Figure 3-3):

- All Devices
- Memory Devices
- Message Devices
- Slot Report (one for each mainframe)
- Commander/Servant Hierarchy
- Interrupt Map
- ULA usage and bus traversal map
- A24 usage and bus traversal map
- A32 usage and bus traversal map

```
--All Devices--
NAME        ULA   BUS.SL    MANUFACTURER    MODEL        A32/A24 MEMORY
TopCmdr     S000    0.01    RadiSys Corp    EPC-7        400000-7FFFFF
mx          S001    0.00    Hewlett-Pack    HP E1482 s
vdev0       S002    0.06    Tektronix       VX4223
timer       S020    0.05    Racal Dana      2251
vdev1       S032    1.00    Hewlett-Pack    HP E1482 s
proto       S033   32.03    Hewlett-Pack    HP E1326
vme1        N034   32.??
sink2       D035   32.01    RadiSys Corp    EPC-2        18000000-18FFFFFF
vdev2       D036   32.04    Tektronix       VX4236
vdev3       D003    0.03    RadiSys Corp    EPC-2        1A000000-1AFFFFFF
--------------------------------------------------
--Memory Devices--
NAME        TYPE  SUBTYPE   PRIV  SPEED     BLKT      D32
--------------------------------------------------
--Message Devices--                                  BNO  STATES
NAME     CMD'R  SIGREG   MASTER  INT'R  FASTHS  SHMEM  SEL   SVNTS
TopCmdr     x      x       x       x                x   NA    NA
vdev0                              x                    NA    NA
timer                              x                    NA    NA
sink2       x      x       x       x                x   NA    NA
vdev2                              x                   NORM  NORM
vdev3       x      x       x       x                x   NA    NA
--------------------------------------------------
--Slot Report-- root mainframe
                 0   1   2   3   4   5   6   7   8   9   10  11  12
EMPTY/NONVXI             x       x           x   x   x    x   x   x
OPERATING        x           x       x   x
NON-OPERATING
INDETERMINATE        x
VXI 1.3
-------------------------------------------------
--Slot Report-- bus 32
                 0   1   2   3   4   5   6   7   8   9   10  11  12
EMPTY/NONVXI             x           x   x   x   x   x    x   x   x
OPERATING        x   x       x   x
NON-OPERATING
INDETERMINATE
VXI 1.3                          x
-------------------------------------------------
--Command/Servant Hierarchy--
TopCmdr
  +--vdev0
  +--timer
  +--vdev1
  +--proto
  +--vme1
  +--sink2
  +--vdev2
  \--vdev3
-------------------------------------------------
```

Figure 3-3. VXI System Configuration Report.

```
--Interrupt Map--             IRQ
     Device Name  Interrupter  7 6 5 4 3 2 1      Handler
H TopCmdr                      | | | | | | +------  [H1]
T vdev2                [PI1]   ----------- +
-------------------------------------------------
--ULA usage and bus traversal map--
bus   ula
  0   000   TopCmdr
  0   001   mx
  0   002   vdev0
  0   003   vdev3
      ...
  0   020   timer
      ...
  0   031   *vacant*
  1   032   1...    vdev1
 32   033   1...    32...   proto
 32   034   1...    32...   vme1
 32   035   1...    32...   sink2
 32   036   1...    32...   vdev2
      ...
 32   039   *vacant*
  0   040   *vacant*
      ...
  0   255   *vacant*
-------------------------------------------------
--A24 usage and bus traversal map
bus   low-high address
  0   000000-3fffff    *vacant*
  0   400000-7fffff    TopCmdr
  0   800000-ffffff    *vacant*
-------------------------------------------------
--A32 usage and bus traversal map--
bus   low-high address
  0   00000000-17ffffff   *vacant*
 32   18000000-18ffffff   1...    32...    sink2
 32   19000000-19ffffff   1...    32...    vme1
  0   1a000000-1affffff   vdev3
  0   1b000000-ffffffff   *vacant*
```

Figure 3-3. VXI System Configuration Report -(*continued*).

All Devices

The All Devices portion of the VXI System Configuration report lists all devices in the system. Table 3-3 identifies and describes its six entries.

Entry	Description	Definition
NAME	Device symbolic name	Obtained from the VXIbus Configurator or assigned by the SURM. Device names assigned by the SURM have the form vdevx, where x is a decimal number.
ULA	Unique logical address	Valid first characters are: S = static logical address D= dynamic logical address N = Non-VXIbus device logical address The next three digits are the device's decimal ULA.
BUS	Bus number	The bus where the device is installed. The root mainframe bus number is always zero. All other bus numbers are same as the ULA of the mainframe extender that connects the local bus to the parent bus.
SL	Slot number	The slot where the device is installed. Contains two question marks (??) if the device is external to the root mainframe.
MANUFACTURER	Device manufacturer's name	The device manufacturer's name as contained in the **VXIMANUF** file. If the manufacturer's name is not found in the **VXIMANUF** file, the number read from the device is displayed in brackets.

MODEL	Device model name	The device model name as contained in the **VXIMODEL** file. If the manufacturer's model number is not found in the **VXIMODEL** file, the number read from the device is displayed in brackets.
A32/24 MEMORY	Starting and ending address of the memory space assigned to the device.	Hexadecimal digits. Six-digits represent A24 memory space. Eight digits represents A32 memory space.

Table 3-3. All Devices Configuration Report.

Memory Devices

The Memory Devices portion of the VXI System Configuration Report lists all VXI memory devices in the system. Table 3-3 identifies and defines its seven entries.

Entry	Description	Definition
NAME	Device symbolic name	Obtained from the Configurator or assigned by SURM. Device names assigned by the SURM have the form vdevx, where x is a decimal number.

TYPE	Memory type	Valid values are:
		RAM = random access memory
		ROM = read-only memory
		other = other memory type
		reser = reserved

SUBTYPE	Memory subtype	Valid values are:
		non-volatile = non-volatile memory
		elec-prog = electrically programmable
		nothing = no subtype

PRIV	Privilege	Valid values are:
		supv = responds to supervisor-level address modifiers
		both = responds to user-level and supervisor-level address modifiers

SPEED	Memory device access speed range	Valid values are: Decimal value in nanoseconds or device dep.
BLKT	Block transfer support	Check mark specifies block transfer support.
D32	32-bit transfer support	Check mark specifies that the device supports 32-bit transfers. Otherwise, it supports only 16-bit transfers.

Table 3-3. Memory Devices Configuration Report Table *(continued).*

Message Devices

The Message Devices portion of the VXI System Configuration report lists all the message-based devices in the system. Table 3-4 identifies and defines its nine entries.

Entry	Description
NAME	Device symbolic name. Obtained from the Configurator or assigned by SURM. Device names assigned by the SURM have the form vdevx, where x is a decimal number.
CMD'R	Device is a commander.
SIGREG	Device has a signal register.
MASTER	Device is a bus master.
INT'R	Device is an interrupter.
FASTHS	Device supports fast-handshake mode.
SHMEM	Device supports shared-memory message protocol.

BNO STATES
SELF SVNTS
	Identifies the begin-normal-operation (BNO) states of the device (SELF) and its servants (SVNTS). Valid entries for each field are:
	NA = not applicable (device is not message based)
	CONF = configure state
	NORM = normal operation

Table 3-4. Message Device Report.

Slot Report

The Slot Report of the VXI System Configuration report lists the state of each VXIbus slot. Table 3-5 defines the valid slot states.

Slot State	Description
EMPTY	Slot is empty or contains a port of a multi-slot device.
OPERATING	Slot occupied and operational.
NON_OPERATIN G	Slot occupied but not operational.
INDETERMINAT E	Cannot determine the device's state.
VME	Slot contains a VME device.
VXI 1.3	Slot contains a VXI device that conforms to VXI specification revision 1.3.

Table 3-5. Valid VXIbus Slot States.

Commander/Servant Hierarchy

The Commander/Servant Hierarchy portion of the VXI System Configuration report shows the commander/servant hierarchy of all devices in the system. Its format follows a tree structure. The first device listed is the top-level commander.

Interrupt Map

The Interrupt Map portion of the VXI System Configuration report shows the connections of the interrupters and the handlers in the system to the VXIbus IRQ lines. Table 3-6 identifies and describes its five entries.

Entry	Description
Reason for interrupt connectivity	Valid entries are: C = configuration data required this connection ! = configuration data required this connection but it conflicts with another configured connection 1 = first allocation pass (each PH commander gets one line) 2 = second allocation pass (PH devices get remaining lines) S = servant tracking. The line was allocated to a commander because one of its servants was configured to interrupt on it (1st pass) T = tracking commander. A servant was configured to interrupt on this line because its commander is handling this line. H = hardwired
Device Name	Device symbolic name. Obtained from the Configurator or assigned by SURM. Device names assigned by the SURM have the form vdevx, where x is a decimal number.

Interrupter	The first character specifies the interrupter type:
	I = interrupter
	PI = programmable interrupter
	
	The next digit is the interrupter number. The interrupter number is for devices with multiple interrupt controllers.

IRQ	A map of interrupter and handler to IRQ lines. The horizontal line on the device's name line ends at the connected IRQ line.

Handler	The first character specifies the handler type:
	H = handler
	PH = programmable handler
	
	The next digit is the handler number. The handler number is for devices with multiple handlers.

Table 3-6. Interrupt Map Configuration Report.

ULA Usage and Bus Traversal Map

The ULA usage and bus traversal map portion of the VXI System Configuration report shows ULA to bus mapping. When the system contains mainframe extenders, it also show the busses traversed to reach each device from the root mainframe. Each line can contain up to four pieces of information: bus number, ULA, traversal map, and device name . Bus number, ULA and device name are obvious. The traversal map appears only for mainframe extenders and is displayed between the ULA and device name. It contain bus IDs, one for each bus traversed and in order of traversal from the root mainframe to the device. Vacant ULAs show mainframe extender logical address window boundaries.

A24 and A32 Usage and Bus Traversal Maps

The A24 usage and bus traversal map and A32 usage and bus traversal map display the range of addresses in use by devices mapped to A24 and A32 space, respectively. They also include bus traversal information for devices in mainframe extenders.

3.4.4 Error Messages Report

The Error Messages report lists errors detected by the SURM. It can also contain progress messages if they are enabled when the SURM starts. Table 3-7 lists the error message categories. Refer to Appendix A, *SURM Error Messages*, for a complete error message listing.

Error Message Prefix	Description
E0xx	Fatal error. SURM terminated.
E1xx	Significant error. SURM completed but not all devices configured.
E2xx	Warning message.

Table 3-7. Error Message Categories.

3.4.5 Progress Message Area

The progress message area displays messages as the SURM executes. These messages normally flash by and cannot be read. However, if the SURM hangs, the displayed message is useful in finding the cause.

3.5 Running the SURM

The SURM should be run from the **AUTOEXEC.BAT** file. Syntax for invoking the SURM is:

C:\EPCONNEC\SURM [*options*]

where valid *options* are (if multiple options are specified, only the last *option* is valid):

/NO	Do not write system configuration information to **SURM.LOG**.
/PA	Always prompt.
/PD	Prompt if default symbolic names are assigned to any device or if any errors are detected.
/PE	Prompt if any errors are detected.
/PF	Prompt if fatal errors are detected.
/PN	Never prompt.
/PS	Prompt if significant (or fatal) errors detected.

For example, the following command causes the SURM to stop execution before it enters the termination phase and display the system configuration information if it assigned a default device name or if it detected an error.

C:\EPCONNEC\SURM /PD

This command line forces the SURM to stop execution before it enters the termination phase and to display the system configuration information.

C:\EPCONNEC\SURM /PA

3.5.1 SURM.RC

The **SURM.RC** file is the SURM runtime configuration file. It contains switches that direct SURM operation and treatment specifications that handles unique requirements of certain devices. In most cases, you will not be making changes to this file. It has the following syntax:

```
<line>                  :[<comment>
                         |<switch>
                         |<variable>
                         |<treatment-spec>]<newline>
<comment>               :"#"<non-newline-character>*
<switch>                :<identifier>
<variable>              :<identifier>=<value>
<identifier>            :<letter>[(<letter>|<digit>|"_")*]
<treament-spec>         :<device-id>":"<item>[","<item>]*
<device-id>             :<manuf-code>"."<model-code>
<manuf-code>            :<number>
<model-code>            :<number>
<number>                :<digit>*
<item>                  :<device-switch>|<device-variable>
<device-switch>         :<identifier>
<device-variable>       :(<identifier>=<value>)|
                         (<identifier>=<manf-code>"."<model-code>)
<value>                 :<identifier>|([<sign>]<number>)
<sign>                  :"+"|"-"
```

3.5.2 Switches

The `<switch>` lines direct SURM operation but do not override options entered in the **AUTOEXEC.BAT** SURM command line. Figure 3-3 lists the valid switch entries, the equivalent command line options, and a description of the switch action:

Switch	Command Line Option	Action
TRANSLATE_LOG_FILE_ TO_ASCII (default)	-	Translates IBM extended ASCII characters in **SURM.LOG** into equivalent ASCII characters.
USE-TWELVE_BIT_ MODEL_CODES	-	Truncate all model codes to 12-bits, including model codes read from the **VXIMODEL** file. Useful for certain VXI devices whose upper four bits of the model code are wrong or unpredictable.
NO_OLRM	**/NO**	Do not initialize OLRM, and do not write system configuration to **SURM.LOG**. OLRM will not operate.
IGNORE_OLRM_ERRORS	-	Do not report errors that occur during initialization.

Table 3-8. Valid Switch Entries for SURM.RC.

Switch	Equivalent Command Line Option	Action
DC_BEFORE_CHILD_ BUSSES	-	Perform dynamic configuration before configuring any child busses.
DO_MX_PRESCAN (default)	-	Scan for mainframe extenders before configuring any bus and close all found windows. Use when running SURM without removing power from VXI mainframes.
NEVER_SEND_ANO	-	Prevents the SURM from sending ABORT NORMAL OPERATION (ANO) to return devices to the CONFIGURE state. Send ANO is the default.
PROMPT_ALWAYS (default)	/PA	Always display SURM screen before exiting.
PROMPT_IF_DEFAULT _NAMES_USED	/pd	Display SURM screen if any default symbolic names are assigned any device or if any errors are detected.
PROMPT_IF_ANY_ERRORS	/pe	Display SURM screen if any errors are detected.
PROMPT_IF_FATAL	/pf	Display SURM screen only if fatal errors are detected.
NEVER_PROMPT	/pn	Never display SURM screen.
PROMPT_IF_FATAL_ OR_SIGNIFICANT	/ps	Display SURM screen only if fatal or significant errors are detected.

Table 3-8. Valid Switch Entries for SURM.RC (*continued*).

3.5.3 Variables

The <variable> line sets timeout limits for selftest and word serial communications. Table 3-9 lists the valid <variable> constants.

Variable	Valid Values	Definition
NO_MODID_OK	EPC-8, EPC-6, EPC-5, EPC-4	Lists the EPCs that don't have MODID capability.
SELFTEST_TIMEOUT= (default is 30)	Any decimal number	Specifies the number of seconds to wait for a device selftest to pass. SURM forces this value to be at least 5.
WORD_SERIAL_ TIMEOUT= (default is 30)	Any decimal number	Specifies the number of seconds to wait before generating a word serial communications timeout. Set it large enough to allow all servants to boot (see VXI specification 1.4)

Table 3-9. Valid Variables for SURM.RC

3.5.4 Treatment Specifications

The `<treatment-spec>` line configures system devices according to `<device-switch>` and `<device-variable>`. The device to configure is identified by `<device-id>`. Treatment specifications apply to all devices in the system with the specified `<device-id>`. These treatment specifications allow the SURM to handle unique features of certain VXI devices during configuration.

Table 3-10 lists the valid `<device-switch>` constants and Table 3-11 lists the valid `<device-variable>` constants.

Device Switch	Action
NO_WS_COMMANDS	SURM does not send most word serial commands to the device. Only the BEGIN NORMAL OPERATION command is sent. No response is collected.
CONTRL_REG_RMW	Causes SURM to access the device control register with read-modify-write cycles. Device dependent bits remain unchanged after the VXI defined bits are modified.
IGNORE_WS_ERRORS	SURM ignores word serial command errors during interrupt configuration.
CONTROL_REG_DEF_ZERO	SURM writes zeros to the device dependent bits when information is written to its control registers.
OFFSET_REG_EPC	Applies special memory handling procedures to the device. Set by the default **SURM.RC** file where necessary.
ASSUME_MESSAGE_DEVICE	SURM treats the device as a message-based device regardless of its device class bits.
EXTENDED_SELFTEST_OK	Do not warn about the device being in extended selftest.

Table 3-10. Valid Device Switches for SURM.RC.

Device Variable	Valid Values	Definition
ASSUME_HANDLERS	0 to 7	The number of handlers assigned to this device
ASSUME_INTERRUPTERS	0 to 7	The number of interrupters assigned to this device.
SLOT_0_DEVICE	RADISYS_SZM	The type of slot zero controller
REPLACEMENT_IDS	Any arbitrary manufacturer model number	Replaces the manufacturer model number with another arbitrary pair. When used, no other device variable or device switch is valid.

Table 3-11. Valid Device Variables for SURM.RC.

3.6 Changing System Device Names

You can temporarily change the name of any system device before the SURM enters the exit operation. Use the Configurator to make the name change permanent. To temporarily change a device name:

1. From the Extended Key usage block, choose F3.

2. Enter the device name to change and press enter. The SURM displays an error if the device name to change is not valid.

3. Enter the new device name and press enter.

4. Verify the new device name in the VXI System Configuration report.

3.7 Obtaining Help

Help contains information about error messages, the VXI System Configuration report, the runtime configuration file (**SURM.RC**), and the SURM command line options. To obtain help, press F1.

3.8 Quitting SURM

By default, the SURM stops execution and displays system configuration information before it writes this information to **SURM.LOG** and **SURM.ERR**. To exit the SURM program:

1. Press the ESC key until the SURM displays the Extended Key usage block.

2. Press the ESC key.

3.9 Troubleshooting

The SURM performs error checking but under rare circumstances the SURM cannot complete execution and hangs. You can recognize this situation when the screen still displays the SURM banner after about 30 seconds. Thirty seconds is an approximate interval. In most cases, the interval to wait depends on the longest device timeout.

3.9.1 Probable Causes

When the SURM hangs, the following lists the most common causes:

1. The EPC is not properly seated in the backplane.
2. The EPC has not been configured as the slot-0 controller (e.g., VX-MXI is not seated). [tempting to leave this in ... NOT!]
3. The system contains one or more devices that are also configured as the slot-0 controllers.
4. There is a fault on the VXIbus backplane. Check the bus-grant, bus-request, and SYSCLK lines.

3.9.2 Displaying Progress Messages

Progress messages report SURM progress and decisions it makes as it executes. Normally they flash by and you cannot read them, unless the SURM stop execution. To add the progress messages to the **SURM.ERR** file, execute the SURM with this command line:

surm /v [*options*]

3.9.3 Mainframe Extenders

The SURM routes memory regions to devices in remote mainframes if the **DEVICES** file contains a bus number in a device record. If there is no bus number, SURM assumes that the device is in the root mainframe. The Configurator cannot set the bus number.

Each VXIbus or MXIbus has a unique identification that refers to the physical location of devices. The bus identification of the root bus is zero. The identification of the non-root bus is equal to the ULA of the mainframe extender which connects the non-root bus to its parent bus.

For each mainframe extender, the SURM sets ACFIN, ACFOUT, SFIN, SFOUT, SRIN, and SROUT. This forwards the utility bus in both directions. Refer to your mainframe extender manual for additional information.

All interrupt lines are forwarded across every mainframe extender that has an interrupt configuration register. Interrupt lines are forwarded upward unless the handler for an interrupt line is located downward from the mainframe extender being configured. The result is that each interrupt handler can be interrupted from any bus.

The SURM does not perform trigger configuration.

For additional information about configuring mainframe extenders refer to Appendix C, *Configuring Multi-Mainframe Systems.*

NOTES

4. Using the Configurator

4.1 Introduction

The Configurator is an interactive Windows application that defines devices that EPConnect programs can access. It is especially useful with systems that also contain GPIB instruments and/or VME modules. Use the Configurator to:

- Name system devices.
- Set system device parameters that cannot be obtained dynamically.
- Set commander/servant hierarchy.
- Map interrupters and handlers to devices.
- Modify manufacturer codes.
- Modify manufacturer model codes.
- Set up the VXLink interface.

The Configurator modifies the **DEVICES** file which then communicates device names and parameters to EPConnect programs and the SURM. The SURM uses **DEVICES** file information to initialize system devices. In addition, the Configurator can modify the manufacturer code and manufacturer model database files **C:\EPCONNEC\DB\VXIMANUF** and **C:\EPCONNEC\DB\VXIMODEL**, respectively. For more information about the SURM refer to Chapter 3, *Using the Start-Up Resource Manager*. For more information about the **DEVICES** file refer to Appendix B, *DEVICES File*.

4.2 Device Naming

To make your system easier to manage, give all the system devices unique names rather than using the default names assigned by the SURM. When assigning names to any device the device names must conform to IEEE 488.2 naming conventions. According to these conventions, device names must:

- Begin with a letter and not end with an underscore.
- Be between 1 to 12 characters in length.
- Contains only letters, digits, and underscore.

Once a name is assigned to a device, that name may be used throughout EPConnect to refer to the device. The assigned names do not take effect until the SURM is re-run.

4.3 Configurator Start-up Window

Figure 4-1 shows the Configurator start-up window. It consists of five pull-down menus, the Windows Control-menu, and a help menu.

The File menu allows you to exit the Configurator window application and to display information about the Configurator.

The Devices menu allows you to name and configure VXI, VME, and, GPIB.

The Interface menu allows you to test, set up or modify the settings of the VXLink interface.

The VXI Control menu allows you to configure the VXI commander/servant hierarchy. You can also map interrupters and interrupt handlers to the VXI interrupt lines.

The Database menu allows you to modify the manufacturer code database and the manufacturer model name database and manufacturer model number database.

The Help menu provides online information for the Configurator.

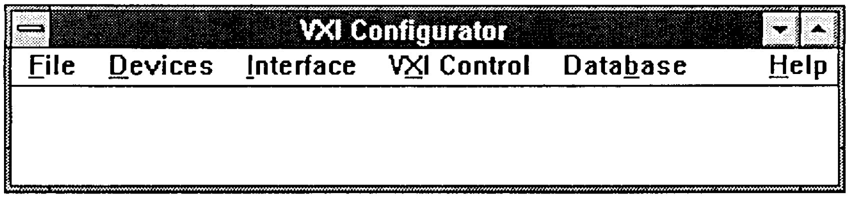


Figure 4-1. Configurator Start-up Window.

4.4 Configuring the VXLink Interface

To configure the VXLink interface, choose the VXLink item under the Interface menu. The "Edit VXLink Interface" dialog box displays.

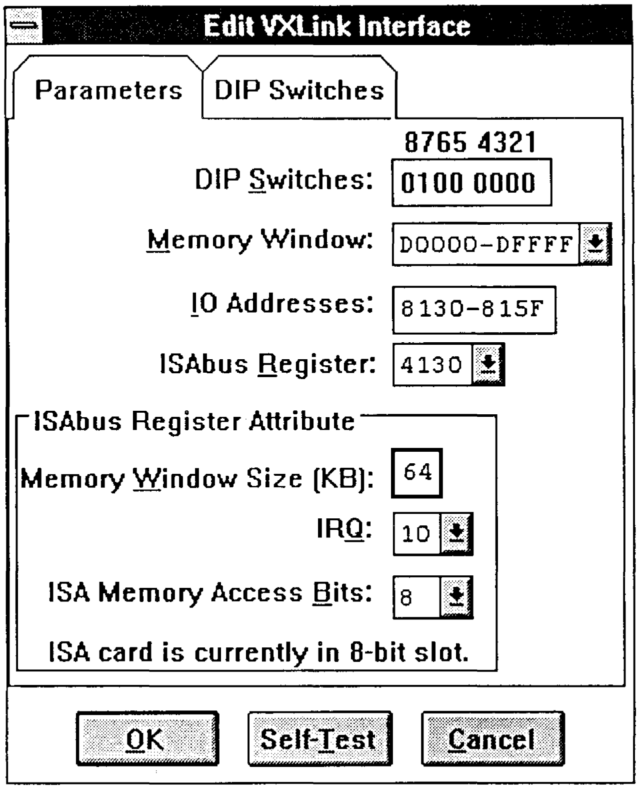
The configuration of a VXLink ISA card is determined by the DIP Switch settings on the ISA card. The ISA card settings are reflected in the Windows **SYSTEM.INI** file in the [vxlink0] section. These settings must match for the VXLink interface to operate properly.

A "self-test" button is provided in the dialog box to verify that the system works properly with the chosen settings.

You can use the VXLink dialog box to experiment with settings and configurations and to determine what the optimal DIP switch settings should be. Once you select the proper settings, the dialog box updates the SYSTEM.INI file. You should then exit Windows and configure the DIP switches according to your selected values in the dialog box. Once you restart Windows, the VXLink interface then operates according to your selections.

For more information, refer to the VXLink Interface (ISA-to-VXI) HP E1383/E1483A Hardware Reference Manual (RadiSys part number 07-0253-xx).

The VXLink interface dialog box displays as follows:



Edit VXLink Interface

```
                    8765 4321
DIP Switches:       0100 0000
Memory Window:      D0000-DFFFF
IO Addresses:       8130-815F
ISAbus Register:    4130

ISAbus Register Attribute
Memory Window Size (KB):   64
IRQ:                       10
ISA Memory Access Bits:    8
ISA card is currently in 8-bit slot.
```

Figure 4-2. VXLink Interface Dialog Box.

The dialog box displays the ISA card's parameters as defined in the **SYSTEM.INI** file in the section [vxlink0]. If the **SYSTEM.INI** file has not been set for the VXLink interface, the factory default settings display.

DIP SWITCH	0100 0000
Memory Address	0xD0000
IO Address	8130
IRQ	10
Memory Window Size	64
ISA Memory access bits	8

When you select the OK button, the [vxlink0] section of the **SYSTEM.INI** file is updated to reflect the entries in this dialog box. The old **SYSTEM.INI** file is saved as **SYSTEM.SAV**.

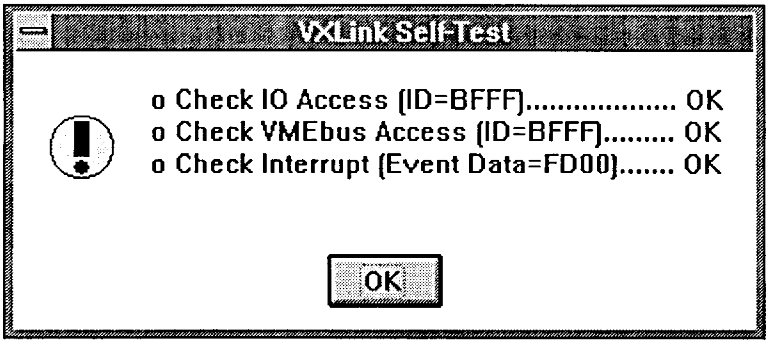
There are two sections to the dialog box: parameters and DIP switch. Click on the parameters tab to make changes to the memory window and ISAbus register.

Click on the DIP Switches tab to make changes to the DIP Switch. Note that the software cannot actually change the DIP switch on the ISA card -- this must be done manually. As you select different DIP switch settings, the resulting changes to the memory window, IO address, and ISAbus register are displayed.

Running the Self-Test

When the "Edit VXLink Interface" dialog box opens, you can run the self-test to verify the current settings and the connection are valid.

The Self-Test runs three tests: IO access, VMEbus access, and interrupt handling. The following displays:



```
o Check IO Access (ID=BFFF).................. OK
o Check VMEbus Access (ID=BFFF)......... OK
o Check Interrupt (Event Data=FD00)....... OK
```

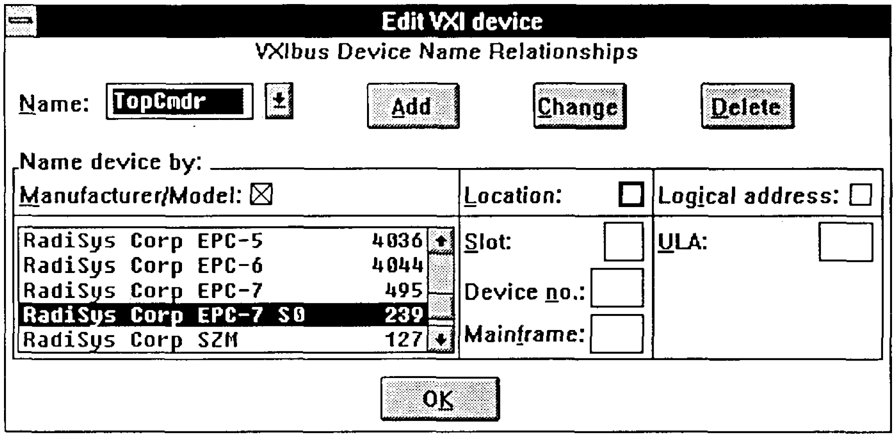
Figure 4-3. VXLink Self-Test.

If you make changes but do not want to accept them, click on the Cancel button or close the window. If you want your changes to be written to the **SYSTEM.INI** file, click on the OK button. Note that the changes do not take effect until you restart Windows. The self-test button disappears once changes are made, and cannot be run until Windows is restarted.

After you click OK, messages display informing you that restarting Windows is required. You should also manually change the DIP switches once you exit Windows. The DIP switches are readily accessible on the ISA card.

4.5 Naming VXI Devices

Selecting the VXI item from the Devices menu displays the Edit VXI device dialog box (see Figure 4-1). This dialog box names or renames VXI devices. It is not necessary that the device physically exist in the system when you name it. This is particularly useful when the system configuration often changes.


Edit VXI device
VXIbus Device Name Relationships
Name: TopCmdr
Name device by:
Manufacturer/Model: ☒
Location: ☐
Logical address: ☐

```
RadiSys Corp EPC-5      4036
RadiSys Corp EPC-6      4044
RadiSys Corp EPC-7       495
RadiSys Corp EPC-7 S0    239
RadiSys Corp SZM         127
```

Slot:
Device no.:
Mainframe:
ULA:

Figure 4-4. Edit VXI Device Dialog Box.

The Edit VXI device dialog box contains two main boxes: the Name list box and the Name device by box. The Name list box lists the names of all defined devices. It is also the box you use to enter a new device's name. The Name device by box contains three naming options. You can use none, one, two, or three of the naming options.

Named VXI devices do not consume bus resources when they are not physically in the system, so you can name many VXI devices without affecting bus resources. The device being named is selected by specifying its manufacturer/model pair, its physical location in the system, its ULA, or any combination of these selections.

When using manufacturer/model to name the device, the manufacturer name and model must be in the database. If not, use the Database menu to add the manufacturer and model names.

When using Location to name the device you must also specify at least one of the location options (Slot, Device no., or Mainframe). Slot is the physical location of the device in the mainframe. Device no. identifies the number of the device in a multi-device module (the first device is number zero). Mainframe is the ULA of the mainframe extender that contains the device.

It is not recommended to use the Logical address method of naming dynamically configured VXI devices because the device ULA is not predictable.

4.5.1 Adding a VXI Device

To add a VXI device:

1. Select the VXI item from the Devices menu.
2. Enter the new device name in the Name list box.
3. Optionally select the device naming method and enter the appropriate attributes.
4. Choose the Add button.
5. Choose the OK button to update the **DEVICES** file.

4.5.2 Changing a VXI Device Name or Naming Method

To change a VXI device name or its naming method:

1. Select the VXI item from the Devices menu.
2. Select the device name to change from the Name list box.

3. Select the new device naming method and enter the appropriate attributes.
4. Choose the Change button.
5. Choose the OK button to update the **DEVICES** file.

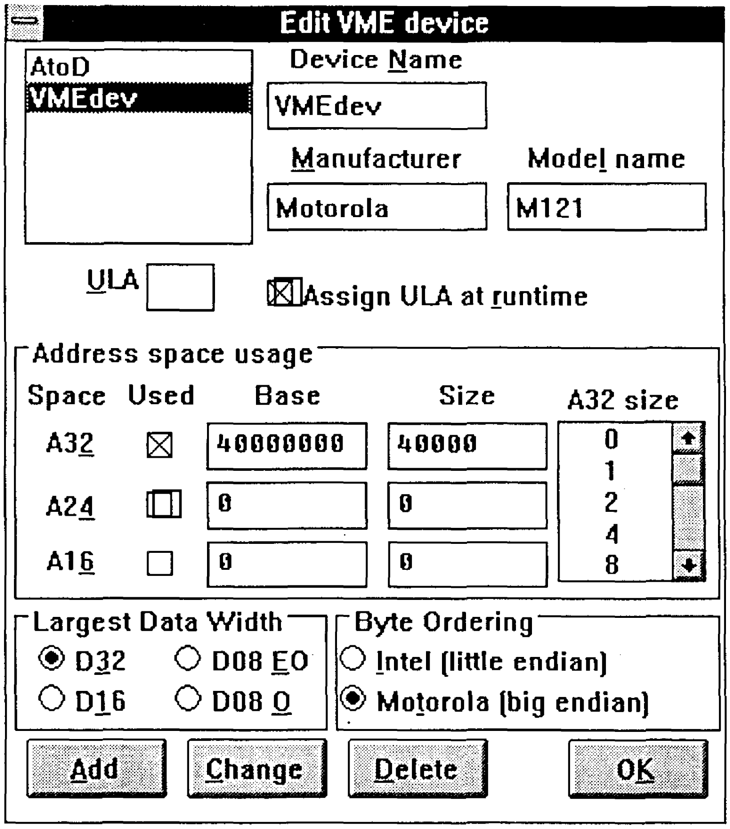
4.5.3 Deleting a VXI Device

To delete a VXI device:

1. Select the VXI item from the Devices menu.
2. Select the device to delete from the Name list box.
3. Choose the Delete button.
4. Choose the OK button to update the **DEVICES** file.

4.6 Configuring VME Devices

Selecting the VME item from the Devices menu displays the Edit VME device dialog box (see Figure 4-1). Use this dialog box to specify the VME module's attributes. Unlike VXI devices, VME modules do not include configuration registers that specify the device's system memory requirements, manufacturer code or model ID. This dialog box also assigns names and logical addresses to VME devices. VME devices consume system resources when not installed. Therefore, it is good practice to only name VME devices that are physically in your system.



Edit VME device

AtoD
VMEdev

Device Name: VMEdev

Manufacturer: Motorola

Model name: M121

ULA:

[✓ Assign ULA at runtime]

Address space usage

Space	Used	Base	Size
A32	✓	40000000	40000
A24		0	0
A16		0	0

A32 size: 0, 1, 2, 4, 8

Largest Data Width: [✓ D32] D16 D08 EO D08 O

Byte Ordering: Intel (little endian) [✓ Motorola (big endian)]

Figure 4-5. Edit VME Device Dialog Box.

Entries in the Manufacturer and Model name text boxes are for reference only and are not used for name matching by EPConnect software.

The ULA box and Assign ULA at runtime are mutually exclusive.

Supplying a VME module's memory requirements is very important since it allows the SURM to avoid any conflicts when assigning memory to the VXI devices in the system. When you move to a Base or Size text box, the name and contents of the scroll box changes to reflect the selected address space and its valid values.

The Largest Data Width buttons display and select the device's data width.

The Byte Ordering buttons display and select the device's byte ordering scheme.

4.6.1 Adding a VME Device

To add a VME device:

1. Select the VME item from the Devices menu.
2. Enter the new device name in the Device Name text box.
3. Optionally enter a manufacturer and model name.
4. Either enter a ULA or check the Assign ULA at runtime box. The OLRM uses this ULA to obtain device information.
5. Enter address space usage information.
6. Select the byte ordering and the largest data width supported by the device..
7. Choose the Add button.
8. Choose the OK button to update the **DEVICES** file.

4.6.2 Changing a VME Device Name or Parameters

To change a VME device name or its parameters:

1. Select the VME item from the Devices menu.
2. Select the device to change from the device list box.
3. Enter the new device name and/or other information.
4. Choose the Change button.
5. Choose the OK button to update the **DEVICES** file.

4.6.3 Deleting a VME Device

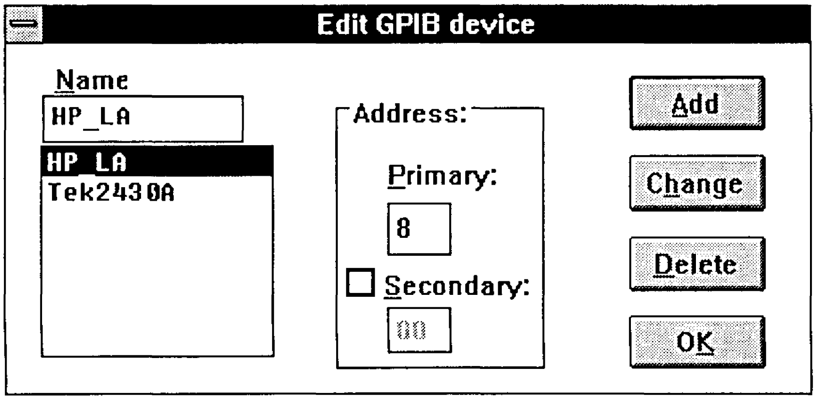
To delete a VME device:

1. Select the VME item from the Devices menu.
2. Select the device to delete from the list box.
3. Choose the Delete button.
4. Choose the OK button to update the **DEVICES** file.

4.7 Configuring Devices

Selecting the GPIB item from the Devices menu displays the Edit GPIB device dialog box (see Figure 4-1). This dialog box allows you to configure a GPIB (IEEE 488) device by setting its names and address. It also allows you to edit this same information for previously defined GPIB devices.

GPIB devices named or edited with this dialog box can be controlled by both the SICL for DOS and SICL for Windows APIs. Refer to the *SICL for DOS Programmer's Reference Manual* or the *SICL for Windows Programmer's Reference Manual* for more information.


Edit GPIB device

Name: HP_LA

HP_LA
Tek2430A

Address:
Primary: 8
☐ Secondary: 00

Figure 4-6. Edit GPIB Device Dialog Box.

Use the Name box to enter a new GPIB device name.

The list box lists all named GPIB devices. Selecting one of the listed devices displays the current device parameters.

Use the Address box to select the GPIB primary and secondary addresses.

4.7.1 Adding a GPIB Device

To add a GPIB device:

1. Select the GPIB item from the Devices menu.
2. Enter the new device name in the Name text box.
3. Enter the GPIB device address (valid values for Primary and Secondary are 0 to 30).
4. Choose the Add button.
5. Choose the OK button to update the **DEVICES** file.

4.7.2 Changing a GPIB Device Name or Parameters

To change a GPIB device name or its parameters:

1. Select the GPIB item from the Devices menu.
2. Select the device name to change from the Name list box.
3. Enter the new device name and/or other information.
4. Choose the Change button.
5. Choose the OK button to update the **DEVICES** file.

4.7.3 Deleting a GPIB Device

To delete a GPIB device:

1. Select the GPIB item from the Devices menu.
2. Select the device name to delete from the Name list box.
3. Choose the Delete button.
4. Choose the OK button to update the **DEVICES** file.

4.8 Setting Commander/Servant Hierarchy

The Commander Hierarchy item from the VXI Control menu causes Configurator to display the Commander/Servant hierarchy dialog box (see Figure 4-1). This dialog box defines and deletes a commander and its servants, adds or deletes a servant, and displays the commander of a selected device. The Commander/Servant hierarchy dialog box has three list boxes: one lists all the system commanders, one that lists all the selected commander's servants, and one that lists all known system devices.

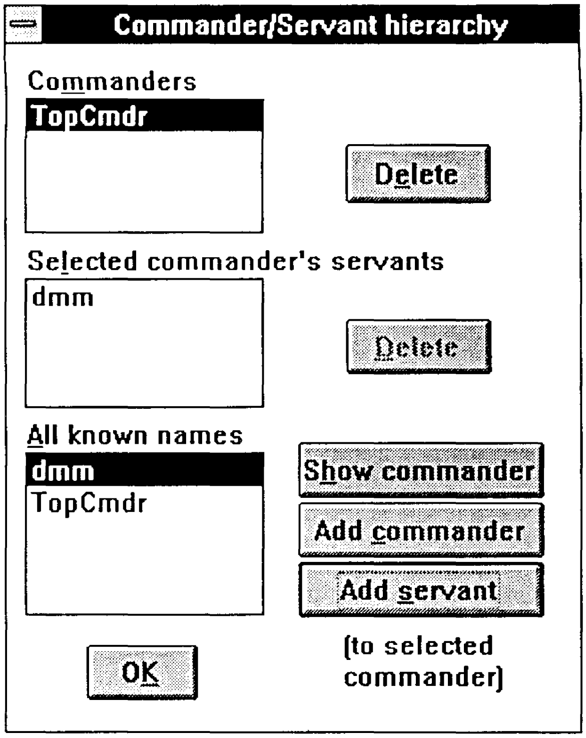


Figure 4-7. Commander/Servant hierarchy Dialog Box.

4.8.1 Adding a Commander

To add a commander:

1. Select the Commander Hierarchy item from the VXI Control menu.

2. Select the commander to add from the All known names list.

3. Choose the Add commander button.

4. Choose the OK button to update the **DEVICES** file.

4.8.2 Adding a Servant

To add a servant to a selected commander:

1. Select the Commander Hierarchy item from the VXI Control menu.
2. Select the servant to add from the All know names list.
3. Select the commander from the Commanders list.
4. Choose the Add servant button.
5. Choose the OK button to update the **DEVICES** file.

4.8.3 Displaying a Commander's Servants

To display the commander's servants:

1. Select the Commander Hierarchy item from the VXI Control menu.
2. Select the commander from the Commanders list.
3. Choose the Show commander button.

4.8.4 Displaying a Servant's Commander

To display a servant's commander:

1. Select the Commander Hierarchy item from the VXI Control menu.
2. Select a servant from the All known names list.
3. Choose the Show commander button.

4.8.5 Deleting a Commander

To delete a commander:

1. Select the Commander Hierarchy item from the VXI Control menu.
2. Select the commander from the Commanders list.
3. Choose the Delete button next to the Commanders list box.
4. Choose the OK button to update the **DEVICES** file.

4.8.6 Deleting a Servant

To delete a servant:

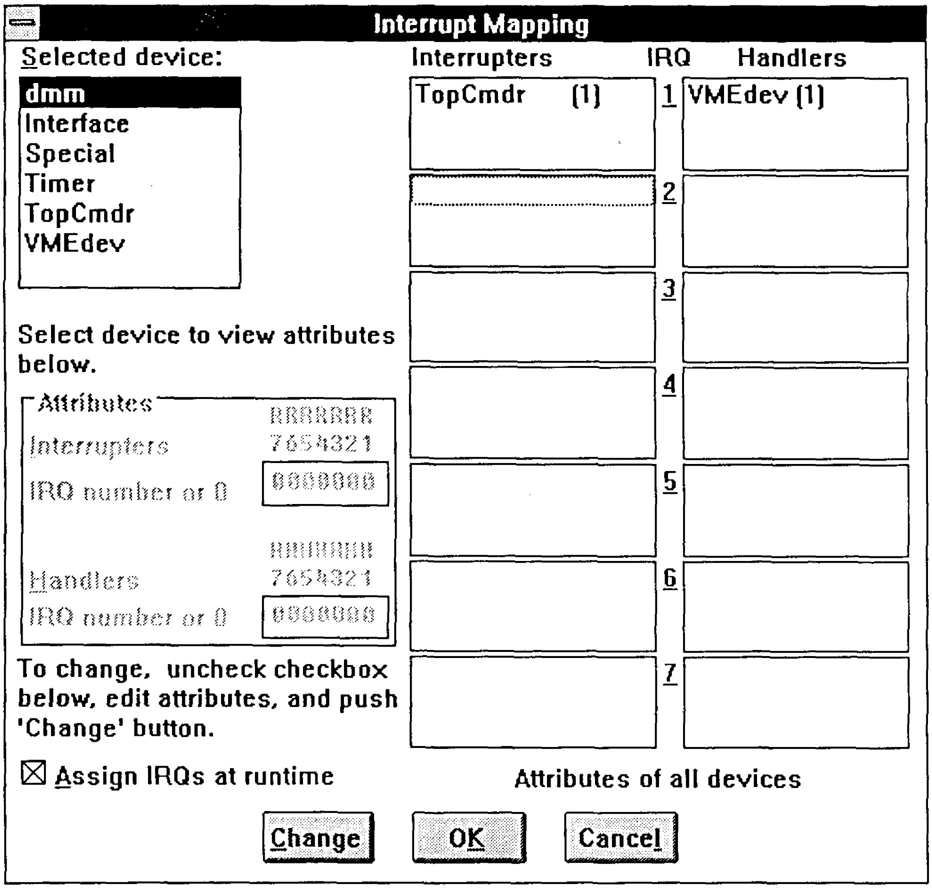
1. Select the Commander Hierarchy item from the VXI Control menu.
2. Select the commander from the Commanders list box.
3. Select the servant from the Selected commander's servants list.
4. Choose the Delete button next to the Selected commander's servants list box.
5. Choose the OK button to update the **DEVICES** file.

4.9 Interrupter and Handler Mapping

Selecting the Interrupt Mapping item from the VXI Control menu displays the Interrupt Mapping dialog box (see Figure 4-1). This dialog box displays interrupter and interrupt handler mapping to VXIbus IRQ lines for the system devices and allows you to edit these mappings.. This mapping information is used by the SURM to override the default configuration when assigning VXI interrupt lines to interrupters and interrupt handlers at system startup.

The left half of the dialog box displays interrupter and handler mapping for the device selected in the Selected device list box, if the Assign IRQs at runtime box is not checked. If the box is checked, the mappings in the Attributes box are dimmed, indicating that you cannot change them.

The right half of the dialog box displays how the VXIbus IRQ lines are mapped to interrupters and handlers for all devices. The display is the current state and includes any changes you have entered.



Interrupt Mapping

Selected device:

- dmm
- Interface
- Special
- Timer
- TopCmdr
- VMEdev

Select device to view attributes below.

Attributes

Interrupters	RRRRRRR 7654321	
IRQ number or 0	0000000	
Handlers	HHHHHHH 7654321	
IRQ number or 0	0000000	

To change, uncheck checkbox below, edit attributes, and push 'Change' button.

[✓] Assign IRQs at runtime

Interrupters	IRQ	Handlers
TopCmdr (1)	1	VMEdev (1)
	2	
	3	
	4	
	5	
	6	
	7	

Attributes of all devices

Figure 4-8. Interrupt Mapping Dialog Box.

To change interrupter or interrupt handler mapping:

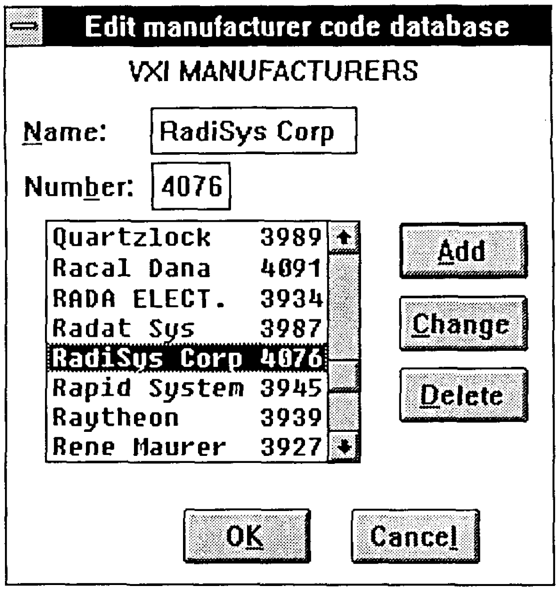
1. Select the device to change interrupter and/or handler mapping from the Select device list box.
2. Uncheck the Assign IRQs at runtime check box.
3. To change an interrupter to IRQ mapping, move to the Interrupters text box and enter the IRQ number to map to the interrupter below the corresponding interrupter number. .
4. To change a handler to IRQ mapping, move to the Handlers text box and enter the IRQ number to map to the handler below the corresponding interrupter number.
5. Choose the Change button.
6. Choose the OK button to record the changes.

4.10 Editing the Manufacturer Code Database

Selecting the Manufacturers item from the Database menu displays the Edit manufacturer code database dialog box (see Figure 4-1). This dialog box adds, changes, or deletes the names and numbers of VXIbus device manufacturers stored in the file **C:\EPCONNEC\DB\VXIMANUF**.

The manufacturers list box displays the most recent contents of the database file plus any changes you have entered. In addition, the Configurator creates the file **VXIMANUF.BAK** that preserves the original file.

Manufacturer names are case sensitive and are limited to 12 characters. Manufacturer codes are limited to four characters.



Name: RadiSys Corp

Number: 4076

```
Quartzlock     3989
Racal Dana     4091
RADA ELECT.    3934
Radat Sys      3987
RadiSys Corp   4076
Rapid System   3945
Raytheon       3939
Rene Maurer    3927
```

Figure 4-9. Edit Manufacturer Code Database Dialog Box.

4.10.1 Adding a Manufacturer Code

To add a manufacturer:

1. Enter the manufacturer's name.
2. Enter the manufacturer's ID number, in decimal. Manufacturer IDs are assigned by the VXIbus consortium.
3. Choose the Add button.
4. Choose the OK button to update the database file.

4.10.2 Changing a Manufacturer Code

To change a manufacturer or manufacturer number:

1. Select the manufacturer name/number to change from the list box.
2. Enter the new manufacturer name and/or the new manufacturer's number.
3. Choose the Change button.
4. Choose the OK button to update the database file.

4.10.3 Deleting a Manufacturer Code

To delete a manufacturer name/number:

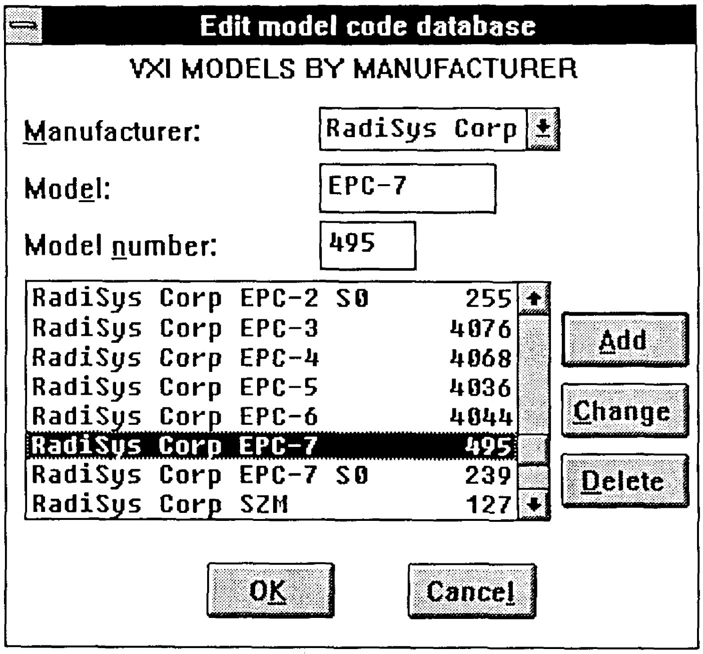
1. Select the manufacturer code to delete from the list box.
2. Choose the Delete button.
3. Choose the OK button to update the database file.

4.11 Editing the Model Code Database

Selecting the Model item from the Database menu displays the Edit model code database dialog box (see Figure 4-1). This dialog box adds, changes, or deletes manufacturer model name and model numbers of VXI devices stored in the database file **C:\EPCONNEC\DB\VXIMODEL**.

The Manufacturer drop-down list box contains all the manufacturers described in the database file **C:\EPCONNEC\DB\VXIMANUF** (you cannot change the contents of this box). The main list box displays the current contents of the database file plus any changes you have entered. In addition, the Configurator creates the file **VXIMODEL.BAK** that preserves the original file.

Manufacturer models are limited to 10 characters and manufacturer model number are limited to five characters.



VXI MODELS BY MANUFACTURER

Manufacturer: RadiSys Corp

Model: EPC-7

Model number: 495

```
RadiSys Corp EPC-2 S0       255
RadiSys Corp EPC-3         4076
RadiSys Corp EPC-4         4068
RadiSys Corp EPC-5         4036
RadiSys Corp EPC-6         4044
RadiSys Corp EPC-7          495
RadiSys Corp EPC-7 S0       239
RadiSys Corp SZM            127
```

Figure 4-10. Edit Model Code Database Dialog Box.

4.11.1 Adding A Model Code

To add a model code:

1. Select the manufacturer from the Manufacturer list box or the main list box. If the manufacturer is not listed, edit the manufacturer code database to add the new manufacturer.

2. Enter the new model and model numbers. Model numbers are assigned by the device's manufacturer and is usually included in the device documentation.

3. Choose the Add button.

4. Choose the OK button to update the database.

4.11.2 Changing a Model Code

To change a device model name and/or model number::

1. Select the model to change from the main list box.
2. Enter the new model name and/or model number.
3. Choose the Change button.
4. Choose the OK button to update the database.

4.11.3 Deleting a Model Code

To delete a model name/number pair:

1. Select the model and name/number pair to delete from the main list box.
2. Choose the Delete button.
4. Choose the OK button to update the database file.

5. Using BusMonitor

5.1 Introduction

BusMonitor is a Windows application that displays VXIbus backplane signal and EPC interface information. Its primary use is to run concurrently with VXI applications to aid debugging during system configuration and software development. BusMonitor is an output-only application and cannot change the value of any bus information it displays. However, you can change the rate at which it samples the VXIbus. BusMonitor displays:

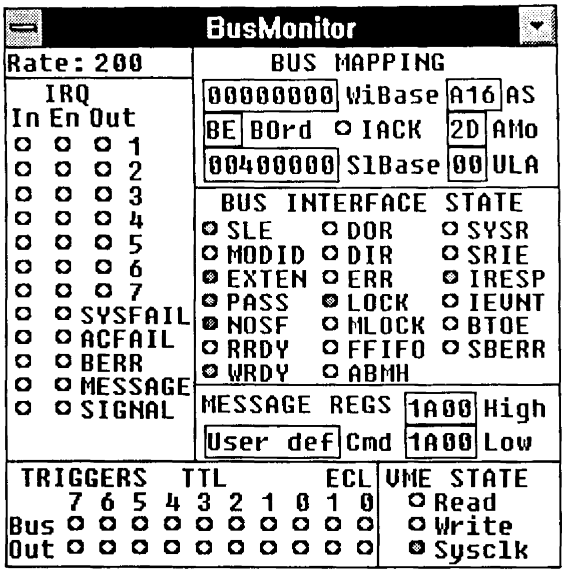
- The state of EPC address mapping hardware.
- The state of 20 VXIbus interface signal.
- The contents of EPC word serial registers.
- The state of TTL and ECL triggers.
- The states of SYSFAIL, ACFAIL, and BERR.
- The state of VXIbus interrupt lines.
- The VXIbus sample rate.

5.2 The BusMonitor Window

Figure 5-1 shows the BusMonitor window. It displays VXIbus information in seven boxes:

- Bus Mapping
- Bus Interface State
- Message Regs
- VME State
- Triggers
- IRQ
- Rate

Within the boxes, simulated LEDs represent signal states and numeric readouts provide register information. When an LED is on, it represents an asserted condition, regardless of the hardware level that represents an assertion. Color monitors display an on LED as red, green, or yellow. A red LED represents an error condition, a yellow LED represents an unusual assertion, and a green LED represents a normal assertion. On monochrome monitors, only signal asserted or signal not asserted conditions are displayed. Numeric readouts are hexadecimal, except in the Rate dialog box, where the readout is decimal.



```
                         BusMonitor
Rate: 200       |             BUS MAPPING
    IRQ         | [00000000] WiBase [A16] AS
In En Out       | [BE] BOrd  o IACK [2D] AMo
o  o  o 1       | [00400000] SlBase [00] ULA
o  o  o 2       |
o  o  o 3       |     BUS INTERFACE STATE
o  o  o 4       | ● SLE      o DOR    o SYSR
o  o  o 5       | o MODID    o DIR    o SRIE
o  o  o 6       | ● EXTEN    o ERR    ● IRESP
o  o  o 7       | ● PASS     ● LOCK   o IEUNT
o  o SYSFAIL    | ● NOSF     o MLOCK  o BTOE
o  o ACFAIL     | o RRDY     o FFIFO  o SBERR
o  o BERR       | ● WRDY     o ABMH
o  o MESSAGE    | MESSAGE REGS [1A00] High
o  o SIGNAL     | [User def] Cmd [1A00] Low

TRIGGERS  TTL           ECL | VME STATE
    7 6 5 4 3 2 1 0  1 0    |  o Read
Bus o o o o o o o o  o o    |  o Write
Out o o o o o o o o  o o    |  ● Sysclk
```

Figure 5-1. BusMonitor Window.

5.2.1 Bus Mapping

The BUS MAPPING box displays the state of the EPC's VXIbus master and slave hardware. Table 5-1 describes each entry in the BUS MAPPING box.

Entry	Definition	Description
WiBase	Window base address	The current VXIbus base address of the E-page window. The base address must begin on 64 KB boundaries. *(This value is defined by the VxD and does not reflect VXIbus memory mappings.)*
AS	Address space	Valid values are A32, A24, A16 and ? (unknown address space). *(This value is defined by the VxD and does not reflect VXIbus memory mappings.)*
BOrd	Byte order	Valid values are LE (little-endian) for INTEL microprocessors and BE (big-endian) for Motorola microprocessors. *(This value is defined by the VxD and does not reflect VXIbus memory mappings.)*
IACK	Interrupt acknowledge	On when an interrupt acknowledge (IACK) is in process.
AMo	Address modifier	The hexadecimal encoding of the VXIbus address modifier used in VXIbus accesses from the EPC. *(This value is defined by the VxD and does not reflect VXIbus memory mappings.)*
SlBase	Slave base address	The base VXIbus address to which the EPC maps its on-board memory so that other VXIbus masters can access it. *(Grayed out on an EPC-8 or if slave memory is disabled.)*
ULA	Unique logical address	The EPC's unique logical address assigned to the EPC.

Table 5-1. Bus Mapping Box.

5.2.2 Bus Interface State

The BUS INTERFACE STATE box consists of 20 simulated LEDs that are on when the referenced interface is asserted. Table 5-1 describes each entry in the BUS INTERFACE STATE box.

Entry	Definition	Description
SLE	Slave memory enable	If lit, the EPC's VXIbus slave interface is enabled and the EPC can respond as a slave to incoming bus requests.
DOR	Data output ready	The EPC can respond to Byte Request commands.
SYSR	System reset	The SYSRESET line is being driven by the EPC.
MODID	Module identification	The EPC's MODID line is being driven.
DIR	Data input ready	The EPC is ready to accept Byte Available commands.
SRIE	SYSRESET input enable	The EPC is enabled to recognize the VXIbus SYSRESET signal.
EXTEN	Extended self-test	Extended self-test in progress.
ERR	Error	The EPC is asserting an error condition to its commander.
IRESP	Interrupt response	The EPC is ready to respond to an IACK cycle with a Response status/ID word. Always "on" on EPC-8.
PASS	-	The initial self-test passed.
LOCK	-	The EPC's commander is locked.
IEVNT	Interrupt event	The EPC will respond to an IACK cycle with an Event Status/ID word. Fixed as "off" on EPC-8.

Table 5-1. Bus Interface State Box.

Entry	Definition	Description
NOSF	Inhibit SYSFAIL line	The EPC cannot drive the VXIbus SYSFAIL line.
MLOCK	Message register lock	The EPC's word serial register is locked.
BTOE	Bus time out error	The EPC's bus time out timer is enabled.
RRDY	Read ready	The EPC word serial register contains output data.
FFIFO	Full FIFO	The EPC's signal register FIFO is full.
SBERR	Sticky-bus-error	Sticky-bus-error flag is set. This flag is set by a bus access that causes BERR to be asserted. Software clears it.
WRDY	Write ready	The EPC can accept VXIbus word serial commands.
ABMH	Access bit message high	On when the upper word serial data high register has been accessed. Always "off" on an EPC-8.

Table 5-2. Bus Interface State Box (*continued*).

5.2.3 Message Regs

The MESSAGE REGS box displays the contents of the EPC word serial registers. Table 5-3 describes each entry in the MESSAGE REGS box.

Entry	Description
Cmd	The symbolic name of the command in the word serial register. Other valid values are: Unknown = encoding is not defined User def = encoding is user defined
High	Word serial data high register contents. Fixed as "000s" on EPC-8.
Low	Word serial data low register contents.

Table 5-3. Message Regs Box.

5.2.4 VME State

The VME STATE box contains three simulated LEDs. Two LEDs display the VXIbus read/write activity and one LED turns on when the EPC's 16 MHz clock is active.

5.2.5 Triggers

The TRIGGERS box displays the state of the TTL triggers 0-7 and ECL triggers 0 and 1. When the simulated LED in the `Bus` row is on, the corresponding VXIbus trigger line is asserted. When the simulated LED in the `Out` row is on, the EPC is driving the corresponding VXIbus trigger line. Refer to the chart below for more information:

Trigger	Supported On:
TTL 0-7	EPC-7
ECL 0-1	EPC-7

Trigger LEDs are always OFF on all other machines.

5.2.6 IRQ

The IRQ box displays the state of the seven VXIbus interrupt lines, the SYSFAIL line, the ACFAIL line, the BERR line, the MESSAGE register, and only on VXI-SIGNAL-PRESENT events. VXI-SIGNAL-PRESENT is only valid for an EPC-2.

The `In` (left-hand) column displays which interrupts and events are asserted. The seven interrupts, SYSFAIL, and ACFAIL are bus wide conditions. BERR indicates that the last bus operation caused a bus error. MESSAGE indicates that the EPC word serial registers contain an unread message. SIGNAL indicates that this EPC's signal register queue is not empty.

The `En` (center column) displays which interrupts and events are enabled for this EPC.

The `Out` (right-hand column) displays which interrupt this EPC is asserting.

5.2.7 Rate Box

The rate box displays the interval at which the VXIbus is sampled for information. The sample interval is in milliseconds, with a default value of 200. The legal valid sample rate range is 18 to 65535 milliseconds. However, the lower limit is 54 milliseconds because the DOS clock operates at 18.2 Hz.

Changing the Sample Interval With the Mouse

1. Click the Control-menu box in the upper-left corner of the window.

2. From the Control-menu, choose Rate.

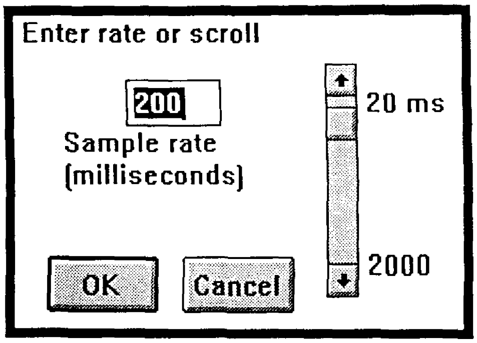


Figure 5-2. Rate Dialog Box.

3. In the Rate dialog box, enter the new sample rate or use the vertical scroll bar to select a new rate.

4. Choose the OK button.

Changing the Sample Interval With the Keyboard

1. Press ALT+SPACE to display the Control-menu.
2. Use the UP ARROW and DOWN ARROW keys to select Rate.
3. Press ENTER to display the rate dialog box.
4. In the Rate Dialog box, enter the new sample rate.
5. Press ENTER to apply the change.

Setting the Sample Interval Start-up Default

The sample interval start-up default is 200 milliseconds. To change it add the following lines to **WIN.INI** file, where *sampleinterval* is a decimal number that specifies the sample interval in milliseconds:

```
[busmon]
time=sampleinterval
```

6. Using BusProbe

6.1 Introduction

BusProbe is an interactive Windows application that directly controls the VXIbus and interacts with system devices. Its primary use is debugging during system configuration and software development

BusProbe interfaces with system devices using word serial or IEEE-488 messages (VXI menu) or a low-level interface (VME menu). It also controls system interrupts, system reset, triggers, and device initialization. These features reduce the need to write code specifically for debugging. BusProbe also maintains a log of user interactions which can be saved in a user-specified file.

6.2 The BusProbe Window

Figure 6-1 shows the BusProbe window at start-up. It consists of a text area and three pull-down menus: File, VME, and VXI. A Control-menu is also available.

The text area fills with lines that summarize requested BusProbe actions and their respective results. These lines be stored in ASCII format log file for later use or analysis. The form of the log file is identical to the text area display. BusProbe always appends requested actions and their respective results to the text area so no action is required to view the requested action and its results..

The File menu provides control of the log file in the text area. The VME menu provides access to the low level interface. The VXI menu provides access to the message interface.

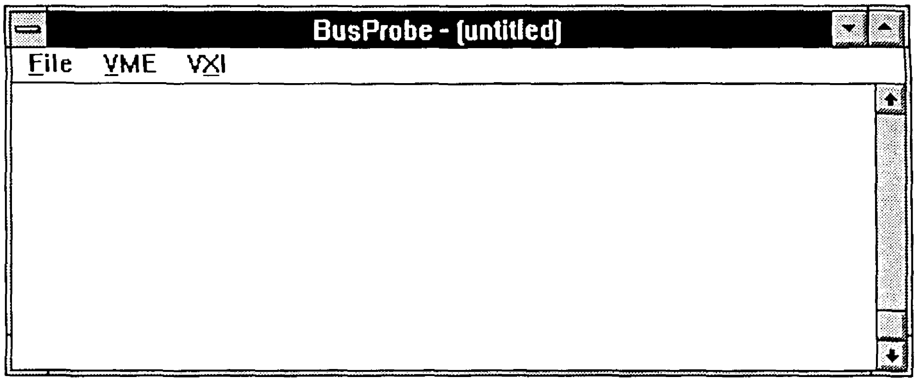
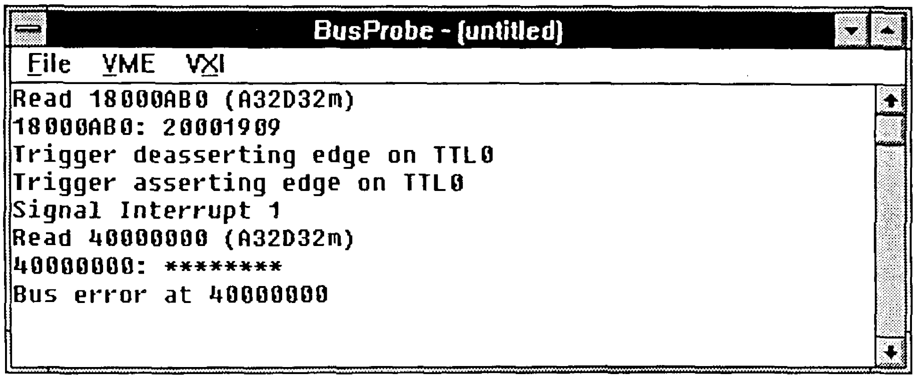


Figure 6-1. BusProbe Window.

6.3 Log files

Figure 6-2 shows a sample log file in the BusProbe window text area. Use File menu items to open a new log file, open an existing log file, or save the displayed text in a log file. BusProbe commands and results are always appended to the text area. The title bar changes from (untitled) to the file name when you save the text area or display an existing log file.



```
Read 18000AB0 (A32D32m)
18000AB0: 20001909
Trigger deasserting edge on TTL0
Trigger asserting edge on TTL0
Signal Interrupt 1
Read 40000000 (A32D32m)
40000000: ********
Bus error at 40000000
```

Figure 6-2. Sample Log File in Text Area.

6.4 Using the VME Menu

VME menu items allow you to read and write multiple memory locations and perform searches across the VXIbus address spaces using a low level interface. It also allows you to assert an interrupt, or reset the system. To use the low level interface:

1. Select one of the BusProbe VME menu items. If you select one of the data transfer items (Read Data, Write Data, Edit Data, or Find Pattern), the Bus Access Parameters dialog box appears. If you select either Send Interrupt or Reset System, that item's dialog box appears.

2. If it appears, select the bus access parameters and choose the OK button. A command dialog box for the type of transfer then appears.

3. Complete the appropriate command dialog box.

4. Observe the results in the main BusProbe window.

6.4.1 Selecting Bus Access Parameters

The Bus Access Parameters dialog box requests information about the data transfer. Figure 6-3 shows the Bus Access Parameters dialog box. You must complete this dialog box before you can execute a data transfer.

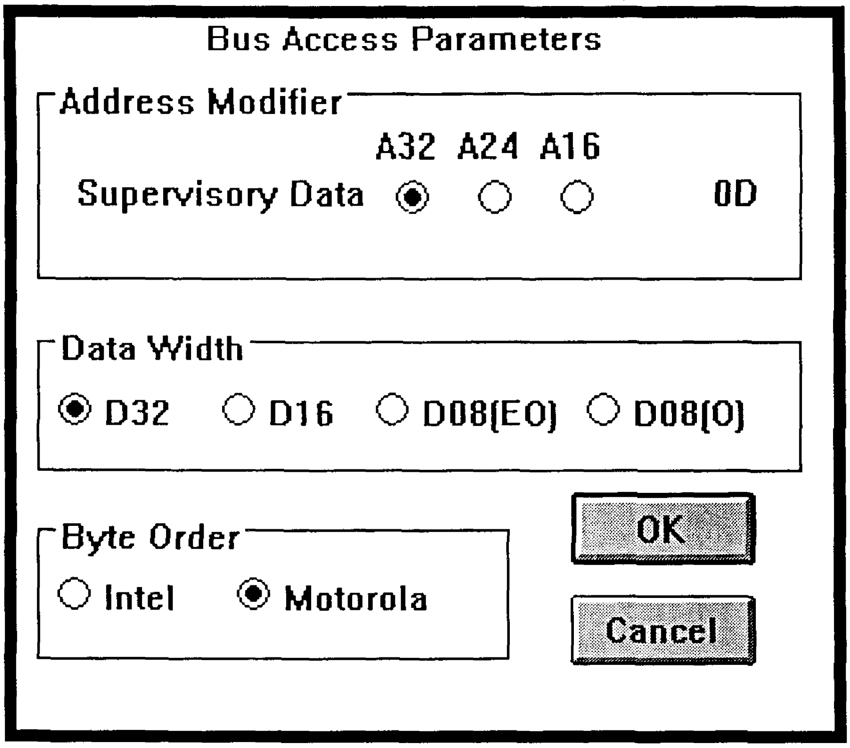
To complete the Bus Access Parameters dialog box:

1. In the Address Modifier box select the type of transfer. The two digit number below A16 is the hexadecimal encoding of the selected address modifier.

2. In the Data Width box select the data width of the transfer. Four selections are available: D32 selects a 32-bit data transfer, D16 selects a 16-bit data transfer, D08(EO) selects 8-bit data transfer, and D08(O) selects 8-bit odd only data transfers.

3. In the Byte Order box select the byte order convention to use for the transfer. Intel selects little-endian and Motorola selects big-endian. The selection depends on the byte-ordering supported by the device being accessed.

4. Choose the OK button and complete the data transfer command dialog box to complete the data transfer.


Bus Access Parameters

Address Modifier

	A32	A24	A16	
Supervisory Data	●	○	○	0D

Data Width

[✓ D32] D16 D08(EO) D08(O)

Byte Order

Intel [✓ Motorola]

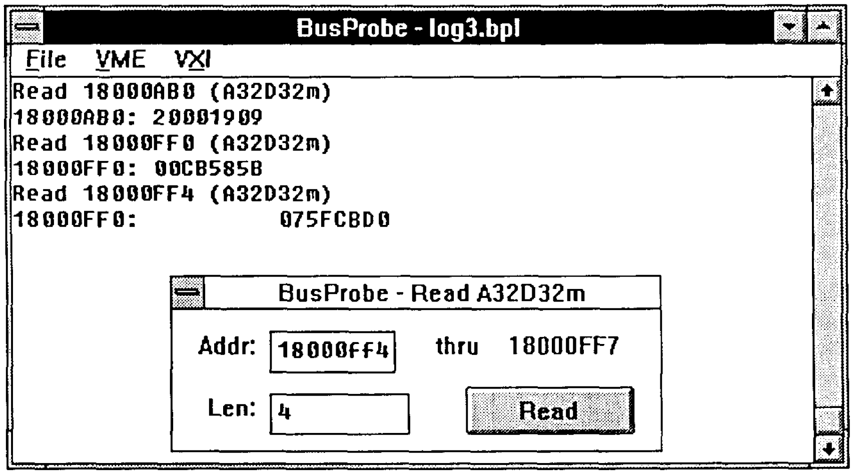
Figure 6-3. Bus Access Parameters Dialog Box.

6.4.2 Reading From VME Memory

The BusProbe - Read dialog box (see Figure 6-4) allows you to read VME data from the VXIbus system memory. It requests the starting address and suggests a number of bytes to read. The suggested byte length is the default number of bytes for one data read. It is based on the data width selection made in the Bus Access Parameters dialog box. Included in the title bar is a summary of the entries made in the Bus Access Parameters dialog box.

The starting address must be aligned on an even address boundary for 16-bit accesses or aligned on a 4-byte boundary for 32-bit accesses. In addition, the byte length must be a multiple of the selected data width. When the starting address and byte length are consistent, BusProbe fills in the ending address.

Use the Control menu to return to the Bus Access Parameters dialog box to change bus access parameters. The Control menu also controls dialog box moving and closing.



```
Read 18000AB0 (A32D32m)
18000AB0: 20001909
Read 18000FF0 (A32D32m)
18000FF0: 00CB585B
Read 18000FF4 (A32D32m)
18000FF0:          075FCBD0
```

Addr: 18000FF4 thru 18000FF7

Len: 4

Figure 6-4. BusProbe - Read Dialog Box.

To read VME data from the VXIbus:

1. Select Read Data item from the VME menu.

2. Complete the Bus Access Parameters dialog box and choose the OK button. BusProbe now displays the BusProbe - Read dialog box.

3. Enter the starting address to read and the number of bytes to read in hexadecimal. BusProbe enters the ending address when the entered address and number of bytes to transfer are consistent. Inconsistent entries cause an error message to appear.

4. Choose the Read button. BusProbe displays the read results in the main BusProbe window.

If a VXIbus error occurs while reading, the read operation terminates. In addition, the displayed data field of the address where the bus error occurred fills with asterisks.

6.4.3 Writing to VME Memory

The BusProbe - Write dialog box (see Figure 6-5) allows you to write VME data to the VXIbus system memory. It requests the starting address, the data to write at each address, and suggests a number of bytes to write. The suggested byte length is the default number of bytes for one data write. It is based on the data width selection made in the Bus Access Parameters dialog box. Included in the title bar is a summary of the entries made in the Bus Access Parameters dialog box.

The address you specify to write must be aligned on an even address boundary for 16-bit accesses or aligned on a 4-byte boundary for 32-bit accesses. In addition, the byte length must be a multiple of the selected data width. When the starting address and byte length are consistent, BusProbe fills in the ending address.

If the selected length is larger than a single data width, the selected data item is written repeatedly across the selected block.

Use the Control menu to return to the Bus Access Parameters dialog box to change bus access parameters. The Control menu also controls dialog box moving and closing.

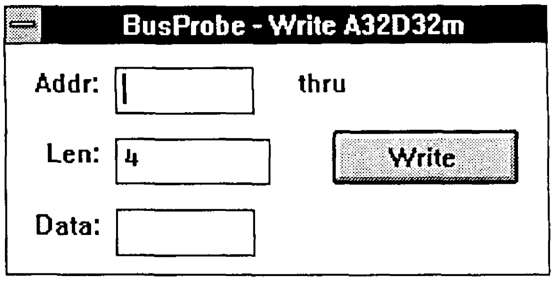


Figure 6-5. BusProbe - Write Dialog Box.

To write VME data to the VXIbus:

1. Select Write Data item from the VME menu.
2. Complete the Bus Access Parameters dialog box and choose the OK button. BusProbe now displays the BusProbe - Write dialog box.
3. Enter the data to write in hexadecimal (the data is automatically left-padded with zeros).
4. Enter the starting address to write and the number of bytes to write in hexadecimal. BusProbe enters the ending address when the entered address and byte length values are consistent.
5. Choose the WRITE button. BusProbe displays the results in the main BusProbe window.

If a VXIbus error occurs while writing, the write operation terminates. In addition, the displayed data field of the address where the bus error occurred fills with asterisks.

6.4.4 Editing VME Memory

The BusProbe - Edit dialog box (see Figure 6-6) allows you to read and write successive VME addresses. It requests a starting address and the data to write at the selected address. First, the address you select is read. Then, you have the option to change that data, or leave it unchanged and continue to the next address. The next address depends on the data width selected in the Bus Access Parameter dialog box.

The starting address must be aligned on an even address boundary for 16-bit read or on a 4-byte boundary for 32-bit reads.

Use the Control menu to return to the Bus Access Parameters dialog box to change bus access parameters. The Control menu also controls dialog box moving and closing.

To edit VME data:

1. Select Edit Data item from the VME menu.

2. Complete the Bus Access Parameter dialog box and choose the OK button. The BusProbe - Edit dialog box then appears.

3. Enter the starting address to read in hexadecimal.

4. Press the TAB key. BusProbe displays the data at the selected address.

5. Choose the NEXT button to display the data at the next address.

Or

6. Edit the data at the selected address by entering the new data in the Data text box and choose the WRITE button or press the ENTER key. BusProbe writes the new data and increments to the next data address.

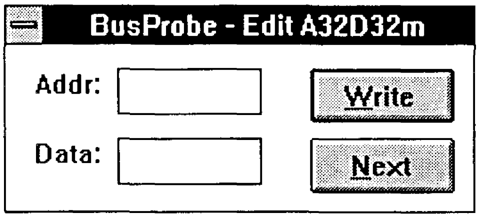


Figure 6-6. BusProbe - Edit Dialog Box.

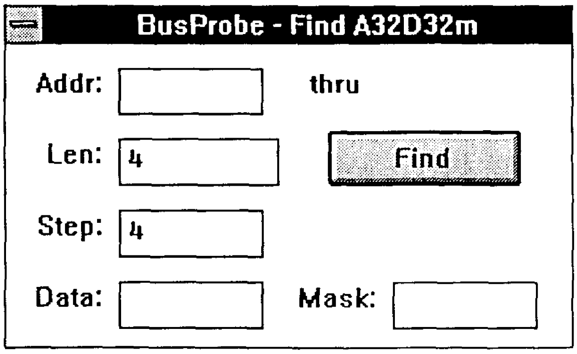
6.4.5 Finding Patterns

The BusProbe - Find dialog box (see Figure 6-7) allows you to search an address range for a specific or masked value. It requests a starting address, the data pattern to find, and the search mask. It also suggests the number of bytes to search and the number of bytes to increment the address before searching again. The suggested byte length and address step are default values for one data search. These suggestions are based on the data width selection made in the Bus Access Parameters dialog box.

The address you specify to start the search must be aligned on an even address boundary for 16-bit accesses or aligned on a 4-byte boundary for 32-bit assesses. In addition, byte length and address step must be a multiple of the selected data width. When the starting address, byte length, and step are consistent, BusProbe fills in the ending address.

The pattern is found when the Data value equals the logical AND operation between the read data and the Mask value.

BusProbe displays a percentage of address range completed below the Find button while the address range is being searched.


BusProbe - Find A32D32m

Addr: thru

Len: 4

Step: 4

Data: Mask:

Figure 6-7. BusProbe - Find Dialog Box.

To find a pattern:

1. Select Find Pattern item from the VME menu.
2. Complete the Bus Access Parameters dialog box and choose the OK button. BusProbe now displays the BusProbe - Find dialog box.
3. Enter the starting search address in hexadecimal.
4. Enter the number of bytes to search in hexadecimal.
5. Enter the increment to step the search address before comparing again. BusProbe enters the ending address when the starting search address, the number of bytes to search, and the step increment are consistent.
6. Enter the data pattern to find.
7. Enter the search mask (one is required for a comparison).
8. Choose the Find button. As the search proceeds, the BusProbe main window displays locations where a match is found. In addition, the percentage of search complete is displayed in the dialog box.

BusProbe displays an error if the data and mask combination is such that there will never be a match (e.g., F for data and 0 for mask).

6.4.6 Asserting or Clearing Interrupts

The Send Interrupt dialog box allows you to assert any one of the seven VXI interrupts or clear all interrupts (see Figure 6-8). You cannot assert a second interrupt until the first interrupt has been acknowledged.

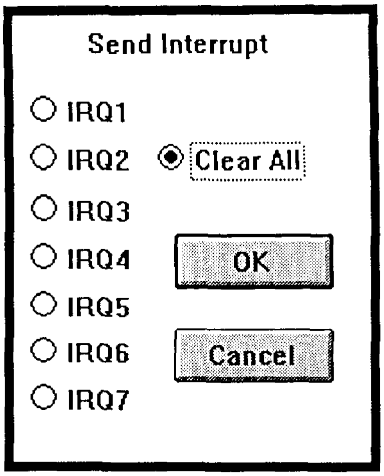


Figure 6-8. Send Interrupt Dialog Box.

To assert or clear an interrupt:

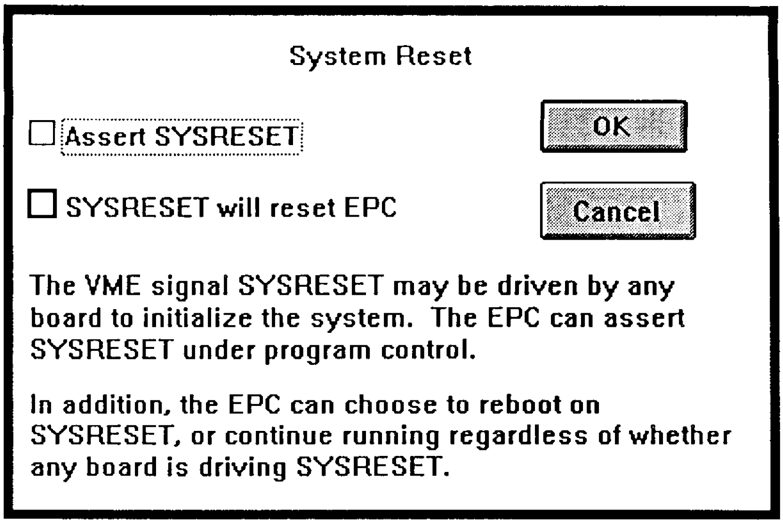
1. Select Send Interrupt item from the VME menu.

2. Select the interrupt to assert or Clear All.

3. Choose the OK button.

6.4.7 Resetting the System

The Reset System dialog box displays the state of SYSRESET and allows you to change its state (see Figure 6-9). This dialog box also selects whether or not the EPC reboots when SYSRESET is asserted. When you assert SYSRESET, BusProbe asserts SYSRESET until you deassert it, unless you also choose to reboot the EPC when SYSRESET is asserted.


System Reset

☐ Assert SYSRESET

☐ SYSRESET will reset EPC

The VME signal SYSRESET may be driven by any board to initialize the system. The EPC can assert SYSRESET under program control.

In addition, the EPC can choose to reboot on SYSRESET, or continue running regardless of whether any board is driving SYSRESET.

Figure 6-9. System Reset Dialog Box.

To assert or deassert SYSRESET:

1. Select Reset System item from the VME menu. The Assert SYSRESET check box displays the state of SYSRESET (checked means SYSRESET is asserted).

2. Check or clear the Assert SYSRESET check box.

3. Choose the OK button.

To assert SYSRESET and reboot the EPC:

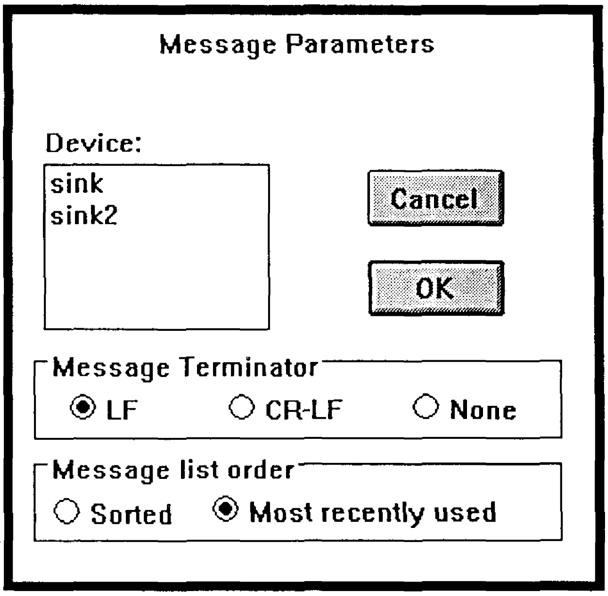
1. Select Reset System item from the VME menu. The Assert SYSRESET check box displays the state of SYSRESET (checked means SYSRESET is asserted).
2. Check the Assert SYSRESET check box.
3. Check the SYSRESET will reset EPC check box.
4. Choose the OK button.

6.5 Using the VXI Menu

VXI menu items allow you to interactively send messages to and receive replies from message-based VXI devices in the system. It also allows you to send and receive word serial commands, assert TTL and ECL triggers, and to initialize the system's VXI devices.

6.5.1 Sending and Receiving Messages

Selecting the Send/Receive Message VXI menu item allows you to send and receive messages to message-based devices on the VXIbus, as well as, IEEE-488 (GPIB) instruments. When you select this menu item the Message Parameters dialog box (see Figure 6-10) appears first. It contains a list of all devices in the system and buttons for selecting the message terminator and the message list order. You must complete this dialog box before you can continue.



Message Terminator
[✓ LF] CR-LF None

Message list order
Sorted [✓ Most recently used]

Figure 6-10. Message Parameters Dialog Box.

After completing the Message Parameters dialog box, the Send/Receive Message dialog box (see Figure 6-11) appears. It requests the message to send. As messages are sent they are added to the Previous Messages list box so you can send them again without reentry. If you select Auto-Receive after Send, the added message is tagged with an asterisk. Messages and message results are also sent to the main BusProbe window.

Checking Send CONFIG? causes BusProbe to query the Message Delivery System (MDS) Router for the names of devices that can be reached from the selected device. Selecting one of the reachable devices adds it to the path displayed in the list box. Checking Send CONFIG? for devices not in the MDS router causes and error.

The Send/Receive Message dialog box also contains five word serial status information indicators. Table 6-1 defines the word serial status indicators. The title bar displays the device to receive or send the message. If the title bar is blank, it is likely that the BusProbe is not connected to a device.

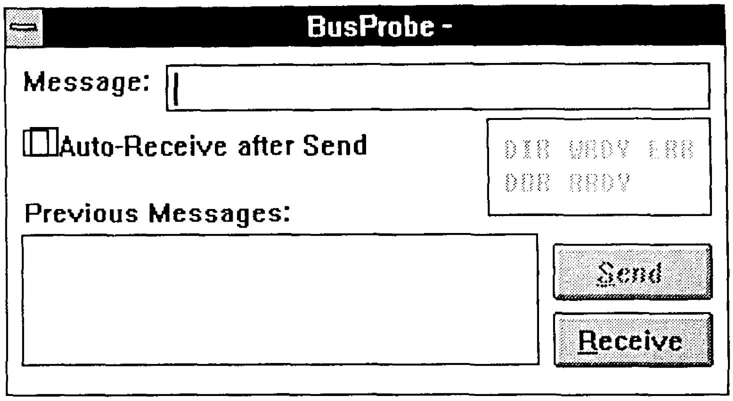


Figure 6-11. Send/Receive Message Dialog Box.

Indicator	Definition
DIR	Data Input Ready. When lit, the servant will accept BYTE AVAILABLE commands.
DOR	Data Output Ready. When lit, the servant will accept BYTE REQUEST commands.
WRDY	Write Ready. When lit, the servant will accept a VXIbus command.
RRDY	Read Ready. When lit, the servant's registers contain data.
ERR	Error. When lit, the servant is asserting an error condition.

Table 6-1. VXIbus Word Serial Status Indicators.

To send/receive a message:

1. Select the Send/Receive Message item from the VXI menu. The Message Parameters dialog box appears.

2. In the Device list box select the device for message communicating (double-click the device name to display additional device information). With each selection, Path is modified to reflect the selected device and the Device list box is modified to contain only those devices accessible by the selected device.

   The device `(commander)` represents the commander of the device specified by Path. Selecting `(commander)` renames the last device from the MDS path being constructed. The device `(this EPC)` represents the EPC on which BusProbe is running.

3. In the Message Terminator box select the termination character(s) to be appended to the message before it is sent. The appropriate terminator depends on the device itself. Refer to the device's documentation for details.

4. In the Message list box select the order in which to display device messages in the Send/Receive Message dialog box (next dialog box to appear).

5. Choose the OK button. The Send/Receive Message dialog box now appears.

6. Enter the message in the Message text box.

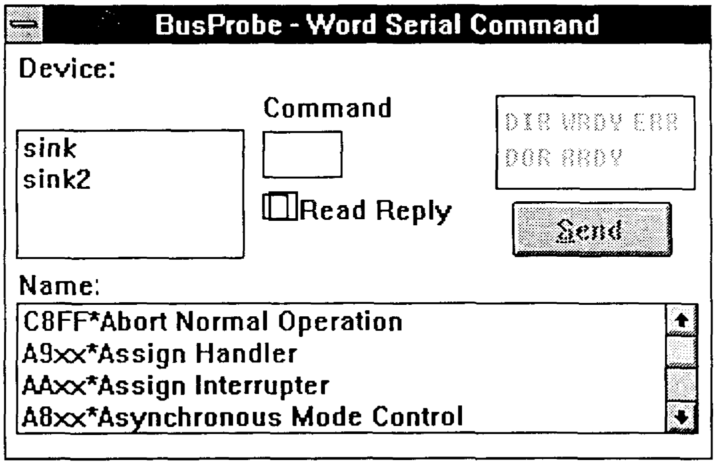
7. Check Auto-Receive after Send if you want to automatically receive a result message.

8. Choose the Send or Receive button.

Choosing the Receive button causes BusProbe to read a message. A timeout occurs if there is no message waiting to be received from the device.

6.5.2 Sending Word Serial Commands

The BusProbe - Word Serial Command dialog box allows you to send 16-bit word serial commands to a device and optionally read a reply. It requests the device to receive the word serial command (see Figure 6-12). When a device is selected, the Name list box is filled with the available word serial commands (defined by the VXIbus specification). The dialog box also contains the word serial status indicators (refer to Table 6-1).



sink
sink2

C8FF*Abort Normal Operation
A9xx*Assign Handler
AAxx*Assign Interrupter
A8xx*Asynchronous Mode Control

Figure 6-12. Word Serial Command Dialog Box.

To send a 16-bit word serial command:

1. Select the Word Serial Command item from the VXI menu.

2. In the Device list box, select the device to receive the word serial command (double-click the device name to display additional device information). With each selection, Path is modified to reflect the selected device and the Device list box is modified to contain only those devices accessible by the selected device.

   The device `(commander)` represents the commander of the device specified by Path.

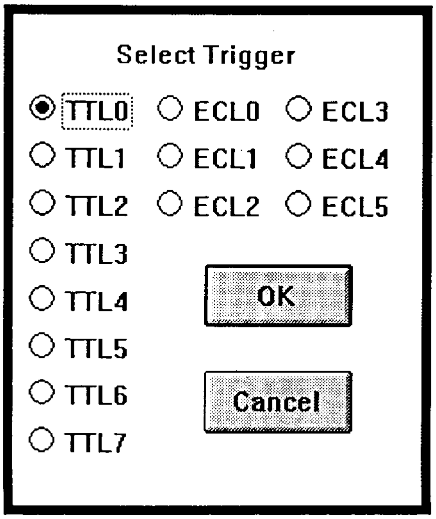
3. Select the word serial command to send from the Name list box or enter the hexadecimal value of the word serial command in the Command text box.

4. Check the Read Reply check box if you want to read the device's reply. BusProbe automatically checks Read Reply if you select a word serial command from the Name list that generates a reply (word serial commands that generate a reply are tagged with an asterisk).

5. Choose the Send button. You can also double-click the command name in the Name list to send the command.

6.3.6 Monitoring, Asserting, and Deasserting Triggers

The Select Trigger dialog box allows you to monitor, assert, or deassert TTL and ECL triggers. (see Figure 6-13). It contains a list of triggers and you must complete this dialog box before you can continue.


Select Trigger

[✓ TTL0] ECL0 ECL3
TTL1 ECL1 ECL4
TTL2 ECL2 ECL5
TTL3
TTL4
TTL5
TTL6
TTL7

Figure 6-13. Select Trigger Dialog Box.

After completing the Select Trigger dialog box, the Trigger dialog box for the selected trigger (see Figure 6-14) appears. Some triggers may not be available on your machine; if so, they are greyed out here.

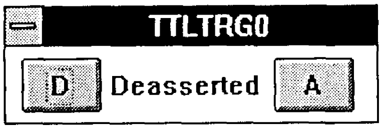


Figure 6-14. Trigger Dialog Box.

To monitor, assert, or deassert a trigger:

1. Select the Trigger item from the VXI menu.
2. Select the trigger to monitor, assert, or deassert.
3. Choose the OK button. The Trigger dialog box appears.
4. To assert the selected trigger click the A button. To deassert the selected trigger click the D button.

You can have multiple trigger dialog boxes open at one time.

6.5.7 Initializing a Device

The Device Initialization dialog box allows you to soft-reset, diagnose, or disable the selected device (see Figure 6-15).

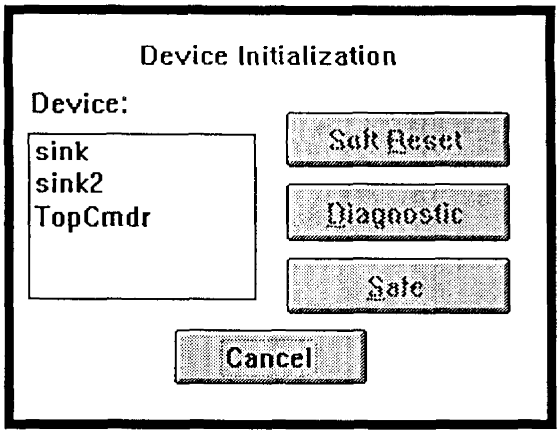


Figure 6-15. Device Initialization Dialog Box.

To initialize a device:

1. Select the Initialize Device item from the VXI menu.
2. In the Device list box select the device to initialize.
3. To reset the selected device, choose Soft Reset. This selection clears SYSFAIL INHIBIT and pulses RESET causing the selected device to reset and execute self-test.
4. To cause the selected device to enter the diagnostic state, choose Diagnostic. This selection asserts SYSFAIL INHIBIT and pulses RESET causing the selected device to enter the diagnostic state.
5. To turn-off the selected device, choose Safe. This selection asserts SYSFAIL INHIBIT and RESET causing the selected device to enter the safe state.

7. Support and Service

7.1 In North America

7.1.1 Technical Support

RadiSys maintains a technical support phone line at (503) 646-1800 that is staffed weekdays (except holidays) between 8 AM and 5 PM Pacific time. If you have a problem outside these hours, you can leave a message on voice-mail using the same phone number. You can also request help via electronic mail or by FAX addressed to RadiSys Technical Support. The RadiSys FAX number is (503) 646-1850. The RadiSys E-mail address on the Internet is *support@radisys.com*. If you are sending E-mail or a FAX, please include information on both the hardware and software being used and a detailed description of the problem, specifically how the problem can be reproduced. We will respond by E-mail, phone or FAX by the next business day.

Technical Support Services are designed for customers who have purchased their products from RadiSys or a sales representative. If your RadiSys product is part of a piece of OEM equipment, or was integrated by someone else as part of a system, support will be better provided by the OEM or system vendor that did the integration and understands the final product and environment.

7.1.2 Bulletin Board

RadiSys operates an electronic bulletin board (BBS) 24 hours per day to provide access to the latest drivers, software updates and other information. The bulletin board is not monitored regularly, so if you need a fast response please use the telephone or FAX numbers listed above.

The BBS operates at up to 14400 baud. Connect using standard settings of eight data bits, no parity, and one stop bit (8, N, 1). The telephone number is (503) 646-8290.

7.2 Other Countries

Contact the sales organization from which you purchased your RadiSys product for service and support.

Appendix A. SURM Error Messages

This appendix contains information about interpreting SURM error messages and tables that list, define, and, in some cases, give corrective action for SURM error messages.

Error Message Interpretation

SURM error messages fall into three categories that are identified as follows (xx is an error number not a returned error code):

E0xx	Fatal errors that terminate the SURM program prematurely.
E1xx	Significant errors that allow the SURM program to complete, but some devices may not be configured correctly
E2xx	Warning messages that inform you of possible error conditions.

A SURM error message may include a return code. The return code is specific to the underlying EPConnect program that generated the error (e.g., BusManager).

SURM Error Messages

Table A-1 contains the SURM fatal error messages.

Table A-2 contains the SURM significant error messages.

Table A-3 contains the SURM warning messages.

Table A-4 lists the return codes that may be included in any SURM error messages.

Table A-1. SURM Fatal Error Messages.

Error #	Description	Corrective Action
E003	The computer running the SURM encountered a bus error while trying to access itself.	Seat computer module properly in backplane or replace computer module.
E004	The BusManager device driver is not installed.	DOS-only users should install the BusManager device driver, **BIMGR.SYS**. Refer to Chapter 2, *Installation and Configuration* for information.
E009	The file **SURM.RC** is missing. The error message tells where the SURM expected to find it.	Place **SURM.RC** file in **C:\EPCONNEC\SURM** or in the directory specified by the EPCONNECT environment variable.
E010	Another device has the same ULA as the computer running the SURM.	Reconfigure the other device to use a different ULA.

Table A-2. SURM Significant Error Messages.

Error #	Description	Corrective Action
E101	Not enough logical address space to assign a contiguous block of logical addresses for the devices on the module in the specified slot.	Move specified device to a lower numbered slot.
E102	The device failed its self-test. The SURM set the device state to safe.	Check for correct operation of device and correct it's problem.
E103	The SURM could not find enough memory space for the device.	Remove one or more modules.
E104	The SURM could not correctly set the device's offset (address base) register.	Check for correct operation of device and correct it's problem.
E105	The SURM could not correctly set the device's control or status register.	Check for correct operation of device and correct it's problem.
E106	The specified device is not enabled to receive messages. If the slot number is shown as "??" the SURM cannot determine the device's slot.	Check device MODID line.
E107	The specified device identified itself as dynamically configured, but could not be found after a new ULA was assigned.	Check for correct operation of device and correct it's problem.
E109	The **DEVICES** file describes a non-VXIbus device with the same ULA as a static VXIbus module or another non-VXIbus device. The device is ignored.	Move the static device or the non-VXI device.

Error #	Description	Corrective Action
E110	The data structures of the On-Line Resource Manager could not be created. The /NO option is therefore assumed.	Increase the spaced reserved for the OLRM. Requires reconfiguration and reboot.
E113	A commander defined in the **DEVICES** file could not be found in the system, but its servant devices were found. The servant devices were made servants of the resource manager.	Change the configuraqion, or install the commander.
E114	A commander defined in the **DEVICES** file was found in the system, but its physical characteristics are not those of a valid VXIbus commander. The commander's servant devices were made servants of the resource manager.	Check device documentation for commander capability.
E115	A commander defined in the **DEVICES** file was found in the system and appears to be a valid VXIbus commander device, but the device was not ready to accept a GRANT DEVICE command.	May have resulted from a word serial timeout. Try increasing WORD_SERIAL_ TIMEOUT in **SURM.RC**, then re-run.
E117	The specified device was not the only device configured to use the specified IRQ. The device appears to be hard-wired, so the SURM could not disconnect it. This problem requires manual intervention to fix.	Reconfigure the offending device, and update the **DEVICES** file to reflect the change.

Error #	Description	Corrective Action
E118	The named device returned an error in its response to the READ INTERRUPTER LINE or READ HANDLER LINE command. SURM assumes that the line in question is disconnected.	Consult the device's documentation or increase the WORD_SERIAL_ TIMEOUT in the **SURM.RC** file.
E119	An error occurred while sending (or getting the response to) the READ INTERRUPTER LINE or READ HANDLER LINE command to (or from) the named device. SURM assumes that the line in question is disconnected.	Consult the device's documentation or increase the WORD_SERIAL_ TIMEOUT in the **SURM.RC** file.
E120	An error occurred while sending (or getting a response to) the READ INTERRUPTER LINES or READ HANDLER LINES command to (or from) the named device. SURM assumes that the number of lines for this device is zero.	Consult the device's documentation or increase the WORD_SERIAL_ TIMEOUT in the **SURM.RC** file.
E121	The named device returned an error in its response to the ASSIGN INTERRUPTER LINE or ASSIGN HANDLER LINE command. SURM assumes that this line has its default configuration.	Consult the device's documentation or increase the WORD_SERIAL_ TIMEOUT in the **SURM.RC** file.
E122	An error occurred while sending (or getting the response to) the ASSIGN INTERRUPTER LINES or ASSIGN HANDLER LINES command to (or from) the named device. SURM assumes that this line has its default configuration.	Consult the device's documentation or increase the WORD_SERIAL_ TIMEOUT in the **SURM.RC** file.

Error #	Description	Corrective Action
E123	An error occurred while waiting for the response to the BEGIN NORMAL OPERATION command.	Consult the device's documentation or increase the WORD_SERIAL_ TIMEOUT in the **SURM.RC** file. If the device is a controller, verify that the servant program is being run.
E124	An error occurred while sending the IDENTIFY COMMANDER command to the specified device. Only servants of the resource manager can receive this command.	Consult the device's documentation or increase the WORD_SERIAL_ TIMEOUT in the **SURM.RC** file.
E125-E131	The specified device responded to the BEGIN NORMAL OPERATION with the indicated error.	Refer to the explanation for the specified error.
E132	Can't read **SYSTEM.INI** file.	SURM will assume this is a RadiSys EPC and continue. If it is not an EPC, SURM will fail later, noting the presence of an unknown EPC type. Correct the bad filename for **SYSTEM.INI**, or fix the permission problem.
E136	The maximum number of devices (256) in a VXIbus system has been exceeded. Additional devices are ignored.	Remove extra device(s).
E137	The EPC's internal jumper is not set as a slot-0 controller, and there is no other slot-0 controller.	1. Set jumper correctly. 2. Install the EPC in slot-0, or check other slot 0 controller jumpers.

Error #	Description	Corrective Action
E138	The A24 VXIbus memory defined in the **DEVICES** file for the specified device is illegal.	1. Set base address to less than 0xFFFFFF 2. Set memory size so no location exceeds 0xFFFFFF.
E139	The A32 VXIbus memory defined in the DEVICES file for the specified device is illegal.	1. Set base address to less than 0xFFFFFFFF 2. Set memory size so no location exceeds 0xFFFFFFFF.
E140	This EPC's internal jumper is set to define it as a slot-0 controller, but it is not installed in slot 0.	1. Jumper set correctly. 2. Put module in slot 0. 3. Seat module in backplane 4. Check MODID00 line.
E141	A device was found with a model code greater than 256 (indicating a slot 0 controller), but it is not installed in slot 0.	Install the device in slot 0 or reconfigure device.
E142	More than one slot 0 controller was found.	Reconfigure system so there is only one slot 0 controller. Ensure slot 0 controller is installed in slot 0.
E144	A word serial error occurred when send a SET UPPER MODID or SET LOWER MODID command.	Verify that slot 0 controller is operating correctly.
E145/ 146	The SET UPPER MODID or SET LOWER MODID command failed.	Verify that slot 0 controller is operating correctly.
E147	The SURM.RC treatments specification SLOT_0_DEVICE contains an invalid value.	Correct SLOT_0_DEVICE variable.
E148	There is a statically configured device at address 255 that prevents the mainframe extender from mapping address 255 to the child bus for dynamic configuration.	Assign the statically configured device another address.

Error #	Description	Corrective Action
E149	Devices on the ancestor bus prevent enlarging its windows to include all devices on the ancestor bus.	Consult Appendix C concerning mainframe extenders, then reconfigure system correctly.
E150	The current slot 0 device is of class extended, but not subclass mainframe extender.	No other extended subclass is defined as a slot 0 controller. Install a slot 0 controller.
E151	The slot 0 device is a memory device not a slot 0 controller.	Install a slot 0 controller.
E155/ 156	The window value read from the mainframe extender does not match the window set by the SURM.	Replace mainframe extender.
E157	SURM is unable to update the file ..**\DB\RESRCMGR**.	Check file attributes and path set by EPCONNECT variable, if it is used.
E158	MXIbus topology is too deep.	Flatten MXIbus tree or free more memory for use by SURM. SURM requires about 450 KB of free memory to execute. Use MODE to reduce the screen size to 25 lines.
E159	The sent ABORT NORMAL OPERATION timeout or generated an error.	Try using the **SURM.RC** treatment specification NEVER_SEND_ANO.
E160	A device on a mainframe extender became inaccessible because the mainframe extender window could not be expanded.	Manually set ULA's of SC devices in ancestor buses. Refer to Appendix C.

Table A-3. SURM Warning Messages.

Error #	Description	Corrective Action
E201	A device was found with fixed logical address 255. The SURM has bypassed dynamic configuration.	If your system contains dynamically configured devices, remove the device with the fixed logical address 255, or change its logical address.
E202	The specified device was still executing its extended selftest at the time the SURM examined it.	Wait for the device to complete the selftest, then try again.
E203	Cannot find the file **VXIMANUF**. All manufacturer names are shown as [nnnn].	Install the file **VXIMANUF** in the directory **C:\EPCONNEC\DB** or in the directory specified by the EPCONNECT environment variable.
E204	Cannot find the file **VXIMODEL**. All manufacturer and model names are shown as [nnnn].	Install the file VXIMODEL in the directory **C:\EPCONNEC\DB** or in the directory specified by the EPCONNECT environment variable.
E208	An error occurred while sending (or getting the response to) the READ PROTOCOL to (or from) the named device. SURM assumes the device in question conforms to version VXI 1.2, or earlier, of the VXI specification.	Check devices for correct installation and operation.
E209	A bad character was found in the **SURM.RC** file.	Fix the indicated line with your editor.

Error #	Description	Corrective Action
E210	The SURM ran out of memory to store the treatment specs from SURM.RC. Call the factory if you need more treatment specs.	Shorten **SURM.RC** file by removing unnecessary switches and treatment specifications.
E211	A **SURM.RC** file treatment specification switch is not defined.	Fix the indicated line with your editor.
E212	A **SURM.RC** file treatment specification has a syntax error.	Fix the indicated line with your editor.
E213	The **SURM.RC** file has an undefined switch.	Fix the indicated line with your editor.
E214	The **SURM.RC** file has an undefined treatment specification variable.	Fix the indicated line with your editor.
E215	The **SURM.RC** file has a syntax error in a treatment specification variable.	Fix the indicated line with your editor.
E216	The **SURM.RC** file has an out-of-range numeric values assigned to a treatment specification variable.	Fix the indicated line with your editor.
E217	The **SURM.RC** file has a syntax error in a treatment specification switch.	Fix the indicated line with your editor.
E218	The **SURM.RC** file has a syntax error in a treatment specification item. The error was detected before the SURM could distinguish the item as a switch or a variable.	Fix the indicated line with your editor.
E220	The specified device is not the only device configured to use the specified IRQ. The device was disconnected from that IRQ. If it is a commander, the device will probably be made a handler of a different line.	Correct the system configuration in the **DEVICES** file.

Error #	Description	Corrective Action
E222	A corrupted interrupter and/or handler map field appears in the **DEVICES** file.	Use the Configurator to correct interrupter and/or handler map.
E224	The specified commander will not be allocated a line to handle because all lines are already handled.	Delete some handlers on other devices with the Configurator.
E225	The specified commander did not advertise PH capability and was not configured to handle all the lines its servants were interrupting on, so interrupts from the specified servant will be ignored.	Use a different commander, or change the configuration if the commander has hard-wired handlers.
E226	The specified commander is PH capable, but did not have enough handlers to cover all the lines its servants were interrupting on.	Reconfigure servants which are hard-wired.
E230	The specified VMEbus device has been assigned A16 memory within the range reserved for VXIbus devices. This causes problems, because the logical address of a VXIbus device is configured (either dynamically or statically) such that its A16 memory overlaps with the memory specified for this device. VXIbus A16 reserved memory range is 0xC000 - 0xFFFF.	Reconfigure servants which are hard-wired.
E231	The number of allowable devices within the On-Line Resource Manager (OLRM) has been exceeded. Additional devices are ignored.	Increase the number of allowable devices using the /S command line option for the **BIMGR.SYS** device driver in the **CONFIG.SYS** file.

Error #	Description	Corrective Action
E232	This EPC has not been assigned any slave VMEbus memory. This is caused by the memory selections for VMEbus devices and/or custom devices in the **DEVICES** file. No memory chunk large enough to accommodate the EPC memory remained.	Reconfigure the VME devices to leave more space for the EPC.
E237	The SURM used a obsolete invocation option (switch) . A new switch was substituted for the obsolete one.	Use the new switch in subsequent SURM invocations in the **AUTOEXEC.BAT** file.
E238	The specified bus has no slot 0 controller.	Install a slot 0 controller on the specified bus.
E241	Missing base or size variable for an address space in **DEVICES**.	Edit the **DEVICES** file with your editor.
E242	Unable to read **DEVICES** file.	Check file attributes and path set by EPCONNECT variable if it is used.
E243	Invalid character in **SURM.RC** variable.	Fix the indicated line.
E244	Invalid SURM invocation variable.	Fix the indicated line.
E245	Can't get the servant area for the named device. Servant area assumed to be 0.	Check deivce. There is probably a word serial error.

Table A-4. EPConnect Return Codes Displayed by SURM.

Return Code	Returned Error
(-1)	ERR_FAIL
(-2)	ERR_TIMEOUT
(-3)	ERR_BERR
(-4)	ERR_WS
(-5)	ERR_RTIMEOUT
(-6)	ERR_RBERR
(-7)	ERR_RWS
(-8)	ERR_COLLISION
(-9)	ERR_BUS_BUSY
(-10)	ERR_BUS_UNOWNED
(-20)	ERR_BUFFER_FULL
(-21)	ERR_DUPLICATE_NAME
(-23)	ERR_INVALID_MESSAGE_TYPE
(-24)	ERR_MEDIA_TABLE_FULL
(-25)	ERR_ROUTER_TABLE_FULL
(-26)	ERR_ILLEGAL_NAME
(-27)	ERR_UNSUPPORTED_MEDIA
(-28)	ERR_ILLEGAL_OPERATION
(-29)	ERR_ILLEGAL_COMMAND
(-30)	ERR_UNSUPPORTED_FNCT
(-31)	ERR_NETWORKBUSY
(-32)	ERR_SESSIONCLOSED
(-33)	ERR_TOOMANYSESSIONS

Return Code	Returned Error
(-34)	ERR_ILLEGAL_PARAMETER
(-35)	ERR_BRIDGE_TABLE_FULL
(-36)	ERR_BEYOND_BOUNDARY
(-37)	ERR_INTERRUPTED

Table A-1. EPConnect Return Codes Displayed by SURM.

Appendix B. DEVICES File

Introduction

The **DEVICES** file is key to the EPConnect environment. It defines the names and other parameters of devices that EPConnect programs can access. Devices are defined by a series of records that consist of variables and value parameters. It is not required that an instrument be physically present in the system when you define it. This is handy when you often change the system configuration. A **DEVICES** file record has this form:

```
<name-assignment>[,<variable-assignment>]*<newline>
```

Where:

<name-assignment>	:	"*name*" "=" *<devicename>*
<variable-assignment>	:	<variable> ["=" <value>]
<variable>	:	<string>
<value>	:	<string>

Creating Records

The install process places a **DEVICES** file in the **C:\EPCONNEC** directory; however, the file contains no records for the system devices. You can create the required **DEVICES** file records by using either the VXIbus Configurator, a Windows application, or by manually editing the **DEVICES** file.

The VXI Configurator is the recommended method. When you use this method the required file syntax is automatically established.

When you manually edit the **DEVICES** file, observe these syntax rules:

- New-line characters delimit records. Multi-line records require a backslash character at the line's end for continuation onto another line.
- Lines beginning with # are comments and are ignored.
- A line cannot contain more than 80 characters. Use \ to break up a definition line longer than 80 characters.
- Quote a single character (e.g., \= or \,) by placing a backslash (\) in front of it. Quote an entire string by enclosing the string with double quotation characters (""). Quoting a string is necessary to ensure leading and trailing spaces are not removed. Character or string quoting is also necessary to embed an equal sign character or comma character.
- Redundant variable name handling is application-dependent. The SURM and the SICL **iopen** function always use first assignment of a given variable name and ignore any others.
- A string is a series of any characters except a non-quoted comma or equal sign. Non-quoted whitespace characters will be removed from the beginning and end of strings.
- Valid variables depend on the device type. Incorrectly named variables are ignored. The form of the value assigned depends on the variable.

- Variable name matching is not case sensitive. If the value is an enumerated type (e.g., VME, VXI, GPIB), value matching is also not case sensitive.

Variables

DEVICES file variables are not case sensitive and include:

Variable	Description
name	Required variable and must: 1. Be 1 to 12 characters in length. 2. Limit characters to letters, digits, and underscore. 3. Begin with a letter and not end with an underscore.
media	Required variable. Valid values are **VXI** , **GPIB**, **VME**, **NETLINK**, and **CUSTOM**. This variable's value affects the interpretation of other variables and determines how SURM and SICL treat the device.
ifname	Optional variable that associates a device driver to an interface (via the SICLIF file) other than a predefined interface. Cannot be set with the VXI Configurator and only used by SICL.
make *model*	Optional variables that specify the device's make and model. Used for name binding by SURM and reference by user applications. For VXI devices, *make* and *model* must match exactly one of the make or model names defined in the files ...**\EPCONNEC\DB\VXIMANUF** and ...**\EPCONNEC\DB\VXIMODEL**, respectively.
slot	Optional variable that defines the VXI slot number. Valid range is 0 to 12. Specify in decimal. For name binding purposes only.

bus	Optional variable that defines the VXIbus in which the device resides. Valid range is 0 to 255. Specify in decimal. For VXI devices, this variable is for name binding purposes only. For VME devices, this variable is used for mapping the device's memory across the appropriate mainframe extenders.
device	Optional variable for name binding a VXI device when multiple logical devices exist in a single VXI module. This variable provides a way to distinguish them by name. Specify in decimal. The first device in a module is number 0.
ula	Optional variable whose value is the device's statically configured logical address used for name binding a VXI device. Specify in decimal. Using this variable is not the recommended method to name dynamically configured devices, because these ULAs are not predictable. All devices are given ULAs regardless of whether they have configuration registers in A16 or not. For non-VXI devices, *ula* is assigned based on this variable if it is defined. Otherwise, SURM selects a value.
commander	Optional variable whose value is the device name of the commander. Any device may have a commander variable defined. If the device and its commander are both VXI, SURM makes the device a VXI servant of the named commander. For other media types, the relationship is stored for reference only.

handlermap *interruptermap*	Optional variable that defines interrupt maps for any **VXI, VME**, or **CUSTOM** device. The *handlermap* variable selects the IRQ line(s) on which a device is to handle interrupts. If the device is VXI and has programmable handler (PH) capability, it will be programmed by SURM to handle the interrupts specified here. *Interruptermap* selects the IRQ line on which the device generates interrupts. If the device is VXI and has programmable interrupter (PI) capability, it will be programmed by SURM to interrupt as specified here. IRQ lines specified by *handlermap* are not assigned by SURM to any other PH capable VXI device. Maps for VXI devices not found in the system are ignored (no IRQs reserved). **VME** and **CUSTOM** devices are always assumed to be present. The value of these variables is a string of 7 digits. The rightmost digit corresponds to *interrupter* or *handler* number 1 on the device (this is for devices with multiple handlers or interrupters). The digits are 1 through 7, and indicate the IRQ line to which the interrupter or handler is connected. Zero indicates the interrupter or handler is not connected to an IRQ. Unspecified digits are assumed to be zero. The SURM automatically configures PI/PH capable devices with no maps specified. Once an interrupt number appears in the *handlermap* of a device that exists, that interrupt assignment is used and no other devices in the system can handle that interrupt (*Handlermap* for VXI devices that are configured but not present do not have this effect).
a16base *a16size* *a24base* *a24size* *a32base* *a32size*	Optional variables for **VME** and **CUSTOM** devices. Defines which section of the address space to assign to the device. Values are in hexadecimal. If either **size* or **base* is missing, it is assumed that the address space is not used by the device.

byteorder	Optional variable for **VME** and **CUSTOM** devices. Valid entries are: **I** for little-endian (Intel), or **M** for big-endian (Motorola).
datawidth	Optional variable for **VME** and **CUSTOM** devices. Valid entries are: **D08** (8-bit), **D08O** (8-bit odd address only), **D16** (16-bit), or **D32** (32-bit).
bridge *mdsparams*	Optional variables for **CUSTOM** devices (both must appear). *Bridge* is the mnemonic for the user-supplied bridge, and *mdsparams* is the string of initialization parameters for the device. Devices on other **media** will be registered with the appropriate bridge by SURM with an equivalent of *mdsparams* derived by the SURM.
primary *secondary*	Required entry for **GPIB** devices that forms the device's address on the GPIB bus. *Primary* is a required parameter, *secondary* is not required and if not entered defaults to 0.
sendeoioneos *sendeoiwithlast* *terminatereadoneos*	Optional variables for **GPIB** devices. No value is required. When specified, the action defined by variable takes place.
eosbitscompared	Optional variable for **GPIB** devices that controls the number of bits that identify the EOS character in messages. Valid values are **7** or **8**.
eoschar	Required variable for **GPIB** devices that specifies the hexadecimal value of the EOS character.
timeout	Required variable for **GPIB** and **NETLINK** devices. For **NETLINK** devices, this is the decimal value of the network time-out for the device in half seconds. For GPIB devices, it is the GPIB time-out value for the device. For GPIB devices, valid values are: **none**, **10us**, **30us**,**100us**, **300us**, **1ms**, **3ms**, **10ms**, **30ms**, **100ms**, **300ms**, **1s**, **3s**, **10s**, **30s**, **100s**, **300s**, and **1000s**.

Sample DEVICES Files

The following are sample **DEVICES** file records:

```
name=vxidev, \
        media=      VXI, \
        make=       Tektronix, \
        model=      VX4236



name=GPIBdev, \
        media=GPIB, \
        make=HPAF6, \
        model=E1445A, \
        primary=1, \
        eoschar=0d, \
        terminatereadoneos, \
        sendeoiwitheos, \
        sendeoiwithlast, \
        eosbitcompare=8, \
        timeout = "300 ms"

#  In this example TopCmdr handles all interrupts (0-7).
#  This example also
#  enables TopCmdr to assert interrupts on IRQ2.  Unlike
#  handler assignment other devices can specify duplicate
#  interrupt map IRQs.

name=TopCmdr, \
        media=VXI, \
        make="RadiSys Corp", \
        model=EPC-7 S0, \
        handlermap=7654321, \
        interruptermap=2

#  This is a sample netlink DEVICE record:  */

name=netdev, \
        media=NETLINK, \
        make="RadiSys Corp", \
        model=RIC386, \
        timeout=5

#  This is minimal VME DEVICE record for a device
#  occupying 64 KB in A24 address space:

name=vmedev, \
        media=VME, \
        make=Motorola, \
        model=MVME121, \
        a24base=8000, \
        a24size=10000, \
        datawidth=D16, \
        byteorder=M

#  This is a custom DEVICE record where interactions
#  with the device are handled by a user-supplied bridge:

name=custdev, \
        media=CUSTOM, \
        mdsparms="init string", \
        bridge=USER
```

Appendix C. Configuring Multi-Mainframe Systems

C

Introduction

This appendix describes the hardware installation requirements for multi-mainframe systems. It also explains how to set system ULAs so that the SURM recognizes all the system devices.

Multi-Mainframe Topology

If your system requires more than 13 devices you need to use another mainframe. Figure C-1 show the interconnect topology of a multiple mainframe system.

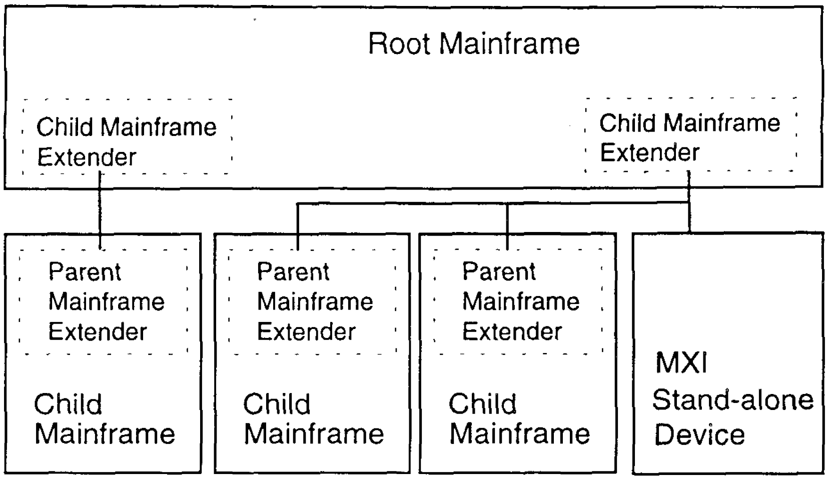


Figure C-1. Multiple Mainframe Interconnect Diagram.

Hardware Configuration

It is recommended that you operate the mainframe extender in Interlocked Arbitration Mode to prevent deadlocks. This mode requires that you install the mainframe extenders in slot 0. When moving devices about in the mainframe, ensure that all backplane jumpers are removed from slots that will be occupied. To configure multi-mainframe system hardware:

1. Configure the EPC to be a non-slot-zero controller, and remove it from slot 0 in the root mainframe.
2. Install the root mainframe extender in slot 0.
3. Install additional mainframe extenders next to the previous one.
4. Install the EPC next to the last mainframe extender.
5. In the child mainframe install the parent bus extender in slot 0 if there is no slot 0 controller or in the slot next to the slot 0 controller.

ULA Assignment

ULA assignment becomes extremely important with multi-mainframe systems. If it is not done correctly the SURM may not recognize the mainframe extender or its devices. When this happens your application cannot access those devices.

Key to correct ULA assignment is the mainframe extender logical address window. A logical address window maps a range of ULAs out to child mainframes or in from parent mainframes. This range of ULAs begins at a logical address window base ULA and extends through the number of ULAs in the logical address window. The SURM determines both the logical address window base and its size (number of ULAs in the logical address window) based on the mainframe extender's device ULAs.

A logical address window base ULA is always at a multiple of the logical address window size. Depending on ULA assignment, a logical address window may or may not contain all mainframe extender devices.

Child mainframe extender logical address windows must be contained within the parent's (root) mainframe logical address window. The mainframe extender ULA may be or may not be part of the logical address window.

The SURM defers opening logical address windows until the parent bus is configured, which allows statically configured devices to occupy high ULAs.

If the child mainframe contains only dynamically configured devices, you only need to ensure that the ULAs assigned to the parent mainframe extender are located at a ULA that is within an appropriately aligned logical address window that is large enough to contain all devices. Failure to do so can result in a logical address window that is larger than necessary, which in turn, could cause devices in other mainframe to be unreachable.

If the child mainframe contains statically configured devices, you must also ensure that their ULAs are located within an appropriately aligned logical address window.

Figure C-2 is an example system with two child mainframe extenders and a stand-alone MXIbus device. All the system devices are statically configured. Table C-1 shows the ULAs and logical address windows assigned to the system. Note that child mainframe 36 devices dmm, cntr0, cntr1, and osc0 are unreachable and child mainframe 65 device vdev7 is unreachable.

The process of determining a logical address window involves determining a target range of ULAs and trying different logical address window sizes and bases until a size and base are found that enclose the target range or the ULA of a device in another mainframe. The following paragraphs briefly explain the process for child mainframe 36.

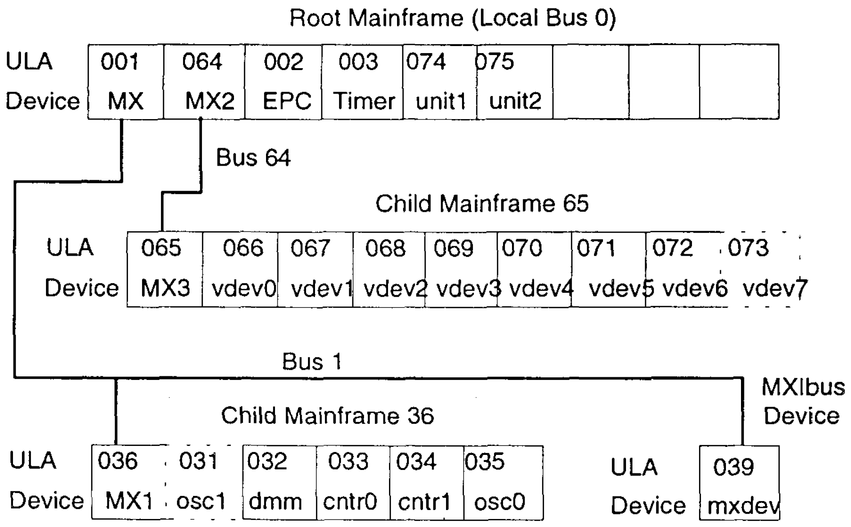


Figure C-2. Example System With Mainframe Extenders

1. The SURM first determines a target range of ULAs. For child mainframe 36, the target range of ULAs is 031 to 039.

2. The SURM next determines a beginning logical address window size and a beginning base ULA. The first trial size is always two. For child mainframe 36 the default beginning base ULA is 030. The default base is always the smallest ULA assigned to the mainframe extender devices. If the smallest ULA is odd the SURM rounds it down to the next even number (logical address windows cannot begin on an odd ULA). If the smallest ULA is even it uses it.

3. As can be seen by Table C-1, the first test logical address window (base ULA 30 and size 2) does not include all child mainframe 36 devices. The SURM now doubles the size to four and adjusts the base ULA down to the closest multiple of the new size. In this case the new base ULA would be 28. Again, this logical address window does not include all child mainframe 36 devices.

4. This process of doubling the logical address window size and adjusting the base continues until the SURM finds a window that includes all child mainframe 36 devices or includes a ULA of a device in another mainframe. When the SURM tries logical address window base ULA 000 and size 32, it finds root mainframe devices within the trial logical address window.

5. When other mainframe devices are found within the trial logical address window, SURM returns the logical address window to the previous test case (logical address window base 16 with size 16) and assigns these values as the logical address window. Table C-1 shows that this makes devices at ULAs 032, 033, 034, 035, 036, and 039 unreachable.

If you change the ULA assignment of osc1 from 031 to 037 all devices will be reachable.

The case with vdev7 is very similar to osc1, but it occurs at a different logical address window size and base. In this case, when the SURM tries a logical address window size of 16 at ULA base address 64 it finds ULA collisions with the root mainframe devices Unit1 and Unit2 at ULAs 74 and 75, respectively. As before, the SURM returns to the next smaller logical address window size (8), where ULA 74 is unreachable.

ULA	Device	Child LA Window	Parent LA Window	Remarks
000				
001	MX			
002	EPC			
003	Timer			
.				
.				
.				
016	*vacant*			
.				
.				
.				
031	osc1			
032	dmm	mx1	mx	Unreachable
033	cntr0	mx1	mx	Unreachable
034	cntr1	mx1	mx	Unreachable
035	ocs0	mx1	mx	Unreachable
036	MX1		mx	Unreachable
037	*vacant*		mx	
038	*vacant*		mx	
039	mxdev		mx	Unreachable
.				
.				
.				
064	MX2		MX2	
065	MX3	MX3	MX2	
066	vdev0	MX3	MX2	
067	vdev1	MX3	MX2	
068	vdev2	MX3	MX2	
069	vdev3	MX3	MX2	
070	vdev4	MX3	MX2	
071	vdev5	MX3	MX2	
072	vdev6	MX3	MX2	
073	vdev7		MX2	Unreachable
074	unit1		MX2	
075	unit 2		MX2	
076	*vacant*		MX2	
077	*vacant*		MX2	
078	*vacant*		MX2	
079	*vacant*		MX2	

.
.
.
0255

Table C-1. Example System Logical Address Windows (ctd).

NOTES

Index

NOTES