XVME-984

Windows VMEbus Toolkit

74984-001E

Printed in the United States of America
Part Number 74984-001E

XYCOM
750 North Maple Road
Saline, Michigan 48176
(313) 429-4971

XYCOM REVISION RECORD

Revision	*Description*	*Date*
A	Manual Released	9/91
B	Manual Updated	4/92
C	Manual Updated	10/93
D	Manual Updated	2/94
E	Manual Updated to Include PCN 170	5/94

Address comments concerning this manual to:

xycom

Technical Publications Department
750 North Maple Road
Saline, Michigan 48176

Part Number: ***74984-001E***

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.1 MANUAL STRUCTURE

This manual is designed to help you understand and use the Xycom XVME-984 Windows VMEbus Toolkit.

This outline of the manual structure will help you find the specific sections containing information relevant to your system needs:

Chapter 1	**Introduction:** gives a general overview of the package, and general directions on how to start using the Toolkit.
Chapter 2	**VMEbus Manager:** details the operation of the VMEbus Manager, VMEbus monitor application.
Chapter 3	**LIBDEMO.EXE:** details the demonstration program, a Windows application.
Chapter 4	**XVME Dynamic Link Library:** describes the General Purpose, Analog, Counter, and Digital routines, along with the real mode window, the VMEbus PC/AT processor interrupt capability, and the XVME984.SYS MS-DOS device driver.

1.2 OVERVIEW

The XVME-984 Windows VMEbus Toolkit is a comprehensive package that simplifies the process of designing VMEbus application programs for Xycom VME PC/AT hardware in a Microsoft Windows 3.1 environment.

The XVME-984 is composed of three main components: the I/O Dynamic Link Library; the VMEbus Manager; and a series of installation and demonstration programs. These main components are distributed on a 3.5" diskette provided in the kit. Microsoft Windows is also included in a separate version of the XVME-984.

The Dynamic Link Library (DLL) contains a library of routines which simplifies development of application programs to run under Microsoft Windows on your Xycom VMEbus PC/AT products. The DLL includes the low level code required to configure and use Xycom I/O modules in a Windows application.

Xycom's VMEbus Manager (VMEMAN.EXE) is a Windows application program which gives you a simple mechanism for accessing and monitoring the VMEbus. It lets Xycom PC/AT processors read from and write to VMEbus memory, and monitor the status of VMEbus interrupts and PC/AT registers, so you can test your VMEbus system configuration easily and efficiently.

The installation program is a Windows application which will install the Toolkit for you. This installation includes copying appropriate files to a hard disk, modifying Windows system files, and modifying your CONFIG.SYS. file.

The demonstration program (LIBDEMO.EXE) provides examples of how to use the DLL routines in your application. Since the program knows the Xycom I/O memory maps, you can also use it to interactively configure a Xycom I/O module on your system.

The XVME-984 Windows VMEbus Toolkit is released in two configurations on 3.5" diskettes: XVME-984/1 and XVME-984/2. The XVME-984/1 includes Microsoft Windows (including a user manual) along with the software described in this manual. This version is available in the U.S. and Canada, only. The XVME-984/2 version is designed for customers who already own Microsoft Windows; it is the same as the XVME-984/1 version except that it does not include Microsoft Windows.

Table 1-1. VMEbus Boards Supported in the Dynamic Link Library

PC/AT Processors:			
•	XVME-674	-	33 MHz 80486DX or 66 MHz 80486DX2 processor, 4, 16, or 32 Mbytes of dual-access DRAM, Super VGA graphics controller, floppy and IDE hard disk controllers
•	XVME-677	-	33 MHz 80486SX processor, 4, 16, or 32 Mbytes of dual-access DRAM, Super VGA graphics controller, floppy and IDE hard disk controllers
•	XVME-678	-	25 MHz 486SLC Processor 0, 1, or 4 Mbytes DRAM Super VGA graphics, two serial, one parallel port, floppy and hard disk controllers
•	XVME-684	-	25 or 33 MHz 80486 processor, 4 or 16 Mbytes of dual-access DRAM, Super VGA graphics controller, floppy and IDE hard disk controllers
•	XVME-686	-	20 MHz 80386SX processor, 1, 4 or 8 Mbytes of dual-access DRAM, VGA graphics, two serial/one parallel port, floppy and hard disk controllers
•	XVME-687	-	20 MHz 80486SX processor, 0, 1 or 4 Mbytes of dual-access DRAM, Super VGA graphics controller, floppy and IDE hard disk controllers
•	XVME-688	-	25 MHz 80386SX processor, 4 or 16 Mbytes of DRAM, VGA graphics, two serial/one parallel port, floppy and hard disk controllers
Digital I/O:			
•	XVME-200	-	32-channel DIO with interrupts
•	XVME-201	-	48-channel DIO without interrupts
•	XVME-202	-	PAMUX controller
•	XVME-210	-	32-channel optically isolated digital input
•	XVME-212	-	32-channel isolated digital input module with change-of-state detection

Table continued on the following page.

Table 1-1. VMEbus Boards Supported in the Dynamic Link Library (*continued*)

Digital I/O:			
•	XVME-220	-	32-channel isolated digital output module
•	XVME-240	-	80-channel TTL I/O module
•	XVME-244	-	64-channel optically isolated digital I/O
•	XVME-260	-	32-channel relay output module
•	XVME-290	-	6U version of the XVME-200 with I/O on the P2 connector
Analog I/O:			
•	XVME-500	-	16SE/8DI-channel analog input
•	XVME-505	-	4-channel analog output
•	XVME-530	-	8-channel isolated analog output module
•	XVME-540	-	32/16-channel analog input, 4-channel analog output, 12-bit A/D
•	XVME-560	-	64/32-channel analog input module
•	XVME-566	-	100 KHz, 32/16-channel analog input module
•	XVME-590	-	6U version of the XVME-500 with I/O on the P2 connector
•	XVME-595	-	6U version of the XVME-505 with I/O on the P2 connector
Counter Modules:			
•	XVME-203	-	10-channel counter
•	XVME-230	-	16-channel intelligent counter module
•	XVME-293	-	6U version of the XVME-203 with I/O on the P2 connector

1.3 INSTALL PROGRAM

To install the XVME-984 software on a hard disk, follow these steps:

1. Insert the XVME-984 diskette in drive A or B.

2. Run Windows.

3. When the Program Manager window opens, select Run in the file menu.

4. Type A:install or B:install, depending on which drive you inserted the diskette into in step 1. Then press enter.

> **NOTE**
> It is recommended that the SYSTEM.INI file contain "EMMExclude=E000-EFFF" to avoid problems while running the VMEbus Manager program. The install program will perform this modification by default. See the latest version of the Microsoft Windows manual for further information about the SYSTEM.INI file.
>
> If the memory manager EMM386.EXE is being used, it must contain the option X=E000-EFFF. The install program **will not** perform this modification.

A windows based program is provided to install the XVME-984 software. This program consists of several dialog boxes that prompt the user for installation options. The first set of dialog boxes prompts you for target drive and target directory. Selecting "OK" performs the desired function and continues the installation.

After target drives and directories have been established, the install program prompts you for the files you wish to install. The default files to be loaded are the Windows VMEbus Toolkit software (XVME984/001), the Dynamic Link Library software (XVME984.DLL), and the Source directory. The default is to load the XVME984/001 and Source directory in the target drive/directory previously selected. The XVME984.DLL will be loaded in the C:\WINDOWS directory per Windows convention for .DLL files. The installation program will search your target directory for an existing XVME984.DLL file. If an earlier version is found, you will have the option of retaining or updating this file.

If a memory manager is being used on your system, you must exclude the addresses E000-EFFF. Consult the manual of your memory manager on how to exclude addresses. The install program provides notice that you must perform this modification. The install program **does not** do this for you.

In order for the Windows VMEbus Toolkit to access V<Ebus memory, the following line

must be added to the WINDOWS\SYSTEM.INI file under the [386 enh] section:

EMMExclude = E000-EFFF

Select "OK" if you want the install program to perform this modification for you.

The install program creates a "VME Tools" Program Group with Items and Icons if desired.

For proper operation of the dual-access memory portion of the Windows VMEbus Toolkit, the driver XVME984.SYS must be installed in your CONFIG.SYS file. By installing this driver, the dual-access memory mailbox driver will be loaded when your system is powered up or reset. Check "OK" if you want the install program to make the proper modification to your CONFIG.SYS file.

Select "OK" to exit the install program.

1.4 DISK CONTENTS

The XVME-984 Windows VMEbus Toolkit diskette contains directories and files.

1.4.1 \Directory (Root)

This directory contains the following:

SOURCE	-	Subdirectory
XVME984.DLL	-	Described in Chapter 4
INSTALL.EXE	-	Described in Chapter 1
LIBDEMO.EXE	-	Described in Chapter 3
VMEMAN.EXE	-	Described in Chapter 2
XVME984.SYS	-	Described in Chapter 4

NOTE

These are the only files that are installed with the install program. The source code for the demonstration program is located under the \SOURCE\LIBDEMO directory.

1.4.2 \SOURCE DIRECTORY

This directory contains the following:

LIB	-	Subdirectory
LIBDEMO	-	Subdirectory

1.4.3 \SOURCE\LIB

This directory contains the following:

INCLUDE	-	Subdirectory
XVME984 LIB	-	Import library for customers writing their own code using XVME-984 routines to link into

1.4.4 \SOURCE\LIB\INCLUDE

The \SOURCE\LIB\INCLUDE directory includes files which contain defined identifiers for the library. This subdirectory contains the following files:

XANALOG.H	-	Specific to Analog routines
XCOUNTER.H	-	Specific to Counter routines
XDIGITAL.H	-	Specific to Digital routines
XVME984.H	-	Specific to General Purpose routines

1.4.5 **\SOURCE\LIBDEMO**

The \SOURCE\LIBDEMO directory includes files which contain source code and build files for LIBDEMO.EXE. This subdirectory contains the following files:

BD.BAT
ANALOG.C
COUNTER.C
DIGITAL.C
GLOBALS.C
LIBCMD.C
LIBDEMO.C
LIBDEMO.DEF
ANALOG.DLG
COUNTER.DLG
DIGITAL.DLG
ANALOG.H
COUNTER.H
DIGITAL.H
LIBDEMO.H
XYCOM.ICO
LIBDEMO.LNK
LIBDEMO.MAK
LIBDEMO.RC
ANALOG.RES
COUNTER.RES
DIGITAL.RES

2.1 INTRODUCTION

The Xycom VMEbus Manager (VMEMAN.EXE) is a Windows application program which provides a simple mechanism for accessing and monitoring the VMEbus. When using a Xycom PC/AT processor, the program allows you to quickly test out your VMEbus system configuration by providing a convenient mechanism for reading and writing to VMEbus memory and monitoring the status of VMEbus interrupts and the PC/AT status registers. This program, like all Windows programs, is menu driven. The menus and their functional descriptions are as follows:

MENU	**SUBMENU**
Monitor	System Status
	VME IDs
Probe	Read VME
	Write VME
	Dual-Access RAM (-674, -677, -684, -686, -687)
	Pass/Fail LEDs
	Lock VMEbus
Interrupts	Enable/Disable Interrupts
	Auxiliary NMI Sources (-674, -677, -678, -688)
	VME Interrupt Sources (-674, -677, -678, -688)
	Set VME Interrupt Notify Levels
	Generate VME Interrupts (-674, -677, -684, -686, -687)
	Acknowledge VME Interrupts
About	About VMEMAN...

Figure 2-1 below shows the main VMEbus Manager Window. Select a menu by clicking the mouse on the menu heading desired. Each of the menus provide additional submenus. The following sections describe the options available for each menu.

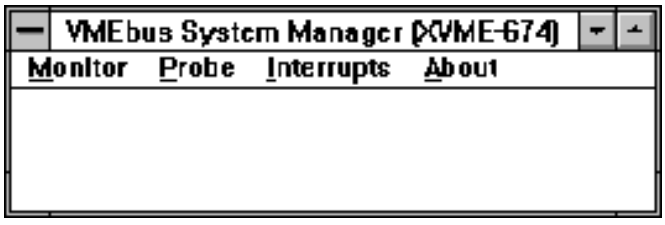

Figure 2-1. VMEbus Manager Window

2.2 MONITOR MENU

2.2.1 Monitor->System Status

This function opens a window and displays the contents of the PC/AT status registers. This window is updated every one second (default). The update rate can be changed by selecting the Timer menu option in the menu bar of the System Status window. You can force an immediate read of the status registers by selecting the Read option in the menu bar.

Figure 2-2 below shows the VMEbus Manager System Status window that appears on the screen. A description of each of section of the display is provided following Figure 2-2.

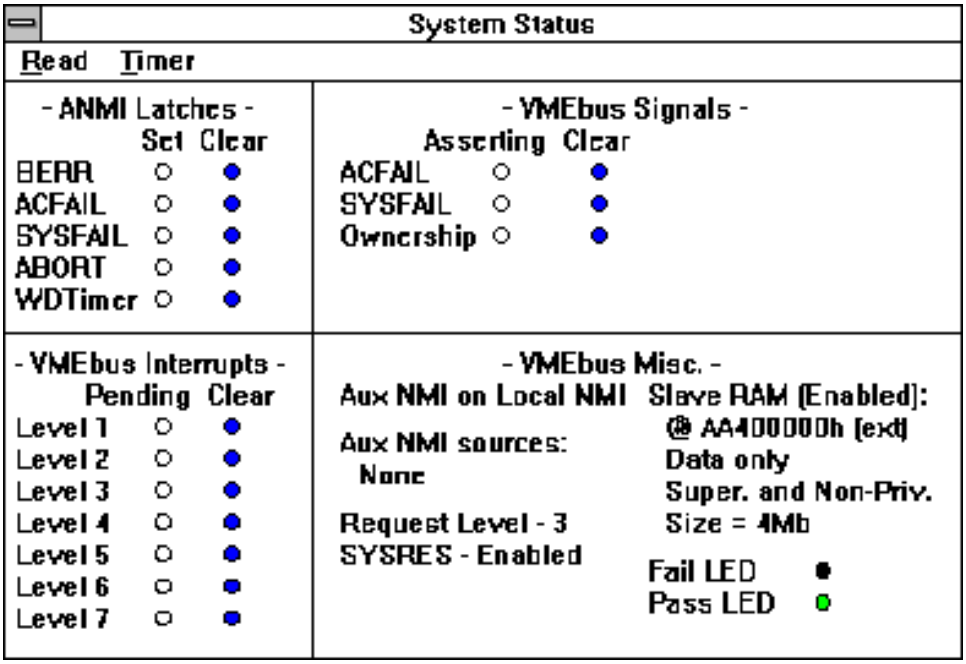

Figure 2-2. VMEbus Manager System Status Window

2.2.1.1 Read and Timer

By clicking on Read, the status of the items displayed in the System Status window are updated.

By clicking on Timer, you are given the option of entering the time interval with which the window is updated on a regular basis. Timer resolution is 1 msec, and the recommended minimum timer resolution is 50 msec. The timer interval you specify is always rounded down to an integral multiple of clock ticks. Each clock tick occurs every 54.925 msec. Any timer interval specified that is less than 55 msec will be treated as one clock tick.

2.2.1.2 **ANMI Latches**

This section of the window shows the current state of the auxiliary non-maskable interrupt latches. The circles next to each latch indicate the status (i.e., if the latch is clear, the circle under clear will be solid; alternatively, if the latch is set, the circle under set will be solid). There is no WDTimer or ACFAIL latch on the XVME-688 and XVME-678.

2.2.1.3 **VMEbus Signals**

This section of the window shows whether the signal on the VMEbus is currently asserted. A solid circle indicates asserted; an empty circle indicates not asserted.

The Ownership category shows whether the PC/AT processor currently has possession of the VMEbus.

2.2.1.4 **VMEbus Interrupts**

This section of the window shows the current state of the VMEbus interrupts (levels 1 through 7). A solid circle under the pending heading indicates asserted; a solid circle under the clear heading indicates no interrupt pending. An asterisk is placed next to the levels which are enabled for VMEbus interrupts on the XVME-674, -677, -678, -688.

2.2.1.5 **VMEbus Miscellaneous**

If you are using the XVME-686 board, the Slave RAM will indicate the VMEbus address of on-board dual access RAM **whatever** the bit address is—either 24 or 32.

If you are using the XVME-674, -677, -678, -688 the auxiliary NMI level, auxiliary NMI sources, VMEbus request level, and system resource provisions are displayed.

The Pass and Fail portion of the window indicates the state of the PC/AT front panel LEDs. The circle following the fail or pass indicators, by being solid or empty, indicates the state.

The Battery OK portion of the display uses on-board circuit to test whether the battery is functioning.

2.2.2 **Monitor->VME IDs**

This function searches the VMEbus Short I/O address space for Xycom ID PROMs. If any are found, a window is opened and a description of the board found is displayed. This description includes the Short I/O address where the board resides. If the board found is intelligent, the result of its power-on self test is displayed (PASS/FAIL). If the board is not intelligent, its green LED is turned on and its red LED is turned off.

Figure 2-3 below shows the VMEbus Manager System IDs window that appears on the screen.

Figure 2-3. System IDs Window

2.3 PROBE MENU

2.3.1 Probe->Read VME

This function allows you to display VMEbus memory. A dialog box is provided to give you the following options:

Option	**Description**
VME Access Type	This specifies the type of VMEbus cycle: IACK, Short I/O, Standard Address, EPROM, Extended Address.
Address Modifier	Choose Supervisory or Non-privileged.
Data Width (bits)	Choose 8, 16, or 32.
Byte Order	Choose Little Endian (80x86) or Big Endian (680x0).
Sustain VMEbus lock	Check box to keep VMEbus locked for entire "Number of bytes" transfer.
Monitor Address	Check box to read continually at intervals.
Memory Address offset	Edit field for memory address offset.
Number of Bytes	Edit field (1 - 384).
Title	Edit field for read window title.

Once the dialog box selections are made and you click on OK or press enter, a window opens containing the title you provided. If you selected Monitor Address in the dialog box, the window starts displaying data (or BUS ERROR). The window includes a menu bar with the following options:

Options	**Description**
Parameters	Changes the original parameters associated with the current read window
Read	Forces an immediate read.
Search	Opens a dialog box to perform a VMEbus memory search. This dialog box includes search options such as: Start Address, End Address, Address Increment, Data (hexadecimal or ASCII), Access Type.
Timer	Changes the update interval when monitoring the VMEbus address (default is one second).

Figure 2-4 shows the VME Read Parameters dialog box that appears on the screen when you select Read VME.

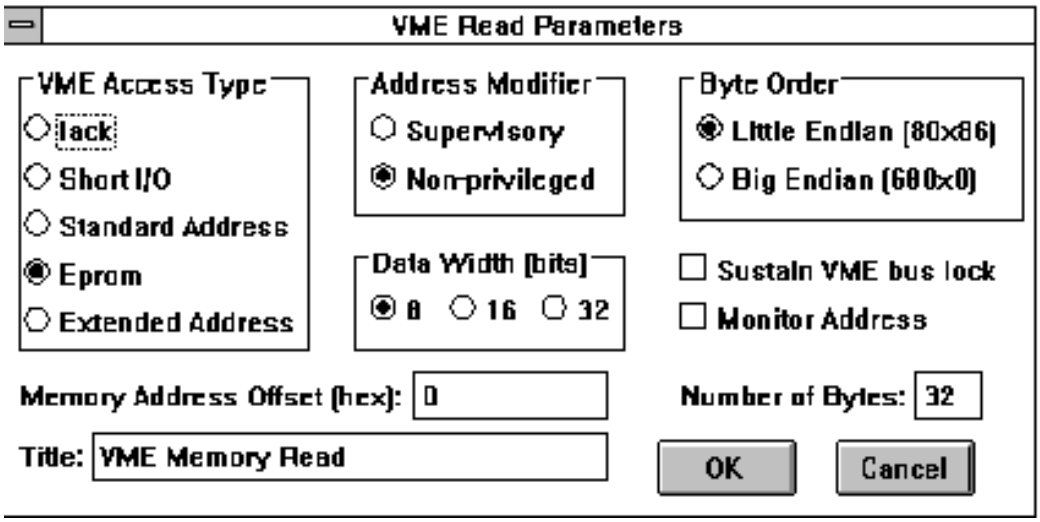

Figure 2-4. VME Read Parameters Dialog Box

2.3.2 **Probe->Write VME**

This function allows you to write to VMEbus memory. A dialog box is provided to give you the following options:

Option	**Description**
VME Access Type	This selects the type of VMEbus cycle: IACK, Short I/O, Standard Address, EPROM, Extended Address.
Address Modifier	Choose Supervisory or Non-privileged.
Data Width (bits)	Choose 8, 16, or 32.
Byte Order	Choose Little Endian (80x86) or Big Endian (680x0).
Continuous Write	Check box to write at intervals.
Increment Address	Check box to increment the address after each write.
Increment Data	Check box to increment the data after each write.
Title	Edit field for write window title.

Once the dialog box selections are made and you click OK or press enter, a window opens containing the title provided by you. If Continuous Write was selected in the dialog box, the window displays the number of successful writes. If the address supplied is not a valid VME address, the write window displays an error message. The window includes a menu bar with the following options:

Options	**Description**
Parameters	Changes the original parameters associated with the current write window.
Write	Forces an immediate write.
BlockFill	Opens a dialog box to perform a block fill. This dialog box includes options such as: Access Type, Memory Address Offset, Number of Bytes, Data.
Timer	Changes the update interval when writing to the VMEbus address (default is one second).

Figure 2-5 shows the VME Write Parameters dialog box that appears on the screen when VME Write is selected.

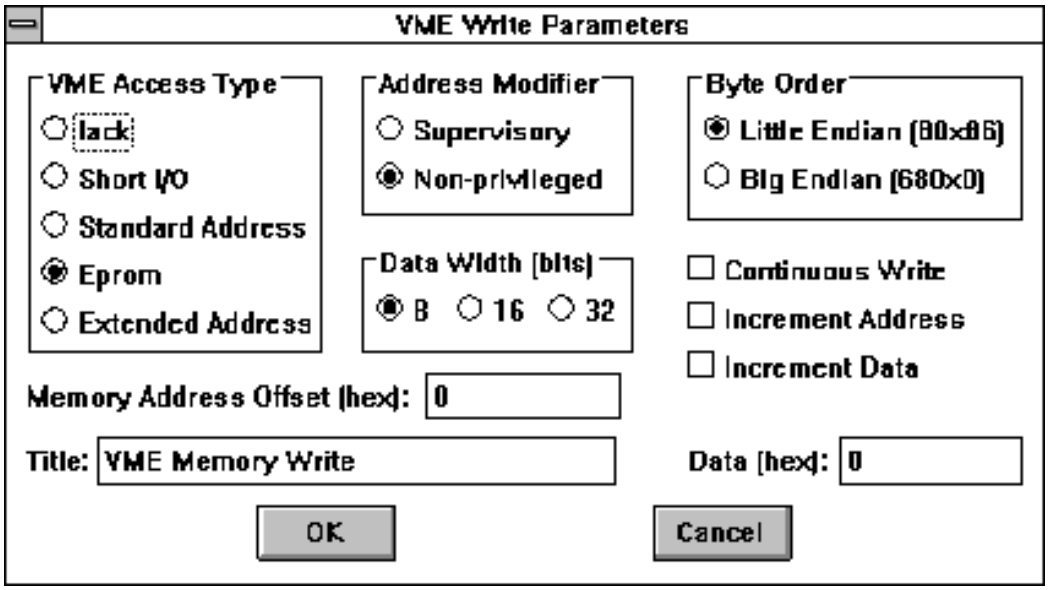

Figure 2-5. VME Write Parameters dialog box

2.3.3 **Probe->Dual-Access RAM (-674, -677, -684, -686, -687)**

This function allows you to allocate and display PC/AT memory. It uses the XVME984.SYS device driver to notify an external processor of the address of the dual-access window. Refer to Chapter 4 for more information on the operation of the XVME984.SYS device driver.

This function provides you with a dialog box to give you the following options:

Option	**Description**
Number of Bytes	Edit field for size of buffer.
Title	Edit field for dual-access window title.

Once the dialog box selections are made and OK has been clicked on or enter was pressed, a window opens containing the title provided by you. If Monitor Address was selected in the dialog box, the window starts displaying data. The window includes a menu bar with the following options:

Options	**Description**
Parameters->Read	Opens a dialog box to set or change the way the data is read and displayed. It includes options such as: Data Width, Byte Order, Buffer Offset, Monitor Address.
Parameters->Write	Opens a dialog box to allow for writes into the buffer. It includes options such as: Data Width, Byte Order, Buffer Offset, Continuous Write, Increment Address, Increment Data, Data.
Read	Forces an immediate read.
Write	Forces an immediate write
BlockFill	Opens a dialog box to perform a block fill of data in the buffer. It includes options such as: Buffer Offset, Number of Bytes, Data.
Timer	Changes the update interval when reading and/or writing to the buffer (default is one second).

Figure 2-6 on the following page shows the Dual-Access Parameters dialog box that appears on the screen when Dual Access RAM is selected.

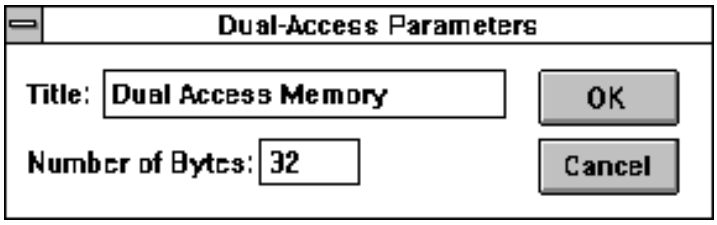

Figure 2-6. Dual-Access Parameters Dialog Box

2.3.4 **Probe->Pass/Fail LEDs**

This function opens a dialog box to allow you to change the status of the front panel red and green LEDs on the Xycom PC/AT VME processor.

2.3.5 **Probe->Lock VMEbus**

This function requests the VMEbus on behalf of the VMEbus PC/AT. Once possession of the bus has been obtained, the VMEbus PC/AT keeps possession of the bus until this option is selected again. A check mark next to this menu selection indicates that the bus is being held.

> **NOTE**
> If a read or write VMEbus window is opened and is accessing the VMEbus at intervals, the VMEbus will be released by its control. For example, with the bus lock mechanism enabled, starting a read window with monitor address selected will force the bus to be unlocked. This occurs because the low level routine for read window obtains the bus, then releases before returning.

Figure 2-7 shows the Pass/Fail LEDs dialog box that appears on the screen when the Lock VMEbus option is selected.

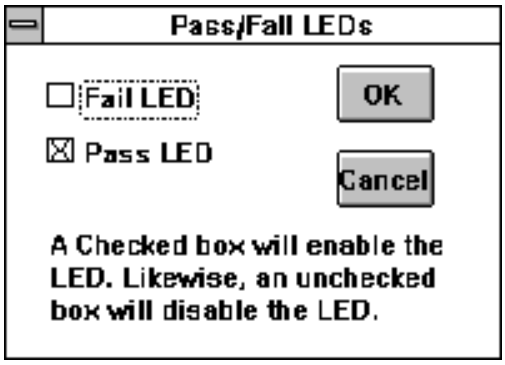

Figure 2-7. Pass/Fail LEDs Dialog Box

2.4 INTERRUPT MENU

2.4.1 Interrupts->Enable/Disable Interrupts

This function provides a dialog box to allow you to enable/disable the following types of interrupts:

Auxiliary Maskable	VMEbus interrupts on IRQ 10
Auxiliary Non-Maskable	ACFAIL, SYSFAIL, ABORT
BERR/IOCHCK	Bus error
Watchdog Timer	Watchdog timer interrupts

NOTE
For proper operation of interrupts, the corresponding enable jumpers must be installed. Refer to the manual of the PC/AT board you are using for further information.

Figure 2-8 shows the VMEbus Enable/Disable Interrupts dialog box that appears on the screen.

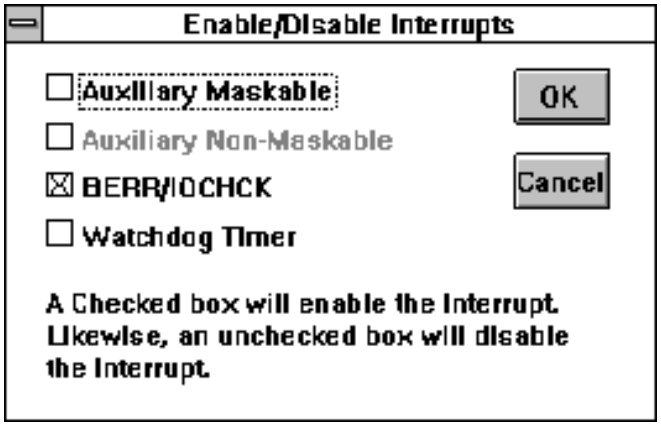

Figure 2-8. VMEbus Enable/Disable Interrupts Dialog Box

2.4.2 **Interrupts->Set VME Interrupt Notify Levels**

This function provides a dialog box to allow you to choose the VMEbus interrupt levels on which to be notified. If auxiliary maskable interrupts are enabled, a message box is displayed whenever a VMEbus interrupt at the level(s) chosen in this dialog box occurs. This message box displays the interrupt level and vector.

The Interrupt Service Routine (ISR) re-enables VMEbus interrupts. Clicking on this box allows the processor to re-enable the VMEbus interrupt.

NOTE

The ISR VMEbus Interrupt option should be used only when the interrupt source can be cleared via a VMEbus IACK cycle. If the interrupt source cannot be cleared via the VMEbus IACK cycle, then you should clear the interrupt source before re-enabling the auxiliary maskable interrupts.

Figure 2-9 shows the VMEbus Interrupt Notification dialog box that appears on the screen.

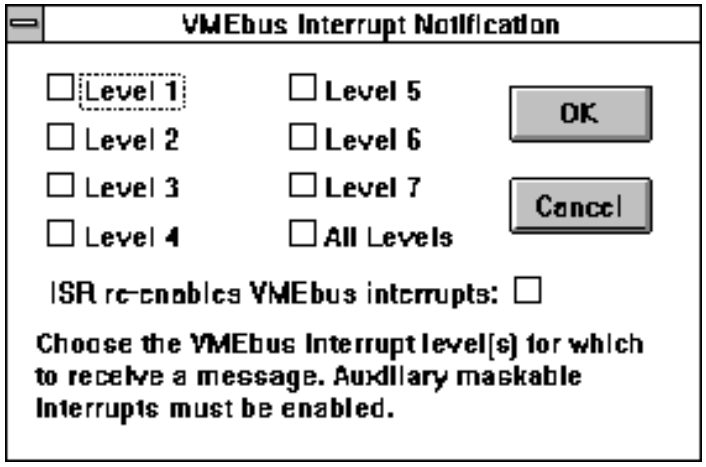

Figure 2-9. VMEbus Interrupt Notification Dialog Box

2.4.3 **Interrupts->Generate VME Interrupts (-674, -677, -684, -686, -687)**

This function provides a dialog box to allow you to choose the interrupt level and the interrupt vector used to generate a VMEbus interrupt. When OK is selected, a VMEbus interrupt is generated.

2.4.4 **Interrupts->Acknowledge VME Interrupts**

This function provides a dialog box to allow you to acknowledge a pending VMEbus interrupt. The dialog box contains the interrupt levels to choose from. When OK is selected, the VMEbus interrupt is acknowledged and the results are displayed in the VMEbus System Manager window. The result includes the interrupt vector read.

2.4.5 **Interrupts->Auxiliary NMI Sources (-674, -677, -678, -688)**

This function provides a dialog box allowing you to set the auxiliary NMI level (NMI or IRQ10) and to enable the auxiliary NMI sources (BERR, SYSFAIL, Abort). A checkbox is provided which, if checked, will cause a notification box to be displayed whenever an auxiliary NMI occurs (IRQ10 only).

2.4.6 **Interrupts->VME sources (-674, -677, -678, -688)**

This function provides a dialog box allowing you to individually enable or disable the VMEbus level sources.

2.4.7 **About->About VMEMAN…**

This function displays a message box containing the version number of VMEMAN.EXE.

3.1 INTRODUCTION

The Xycom Demonstration program (LIBDEMO.EXE) is a Windows application program which provides a reference showing how to make calls into the Xycom XVME-984 Windows library XVME984.DLL. It provides a dialog box for each library function. The dialog box contains the syntax of the function, a description of the parameters, a description of what the function provides, and edit field entries to actually make the function call on-line. This program is menu driven just like most Windows applications. A main window is opened immediately with a menu bar showing the three main categories of XVME-984 I/O functions. These three menu items are Analog, Counter, and Digital. The main window also shows the Interrupt and About menus. The menu structure is organized as follows:

Analog	->XVME-500->All the XVME-500\590 functions
	->XVME-505->All the XVME-505\595 functions
	->XVME-530->All the XVME-530 functions
	->XVME-540->All the XVME-540 functions
	->XVME-545->All the XVME-545 functions
	->XVME-560->All the XVME-560 functions
	->XVME-566->All the XVME-566 functions
Counter	->XVME-203->All the XVME-203\293 functions
	->XVME-230->All the XVME-230 functions
Digital	->XVME-200->All the XVME-200\290 functions
	->XVME-201->All the XVME-201 functions
	->XVME-202->All the XVME-202 functions
	->XVME-210->All the XVME-210 functions
	->XVME-212->All the XVME-212 functions
	->XVME-220->All the XVME-220 functions
	->XVME-240->All the XVME-240 functions
	->XVME-244->All the XVME-244 functions
	->XVME-260->All the XVME-260 functions

The Interrupt menu selection provides a dialog box to allow you to choose the VMEbus interrupt levels on which to be notified. If auxiliary maskable interrupts are enabled, a message box is displayed whenever a VMEbus interrupt at the level(s) chosen in this dialog box occurs. This message box will display the interrupt level and vector.

The About menu item displays the version number of the Demo program.

3.2 RUNNING THE DEMONSTRATION PROGRAM

To run the Library Demonstration programs, click on the LIBDEMO icon. A window appears with a menu bar containing the Analog, Counter, Digital, Interrupts, and About menus. Clicking on one of these menus will provide you with a list of Xycom I/O modules supported in that menu. The Library Demo routines have been grouped according to I/O module family type. Figure 3-1 shows an example of the Analog menu selection.

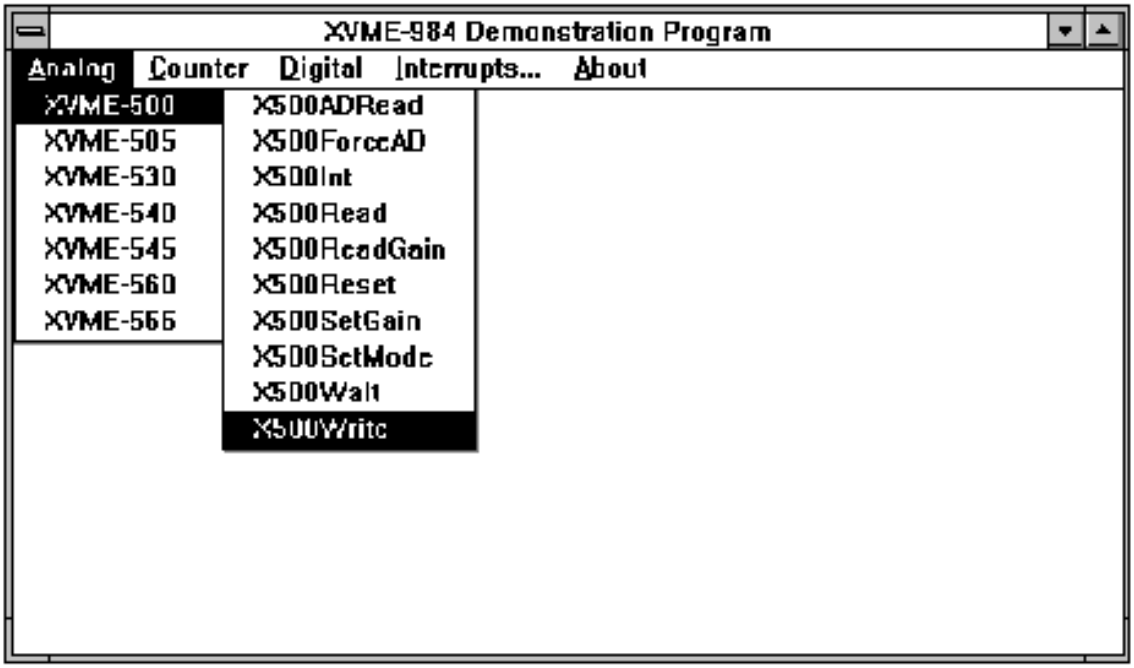

Figure 3-1. Analog Menu Selection

By selecting one of the Xycom I/O modules in the list (in this case the XVME-500) a submenu will appear. This submenu lists the names of the routines that support the XVME-500.

A dialog box appears after you select a routine name such as X500Write. Figure 3-2 below shows the dialog box for the X500Write routine.

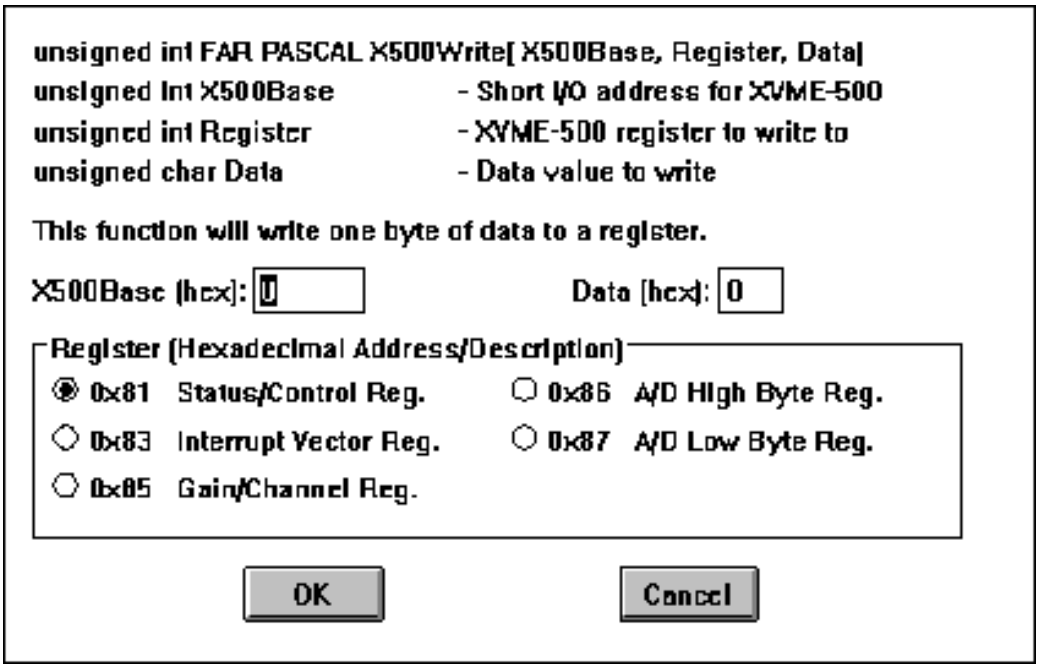

Figure 3-2. X500Write Routine Dialog Box

There are several items that are common to all routine dialog boxes. The following sections describe these items.

3.2.1 **Unsigned int FAR PASCAL X500Write(X500Base, Register, Data)**

The first line of the dialog box shows the definition of the function. Immediately below this line are the descriptions of the parameters passed to the routine. Refer to Chapter 4 for more information on the parameters. After the parameter list is a brief description of what operation the routine will perform.

3.2.2 **X500Base (hex):**

Each routine provides a text box for the module address. This address should be provided as the Short I/O address of the module. Once this address is given, other subsequent routines for that module assume the same address. To change this address, simply type in the value in any of the dialog boxes for routines that access that module.

3.2.3 **Data (hex):**

Routines that write data to the I/O module provide this text box. Enter the data you want written to the module in the text box and click the left mouse button or press the tab key to advance to the next field.

3.2.4 **Register (Hexadecimal Address/Description)**

This option is available on read or write routines. To make a selection, simply place the cursor on the radio button next to the selection desired then click the left mouse button or press the space bar.

3.2.5 **OK**

After all the options have been set as desired, move the cursor to OK and click the left mouse button or press enter. The routine is then executed. A message is displayed indicating that successful execution was made or that an error was encountered.

3.2.6 **Cancel**

Select Cancel to close the dialog box without performing the operation.

4.1 INTRODUCTION

The Xycom Dynamic Link Library (DLL) contains library routines with all the low-level codes required to configure and communicate with Xycom VMEbus I/O modules. The specific library routines can be categorized as follows:

- VMEbus General Purpose routines
- Analog I/O VME product routines
- Counter I/O VME product routines
- Digital I/O VME product routines

This chapter discusses these routines, providing a description of each. This chapter also discusses the real mode window, the VMEbus PC/AT processor interrupt capability, and the XVME984.SYS MS-DOS device driver.

4.2 REAL MODE WINDOW

Some of the General Purpose routines use the real mode window. The real mode window provides a mechanism for addressing the entire VMEbus memory space without the need to run the CPU in protected mode. It is 64 Kbytes long and resides at addresses 0E0000-0EFFFF. The window can be configured, via software, to address one of the following: VMEbus Short I/O, VMEbus Standard address space, VMEbus IACK space, or EPROM. You can also configure some PC/AT modules to address VMEbus Extended address space. Refer to your PC/AT module manual to determine whether you can access Extended address space.

When the window is configured for VMEbus short I/O, the 64 Kbyte Short I/O address space may be accessed through the 64 Kbyte window. Any references to the window will map directly into the VMEbus Short I/O space.

When the window is configured for VMEbus Standard address space, the 64 Kbyte window may be used to access any 64 Kbyte block of the Standard VMEbus address space. In this mode, the 16 Mbyte Standard address space is logically divided into 256 64 Kbyte blocks which are configured through software.

When the window is configured for VMEbus IACK cycle, a byte read from specific locations in the window will cause the PC/AT processor to perform a VMEbus IACK cycle. The data returned from the byte read will be the status ID vector returned from the responding interrupter.

When the window is configured for EPROM, the lower 64 Kbytes of the PC/AT's EPROM will appear in the window. This is the mode selected after reset. This mode is compatible with the IBM PC/AT architecture.

When the window is configured for VMEbus Extended address space, the 64 Kbyte window may be used to access any 64 Kbyte block of the VMEbus Extended address space. In this mode, the 4 Gbyte Extended address space is logically divided into 65,536 64 Kbyte blocks which are configured through software.

4.3 **INTERRUPTS**

All Xycom VMEbus PC/AT processor modules are capable of handling interrupts on all seven VMEbus interrupt levels. The XVME-674, -677, -684, -686, and -687 also contain a VMEbus interrupter circuit. This local interrupter allows the local CPU to generate a VMEbus interrupt on any of the seven VMEbus interrupt levels.

4.4 **XVME984.SYS**

The XVME984.SYS is a MS-DOS device driver which, in conjunction with the XVME-984 library function SendA32ToXVMEDosDeviceDriver(), provides support for Windows applications that need to share the dual-ported RAM on the XVME-CPU with another off-board CPU on the VMEbus. This driver provides the off-board CPU with a consistent method for obtaining an address of dual-ported RAM that was allocated by a Windows application. When loaded it reserves an internal data area to store the address passed to it from the Windows application.

The driver stores the address of this internal data area at physical memory offset 200H. Both of these addresses are stored in 4 bytes with the MSB at the lowest address and the LSB at the highest address. For an off-board CPU to obtain the address of the memory allocated by a Windows application, it must perform two reads from the dual-ported RAM. It must first read offset 200H to get the address of the data area in the driver. Then it must read the data area in the driver to get the address of the buffer to be shared with the Windows application. The reason that two reads are necessary is because Windows will overwrite the vector table (which includes 200H) with its original contents whenever a DOS session is executed. Whatever is written to the data area of the device driver will be preserved until the next SendA32ToXVMEDosDeviceDriver() call.

NOTE

It is important to note that Windows applications must keep the shared memory **locked** under the operating system. If an off-board CPU blindly writes into the dual-ported RAM, there is a great chance that the system will lock up.

The device driver is loaded by installing the following line (which can be done with the XVME-984 install program) to CONFIG.SYS:

device=pathname\XVME984.SYS

4.5 XYCOM I/O DYNAMIC LINK LIBRARY - XVME 984.DLL

The Xycom I/O Dynamic Link Library was developed according to specification as described in the Microsoft Windows Software Development Kit (SDK) for dynamic link libraries (DLLs). If you are writing Windows applications, you can interface to the XVME984.DLL just as you would any other Windows or third party dynamic link libraries.

A header file XVME984.H is provided on the diskette which contains all function prototypes and special constant names. A link file XVME984.LIB is also included on the diskette which provides the dynamic link information necessary at application link time.

The specific library routines can be categorized and are described as follows:

- VMEbus General Purpose routines
- Xycom Analog I/O VME product routines
- Xycom Counter I/O VME product routines
- Xycom Digital I/O VME product routines

The error codes are defined in the XVME984.H header file for customer convenience. They are described in Table 4-1 on the following page.

Table 4-1. XVME984.H Error Codes

Error	**Description**
NOERROR	No errors.
ERRORCODE_BERR	Bus error occurred.
ERRORCODE_BLOCKSIZE	Invalid BlockSize parameter for Read/Write VMEbusMemoryRM.
ERRORCODE_INVALIDPARM	General purpose invalid parameter
ERRORCODE_MEMORYALLOC	The function couldn't allocate the necessary memory.
ERRORCODE_NOTCPUSUPPORTED	The function is not supported for the current CPU board.
ERRORCODE_ODDALIGNMENT	Xycom PC/AT processor modules do not support odd alignment 16 or 32 bit transfers with byte swapping enabled.
ERRORCODE_TIMEOUT	Couldn't acquire the VMEbus in the specified time. NOTE: All library functions that access the VMEbus employ a two second time-out period.
ERRORCODE_TRANSFERTYPE	Invalid transfer parameter for Read/WriteVMEBusMemoryRM.

4.6 GENERAL PURPOSE ROUTINES

The routines in this library section are designed to handle the Watchdog timer, process interrupts, or transfer memory on any Xycom VMEbus PC/AT processor modules. Functions unique to a specific CPU are indicated.

The parameters you use when implementing the General Purpose routines must match the type expected by the routine.

The type definitions for the parameters as described in this document are as follows:

Table 4-2. Parameter Definitions

Type Definition	Size	C Equivalent
BYTE	1 byte	(unsigned char)
WORD	2 bytes	(unsigned int)
LPSTR	4 bytes	(char far *)
HWND	2 bytes	(unsigned int)
DWORD	4 bytes	(unsigned long)
LPWORD	4 bytes	(WORD far *)
LPBYTE	4 bytes	(BYTE far *)
LPDWORD	4 bytes	(DWORD far *)

Certain parameters for the General Purpose routines have reserved constants, as defined in XVME984.H and shown in Table 4-3:

Table 4-3. Reserved Constants

Parameter	**Constants**
AccessType	IACK_ACCESS: Interrupt acknowledge space SHORT_IO_ACCESS: Standard address space PROM_ACCESS: XVME-CPU ROM address space EXTEND_ADDR_ACCESS: Extended address space
EnableFlag	DISABLE ENABLE
EndianType	BIGENDIAN: 680X0 byte ordering LITTLEENDIAN: 80X86 byte ordering
DataWidth	TRANSFER8: 8 bits TRANSFER16: 16 bits TRANSFER32: 32 bits
SustainVMEbus	LOCK: Keep the VMEbus for entire blocksize transfer UNLOCK:Release and then reacquire the VMEbus for each DataWidth transfer within the blocksize

The remainder of this section provides a list of the name and description for each General Purpose routine.

4.6.1 **Allocate Dual Port Memory**

Syntax:

BOOL AllocDualPortMem (DPDescAddr)

Function: This routine allocates a contiguous block of memory to use for dual access. This memory is paged locked in physical memory until it is freed using FreeDualPortMem (refer to Section 4.6.8). Four Kbytes is the maximum amount of physically contiguous memory that can be reliably allocated.

Parameter	**Type**	**Description**
DPDescAddr	DUALPORT FAR*	Address of a DUALPORT data structure

The DUALPORT data structure is defined in XVME984.H as

```
typedef struct  {
                DWORD dwNumBytes;
                DWORD dwOffset;
                DWORD dwID;
                DWORD dwPhysAddr;
                DWORD dwSegsel;
                } DUALPORT;
```

where:

dwNumBytes	Must be assigned to the number of bytes to allocate.
dwOffset	Reserved.
dwID	Reserved.
dwPhysAddr	Contains the 32-bit physical address of the allocated memory upon return.
dwSegSel	Contains the selector value returned by the Windows API call GlobalAlloc.

This routine will return TRUE if successful.

4.6.2 **Disable NMI Interrupts**

Syntax:

void DisableNMIInt()

Function: This routine disables the auxiliary NMI error conditions.

4.6.3 **Disable VME Interrupts**

Syntax:

void DisableVMEInterrupts()

Function: This routine disables the auxiliary maskable interrupts used for VMEbus interrupt levels 1-7.

4.6.4 **Disable Watchdog Timer**

Syntax:

void DisableWDTimer()

Function: This routine disables the Watchdog timer.

4.6.5 **Enable NMI Interrupts**

Syntax:

void EnableNMIInt()

Function: This routine enables the auxiliary NMI error conditions.

4.6.6 **Enable VME Interrupts**

Syntax:

void EnableVMEInterrupts()

Function: This routine enables the auxiliary maskable interrupts used for VMEbus interrupt levels 1-7.

4.6.7 **Enable Watchdog Timer**

Syntax:

void EnableWDTimer()

Function: This routine enables the Watchdog timer.

4.6.8 **Free Dual Port Memory**

Syntax:

BOOL FreeDualPortMem (DPDescAddr)

Function: This routine frees the memory that was previously allocated using AllocDualPortMem.

Parameter	**Type**	**Description**
DPDescAddr	DUALPORT FAR*	Address of a DUALPORT data structure

DPDescAddr must contain the information filled by AllocDualPortMem (refer to Section 4.6.1). This routine will return TRUE if successful.

4.6.9 **Generate VMEbus Interrupt**

Syntax:

WORD GenVMEBusInt(Level, Vector)

Function: This routine generates an interrupt on the VMEbus.

Parameter	**Type**	**Description**
Level	BYTE	VMEbus interrupt level to generate
Vector	BYTE	Vector number to put out to the VMEbus upon acknowledgement

This routine will return an error code as defined in XVME984.H.

4.6.10 **Get Interrupt Address**

Syntax:

DWORD GetIntAddress(Vector)

Function: This routine returns the address of the interrupt handler for the interrupt vector specified by the vector parameter.

Parameter	**Type**	**Description**
Vector	WORD	Identifies the interrupt vector

The return value is the 32-bit address of the interrupt handler. The high word contains the selector of the address while the low word contains the offset.

4.6.11 Get Real Mode Window Data 8

Syntax:

void GetRMWdata8(DestAddress, NumBytes, Offset)

Function: This routine transfers a number of bytes from the real mode window to a local buffer.

Parameter	Type	Description
DestAddress	LPSTR	Address of the destination buffer
NumBytes	WORD	Number of bytes to transfer
Offset	WORD	Offset into the real mode window to transfer from

This routine does not do any error checking and does not request the VMEbus before accessing the real mode window.

4.6.12 Get Real Mode Window Data 16

Syntax:

void GetRMWdata16(DestAddress, NumWords, Offset)

Function: This routine transfers a number of words from the real mode window to a local buffer.

Parameter	Type	Description
DestAddress	LPSTR	Address of the destination buffer
NumWords	WORD	Number of words to transfer
Offset	WORD	Offset into the real mode window to transfer from

This routine does not do any error checking and does not request the VMEbus before accessing the real mode window.

4.6.13 Get Real Mode Window Data 32

Syntax:

void GetRMWdata32(DestAddress, NumDWords, Offset)

Function: This routine transfers a number of double words from the real mode window to a local buffer.

Parameter	Type	Description
DestAddress	LPSTR	Address of the destination buffer
NumDWords	WORD	Number of double words to transfer
Offset	WORD	Offset into the real mode window to transfer from

This routine does not do any error checking and does not request the VMEbus before accessing the real mode window.

4.6.14 Is XVME DOS Device Driver Installed

Syntax:

WORD IsXVMEDosDeviceDriverInstalled()

Function: This routine determines if the supporting MS-DOS device driver XVME984.SYS is installed. It returns a TRUE (1) value if it is.

4.6.15 Lock VMEbus

Syntax:

BYTE LockVMEBus(TimeOut)

Function: This routine requests the VMEbus and then waits for access.

Parameter	Type	Description
TimeOut	BYTE	Period of time to wait for bus access at 55 ms resolution (0 = infinite)

This routine returns a 0 value if the VMEbus access has been granted. A non-zero value is returned if the TimeOut value expired.

4.6.16 **Mask 8259**

Syntax:

void Mask8259(IRQLevel, EnableFlag)

Function: This routine sends the appropriate mask to the slave 8259 interrupt controller to either set or reset the IRQ level.

Parameter	**Type**	**Description**
IRQLevel	BYTE	IRQ level to mask (8-15)
EnableFlag	BYTE	ENABLE or DISABLE as defined in XVME984.H

4.6.17 **Notify On VME Interrupt**

Syntax:

BYTE NotifyOnVMEInterrupt(hWnd, Levels)

Function: This routine adds the window handle (hWnd) to the list of window handles to be notified when a VMEbus interrupt occurs.

Parameter	**Type**	**Description**
hWnd	HWND	Handle to the window to be notified
Levels	BYTE	Mask of VMEbus interrupt levels for notification
ReEnableVMEINts	BYTE	Tells the ISR whether to re-enable the VMEbus interrupts or not. This value should be either REENABLEVMEINTS or LEAVEVMEINTSDISABLED as defined in XVME984.H

This routine allows for a maximum of 10 windows to be notified. An error code is returned as described in XVME984.H. When this routine is called, the library installs an interrupt handler for IRQ 10 (XVME-CPUs **must** have VMEbus interrupts jumpered for this AT IRQ level). This interrupt handler acknowledges ALL VMEbus interrupts that come in. It will post a message (WM_USER) to all windows in the list whose level mask has its bit set to the VMEbus level that came in. The message contains the VMEbus vector read in the wParam parameter; the VMEbus level (1-7) generated in the high word of the 1Param parameter; and the AT IRQ level in the low word of the 1Param parameter.

4.6.18 Put Interrupt Address

Syntax:

DWORD FAR PASCAL PutIntAddress(Vector, HandlerAddress)

Function: This routine sets an interrupt vector to point to an interrupt handling routine.

Parameter	Type	Description
Vector	WORD	Identifies the interrupt vector
HandlerAddress	DWORD	Identifies the address of the interrupt handler. The upper word contains the selector; the lower word contains the offset.

The return value is the 32-bit address of the original interrupt handler. The upper word contains the selector of the address while the lower word contains the offset. This General Purpose routine will work properly for all AT interrupt vectors. However, Windows will not give up control of some vectors (i.e, NMI, keyboard,timer, etc.).

4.6.19 Put Real Mode Window Data 8

Syntax:

void PutRMWdata8(SourceAddress, NumBytes, DestOffset)

Function: This routine transfers a number of bytes from a local buffer to an offset in the real mode window.

Parameter	Type	Description
SourceAddress	LPSTR	Address of the source buffer
NumBytes	WORD	Number of bytes to transfer
DestOffset	WORD	Offset into the real mode window to transfer to

This routine does not do any error checking and does not request the VMEbus before accessing the real mode window.

4.6.20 Put Real Mode Window Data 16

Syntax:

void PutRMWdata16(SourceAddress, NumWords, DestOffset)

Function: This routine transfers a number of words from a local buffer to an offset in the real mode window.

Parameter	Type	Description
SourceAddress	LPSTR	Address of the source buffer
NumWords	WORD	Number of words to transfer
DestOffset	WORD	Offset into the real mode window to transfer to

This routine does not do any error checking and does not request the VMEbus before accessing the real mode window.

4.6.21 Put Real Mode Window Data 32

Syntax:

void PutRMWdata32(SourceAddress, NumDWords, DestOffset)

Function: This routine transfers a number of double words from a local buffer to an offset in the real mode window.

Parameter	Type	Description
SourceAddress	LPSTR	Address of the source buffer
NumDWords	WORD	Number of double words to transfer
DestOffset	WORD	Offset into the real mode window to transfer to

This function does not do any error checking and does not request the VMEbus before accessing the real mode window.

4.6.22 Quit Notify On VME Interrupt

Syntax:

void QuitNotifyOnVMEInterrupt(hWnd)

Function: This routine removes the window handle (hWnd) from the list of window handles to be notified when a VMEbus interrupt occurs.

Parameter	Type	Description
hWnd	HWND	Handle to the window to be removed from the notification list

4.6.23 Read Time Counter

Syntax:

DWORD Readtimecntr()

Function: This routine returns the system-timer time counter.

4.6.24 Read VMEbus Memory Real Mode

Syntax:

WORD ReadVMEBusMemoryRM(DestAddress, DataWidth, AM2, EndianType, NumBytes, AccessType, SourceAddress, SustainVMEbus)

Function: This routine transfers a block of memory from the specified VMEbus memory to the Xycom PC/AT processor module through the real mode window. This routine has the ability to do byte swapping; 8, 16, or 32 bit transfers; 1 to 64 Kbyte block size moves; and VMEbus locking.

Parameter	Type	Description
DestAddress	LPSTR	Address of destination buffer
DataWidth	BYTE	Width of data transfer. TRANSFER8, TRANSFER16, or TRANSFER32 as defined in XVME984.H.
AM2	BYTE	VMEbus address modifier. Either SUPERVISORY or NONPRIVILEDGED as defined in XVME984.H.
EndianType	BYTE	Designates byte ordering. Either BIGENDIAN or LITTLEENDIAN as defined in XVME984.H.
NumBytes	WORD	Number of bytes to transfer
AccessType	BYTE	Specifies the type of VMEbus access. It must be one of the following as defined in XVME984.H:

IACK_ACCESS	VME interrupt acknowledgement
SHORT_IO_ACCESS	Short I/O VME address
STAND_ADDR_ACCESS	Standard VME address
EPROM_ACCESS	XVME-CPU EPROM address
EXTEND_ADDR_ACCESS	Extended VME address

SourceAddress	DWORD	VMEbus address from which to transfer
SustainVMEbus	BYTE	If the low bit of this parameter is set, then this routine will sustain the VMEbus for the entire NumBytes transfer. If the low bit is not set, the routine releases and then reacquires the bus for each DataWidth transfer. If the high bit of this parameter is set, then the two second timeout (default) will be overridden and the routine will wait infinitely for the bus.

This routine returns an error code as defined in XVME984.H.

4.6.25 **Read_Flag_Register**

Syntax:

WORD Read_Flag_Reg()

Function: This routine returns the contents of the CPU flag register.

4.6.26 **Release VMEbus**

Syntax:

void ReleaseVMEBus()

Function: This routine releases the VMEbus from XVME-CPU control.

4.6.27 **Reset Watchdog Timer**

Syntax:

void ResetWDTimer()

Function: This routine resets the Watchdog timer after it has timed out.

4.6.28 **Restore_Flag_Register**

Syntax:

void Restore_Flag_Reg(FlagRegVal)

Function: This routine changes the contents of the CPU flag register.

Parameter	**Type**	**Description**
FlagRegVal	WORD	Flag register contents

4.6.29 **Send Address to Device Driver**

Syntax:

WORD SendA32ToXVMEDosDeviceDriver (Address)

Function: This routine sends a 32-bit address to the XVME948.SYS MS-DOS Device Driver.

Parameter	**Type**	**Description**
Address	DWORD	32-bit address of an allocated dual access buffer

The parameter for this routine should be obtained by the XVME-984 AllocDualPortMem routine. This routine will return FALSE if the XVME984.SYS driver is not installed.

4.6.30 **Set_Real Mode_Window**

Syntax:

Set_RM_Window(AccessType, Block64k)

Function: This routine sets the real mode window to the desired VMEbus address space.

Parameter	**Type**	**Description**
AccessType	BYTE	Specifies the type of VMEbus access. It must be one of the following as defined in XVME984.H:
	IACK_ACCESS	VME interrupt acknowledgement
	SHORT_IO_ACCESS	Short I/O VME address
	STAND_ADDR_ACCESS	Standard VME address
	EPROM_ACCESS	XVME-CPU EPROM address
	EXTEND_ADDR_ACCESS	Extended VME address
Block64k	WORD	Specifies which 64 Kbyte block to map in if AccessType is either

		STAND_ADDR_ACCESS or EXTEND_ADDR_ACCESS.

4.6.31 Strobe Watchdog Timer

Syntax:

void StrobeWDTimer()

Function: This routine will re-trigger the Watchdog timer. It must be executed at least once every 150 ms to keep the Watchdog timer from generating an ANMI (if enabled).

4.6.32 Write VMEbus Memory Real Mode

Syntax:

WORD WriteVMEBusMemoryRM(SourceAddress, DataWidth, AM2, EndianType, NumBytes, AccessType, DestAddress, SustainVMEbus)

Function: This routine transfers a block of memory from the Xycom PC/AT processor module to the specified VMEbus memory through the real mode window. This routine has the ability to do byte swapping; 8, 16, or 32 bit transfers; 1 to 64 Kbyte block size moves; VMEbus locking.

Parameter	**Type**	**Description**
SourceAddress	LPSTR	Address of source buffer
DataWidth	BYTE	Width of data transfer. TRANSFER8, TRANSFER16, or TRANSFER32 as defined in XVME984.H.
AM2	BYTE	VMEbus address modifier. Either SUPERVISORY or NONPRIVILEDGED as defined in XVME984.H.
EndianType	BYTE	Designates byte ordering. Either BIGENDIAN or LITTLEENDIAN as defined in XVME984.H.
NumBytes	WORD	Number of bytes to transfer.
AccessType	BYTE	Specifies the type of VMEbus access. It must be one of the following as defined in XVME984H:
	IACK_ACCESS	VME interrupt acknowledgement
	SHORT_IO_ACCESS	Short I/O VME address
	STAND_ADDR_ACCESS	Standard VME address
	EPROM_ACCESS	XVME-CPU EPROM address
	EXTEND_ADDR_ACCESS	Extended VME address
DestAddress	DWORD	VMEbus address to which to transfer

SustainVMEbus	BYTE	If the low bit of this parameter is set, then this routine sustains the VMEbus for the entire NumBytes transfer. If the low bit is not set, the routine releases and then reacquires the bus for each DataWidth transfer. If the high bit of the parameter is set, then the two-second timeout (default) will be overridden and the routine will wait infinitely for the bus.

This routine will return an error code as defined in XVME984.H.

4.6.33 XVME CPU Type

Syntax:

WORD XvmeCPUType()

Function: This routine returns the CPU type (as defined in XVME984.H) of the XVME-CPU board as determined upon initialization.

4.7 ANALOG LIBRARY

This library section provides a set of routines specific to Xycom Analog I/O VME products. They all return an error code as defined in XVME984.H and described in Section 4.5.

4.7.1 X500 AD Read

Syntax:

WORD X500ADRead(X500Base, WordAddress)

Function: This routine returns a 2 byte value containing the A/D reading for a channel.

Parameter	**Type**	**Description**
X500Base	WORD	Base address in Short I/O
WordAddress	LPWORD	Address for result

4.7.2 X500 Force AD

Syntax:

WORD ForceAD(X500Base)

Function: This routine sets the A/D busy bit in the status/control register which causes an A/D conversion to take place on the present channel.

Parameter	Type	Description
X500Base	WORD	Base address in Short I/O

4.7.3 X500 Interrupt

Syntax:

WORD X500Int(X500Base, IntFlag)

Function: This routine sets or resets the interrupt bit in the status/control register to either set or reset the XVME-500 interrupts.

Parameter	Type	Description
X500Base	WORD	Base address in Short I/O
IntFlag	BYTE	0 = Disable, >0 = Enable

4.7.4 X500 Read

Syntax:

WORD X500Read(X500Base, Register, ByteAddress)

Function: This routine reads a byte from a register from the XVME-500 board.

Parameter	Type	Description
X500Base	WORD	Base address in Short I/O
Register	WORD	Register to read
ByteAddress	LPBYTE	Address for result

4.7.5 X500 Read Gain

Syntax:

WORD X500ReadGain(X500Base, Channel, ByteAddress)

Function: This routine reads the gain factor stored for a channel. The return value is one of four gain factors.

Parameter	Type	Description
X500Base	WORD	Base address in Short I/O
Channel	BYTE	Channel to read
ByteAddress	LPBYTE	Address for result

4.7.6 X500 Reset

Syntax:

WORD X500Reset(X500Base)

Function: This routine performs a software reset on the XVME-500 board. The A/D busy (bit 7) and the interrupt pending (bit 2) are reset in the status/control register.

Parameter	Type	Description
X500Base	WORD	Base address in Short I/O

4.7.7 X500 Set Gain

Syntax:

WORD X500SetGain(X500Base, Channel, GainSelect)

Function This routine programs the gain RAM for a channel.

Parameter	Type	Description
X500Base	WORD	Base address in Short I/O
Channel	BYTE	Channel to set the gain RAM for
GainSelect	BYTE	Gain factor as related to the jumper settings (0 - 3)

4.7.8 X500 Set Mode

Syntax:

WORD X500SetMode(X500Base, Mode)

Function: This routine sets the XVME-500 module to one of the four analog conversion modes.

Parameter	Type	Description
X500Base	WORD	Base address in Short I/O
Mode	BYTE	0 - Single channel 1 - Sequential channel 2 - Random channel 3 - External channel

4.7.9 X500 Wait

Syntax:

WORD X500Wait(X500Base)

Function: This routine waits for an A/D conversion to be completed before continuing.

Parameter	Type	Description
X500Base	WORD	Base address in Short I/O

4.7.10 X500 Write

Syntax:

WORD X500Write(X500Base, Register, Byte)

Function: This routine writes 1 byte of data to the XVME-500.

Parameter	Type	Description
X500Base	WORD	Base address in Short I/O
Register	WORD	Register to write to
Byte	BYTE	Byte of data to write

4.7.11 X505 Output

Syntax:

WORD X505Output(X505Base, Channel, OutLevel)

Function: This routine writes a 12 bit value into the channel output register which causes a voltage output corresponding to the 12 bit value.

Parameter	Type	Description
X505Base	WORD	Base address in Short I/O
Channel	BYTE	Channel to program (0 - 3)
OutLevel	WORD	12 bit value to write

4.7.12 X530 Channel Output

Syntax:

WORD X530ChanOut(X530Base, Channel, OutLevel)

Function: This routine writes a 12 bit value into the channel output register which causes a voltage output corresponding to the 12 bit value.

Parameter	Type	Description
X530Base	WORD	Base address in Short I/O
Channel	BYTE	Channel to program (0 - 7)
OutLevel	WORD	12 bit value to write

4.7.13 X530 Read

Syntax:

WORD X530Read(X530Base, Register, ByteAddress)

Function: This routine reads the byte value at the register address.

Parameter	Type	Description
X530Base	WORD	Base address in Short I/O
Register	WORD	Register to read
ByteAddress	LPBYTE	Address for result

4.7.14 X530 Reset

Syntax:

WORD X530Reset(X530Base)

Function: This routine resets all eight channel outputs to 0 volts (or 4 mA if in current mode) and turns the green LED on and the red LED off.

Parameter	Type	Description
X530Base	WORD	Base address in Short I/O

4.7.15 X530 Wait

Syntax:

WORD X530Wait(X530Base)

Function: This routine waits for a D/A conversion to be completed on an output channel.

Parameter	Type	Description
X530Base	WORD	Base address in Short I/O

4.7.16 X530 Write

Syntax:

WORD X530Write(X530Base, Register, Byte)

Function: This routine writes a byte of data at the register address.

Parameter	Type	Description
X530Base	WORD	Base address in Short I/O
Register	WORD	Register address to write to
Byte	BYTE	Byte of data to write

4.7.17 X540 AD Read

Syntax:

WORD X540ADRead(X540Base, WordAddress)

Function: This routine reads the A/D value for a channel.

Parameter	Type	Description
X540Base	WORD	Base address in Short I/O
WordAddress	LPWORD	Address for result

4.7.18 X540 Channel Output

Syntax:

WORD X540ChanOut(X540Base, Channel, OutLevel)

Function: This routine writes a 12 bit value into the channel output register which causes a voltage output corresponding to the 12 bit value.

Parameter	Type	Description
X540Base	WORD	Base address in Short I/O
Channel	BYTE	Channel to program (0 - 3)
OutLevel	WORD	12 bit value to write

4.7.19 X540 Force AD

Syntax:

WORD X540ForceAD(X540Base)

Function: This routine sets the A/D busy bit in the status/control register which causes an A/D conversion to take place on the present channel.

Parameter	Type	Description
X540Base	WORD	Base address in Short I/O

4.7.20 X540 Interrupt

Syntax:

WORD X540Int(X540Base, IntFlag)

Function: This routine sets or resets the interrupt bit in the status/control register to either set or reset the XVME-540 interrupts.

Parameter	Type	Description
X540Base	WORD	Base address in Short I/O
IntFlag	BYTE	0 = Disable, >0 = Enable

4.7.21 X540 Read

Syntax:

WORD X540Read(X540Base, Register, ByteAddress)

Function: This routine reads the byte at a register address.

Parameter	Type	Description
X540Base	WORD	Base address in Short I/O
Register	WORD	Register to read
ByteAddress	LPBYTE	Byte address for result

4.7.22 X540 Read Gain

Syntax:

WORD X540ReadGain(X540Base, Channel, ByteAddress)

Function: This routine reads the gain factor stored for a channel.

> **NOTE**
> An A/D conversion is initiated for the channel when the gain RAM is read.

Parameter	Type	Description

X540Base	WORD	Base address in Short I/O
Channel	BYTE	Channel to read
ByteAddress	LPBYTE	Byte address for result

4.7.23 X540 Reset

Syntax:

WORD X540Reset(X540Base)

Function: This routine performs a software reset on the XVME-540 board. The A/D busy (bit 7) and the interrupt pending (bit 2) are reset in the status/control register.

Parameter	Type	Description
X540Base	WORD	Base address in Short I/O

4.7.24 X540 Set Gain

Syntax:

WORD X540SetGain(X540Base, Channel, GainSelect)

Function: This routine programs the gain RAM for a channel.

Parameter	Type	Description
X540Base	WORD	Base address in Short I/O
Channel	BYTE	Channel to set the gain RAM for
GainSelect	BYTE	Gain factor related to jumper settings (0 - 3)

4.7.25 X540 Set Mode

Syntax:

WORD X540SetMode(X540Base, Mode)

Function: This routine sets the XVME-540 module to one of the four analog conversion modes.

Parameter	Type	Description
X540Base	WORD	Base address in Short I/O
Mode	BYTE	0 - Single channel
		1 - Sequential channel
		2 - Random channel
		3 - External channel

4.7.26 X540 Wait

Syntax:

WORD X540Wait(X540Base)

Function: This routine waits for an A/D conversion to be completed before continuing.

Parameter	Type	Description
X540Base	WORD	Base address in Short I/O

4.7.27 X540 Write

Syntax:

WORD X540Write(X540Base, Register, Byte)

Function: This routine writes 1 byte of data to a register.

Parameter	Type	Description
X540Base	WORD	Base address in Short I/O
Register	WORD	Register to write to
Byte	BYTE	Byte of data to write

4.7.28 X560 AD Read

Syntax:

WORD X560ADRead(X560Base, WordAddress)

Function: This routine reads the A/D value for a channel.

Parameter	Type	Description
X560Base	WORD	Base address in Short I/O
WordAddress	LPWORD	Address for result

4.7.29 X560 Force AD

Syntax:

WORD X560ForceAD(X560Base)

Function: This routine sets the A/D busy bit in the status/control register which causes an A/D conversion to take place on the present channel.

Parameter	Type	Description
X560Base	WORD	Base address in Short I/O

4.7.30 X560 Interrupt

Syntax:

WORD X560Int(X560Base, IntFlag)

Function: This routine sets or resets the interrupt bit in the status/control register to either set or reset the XVME-560 interrupts.

Parameter	Type	Description
X560Base	WORD	Base address in Short I/O
IntFlag	BYTE	0 = Disable, >0 = Enable

4.7.31 X560 Read

Syntax:

WORD X560Read(X560Base, Register, ByteAddress)

Function: This routine reads the byte at a register address.

Parameter	Type	Description
X560Base	WORD	Base address in Short I/O
Register	WORD	Register to read
ByteAddress	LPBYTE	Byte address for result

4.7.32 X560 Reset

Syntax:

WORD X560Reset(X560Base)

Function: This routine performs a software reset on the XVME-560 board. The A/D busy (bit 7) and the interrupt pending (bit 2) are reset in the status/control register.

Parameter	Type	Description
X560Base	WORD	Base address in Short I/O

4.7.33 X560 Set Gain

Syntax:

WORD X560SetGain(X560Base, Channel, GainSelect)

Function: This routine programs the gain RAM for a channel.

Parameter	Type	Description
X560Base	WORD	Base address in Short I/O
Channel	BYTE	Channel to set the gain RAM for
GainSelect	BYTE	Gain factor related to jumper settings (0 - 3)

4.7.34 X560 Set Mode

Syntax:

WORD X560SetMode(X560Base, Mode)

Function: This routine sets the XVME-560 module to one of the four analog conversion modes.

Parameter	Type	Description
X560Base	WORD	Base address in Short I/O
Mode	BYTE	0 - Single channel 1 - Sequential channel 2 - Random channel 3 - External channel

4.7.35 X560 Wait

Syntax:

WORD X560Wait(X560Base)

Function: This routine waits for an A/D conversion to be completed before continuing.

Parameter	Type	Description
X560Base	WORD	Base address in Short I/O

4.7.36 X560 Write

Syntax:

WORD X560Write(X560Base, Register, Byte)

Function: This routine writes 1 byte of data to a register.

Parameter	Type	Description
X560Base	WORD	Base address in Short I/O
Register	WORD	Register to write to
Byte	BYTE	Byte of data to write

4.7.37 X566 Read

Syntax:

WORD X566Read(X566Base, Register, ByteAddress)

Function: This routine reads a byte from a register on the XVME-566 board.

Parameter	Type	Description
X566Base	WORD	Base address in Short I/O
Register	WORD	Register or read from
ByteAddress	LPBYTE	Address for result

4.7.38 X566 Reset

Syntax:

WORD X566Reset(X566Base)

Function: This routine resets the sequence controller. The module is set to: continuous mode; sequential mode; VMEbus interrupts disabled; red LED off; green LED on. STC channels 2, 4, and 5 are disabled.

Parameter	Type	Description
X566Base	WORD	Base address in Short I/O

4.7.39 X566 Set Clock

Syntax:

WORD X566SetCK(X566Base, ClockNum, CtrlReg, LoadReg, HoldReg)

Function: This routine programs one of five counters by writing the control mode value into the counter's control register and by writing the desired values into the counter's load and hold registers.

Parameter	Type	Description
X566Base	WORD	Base address in Short I/O
ClockNum	BYTE	Clock channel to program (1-5)
CtrlReg	WORD	Control register value
LoadReg	WORD	Load register value
HoldReg	WORD	Hold register value

4.7.40 X566 Set Sample Clock

Syntax:

WORD X566SetSampCK(X566Base, Period)

Function: This routine programs the sample clock (STC counter 4) period which controls the A/D conversion frequency on the XVME-566 board. The period must be ≥10 microseconds for 12 bit A/D conversions and ≥7 microseconds for 8 bit conversions.

Parameter	Type	Description
X566Base	WORD	Base address in Short I/O
Period	WORD	Sampling frequency in microseconds

4.7.41 X566 Write

Syntax:

WORD X566Write(X566Base, Register, Byte)

Function: This routine writes 1 byte of data to a register on the XVME-566.

Parameter	Type	Description
X566Base	WORD	Base address in Short I/O
Register	WORD	Register to write to
Byte	BYTE	Byte of data to write

4.7.42 X566 Write Word

Syntax:

WORD X566WriteWord(X566Base, Register, Word)

Function: This routine writes to two consecutive registers on the XVME-566.

Parameter	Type	Description
X566Base	WORD	Base address in Short I/O
Register	WORD	First of two registers to write to
Word	WORD	Word of data to write

4.8 COUNTER LIBRARY

This library section provides a set of routines specific to Xycom Counter I/O VME products. They all return an error code as defined in XVME984.H and described in Section 4.5.

4.8.1 X203 Interrupt Initialize

Syntax:

WORD X203IntInit(X203Base, IntMask, IntVectors)

Function: This routine initializes the Am9519 interrupt controller as follows: The mode register is set for individual vectors, fixed priority, GINT signal active high, IREQ's active low and the chip is armed. The IntMask value will be written to the IMR and the IntVectors are written to the response memory.

Parameter	Type	Description
X203Base	WORD	Base address in Short I/O
IntMask	WORD	Interrupts enable/disable mask
IntVectors	LPBYTE	First address of eight vectors

4.8.2 X203 Read

Syntax:

WORD X203Read(X203Base, Register, ByteAddress)

Function: This routine reads a byte from a register on the XVME-203.

Parameter	Type	Description
X203Base	WORD	Base address in Short I/O
Register	WORD	Register to read from
ByteAddress	BYTE	Address for result

4.8.3 X203 Reset

Syntax:

WORD X203Reset(X203Base)

Function: This routine resets all counters on STC A and B, turns off quadrature detect circuitry, and resets the Am9519 interrupt controller.

Parameter	Type	Description
X203Base	WORD	Base address in Short I/O

4.8.4 X203 Set Clock

Syntax:

WORD X203SetCK(X203Base, ClockNum, CtrlReg, LoadReg, HoldReg)

Function: This routine programs one of 10 counters by writing the control mode value into the counter's control register and by writing the desired values into the counter's load and hold registers.

Parameter	Type	Description
X203Base	WORD	Base address in Short I/O
ClockNum	BYTE	Clock channel to program (1-10)
CtrlReg	WORD	Control register value
LoadReg	WORD	Load register value
HoldReg	WORD	Hold register value

4.8.5 X203 Write

Syntax:

WORD X203Write(X203Base, Register, Byte)

Function: This routine writes 1 byte of data to a register on the XVME-203.

Parameter	Type	Description
X203Base	WORD	Base address in Short I/O
Register	WORD	Register to write to
Byte	BYTE	Byte of data to write

4.8.6 X230 Build Command Block

Syntax:

X230BldCmdBlk(X230Base, BlkOff, Command, ILevel, IVector, NextBlk, OpSize, OpBuff)

Function: This routine builds a command block for the XVME-230 in Short I/O.

Parameter	Type	Description
X230Base	WORD	Base address in Short I/O
BlkOff	WORD	Offset from the base address for the command block
Command	BYTE	Command byte
ILevel	BYTE	VMEbus interrupt level to use
IVector	BYTE	Interrupt vector to put out
NextBlk	WORD	Address of next command block or 0xFF if none
OpSize	BYTE	Number of operands in OpBuff
OpBuff	LPBYTE	First address of up to six operands

4.8.7 X230 Build Command Block Buffer

Syntax:

X230BldCmdBlkBuf(X230Base, BlkOff, Command, ILevel, IVector, NextBlk, OpSize, BuffAddr)

Function: This routine builds a command block for the XVME-230 in Short I/O.

Parameter	Type	Description
X230Base	WORD	Base address in Short I/O
BlkOff	WORD	Offset from the base address for the command block
Command	BYTE	Command byte
ILevel	BYTE	VMEbus interrupt level to use
IVector	BYTE	Interrupt vector to put out
NextBlk	WORD	Address of next command block or 0xFF if none
OpSize	WORD	Number of operands in OpBuff
OpBuff	WORD	VMEbus address of the command data buffer

4.8.8 X230 Execute Command

Syntax:

WORD X230ExecCmd(X230Base, BlkOff, Channel)

Function: This routine executes a command block.

Parameter	Type	Description
X230Base	WORD	Base address in Short I/O
BlkOff	WORD	Offset from the base address for the command block
Channel	BYTE	Channel/timer to command

4.8.9 X230 Read

Syntax:

WORD X230Read(X230Base, Register, ByteAddress)

Function: This routine reads a byte of data from a register on the XVME-230.

Parameter	Type	Description
X230Base	WORD	Base address in Short I/O
Register	WORD	Register to read from
ByteAddress	LPBYTE	Address for result

4.8.10 X230 Write

Syntax:

WORD X230Write(X230Base, Register, Byte)

Function: This routine writes a byte of data to a register on the XVME-230.

Parameter	Type	Description
X230Base	WORD	Base address in Short I/O
Register	WORD	Register to write to
Byte	BYTE	Byte of data to write

4.9 DIGITAL LIBRARY

This library section provides a set of routines specific to Xycom Digital I/O VME products. They all return an error code as defined in XVME984.H and described in Section 4.5.

4.9.1 X200 Counter Pre-Load

Syntax:

WORD X200CPLoad(X200Base, PITimer, TimerVal)

Function: This routine writes a counter value to the counter pre-load register of the PI/T chip.

Parameter	Type	Description
X200Base	WORD	Base address in Short I/O
PITimer	BYTE	PI/timer (0 or 1)
TimerVal	DWORD	Counter value (0 - 0xFFFFFF)

4.9.2 X200 Initialize

Syntax:

WORD X200Init(X200Base)

Function: This routine initializes port C and the data direction registers on the two PI/T chip.

Parameter	Type	Description
X200Base	WORD	Base address in Short I/O

4.9.3 X200 Port A Direction

Syntax:

WORD X200PortADir(X200Base, PITimer, Direction)

Function: This routine sets the data direction for port A and the transceiver direction for the PI/T chip.

Parameter	Type	Description
X200Base	WORD	Base address in Short I/O
PITimer	BYTE	PI/timer (0 or 1)
Direction	BYTE	0 for input; 1 for output

4.9.4 X200 Port B Direction

Syntax:

WORD X200PortBDir(X200Base, PITimer, Direction)

Function: This routine sets the data direction for port B and the transceiver direction for the PI/T chip.

Parameter	Type	Description
X200Base	WORD	Base address in Short I/O
PITimer	BYTE	PI/timer (0 or 1)
Direction	BYTE	0 for input; 1 for output

4.9.5 X200 Read

Syntax:

WORD X200Read(X200Base, PITimer, Register, ByteAddress)

Function: This routine reads a register from a PI/T chip.

Parameter	Type	Description
X200Base	WORD	Base address in Short I/O
PITimer	BYTE	PI/timer (0 or 1)
Register	WORD	Register to read
ByteAddress	LPBYTE	Address for result

4.9.6 X200 Submode A

Syntax:

WORD X200SubModeA(X200Base, PITimer, SubMode)

Function: This routine sets the port A submode in the port A control register on a PI/T chip.

Parameter	Type	Description
X200Base	WORD	Base address in Short I/O
PITimer	BYTE	PI/timer (0 or 1)
SubMode	BYTE	0 for submode 0 or 0x 1 for submode 1 2 for submode 1x

4.9.7 X200 Submode B

Syntax:

WORD X200SubModeB(X200Base, PITimer, SubMode)

Function: This routine will set the port B submode in the port B control register on a PI/T chip.

Parameter	Type	Description
X200Base	WORD	Base address in Short I/O
PITimer	BYTE	PI/timer (0 or 1)
SubMode	BYTE	0 for submode 0 or 0x 1 for submode 1 2 for submode 1x

4.9.8 X200 Write

Syntax:

WORD X200Write(X200Base, PITimer, Register, Byte)

Function: This routine writes to a register in a PI/T chip.

Parameter	Type	Description
X200Base	WORD	Base address in Short I/O
PITimer	BYTE	PI/timer (0 or 1)
Register	WORD	Register to write
Byte	BYTE	Byte to write

4.9.9 X201 Counter Pre-Load

Syntax:

WORD X201CPLoad(X201Base, PITimer, TimerVal)

Function: This routine writes a counter value to the counter pre-load register of the PI/T chip.

Parameter	Type	Description
X201Base	WORD	Base address in Short I/O
PITimer	BYTE	PI/timer (0 or 1)
TimerVal	DWORD	Timer value (0 - 0xFFFFFF)

4.9.10 X201 Initialize

Syntax:

WORD X201Init(X201Base)

Function: This routine initializes the two PI/T chips to mode 0, submode 1X, and sets port C by disabling timer and port interrupts.

Parameter	Type	Description
X201Base	WORD	Base address in Short I/O

4.9.11 X201 Port Direction

Syntax:

WORD X201PortDir(X201Base, PITimer, Direction)

Function: This routine sets the data direction for ports A and B on the PI/T chip.

Parameter	Type	Description
X201Base	WORD	Base address in Short I/O
PITimer	BYTE	PI/timer (0 or 1)
Direction	BYTE	0 for input; 1 for output

4.9.12 X201 Port C Direction

Syntax:

WORD X201PortCDir(X201Base, PITimer, Direction)

Function: This routine sets the data direction for port C on the PI/T chip.

Parameter	Type	Description
X201Base	WORD	Base address in Short I/O
PITimer	BYTE	PI/timer (0 or 1)
Direction	BYTE	0 for input; 1 for output

4.9.13 X201 Read

Syntax:

WORD X201Read(X201Base, PITimer, Register, ByteAddress)

Function: This routine reads a register from a PI/T chip.

Parameter	Type	Description
X201Base	WORD	Base address in Short I/O
PITimer	BYTE	PI/timer (0 or 1)
Register	WORD	Register to read
ByteAddress	LPBYTE	Address for result

4.9.14 X201 Write

Syntax:

WORD X201Write(X201Base, PITimer, Register, Byte)

Function: This routine writes to a register in a PI/T chip.

Parameter	Type	Description
X201Base	WORD	Base address in Short I/O
PITimer	BYTE	PI/timer (0 or 1)
Register	WORD	Register to write
Byte	BYTE	Byte to write

4.9.15 X202 Initialize

Syntax:

WORD X202Init(X202Base)

Function: This routine initializes the XVME-202 board by deactivating the reset line.

Parameter	Type	Description
X202Base	WORD	Base address in Short I/O

4.9.16 X202 Read

Syntax:

WORD X202Read(X202Base, BankReg, ByteAddress)

Function: This routine reads a register from a PAMUX bank.

Parameter	Type	Description
X202Base	WORD	Base address in Short I/O
BankReg	WORD	Bank register to read
ByteAddress	LPBYTE	Address for result

4.9.17 X202 Reset

Syntax:

WORD X202Reset(X202Base)

Function: This routine asserts the reset line on the XVME-202 causing the attached PAMUX units to be reset.

Parameter	Type	Description
X202Base	WORD	Base address in Short I/O

4.9.18 X202 Write

Syntax:

WORD X202Write(X202Base, BankReg, Byte)

Function: This routine writes to a register in a PAMUX bank.

Parameter	Type	Description
X202Base	WORD	Base address in Short I/O
BankReg	WORD	Bank register to write to
Byte	BYTE	Byte to write

4.9.19 **X210 Read Channel**

Syntax:

WORD X210ReadChannel(X210Base, Channel, ByteAddress)

Function: This routine reads the state of a channel. It returns a 0 if low, or 1 if high in the result address.

Parameter	Type	Description
X210Base	WORD	Base address in Short I/O
Channel	BYTE	Channel number to read (0 - 31)
ByteAddress	LPBYTE	Address for result

4.9.20 **X210 Read Port**

Syntax:

WORD X210ReadPort(X210Base, Port, ByteAddress)

Function: This routine reads the state of a port where:
Port 0 = Channels 8 - 15 (JK-1)
Port 1 = Channels 0 - 7 (JK-1)
Port 2 = Channels 24 - 31 (JK-2)
Port 3 = Channels 16 - 23 (JK-2)

Parameter	**Type**	**Description**
X210Base	WORD	Base address in Short I/O
Port	BYTE	Port to read (0 - 3)
ByteAddress	LPBYTE	Address for result

4.9.21 X210 Read Two Ports

Syntax:

WORD X210Read2Ports(X210Base, StartPort, WordAddress)

Function: This routine reads the state of two consecutive ports where:
Port 0 = Channels 8 - 15 (JK-1)
Port 1 = Channels 0 - 7 (JK-1)
Port 2 = Channels 24 - 31 (JK-2)
Port 3 = Channels 16 - 23 (JK-2)

Parameter	**Type**	**Description**
X210Base	WORD	Base address in Short I/O
StartPort	BYTE	Starting port to read (0 - 2)
WordAddress	LPWORD	Address for result

4.9.22 X210 Read Four Ports

Syntax:

WORD X210Read4Ports(X210Base, DWordAddress)

Function: This routine reads the state of all four ports where:
Port 0 = Channels 8 - 15 (JK-1)
Port 1 = Channels 0 - 7 (JK-1)
Port 2 = Channels 24 - 31 (JK-2)
Port 3 = Channels 16 - 23 (JK-2)

Parameter	Type	Description
X210Base	WORD	Base address in Short I/O
DWordAddress	LPDWORD	Address for result

4.9.23 X212 Initialize

Syntax:

WORD X212Init(X212Base)

Function: This routine clears all the change registers, turns on the green LED and turns off the red LED.

Parameter	**Type**	**Description**
X212Base	WORD	Base address in Short I/O

4.9.24 X212 Interrupt Disable

Syntax:

WORD X212IntDisable(X212Base)

Function: This routine turns off the interrupt enable bit in the status control register.

Parameter	Type	Description
X212Base	WORD	Base address in Short I/O

4.9.25 X212 Read

Syntax:

WORD X212Read(X212Base, Register, ByteAddress)

Function: This routine reads a register from the XVME-212.

Parameter	Type	Description
X212Base	WORD	Base address in Short I/O
Register	WORD	Register to read
ByteAddress	LPBYTE	Address for result

4.9.26 X212 Read Channel

Syntax:

WORD X212ReadChannel(X212Base, Channel, RegisterSet, ByteAddress)

Function: This routine reads a channel from either the change or data register. The value returned in ByteAddress will be 0 if the channel is not set or 1 if the channel is set.

Parameter	Type	Description
X212Base	WORD	Base address in Short I/O
Channel	BYTE	Channel number to read (0 - 31)
RegisterSet	BYTE	0 - Change register 1 - Data register
ByteAddress	LPBYTE	Address for result

4.9.27 X212 Read Scan

Syntax:

WORD X212ReadScan(X212Base, CRStatusAddr, DRStatusAddr, Port)

Function: This routine reads the change and data register pair.

Parameter	Type	Description
X212Base	WORD	Base address in Short I/O
CRStatusAddr	LPBYTE	Address for change register result
DRStatusAddr	LPBYTE	Address for data register result
Port	Byte	Identifies port to read (0 - 3)

4.9.28 X212 Read Word

Syntax:

WORD X212ReadWord(X212Base, Register, WordAddress)

Function: This routine reads two consecutive registers from the XVME-212.

Parameter	Type	Description
X212Base	WORD	Base address in Short I/O
Register	WORD	Starting register to read
WordAddress	LPWORD	Address for result

4.9.29 X212 Read Word Scan

Syntax:

X212ReadWordScan(X212Base, CRStatusAddr, DRStatusAddr, Port)

Function: This routine reads the two pairs of change and data registers.

Parameter	Type	Description
X212Base	WORD	Base address in Short I/O
CRStatusAddr	LPWORD	Address for change register result
DRStatusAddr	LPWORD	Address for data register result
Port	Byte	Identifies starting port to read (0 - 2)

4.9.30 X212 Write

Syntax:

WORD X212Write(X212Base, Register, Byte)

Function: This routine writes to a register in the XVME-212.

Parameter	Type	Description
X212Base	WORD	Base address in Short I/O
Register	WORD	Register to write to
Byte	BYTE	Byte to write

4.9.31 X212 Write Word

Syntax:

WORD X212WriteWord(X212Base, Register, Word)

Function: This routine writes to two consecutive registers in the XVME-212.

Parameter	Type	Description
X212Base	WORD	Base address in Short I/O
Register	WORD	Starting register to write to
Word	WORD	Word to write

4.9.32 X220 Read

Syntax:

WORD X220Read(X220Base, Register, ByteAddress)

Function: This routine reads a register from the XVME-220.

Parameter	Type	Description
X220Base	WORD	Base address in Short I/O
Register	WORD	Register to read
ByteAddress	LPBYTE	Address for result

4.9.33 X220 Read All

Syntax:

WORD X220ReadAll(X220Base, DWordAddress)

Function: This routine reads the status of the four output ports.

Parameter	Type	Description
X220Base	WORD	Base address in Short I/O
DWordAddress	LPDWORD	Address for result

4.9.34 X220 Read Channel

Syntax:

WORD X220ReadChannel(X220Base, Channel, ByteAddress)

Function: This routine reads a channel from the XVME-220. The value returned in ByteAddress will be 0 if the channel is not set or 1 if the channel is set.

Parameter	**Type**	**Description**
X220Base	WORD	Base address in Short I/O
Channel	BYTE	Channel number to read (0 - 31)
ByteAddress	LPBYTE	Address for result

4.9.35 X220 Read Word

Syntax:

WORD X220ReadWord(X220Base, Register, WordAddress)

Function: This routine reads two consecutive registers from the XVME-220.

Parameter	**Type**	**Description**
X220Base	WORD	Base address in Short I/O
Register	WORD	Starting register to read
WordAddress	LPWORD	Address for result

4.9.36 X220 Reset

Syntax:

WORD X220Reset(X220Base)

Function: This routine sets all the XVME-220 output port configuration registers to 0.

Parameter	Type	Description
X220Base	WORD	Base address in Short I/O

4.9.37 X220 Write

Syntax:

WORD X220Write(X220Base, Register, Byte)

Function: This routine writes to a register in the XVME-220.

Parameter	Type	Description
X220Base	WORD	Base address in Short I/O
Register	WORD	Register to write to
Byte	BYTE	Byte to write

4.9.38 X220 Write All

Syntax:

WORD X220WriteAll(X220Base, DWord)

Function: This routine configures all four output ports.

Parameter	Type	Description
X220Base	WORD	Base address in Short I/O
DWord	DWORD	Port configuration data

4.9.39 X220 Write Channel

Syntax:

WORD X220WriteChannel(X220Base, Channel, Byte)

Function: This routine sets a channel in the XVME-220.

Parameter	Type	Description
X220Base	WORD	Base address in Short I/O
Channel	BYTE	Channel number to write (0 - 31)
Byte	BYTE	State to set the channel to (0 or 1)

4.9.40 X220 Write Word

Syntax:

WORD X220WriteWord(X220Base, Register, Word)

Function: This routine writes to two consecutive registers in the XVME-220.

Parameter	Type	Description
X220Base	WORD	Base address in Short I/O
Register	WORD	Starting register to write to
Word	WORD	Word to write

4.9.41 X240 Read

Syntax:

WORD X240Read(X240Base, Register, ByteAddress)

Function: This routine reads a register from the XVME-240.

Parameter	Type	Description
X240Base	WORD	Base address in Short I/O
Register	WORD	Register to read
ByteAddress	LPBYTE	Address for result

4.9.42 **X240 Read Word**

Syntax:

WORD X240ReadWord(X240Base, Register, WordAddress)

Function: This routine reads two consecutive registers from the XVME-240.

Parameter	Type	Description
X240Base	WORD	Base address in Short I/O
Register	WORD	Starting register to read
WordAddress	LPWORD	Address for result

4.9.43 **X240 Reset**

Syntax:

WORD X240Reset(X240Base)

Function: This routine resets the XVME-240 by setting the interrupt mask to 0 and output flags to low, clearing the interrupt latches, and setting all ports as inputs.

Parameter	Type	Description
X240Base	WORD	Base address in Short I/O

4.9.44 **X240 Write**

Syntax:

WORD X240Write(X240Base, Register, Byte)

Function: This routine writes to a register in the XVME-240.

Parameter	Type	Description
X240Base	WORD	Base address in Short I/O
Register	WORD	Register to write to
Byte	BYTE	Byte to write

4.9.45 X244 Read

Syntax:

WORD X244Read(X244Base, Register, Type, ByteAddress)

Function: This routine reads a register from the XVME-244.

Parameter	Type	Description
X244Base	WORD	Base address in Short I/O
Register	WORD	Register to read (0 - 3)
Type	WORD	STATUS_____REG, OUTPUT_____REG, FILTERED_REG or INPUT_REG
ByteAddress	LPBYTE	Address for result

4.9.46 X244 Read All

Syntax:

WORD X244ReadAll(X244Base, Type, DWordAddress)

Function: This routine reads the status of four registers.

Parameter	Type	Description
X244Base	WORD	Base address in Short I/O
Type	WORD	OUTPUT___REG, FILTERED___REG, or INPUT_REG
DWordAddress	LPDWORD	Address for result

4.9.47 X244 Read Channel

Syntax:

WORD X244ReadChannel(X244Base, Channel, Type, ByteAddress)

Function: This routine reads a channel from the XVME-244. The value returned in ByteAddress will be 0 if the channel is not set or 1 if the channel is set.

Parameter	Type	Description
X244Base	WORD	Base address in Short I/O
Channel	BYTE	Channel number to read (0 - 31)
Type	WORD	OUTPUT___REG, FILTERED___REG, or INPUT_REG
ByteAddress	LPBYTE	Address for result

4.9.48 X244 Read Word

Syntax:

WORD X244ReadWord(X244Base, StartReg, Type, WordAddress)

Function: This routine reads two consecutive registers from the XVME-244.

Parameter	Type	Description
X244Base	WORD	Base address in Short I/O
StartReg	WORD	Starting register to read (0 or 2)
Type	WORD	OUTPUT_REG, FILTERED_REG, or INPUT_REG
WordAddress	LPWORD	Address for result

4.9.49 X244 Reset

Syntax:

WORD X244Reset(X244Base)

Function: This routine sets all the XVME-244 output port configuration registers to 0.

Parameter	Type	Description
X244Base	WORD	Base address in Short I/O

4.9.50 X244 Write

Syntax:

WORD X244Write(X244Base, Register, Type, Byte)

Function: This routine writes to a register in the XVME-244.

Parameter	Type	Description
X244Base	WORD	Base address in Short I/O
Register	WORD	Register to write to (0 - 3)
Type	WORD	STATUS_REG or OUTPUT_REG
Byte	BYTE	Byte to write

4.9.51 X244 Write All

Syntax:

WORD X244WriteAll(X244Base, DWord)

Function: This routine configures all four output registers.

Parameter	Type	Description
X244Base	WORD	Base address in Short I/O
DWord	DWORD	Port configuration data

4.9.52 X244 Write Channel

Syntax:

WORD X244WriteChannel(X244Base, Channel, Byte)

Function: This routine sets an output channel in the XVME-244.

Parameter	Type	Description
X244Base	WORD	Base address in Short I/O
Channel	BYTE	Channel number to write (0 - 31)
Byte	BYTE	State to set the channel to (0 or 1)

4.9.53 X244 Write Word

Syntax:

WORD X244WriteWord(X244Base, StartReg, Word)

Function: This routine writes to two consecutive output registers in the XVME-244.

Parameter	Type	Description
X244Base	WORD	Base address in Short I/O
StartReg	WORD	Starting register to write to (0 or 2)
Word	WORD	Word to write

4.9.54 X260 Read

Syntax:

WORD X260Read(X260Base, Register, ByteAddress)

Function: This routine reads a register from the XVME-260.

Parameter	**Type**	**Description**
X260Base	WORD	Base address in Short I/O
Register	WORD	Register to read
ByteAddress	LPBYTE	Address for result

4.9.55 X260 Read All

Syntax:

WORD X260ReadAll(X260Base, DWordAddress)

Function: This routine reads the status of the four output ports.

Parameter	Type	Description
X260Base	WORD	Base address in Short I/O
DWordAddress	LPDWORD	Address for result

4.9.56 X260 Read Channel

Syntax:

WORD X260ReadChannel(X260Base, Channel, ByteAddress)

Function: This routine reads a channel from the XVME-260. The value returned in ByteAddress will be 0 if the channel is not set or 1 if the channel is set.

Parameter	**Type**	**Description**
X260Base	WORD	Base address in Short I/O
Channel	BYTE	Channel number to read (0 - 31)
ByteAddress	LPBYTE	Address for result

4.9.57 X260 Read Word

Syntax:

WORD X260ReadWord(X260Base, Register, WordAddress)

Function: This routine reads two consecutive registers from the XVME-260.

Parameter	Type	Description
X260Base	WORD	Base address in Short I/O
Register	WORD	Starting register to read
WordAddress	LPWORD	Address for result

4.9.58 X260 Write

Syntax:

WORD X260Write(X260Base, Register, Byte)

Function: This routine writes to a register in the XVME-260.

Parameter	Type	Description
X260Base	WORD	Base address in Short I/O
Register	WORD	Register to write to
Byte	BYTE	Byte to write

4.9.59 X260 Write All

Syntax:

WORD X260WriteAll(X260Base, DWord)

Function: This routine configures all four output ports.

Parameter	Type	Description
X260Base	WORD	Base address in Short I/O
DWord	DWORD	Port configuration data

4.9.60 X260 Write Channel

Syntax:

WORD X260WriteChannel(X260Base, Channel, Byte)

Function: This routine sets a channel in the XVME-260.

Parameter	Type	Description
X260Base	WORD	Base address in Short I/O
Channel	BYTE	Channel number to write (0 - 31)
Byte	BYTE	State to set the channel to (0 or 1)

4.9.61 X260 Write Word

Syntax:

WORD X260WriteWord(X260Base, Register, Word)

Function: This routine writes to two consecutive registers in the XVME-260.

Parameter	Type	Description
X260Base	WORD	Base address in Short I/O
Register	WORD	Starting register to write to
Word	WORD	Word to write

E

G