ARTISAN®
TECHNOLOGY GROUP

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

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

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

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

Custom engineering
so your equipment works exactly as you specify.

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

Expert team | Trust guarantee | 100% satisfaction

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

Find the *Tektronix VX5530* at our website: ***Click HERE***



VX5530/VX5535
VXIbus System Controller
BASIC Programmers Manual

070-7438-00 (Book 2 of 3)
First Printing JAN 1990

Tektronix®
COMMITTED TO EXCELLENCE

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Copyright © 1990 Tektronix, Inc. All rights reserved. Contents of this publication may not be reproduced in any form without the written permission of Tektronix, Inc.

Products of Tektronix, Inc. and its subsidiaries are covered by U.S. and foreign patents and/or pending patents.

Printed in the U.S.A. Specification and price change privileges are reserved.

Tektronix, Inc.
VXIbus Products
Jack Murdock Park
P.O. Box 3500
Vancouver, WA 98607-3500

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table of Contents

Section 1 General Information 1-1

Introduction 1-1

Notational Conventions 1-3

Section 2 TekVICS Programming Interface 2-1

Calling TekVICS From Microsoft® BASIC 2-1

Error Codes 2-2

Section 3 TekVICS Message Delivery System 3-1

Device Naming and Addressing 3-2

Program Interface 3-3

Alphabetical Command Listing 3-3

Section 4 TekVICS BusManager 4-1

BusManager Context 4-1

Driver Considerations 4-2

Command Functions 4-3

Interrupt and Error Handling Functions 4-4

Commander Message Passing Functions 4-4

Event/Response Functions 4-5

Servant Message Passing Functions 4-5

Version Information Functions 4-5

Alphabetical Command Listing 4-6

Description: 4-46

Section 5 TekVICS Online Resource Manager 5-1

Introduction 5-1

Calling the OLRM From MS BASIC 5-2

Alphabetical Command Listing 5-3

Section 6 TekVICS Advanced Trigger Functions 6-1

Introduction 6-1

ATF Functions 6-2

Alphabetical Function Listing 6-3

Advanced Trigger Functions Examples 6-28

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix A I/O Control Commands A-1
GPIBRDG Control Commands A-1
VXIbus Bridge Control Commands A-8
VXIbus Word Serial Protocol A-8
Shared-Memory Protocol A-9

Appendix B Writing BusManager Applications B-1
Introduction B-1
Using the BusManager Interface B-2
The Bus Window B-2
Interrupts B-3
Polling Interrupts B-3

Appendix C BusManager QuickBASIC Applications C-1
Using the Microsoft® QuickBASIC TekVICS Interface C-1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

List of Illustrations

Figure 1-1. VX5530/VX5535 software overview. 1-2

List of Tables

Table 2-1 Error Codes 2-2
TekVICS BusManager Functional Listing 4-3
Interrupts 4-4
Errors 4-4
ATF Functional Listing 6-2
Table A-1 GPIBRDG Control Commands A-2
Table A-2 Status Word Layout A-6
Table A-3 GPIB Error Codes A-7
Table A-4 GPIB Timeout Codes A-7
Table A-5 VXIbus Word Serial Control Commands A-8
Table A-6 VXIbus Shared-Memory Protocol Commands A-9

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Section 1
General Information

Introduction

This manual is written for BASIC-language programmers using the TekVICS programming interface on a VX5530/VX5535. The reader is expected to have read the VX5530/VX5535 Users Manual for a global understanding of what is in TekVICS, how it is installed, and how to use BusProbe, a program that is very helpful in learning about the system. The reader is expected to have a basic knowledge of DOS.

The TekVICS programming environment provides programmers access to the TekVICS Message Manager, On Line Resource Manager, and Advanced Trigger Functions (ATF).

The TekVICS Message Manager is a software package which coordinates all communication to instruments in the system. Currently, this includes both VXIbus and GPIB instruments. There are five parts to the Message Manager:

1) **Programming Interface Libraries.** The programming interface libraries are for the C and BASIC programming languages. In the BASIC version, there are two libraries, one for Microsoft® BASIC and one for QuickBASIC.

   The programming interface libraries provide access to the Message Delivery System and the BusManager.

2) **Message Delivery System (MDS).** This is the transport mechanism for application programs to communicate with intelligent instruments. Most programmers use the MDS facilities only.

3) **BusManager**. This is a lower-level facility for communicating over the VXIbus. It manages a bus window by which programs can directly read and write locations on the VXIbus. It also manages VXIbus interrupts and signals. Programmers should use the BusManager only for communicating with VXIbus register-based devices or VMEbus devices.

4) **GPIB Device Driver.** The GPIB device driver is a low-level driver controlling the GPIB hardware on the VX5530/VX5535. Programmers should never have to directly access this driver.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5) **Communication Bridges**. Bridges interface the Message Delivery System to the various interface busses and communication protocols supported by the VX5530/VX5535. There are three bridges. These are the *VXIbus Bridge* for communicating with VXIbus message-based devices using Word Serial or Shared Memory protocol, the *GPIB Bridge* for communicating with GPIB devices, and the *Command Bridge*. The *Command Bridge* represents local capabilities of the VX5530/VX5535 and does not communicate with any physical instruments. All bridges are implemented as terminate and stay resident (TSR) programs.

The Online Resource Manager (OLRM) provides the capability to manage VXIbus system resources during normal operation.

The Advanced Trigger Functions (ATF) provide advanced trigger, signal and timing features within a VXIbus mainframe, as well as the necessary real time links between multiple VXIbus subsystems.

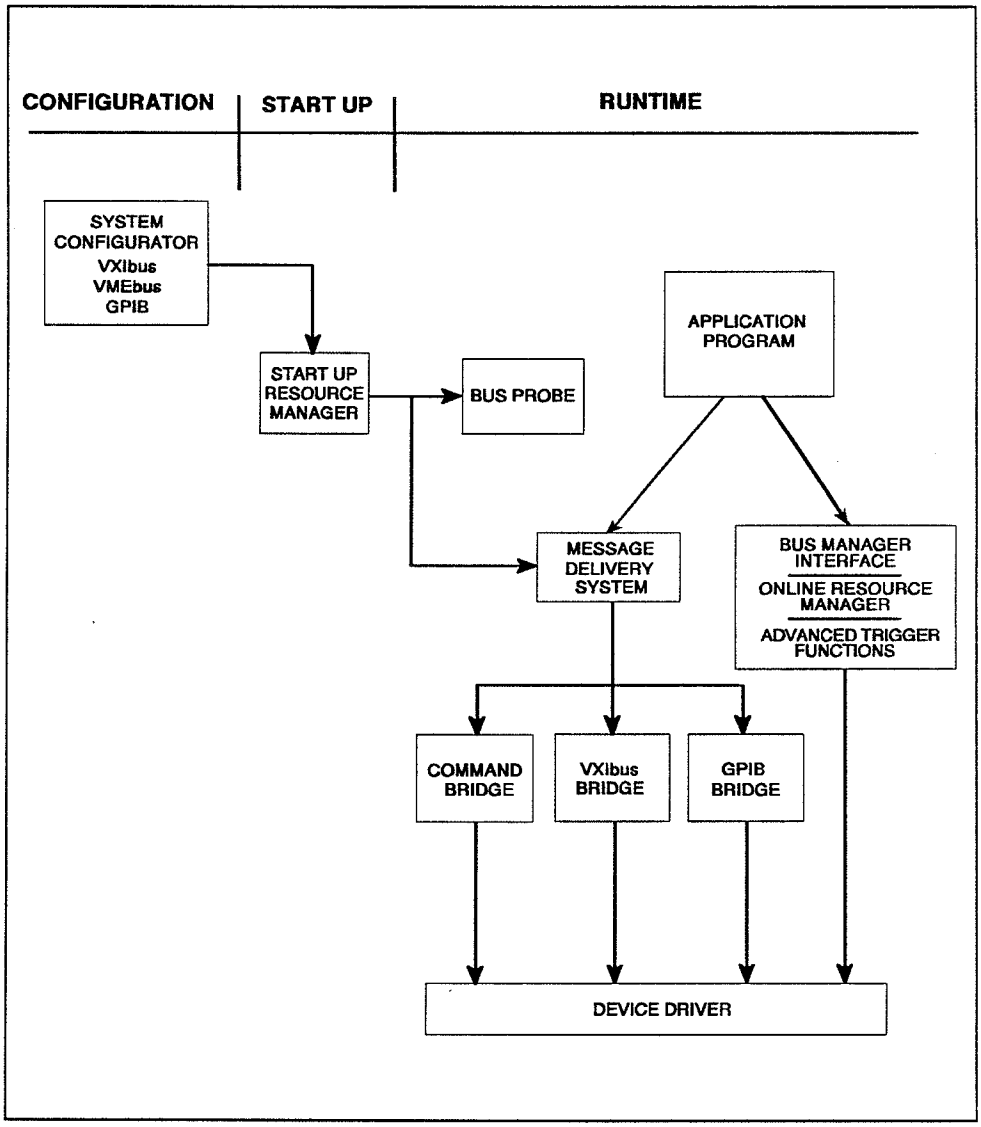


Figure 1-1. VX5530/VX5535 software overview.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Notational Conventions

The following notational conventions are used throughout this manual.

Convention	Description
Filename	The typeface shown at left is used in the text to refer to a DOS file name
`Example`	This typeface is used to simulate the output as seen on the VX5530/VX5535 screen. It is used for examples of commands, the output from commands, and code.
`[optional]`	The square brackets indicate optional fields.
<placeholder>	Words in angle brackets are placeholders for fields that you need to supply.
`(parameter)`	In the assembly language model discussion, words shown in parentheses in this typeface indicate the BASIC language parameter for that assembly function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Section 2
TekVICS Programming Interface

Each program-level function in TekVICS has a number of interfaces. Most have at least one interface for each of the supported applications languages, Microsoft® C and Microsoft® BASIC. For ease of programming, many functions have multiple interfaces.

The BASIC language has its own interface library and corresponding definition file:

Library	Definition file
\tekvics\lib\vcmsb.lib \tekvics\lib\vcmsb.qlb	\tekvics\include\vcmsb.bi \tekvics\include\gpibrdg.bi \tekvics\include\vxicmd.bi \tekvics\include\bdgioctl.bi \tekvics\include\busmgr.bi \tekvics\include\atf.bi

The use of each of these files is discussed more fully below.

Calling TekVICS From Microsoft® BASIC

The Microsoft® BASIC interface consists of the standalone library, **\tekvics\lib\vcmsb.lib** for creating compiled BASIC applications, and the BASIC include files **\tekvics\include\vcmsb.bi, gpibrdg.bi, vxicmd.bi, bdgioctl.bi,** and **busmgr.bi.** All Bus Interface Manager and OLRM calls are accessible through this interface.

The **vcmsb.bi** include file contains the prototypes for the interface functions and subroutines, enabling BASIC to perform typechecking and near to far data conversion.

The definition files **gpibrdg.bi** and **vxicmd.bi** contain I/O control commands and define files for the GPIB and VXIbus bridges, respectively. I/O commands common to both the GPIB and VXIbus bridges are found in **bdgioctl.bi**. The file **busmgr.bi** contains constant definitions for BusManager functions. The file **atf.bi** contains function prototypes and constant definitions for Advanced Trigger Functions.

The QuickBASIC quick library **\tekvics\lib\vcmsb.qlb** is provided for use with the interactive QuickBASIC programming environment. The include files are used in this context as well.

NOTE

Message and devicepath strings are limited to 512 characters in the MS BASIC interface.

See Appendix B, *Writing BusManager Applications*, for more information.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Error Codes

Functions return a success (VCSUCCESS) or failure indication. This status is typically returned as the value of the function. A negative value indicates an error, as shown in Table 2-1. These error codes are defined in the definition file, **vcmsb.bi.**

Table 2-1 Error Codes

Number	Error	Description
-1	ERRFAIL	Undesignated error.
-2	ERRTIMEOUT	A device failed to respond to a command (e.g., Byte Available command while sending data) within a period of time.
-3	ERRBERR	A VXIbus error occurred while trying to access a device.
-4	ERRWS	A word serial protocol error has been detected.
-5	ERRRTIMEOUT	A device failed to generate a reply within a period of time.
-6	ERRRBERR	After successfully sending data to a device, a VXIbus error occurred while trying to access a device to get a reply.
-20	ERRBUFFERFULL	The read buffer is insufficient to hold the incoming data.
-21	ERRDUPLICATENAME	Attempted to add a duplicate name to the router table.
-23	ERRINVALIDMESSAGETYPE	Call was made with an illegal message type.
-24	ERRMEDIATABLEFULL	No room exists in the media table to store the entry.
-25	ERRROUTERTABLEFULL	No room exists in the router table to store the entry.
-26	ERRILLEGALNAME	Call was made with an unrecognized device name. (The name either has invalid syntax or it is not in the routing table.)
-27	ERRUNSUPPORTEDMEDIA	Call was made with an unknown media type.
-28	ERRILLEGALOPERATION	MDS was called with an illegal operation code.
-29	ERRILLEGALCOMMAND	An illegal command was received by the command bridge.
-100	ERRPARA	Parameter error.
-101	ERRFOPEN	File cannot be opened for access.
-102	ERRFCLOSE	Error when closing the file.
-103	ERRFERROR	File read/write error.
-105	ERRMEMORY	Memory allocation error.
-200	ERRHW	Advanced Trigger Functions—hardware not available.
-201	ERRRSNA	Advanced Trigger Functions—routing resource not available.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Section 3
TekVICS Message Delivery System

The Message Delivery System handles the delivery of commands, data write, and data read requests from a client on the VX5530/VX5535 to various server devices. It also handles the delivery of events/responses from server devices to a client on the VX5530/VX5535. A client may be:

- An application running in the VX5530/VX5535 (the most usual case)
- A GPIB gateway, translating requests from GPIB to VXIbus
- A network gateway, unwrapping requests from the network to VXIbus

The initial release of MDS supports only application programs as clients.

A server device may be:

- A VXIbus instrument
- A GPIB instrument
- An embedded servant, which is a local driver acting as a VXIbus device
- A network driver, wrapping VXIbus requests and sending them out the network

VXIbus and GPIB instrument support and provisions for writing embedded servants are provided with the initial release of MDS.

There are four types of information transferred

- Data Writes. These are bounded-size chunks of information that are transmitted from the client to the server. These operations are analogous to a write to a file.
- Data Reads. These are bounded-size chunks of information that are transmitted from the server to the client at the request of the client. The message delivery system does no interpretation of any of the transmitted messages.
- Commands. These are I/O control commands or communication protocol commands (e.g., word serial command in VXIbus or setup EOS in GPIB).
- Event/Response. These are 16-bit quantities that are transmitted asynchronously from the server to the client. An event/response is analogous to an interrupt. **This is the only information flow that is initiated by the server.**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Device Naming and Addressing

Devices are named by the Configurator and Start Up Resource Manager and comply with the device names defined in Section 7.6 of the IEEE 488.1 specification. As defined, device names are ASCII strings where each character in the string is in the set {0x41-0x5a ('A'-'Z'), 0x61-0x7a ('a'-'z'), 0x30-0x39 ('0'-'9'), 0x5f ('_')}. Names must start with a letter, and must not end with an underscore. The maximum device name length is 12 characters.

Messages are addressed to a device by concatenating a *device path* and the message together and sending it. A device path is defined as a list of device names separated by a colon. An optional colon may precede the first device name. This general form of addressing devices is known as *embedded addressing.*

For example, to send message "CLEAR" to device "dvm1" the appropriate line of MS BASIC code would be:

```
ok%  =  mpwrtmsg%("dvm1","CLEAR"  +  CHR$(10))
```

Note that the string length is not needed and that a newline (decimal 10) must be added for 488.2 devices.

The MDS looks only at the first name of the device path to route a message. This name is stripped off before the message is sent. If the destination device of the message is also a VXIbus commander with servants, the rest of the message may contain a further device path. For example, sending the message ":dvm1:scan: CLEAR" would cause the message ":scan:CLEAR" to be sent to device "dvm1", which would in turn send the message "CLEAR" to device "scan".

Just as messages are sent to a device by name, data from that device is read by referencing that device's name. For instance, to read data from "arb", the appropriate line of MS BASIC code would be:

```
OK%  =  mprdmsg%("arb",  buf$,  len&)
```

The library interface creates and returns the string `buf$`. The size of the received message is returned in `len`.

When the MDS determines which device is being addressed, it calls the appropriate *bridge* for that device. A bridge can be thought of as a sort of device driver—it makes sure that the information gets output to (or input from) the specified device.

One important bridge is the *command bridge*. The command bridge is called when there is no address specified in the message. Its purpose is to handle commander-wide issues, such as returning the names of all the devices and their attributes.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Program Interface

Access to the MDS by application programs is through a collection of library routines. These routines allow applications to establish routes and communicate with the various devices in the system. All error values returned by the library routines are defined in the **vcmsb.bi** header file.

Alphabetical Command Listing

Commands listed below with a brief purpose statement are described in detail, in alphabetical order, on the following pages.

mpcheck	Checks whether MDS is loaded
mpctrl	I/O control call for the device
mpctrlnr	I/O control call for the device with no response
mpdtof	Reads data from device and writes to file
mpftod	Reads data from file and writes to device
mppollevent	Polls event from the event queue
mprdmsg	Receives message from device
mpsendsig	Sends a signal/event to device
mpwrtmsg	Sends message to device

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

mpcheck

mpcheck

Purpose: Checks whether MDS is loaded.

Syntax:

```
DECLARE FUNCTION mpcheck%()
ok% = mpcheck%
```

Description: This function determines whether or not MDS is loaded.

Return Value: Returns `VCSUCCESS` if MDS is loaded.

Errors: Returns `ERRFAIL` if MDS is not loaded.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

mpctrl
mpctrlnr

mpctrl
mpctrlnr

Purpose: Performs an I/O control (ioctl) call for the device.

Syntax:

```
DECLARE FUNCTION mpctrl%(devicename$, BYVAL request&,
                         argp%(), SEG result&)
ok% = mpctrl%(devicename$, request&, argp%(), result&)

DECLARE FUNCTION mpctrlnr%(devicename$, BYVAL request&,
                           argp%())
ok% = mpctrlnr%(devicename$, request&, argp%())
```

Where:

`devicename`	device name
`request`	type of ioctl call
`argp`	argument of the ioctl call
`result`	result of the ioctl call

Description: Performs an I/O control call for the destination device. The destination device is specified by `devicename`. For each type of control specified by the `request` field, ioctl arguments are provided by `argp`. The `argp` may be a structure type which may differ for different requests. See Appendix A for more information on I/O control calls.

The `mpctrlnr` function is the same as `mpctrl` except that it doesn't expect a response from the device.

Return Value: Returns `VCSUCCESS` if successful. Otherwise a negative value is returned, as noted below.

Errors:

`ERRILLEGALNAME`	invalid device name
`ERRFAIL`	other error
`ERRTIMEOUT`	timeout
`ERRBERR`	bus error
`ERRWS`	word serial protocol error
`ERRRTIMEOUT`	timeout on receive
`ERRRBERR`	bus error on receive

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

mpdtof mpdtof

Purpose: Reads data from device and writes to file.

Syntax:

```
DECLARE FUNCTION mpdtof%(devicename$, filename$, maxsize&,
                         SEG rsize&, BYVAL access%)
ok% = mpdtof%(devicename$, filename$, maxsize&, rsize&,
                         access%)
```

Where:

`devicename`	device name
`filename`	file to be written to
`maxsize`	maximum number of bytes to read
`rsize`	number of bytes read
`access`	file access code

Description: Reads data from device and writes to a file. If the file does not exist, it will be created. If the file exists, it will be overwritten if the access code is FDEFAULT. If the file exists and the access code FAPPEND is specified, then the data will be appended to the current file.

Return Value: Returns `VCSUCCESS` if successful. Otherwise a negative value is returned, as noted below.

Errors:

`ERRILLEGALNAME`	invalid device name
`ERRTIMEOUT`	device timeout
`ERRBERR`	bus error
`ERRWS`	word serial protocol error
`ERRFOPEN`	file cannot be opened for access
`ERRFCLOSE`	error when closing the file
`ERRPARA`	parameter error
`ERRRTIMEOUT`	timeout on receive
`ERRRBERR`	bus error on receive

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

mpftod

mpftod

Purpose: Reads data from file and writes to device.

Syntax:

```
DECLARE FUNCTION mpftod%(devicename$, filename$)
ok% = mpftod%(devicename$, filename$)
```

Where:

`devicename`	device name
`filename`	file to be read from

Description: Reads data from file and writes to the device. The file should be able to be opened for a read operation. The file is opened in binary mode.

Return Value: Returns `VCSUCCESS` if successful. Otherwise a negative value is returned, as noted below.

Errors:

`ERRILLEGALNAME`	invalid device name
`ERRTIMEOUT`	device timeout
`ERRFOPEN`	file cannot be opened for access
`ERRBERR`	bus error
`ERRWS`	word serial protocol error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

mppollevent mppollevent

Purpose: Polls event from the event queue.

Syntax:

```
DECLARE FUNCTION mppollevent%(devicename$, SEG eventID%)
ok% = mppollevent%(devicename$, eventID%)
```

Where:

`devicename`	device name
`eventID`	event ID

Description: This function polls the event queue to determine whether an event has been received. The device name is the name assigned to a device.

Since interrupt handlers cannot be assigned in BASIC, polling is a way of getting an event. Once there is an event, it will be put into the event queue.

Return Value: Returns a value greater than 0 if an event is dequeued from the event queue. If the event queue is empty or there is no event for the device, 0 is returned. Otherwise, a negative error value is returned, as noted below.

Errors:

`ERRILLEGALNAME`	invalid device name
`ERRFAIL`	undesignated error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

mprdmsg mprdmsg

Purpose: Receives message from device.

Syntax:

```
DECLARE FUNCTION mprdmsg%(devicename$, str$, BYVAL length&)
ok% = mprdmsg%(devicename$, str$, length&)
```

Where:

`devicename`	device name
`str`	received message
`length`	message buffer size

Description: Receives message from the destination device. The destination device is specified by the device name. It will read the data until it receives the end of message character/signal (depending on communication protocol), or until the number of bytes received equals the `length` specified.

The user can check whether a complete message has been received by checking the return value of this function. If a multiple read is required to read a complete message, then the user has to manage the receive buffers. The minimum receiving buffer size is 1.

Return Value: Returns `VCSUCCESS` if a complete message has been received. If the return value is less than 0, an error has occurred. If the return value is `ERRBUFFERFULL`, the receiving buffer is full and the end of message character/signal was not received.

Errors:

`ERRILLEGALNAME`	invalid device name
`ERRBUFFERFULL`	receiver buffer full
`ERRTIMEOUT`	device timeout
`ERRPARA`	parameter error
`ERRBERR`	bus error
`ERRWS`	word serial protocol error
`ERRRTIMEOUT`	timeout on receive
`ERRRBERR`	bus error on receive

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

mpsendsig

mpsendsig

Purpose: Sends a signal/event to a device.

Syntax:

```
DECLARE FUNCTION mpsendsig%(devicename$, BYVAL eventID&)
ok% = mpsendsig%(devicename$, eventID&)
```

Where:

`devicename`	device name
`eventID`	event ID

Description: Sends a signal/event to a device. If the device is a VXIbus instrument, the event ID value is written to the signal register on the device. If the device is not a VXIbus instrument, an error is flagged.

Return Value: Returns `VCSUCCESS` if successful. Otherwise a negative value is returned, as noted below.

Errors:

`ERRILLEGALNAME`	invalid device name
`ERRFAIL`	other error
`ERRBERR`	bus error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

mpwrtmsg mpwrtmsg

Purpose: Sends message to device.

Syntax:

```
DECLARE FUNCTION mpwrtmsg%(devicename$, str$)
ok% = mpwrtmsg%(devicename$, str$)
```

Where:

`devicename`	device name
`str`	message to send

Description: Sends message to the destination device. The destination device is specified by the device name.

Messages with NULL names are routed to the Command Bridge. A subsequent call to `mprdmsg` with a NULL name retrieves the data.

Return Value: Returns `VCSUCCESS` if successful. Otherwise a negative value is returned, as noted below.

Errors:

`ERRILLEGALNAME`	invalid device name
`ERRTIMEOUT`	device timeout
`ERRBERR`	bus error
`ERRWS`	word serial protocol error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Section 4
TekVICS BusManager

The BusManager is the part of the Bus Interface Manager that manages the proprietary VXIbus interface hardware.

It is intended that **all** bus activity through the bus window be coordinated by BusManager rather than going to the underlying hardware, so that

- Software is easier to write
- Drivers for various devices on the bus won't interfere with one another
- Compatibility is maintained with different implementations of the bus interface hardware

The BusManager provides facilities for reading and writing locations on the bus, catching or causing interrupts on the bus, and transferring of data. It is accessible from either user or interrupt level. The MDS itself uses the BusManager.

BusManager Context

There are three parts to the context of the BusManager:

1) Vector of interrupt routines assigned to bus interrupts. This is fairly static.

2) Bus access **mode** which encompasses address space (A16, A24, A32), and byte sex. This is dynamic.

3) The **base** of the bus mapping window—at what bus addresses it is looking. This is dynamic.

The primary means of accessing a device is via its control and status registers. These registers are addressable locations, usually in VMEbus A16 space. They are accessed through the VMEbus window. The VMEbus window is a 64KB region of memory within the first megabyte of memory on the VX5530/VX5535 (and therefore easily accessible) which can be mapped to arbitrary sections of the A16, A24, or A32 address spaces on 64KB boundaries. The bus window is only a *master* to the VMEbus—it has no slave address itself and cannot be used as the destination of accesses originated on other boards. This means, for instance, that you can't have a VMEbus device DMA into the bus window.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The **mode** and **base** together form the **state** of the BusManager. This state is part of the context of the current process, just like the contents of the processor registers are part of the context of the current process. The BusManager handles the saving and restore of this state for VXIbus interrupts; however, if a VMEbus interrupt handler changes the BusManager state (*i.e.*, to access a device on the VXIbus), it must save the previous state before and restore it after. Similarly, in multitasking environments like Windows/386 or DESQview the program must save the state before potential context switches (*e.g.*, waiting for input) and restore it afterwards.

An exception to the rule about saving and restoring state is the case where only A16 space is being used and the bus access mode isn't changed. In this case the BusManager state is never changed since the VXIbus mapping window can encompass all of A16 space without changing the mapping.

Driver Considerations

To speed state changes that access the VXIbus, it is suggested that during initialization the driver set up the appropriate bus mapping and then call save state (`vcsavestate`) to make an internal format descriptor for the mapping. A call to restore state (`vcreststate`), passing that internal format descriptor will quickly set the mapping required by the driver.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command Functions

Following is a functional listing of TekVICS BusManager commands, and a general discussion of selected functions. Detailed descriptions of commands follow later in this section.

Functional Listing

VXIbus and Internal Control	VXIbus and Internal Information
vcvmectrl( ) vcvxictrl( )	vcbiosver( ) vcbmver( ) vcckbm( ) vcgetslavebase( ) vcgetula( ) vchwver( )
Event/Response Handling	**Error Handling**
vcerredir( ) vcerunredir( )	vcdiserr( ) vcenerr( ) vcwatchdog( )
VMEbus Data Transfer	**Interrupt Handling**
vcfromvme( ) vcsetaccmode( ) vcmapbus( ) vcreststate( ) vcsavestate( ) vcgetaccmode( ) vctovme( )	vcckintr( ) vcdisintr( ) vcenintr( ) vcsigintr( ) vcwaitintr( ) vcwaitintr2( )
Commander Word Serial Protocol	**Slave Word Serial Protocol**
vcelwscmd( ) vclwscmd( ) vclwscmdnr( ) vcwscmd( ) vcwscmdnr( ) vcwsrcvstr( ) vcwssndstr( ) vcwsstat( )	vcerservintr( ) vcerservsig( ) vcwsservarm( ) vcwsservpeek( ) vcwsservpeek2( ) vcwsservrcv( ) vcwsservrcv2( ) vcwsservsend( ) vcwsservsend2( )

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Interrupt and Error Handling Functions

It is not possible to set interrupt or error handlers in BASIC. Instead, use polling for interrupts.

Interrupts

Code	Name	Indicates
0	Message	VXIbus word serial servant response received
1	VXIbus interrupt 1	Interrupt asserted by VXIbus device
2	VXIbus interrupt 2	Interrupt asserted by VXIbus device
3	VXIbus interrupt 3	Interrupt asserted by VXIbus device
4	VXIbus interrupt 4	Interrupt asserted by VXIbus device
5	VXIbus interrupt 5	Interrupt asserted by VXIbus device
6	VXIbus interrupt 6	Interrupt asserted by VXIbus device
7	VXIbus interrupt 7	Interrupt asserted by VXIbus device
8*	Event/Response	Event or response has been received in the signal register or via redirected interrupts

*The Event/Response interrupt is used by MDS.

Errors

Code	Name	Indicates
0	SYSFAIL	VXIbus failure signal
1	BERR	VXIbus transaction error signal. Usually indicates an access to non-existent memory
2	ACFAIL	VXIbus signal from the power supply indicating that shutdown is imminent
3	Watchdog	Watchdog timer has elapsed

Commander Message Passing Functions

The VXIbus message functions are:

- Word Serial Protocol
- Longword Serial Protocol
- Extended Longword Serial Protocol
- Error Handling for the Serial Protocols utilizing the ERR* bit in the Bus Response Register (BRE)
- Signals and Interrupts

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Event/Response Functions

VXIbus *events* and *responses* (collectively called *ER's*) get special handling. They arrive through the signal register or as the Status/ID returned in response to an interrupt acknowledge for a VXIbus interrupt. All ER's are queued together, in order, as the sequence of responses and events may be important.

When a value is placed in the signal register, the signal FIFO is emptied into the BusManager-maintained ER queue. The BusManager uses the hardware signal interrupt internally to maintain this queue. Interrupts may be designated to be VXIbus interrupts so that the Status/ID's returned in response to interrupt acknowledges are recognized as ER's and placed in the ER queue as well.

The *ER Interrupt* is asserted by the BusManager when the ER queue is non-empty. This interrupt is enabled and disabled via the `vcenintr()` and `vcdisintr()` functions. The `vcwaitintr()` function can be used to poll the ER Interrupt.

Programs may call a function that dequeues and returns the oldest message in the ER queue.

Servant Message Passing Functions

The BusManager provides the lowest-level interface for messaging operations. These functions support the VX5530/VX5535 as a VXIbus servant.

These functions include facilities for sending Event/Responses to the VX5530/VX5535 commander via VXIbus interrupts or the commander's signal register, and operations for reading/writing to/from the VX5530/VX5535 message registers.

Messages are 16- or 32-bit values that can be *sent* (written in the message registers to be read by another master on the VMEbus) or *received* (read from the message registers after previously having been written by another VMEbus master. For the purposes of this document, we refer to a local send of a message as *putting* a message (since the message is "put" into the local registers).

The VX5530/VX5535 bus interface provides support for message passing in the form of location monitors and a message interrupt. Note that the message interrupt is defined such that the message interrupt is asserted when nothing is happening (neither waiting to send nor receive messages).

Version Information Functions

Version information is available for the BusManager, VX5530/VX5535 hardware, and BIOS. A function that indicates whether the BusManager is currently loaded in the system is also provided.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Alphabetical Command Listing

Commands listed below with a brief purpose statement are described in detail, in alphabetical order, on the following pages.

vcbmver	Returns version number for BusManager.
vcbiosver	Returns version number for BIOS.
vcckbm	Checks if BusManager is loaded.
vcckintr	Reads the number of the interrupt currently being asserted by the VX5530/VX5535.
vcdiserr	Disables the specified error without affecting the handler assignment.
vcdisintr	Disables the specified interrupt without affecting the handler assignment.
vcelwscmd	Sends one extended long word serial command.
vcenerr	Enables the specified error without affecting the handler assignment.
vcenintr	Enables the specified interrupt without affecting the handler assignment.
vcerredir	Assigns specified interrupt to be the VXIbus event/response interrupt.
vcerservintr	Sends an ER from the VX5530/VX5535 as a servant to a commander, using a VXIbus interrupt.
vcerservsig	Sends signal to VX5530/VX5535's commander.
vcerunredir	Deassigns the specified interrupt as VXIbus event/response interrupt.
vcfromvme	Copies data from VMEbus space to the VX5530/VX5535 used as a controller.
vcgetaccmode	Gets current VMEbus access mode.
vcgetslavebase	Returns the current slave memory base address.
vcgetula	Reports VX5530/VX5535 controller's logical address.
vchwver	Returns version numbers for hardware.
vclwscmd **vclwscmdnr**	Sends long word serial command. The **nr** version of this command does not wait for a response.
vcmapbus	Maps VMEbus address to internal controller address.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcreststate	Restores the previously-saved VMEbus access state.
vcsavestate	Saves access mode and bus window base as the VMEbus access state.
vcsetaccmode	Sets VMEbus access mode.
vcsigintr	Signals a bus interrupt.
vctovme	Copies data from VX5530/VX5535 controller memory to VMEbus space.
vcwaitintr **vcwaitintr2**	Waits for interrupt before executing caller. The **2** version differs slightly (see command description).
vcwatchdog	Sets specified watchdog timer service.
vcwscmd **vcwscmdnr**	Sends one word serial command. The **nr** version of this command does not wait for a response.
vcwsrcvstr	Receives word serial string via BYTE REQUEST.
vcwsservarm	Arms the local VXIbus data register for the command receive function.
vcwsservpeek **vcwsservpeek2**	Peek command. Same as **vcwsservrcv** but does not remove incoming command. The **2** version differs slightly (see command description).
vcwsservrcv **vcwsservrcv2**	The VX5530/VX5535 as a servant waits for a command to arrive and returns it. The **2** version differs slightly (see command description).
vcwsservsend **vcwsservsend2**	Sends a word serial command from this servant to its commander. The **2** version differs slightly (see command description).
vcwssndstr	Sends word serial string via BYTE AVAILABLE.
vcwsstat	Gets designated servant's word serial status.
vcvmectrl	VMEbus control. Allows reading, setting and clearing of various control bits in the VMEbus subset of the VXIbus interface.
vcvxictrl	VXIbus control. Allows reading, setting and clearing of various control bits in the VXIbus interface.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcbiosver vcbiosver

Purpose: This function returns the version number for the BIOS.

Syntax:

```
DECLARE FUNCTION vcbiosver%
bmversion% = vcbiosver%
```

Description: The BIOS version number consists of the major and minor version numbers of the BIOS that is installed in the VX5530/VX5535.

Returns: Appropriate version number.

Errors: None.

Assembly: *Entry:*

AH =BMVER

Returns:

CH Major BIOS Version Number
CL Minor BIOS Version Number

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcbmver

vcbmver

Purpose: This function returns the version number for the BusManager.

Syntax:

```
DECLARE FUNCTION vcbmver%
bmversion% = vcbmver%
```

Description: The BusManager version number consists of a major version and minor version number assigned to the BusManager software currently running in the VX5530/VX5535.

Returns: Appropriate version number.

Errors: None.

Assembly: *Entry:*

AH	=BMVER

Returns:

AH	Major BusManager Version Number
AL	Minor BusManager Version Number

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcckbm

vcckbm

Purpose: Check BusManager.

Syntax:

```
DECLARE FUNCTION vcckbm%
ok% = vcckbm%
```

Return Value: Returns `VCSUCCESS` if the BusManager is loaded.

Errors: Returns `ERRFAIL` if the BusManager is not loaded.

Assembly: None.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcckintr

vcckintr

Purpose: Checks VXIbus interrupt.

Syntax:

```
DECLARE FUNCTION vcckintr%
interrupt% = vcckintr%
```

Description: Reads the number of the interrupt currently being asserted by this VX5530/VX5535.

Return Value: The number of the interrupt currently asserted. If no interrupt is being asserted (*e.g.,* the last interrupt has been acknowledged) then zero is returned.

Errors: None

Assembly: *Entry:*

AH = BMCKBI

Returns:

AL VXIbus interrupt (1..7) *or* 0 for no interrupt being asserted

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcdiserr vcdiserr

Purpose: Disables the specified error.

Syntax:

```
DECLARE FUNCTION vcdiserr%(BYVAL errcode%)
ok% = vcdiserr%(errcode%)
```

Where

`errcode` bus event code

Description: Use one of the following error numbers, as defined in **busmgr.bi**, for `errcode`.

```
' * Error Constants

CONST BMSYSFAILERR          = (0)
CONST BMBERRERR             = (1)
CONST BMACFAILERR           = (2)
CONST BMWATCHDOGERR         = (3)
```

Return Value: Returns `VCSUCCESS` if successful.

Errors: Returns `ERRFAIL` if error occurs.

Assembly: *Entry:*

AH = BMDERR

AL Bus event code (`errcode`)

0	SysFail
1	Bus error (BERR)
2	ACFAIL
3	Watchdog timer

Returns:

Carry Flag Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcdisintr

vcdisintr

Purpose: Disables the specified interrupt.

Syntax:

```
DECLARE FUNCTION vcdisintr%(BYVAL interrupt%)
ok% = vcdisintr%(interrupt%)
```

Where

`interrupt` VXIbus interrupt

Description: This function disables the specified interrupt. The parameter `interrupt` is (1..7) for VXIbus interrupt, 0 for message interrupt, or 8 for Event/Response pending.

Return Value: Returns `VCSUCCESS` if successful.

Errors: Returns `ERRFAIL` if error occurs.

Assembly: *Entry:*

AH = BMDINT
AL VXIbus interrupt (1..7), 0 for message interrupt, or 8 for Event/response pending `(interrupt)`

Returns:

Carry Flag Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcelwscmd vcelwscmd

Purpose: Sends one extended longword serial command.

Syntax:

```
DECLARE FUNCTION vcelwscmd%(BYVAL ula%, SEG cmd%, BYVAL
                            ticks%)
DIM cmd%[3]
ok% = vcelwscmd%(ula%, cmd%, ticks%)
```

Where

`ula`	servant's unique logical address
`cmd`	command to send
`ticks`	timeout in 18.2 Hz ticks

Description: A command will be sent only when the servant device's WRDY bit is set.

NOTE

Extended longword serial commands do not generate a reply.

Return Value: Returns `VCSUCCESS` if successful or a negative number on failure.

Errors: Returns one of the following:

`ERRTIMEOUT`	TimeOut
`ERRBERR`	Bus Error
`ERRWS`	Word Serial Protocol Error

Assembly: *Entry:*

AH	= BMELWSCMD
AL	Servant's ULA (`ula`)
ES:SI	Address of command to send (`cmd`)
CX	Number of 18.2 Hz clock ticks to send the command (`ticks`)

Returns:

AL	Return code. One of
	`VCSUCCESS`
	`ERRTIMEOUT`
	`ERRBERR`
	`ERRWS`

CX	Number of clock ticks remaining
Carry Flag	Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcenerr

vcenerr

Purpose: Enables the specified VXIbus error.

Syntax:

```
DECLARE FUNCTION vcenerr%(BYVAL errcode%)
ok% = vcenerr%(errcode%)
```

Where

`errcode` bus event code

Description: This should be used only in conjunction with the `vcdiserr` command .

Use one of the following error numbers, as defined in **busmgr.bi**, for `errcode`.

```
' * Error Constants

CONST BMSYSFAILERR          = (0)
CONST BMBERRERR             = (1)
CONST BMACFAILERR           = (2)
CONST BMWATCHDOGERR         = (3)
```

Return Value: Returns `VCSUCCESS` if successful.

Errors: Returns `ERRFAIL` if error occurs.

Assembly: *Entry:*

AH = BMEERR
AL Bus event code (`errcode`)
- 0 SysFail
- 1 Bus error (BERR)
- 2 ACFAIL
- 3 Watchdog timer

Returns:

Carry Flag Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcenintr

vcenintr

Purpose: Enables the specified interrupt without affecting the handler assignment.

Syntax:

```
DECLARE FUNCTION vcenintr%(BYVAL interrupt%)
ok% = vcenintr%(interrupt%)
```

Where

`interrupt` VXIbus interrupt

Description: This should be used only in conjunction with `vcdisintr`. The parameter `interrupt` is (1..7) for VXIbus interrupt, 0 for message interrupt, or 8 for Event/Response pending.

Return Value: Returns `VCSUCCESS` if successful.

Errors: Returns `ERRFAIL` if error occurs.

Assembly: *Entry:*

AH = BMEINT
AL VXIbus interrupt (1..7), 0 for message interrupt, or 8 for Event/response pending `(interrupt)`

Returns:

Carry Flag Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcerredir

vcerredir

Purpose: Designates an interrupt to be a VXIbus Event/Response interrupt for receipt of events and responses from servants.

Syntax:

```
DECLARE FUNCTION vcerredir%(BYVAL interrupt%)
ok% = vcerredir%(interrupt%)
```

Where

`interrupt` VXIbus interrupt from which to redirect ER's

Description: When an interrupt is redirected, the interrupt is unmasked.

At system restart no interrupts are redirected. Any number (1-7) of interrupts may be redirected.

After a redirected interrupt is asserted and acknowledged, the low 16-bits of the returned Status/ID is placed in the ER queue. An ER interrupt is then asserted (as the queue is now non-empty) .

Return Value: Returns `VCSUCCESS` if successful.

Errors: `ERRFAIL` will be returned if an error occurs.

Assembly: *Entry:*

AH = BMERREDIR
AL Interrupt number (1..7) `(interrupt)`

Returns:

AL Return code. One of
`VCSUCCESS`
`ERRFAIL`
Carry Flag Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcerservintr vcerservintr

Purpose: Sends an ER to a commander utilizing a VXIbus interrupt.

Syntax:

```
DECLARE FUNCTION vcerservintr%(BYVAL interrupt%, BYVAL er%)
ok% = vcerservintr%(interrupt%, er%)
```

Where

`interrupt`	VXIbus interrupt to send this ER through
`er`	ER value to send

Description: This function is for implementing a VXIbus servant interface on the VX5530/VX5535.

In the VX5530/VX5535, the event/response value sent via an interrupt must be stored in the message register that the VX5530/VX5535 uses for word serial communication with its commander before the interrupt is signaled. If there is no word serial activity pending, then the event/response value is just stored in the register and the operation proceeds. If a word serial command from the commander is in the register, then the command is saved and the register used. If there is outgoing data in the register, then this function must wait until the commander has read the data before signaling the interrupt. These various contortions are transparent to the caller.

Return Value: Returns `VCSUCCESS` if successful.

Errors: `ERRFAIL` indicates the VX5530/VX5535 is unable to send because an interrupt is currently being asserted. `ERRTIMEOUT` indicates a time out when sending the ER.

Assembly: *Entry:*

AH	= BMERSNDINTR
AL	Interrupt by which to send Event/Response `(interrupt)`
BX	Event/Response value to send `(er)`

Returns:

AL	Return code. One of
	`VCSUCCESS`
	`ERRTIMEOUT`
	`ERRFAIL`

Carry Flag	Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcerservsig

vcerservsig

Purpose: Signals the VX5530/VX5535's commander by placing a value in its signal register.

Syntax:

```
DECLARE FUNCTION vcerservsig%(BYVAL ula%, BYVAL er%)
ok% = vcerservsig%(ula%, er%)
```

Where

`ula`	unique logical address of commander
`er`	ER value to send

Description: This function is for implementing a VXIbus servant interface on the VX5530/VX5535.

Return Value: Returns `VCSUCCESS` if successful.

Errors: `ERRBERR` or `ERRFAIL` if the commander has no signal register.

Assembly: *Entry:*

AH	= BMERSNDSIG
AL	Unique logical address of the Commander to which to send the signal `(ula)`
BX	Event/Response value to send `(er)`

Returns:

AL	Return code. One of
	`VCSUCCESS`
	`ERRBERR`
	`ERRFAIL`

Carry Flag	Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcerunredir

vcerunredir

Purpose: De-assigns an interrupt as a VXIbus Event/Response interrupt and makes it available as a regular bus interrupt.

Syntax:

```
DECLARE FUNCTION vcerunredir%(BYVAL interrupt%)
ok% = vcerunredir%(interrupt%)
```

Where

`interrupt` VXIbus interrupt number previously redirected by `vcerredir`

Description: None.

Return Value: Returns `VCSUCCESS` if successful.

Errors: `ERRFAIL` will be returned if there is an attempt to un-redirect an interrupt that is not redirected.

Assembly: *Entry:*

AH = BMERUNREDIR
AL Interrupt number (1..7) `(interrupt)`

Returns:

AL Return code. One of
`VCSUCCESS`
`ERRFAIL`
Carry Flag Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcfromvme vcfromvme

Purpose: Copies data from VMEbus space to VX5530/VX5535 controller.

Syntax:

```
DECLARE FUNCTION vcfromvme%(BYVAL width%, BYVAL src&,
                            SEG dest%, BYVAL count%)
DIM dest%[ ... ]
ok% = vcfromvme%(width%, src&, dest%, count%)
```

Where

`width`	copy width
`src`	VXIbus address
`dest`	starting destination address in VX5530/VX5535 memory
`count`	byte count to transfer

Description: Efficiently copies a block of memory to VX5530/VX5535 memory from consecutive bus locations, using the current access mode. This is intended for communicating between two VX5530/VX5535s, or dealing with devices such as network cards that map their memory onto the bus.

The `(width)` parameter specifies how the data is moved: whether data is to be moved in 8-, 16-, or 32-bit blocks, and whether error checking is performed (see below). VXIbus transfers are always done on natural boundary alignment—16-bit quantities are read from the VXIbus only at even addresses, and 32-bit quantities are read from the VXIbus only at addresses divisible by 4. Transfers from non-aligned locations are done by reading a block and extracting the appropriate data. For instance, with a 32-bit copy from a non-aligned address, a leading 32-bit word would be read and the leading bytes copied from it, then whole 32-bit values would be transferred, finally a trailing word would be read and trailing bytes extracted.

If the Fast Copy (`BMFASTCOPY`) is not set in `(width)`, BERR is checked after every transfer. If there is an error, the copy is aborted and the BERR error handler is not called. Any errors are reflected by a non-zero return value of this function. This eliminates the need of requiring the program calling this function to coordinate with the BERR handler.

If the Fast Copy bit *is* set in `(width)`, only transfers to non-aligned locations (if any) are checked. This option takes advantage of the pipelining in the processor and the VXIbus interface hardware to achieve a five times improvement in transfer rate; however, errors may go undetected.

Note that `(count)` always contains the size of the transfer *in bytes* regardless of the width of the transfer. If `(count) = 0`, nothing will be transferred.

Any VXIbus mapping in place when this function is called is restored on return.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following manifest constants are defined in **busmgr.bi** to make life easier:

```
' * Copy defines

CONST BMW8              = &H1       ' * 8-bit xfer width
CONST BMW16             = &H2       ' * 16-bit xfer width
CONST BMW32             = &H4       ' * 32-bit xfer width
CONST BMFASTCOPY        = &H80      ' * fast copy bit
```

Return Value: Returns `VCSUCCESS` (zero) if the copy is successful, otherwise it's the number of bytes remaining when a bus error occurs.

Errors: Any errors are reflected by a non-zero return value.

Assembly: *Entry:*

AH	= BMCPIN
AL	Copy width `(width)`
	1 8-bit wide copy
	2 16-bit wide copy
	4 32-bit wide copy
	Optionally ORed with
	80H Fast Copy bit
ES:DI	Starting destination address in VX5530/VX5535 memory `(dest)`
BX	High 16-bits of VXIbus source address `(src)`
SI	Low 16-bits of VXIbus source address `(src)`
CX	Byte count to transfer, maximum of 64K-1 bytes `(count)`

Returns:

CX	= `VCSUCCESS` if copy successful, otherwise number of bytes remaining on bus error
Carry Flag	Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

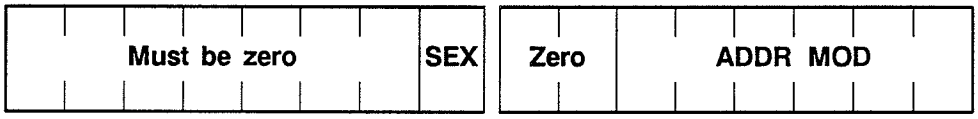
vcgetaccmode vcgetaccmode

Purpose: This gets the bus access parameters currently in effect.

Syntax:
```
DECLARE FUNCTION vcgetaccmode%
oldmode% = vcgetaccmode%
```

Description: This command uses the access mode constants defined in **busmgr.bi**. See `vcsetaccmode`.

Return Value: Current mode bits, in the format shown below.



Where:

SEX	= 0 for Little Endian (386-style)
	= 1 for Big Endian (68000-style)
ADDR MOD	VXIbus address modifier

Errors: None.

Assembly: *Entry:*

AH = BMGMODE

Returns:

DX Current mode bits

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcgetslavebase vcgetslavebase

Purpose: Gets slave memory base address.

Syntax:
```
DECLARE FUNCTION vcgetslavebase&
slavebase& = vcgetslavebase&
```

Description: Returns the current base address of where VX5530/VX5535 memory shows up in A32 space on the VXIbus. Returns a **long** (32-bit) `ERRFAIL` (i.e., -1L) if slave memory is disabled.

Return Value: Current slave memory base address in A32 space.

Errors: Returns `ERRFAIL`, if slave memory is disabled.

Assembly: *Entry:*

AH	= BMGSMB
AL	Reserved. Must be zero.

Returns:

AL	= 0 if slave memory disabled, = 1 if slave memory enabled
BX:DX	Current slave memory base address in A32 space, the most significant 16 bits in BX and the least significant 16 bits in DX

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcgetula vcgetula

Purpose: Reports the current Unique Logical Address of the VX5530/VX5535.

Syntax:
```
DECLARE FUNCTION vcgetula%
ula% = vcgetula%
```

Description: The controller's logical address is a number in the range 00H to FFH.

Return Value: A number in the range 00H to FFH.

Errors: None.

Assembly: *Entry:*

AH	= BMGULA
AL	Reserved. Must be zero.

Returns:

DX	Previous base address for registers in A16 space
Carry Flag	Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vchwver

vchwver

Purpose: This function returns the version number for the hardware.

Syntax:

```
DECLARE FUNCTION vchwver%
bmversion% = vchwver%
```

Description: The hardware version number indicates the VX5530/VX5535 hardware that the BusManager application was created to run on.

Returns: Appropriate version number.

Errors: None.

Assembly: *Entry:*

AH =BMVER

Returns:

BX Hardware Version Number

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vclwscmd
vclwscmdnr

vclwscmd
vclwscmdnr

Purpose: Sends one longword serial command.

Syntax:

```
DECLARE FUNCTION vclwscmd%(BYVAL ula%, BYVAL cmd&, SEG
                                    result&, BYVAL ticks%)
ok% = vclwscmd%(ula%, cmd&, result&, ticks%)

DECLARE FUNCTION vclwscmdnr%(BYVAL ula%, BYVAL cmd&,
                                    BYVAL ticks%)
ok% = vclwscmdnr%(ula%, cmd&, ticks%)
```

Where

`ula`	servant's unique logical address
`cmd`	command to send
`result`	result of the longword serial command
`ticks`	timeout in 18.2 Hz ticks

Description: A command will be sent only when the servant device's WRDY bit is set. The `vclwscmdnr` function is similar to `vclwscmd` except that it will not wait for the response of the long word serial command.

Return Value: Returns `VCSUCCESS` if successful or a negative number on failure.

Errors: Returns one of the following:

`ERRTIMEOUT`	TimeOut
`ERRBERR`	Bus Error
`ERRWS`	Word Serial Protocol Error
`ERRRTIMEOUT`	TimeOut on Receive
`ERRRBERR`	Bus Error on Receive

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Assembly: *Entry:*

AH	= BMLWSCMD
AL	Servant's ULA (`ula`)
BX,DX	Command to send (`cmd`)
ES:DI	Address at which to store result or 0 if no result expected (`result`)
CX	Number of 18.2 Hz clock ticks to send the command (`ticks`)

Returns:

AL	Return code. One of
	VCSUCCESS
	ERRTIMEOUT
	ERRBERR
	ERRWS
	ERRRTIMEOUT
	ERRRBERR
BX,DX	Result value
CX	Number of clock ticks remaining
Carry Flag	Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcmapbus vcmapbus

Purpose: Maps VMEbus address to internal controller address.

Syntax:

```
DECLARE SUB vcmapbus(BYVAL vxiaddr&, SEG busseg%, SEG
                                busoff%)
CALL vcmapbus(vxiaddr&, busseg%, busoff%)
```

Where

`vxiaddr`	VXIbus address to map to
`busseg`	VX5530/VX5535 internal segment address
`busoff`	VX5530/VX5535 internal offset address

Description: Given a bus address, sets up the map base and returns a VX5530/VX5535 pointer to the space. Accesses to the bus may be made via this pointer. All accesses are done according to the current access mode. It is up to the user to handle the situation when a range of addresses goes beyond the 64KB-aligned bus window.

NOTE

Since the VXIbus window is 64KB in size and aligned on a 64KB boundary, the BusManager uses only BX to set the mapping and passes DX back to the caller. ES is set to the location of the VXIbus window. Such arrangement is not guaranteed on future implementations of the bus map hardware.

Return Value: A VX5530/VX5535 pointer to the VXIbus location in `busseg` and `busoff`.

Errors: None.

Assembly: *Entry:*

AH	= BMMAP
DX	Low 16 bits of VXIbus address to which to map `(vxiaddr)`
BX	High 16 bits of VXIbus address to which to map `(vxiaddr)`

Returns:

ES:DX	Pointer to VXIbus location
Carry Flag	Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcreststate vcreststate

Purpose: Restores VMEbus access state.

Syntax:

```
DECLARE SUB vcreststate(SEG statestash&)
CALL vcreststate(statestash&)
```

Where

`statestash` previously saved state in internal format

Description: This function restores the access mode and bus window base that were previously saved via `vcsavestate`. No checking of this internal format is done. It does not affect interrupts.

Return Value: None.

Errors: None.

Assembly: *Entry:*

AH = BMREST
ES:DI Pointer to previously saved state in internal format (`statestash`)

Returns:

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcsavestate vcsavestate

Purpose: Saves access mode and bus window base as the VMEbus access state.

Syntax:

```
DECLARE SUB vcsavestate(SEG statestash&)
CALL vcsavestate(statestash&)
```

Where

`statestash` 4 byte area to save mapping state

Description: This function preserves access mode and bus window base in caller-supplied area. Saved information is in internal format, suitable only for a subsequent command, `vcreststate`, and not to be changed by the caller. This saved state does not include interrupts.

NOTE

The BusManager state is, in effect, part of the context of any program or TSR task that uses the bus. See notes on BusManager Context for considerations in saving and restoring this context.

`vcsavestate` and `vcreststate` are much faster operations than `vcset accmode` and `vcgetaccmode`, and should be used if the information is not being changed.

Return Value: None

Errors: None

Assembly: Entry:

AH = BMSAVE
ES:DI Pointer to a four byte area in which to save the mapping state (`statestash`)

Returns:

Buffer at ES:DI contains internal format mapping state as in `statestash`.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcsetaccmode vcsetaccmode

Purpose: Sets the bus access parameters used on all subsequent operations.

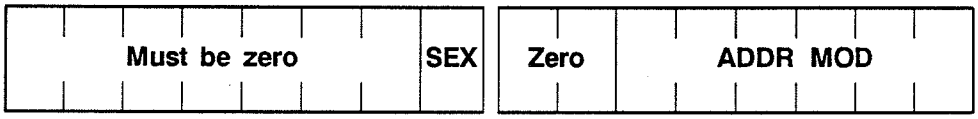
Syntax:

```
DECLARE FUNCTION vcsetaccmode%(BYVAL newmode%)
ok% = vesetaccmode%(newmode%)
```

Where

`newmode` new mode bits

Description: The new mode bit format is.



Where:

SEX	= 0 for Little Endian (386-style)
	= 1 for Big Endian (68000-style)
ADDR MOD	VXIbus address modifier

The hardware does not allow setting certain undefined VXIbus address modes. Setting undefined VXIbus address modes is an error and `ERRFAIL` is returned.

The include file **busmgr.bi** defines the following constants:

```
' * VME Address Modifier Codes

CONST A16N = &H29       ' * A16 non-privileged
CONST A16U = A16N
CONST A16S = &H2d       ' * A16 supervisor
CONST A24ND = &H39      ' * A24 non-privileged data
CONST A24NP = &H3a      ' * A24 non-privileged program
CONST A24N = A24ND
CONST A24U = A24ND
CONST A24SD = &H3d      ' * A24 supervisor data
CONST A24SP = &H3e      ' * A24 supervisor program
CONST A24S = A24SD
CONST A32ND = &H09      ' * A32 non-privileged data
CONST A32NP = &H0a      ' * A32 non-privileged program
CONST A32N = A32ND
CONST A32U = A32ND
CONST A32SD = &H0d      ' * A32 supervisor
CONST A32SP = &H0e      ' * A32 supervisor
CONST A32S = A32SD
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
' * VME Byte Ordering bit to be or'ed in with above addr
    mod

CONST BMMBO       = &H100      ' * Motorola byte ordering
CONST BMIBO       = &H0        ' * Intel byte ordering
```

Return Value: Returns VCSUCCESS on no error or one of the following error codes.

Errors: Returns ERRFAIL if the address mode is undefined.

Assembly: *Entry:*

AH = BMSMODE
DX New mode bits (newmode)

Returns:

Carry Flag Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcsigintr vcsigintr

Purpose: Signals a bus interrupt.

Syntax:

```
DECLARE FUNCTION vcsigintr%(BYVAL interrupt%)
ok% = vcsigintr%(interrupt%)
```

Where

`interrupt`	VXIbus interrupt

Description: If `(interrupt)` is non-zero and the VX5530/VX5535 is not currently asserting an interrupt, then the appropriate bus interrupt (1 thru 7 on VXIbus) is signaled. If `(interrupt)` is non-zero and the VX5530/VX5535 *is* currently asserting an interrupt waiting for an acknowledgement, then an error is returned. If `(interrupt)` is zero and the VX5530/VX5535 is currently asserting an interrupt waiting for an acknowledgement, then the bus interrupt is deasserted. It is not an error to cancel an interrupt when no interrupt is asserted—this function always suceeds if `(interrupt)` is zero.

NOTE

Deasserting an interrupt without waiting for acknowledgement may cause certain hardware configurations to lock up and should be used with extreme care.

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns `ERRFAIL` for failure.

Assembly: *Entry:*

AH	= BMSIG
AL	VXIbus interrupt (1..7) *or* 0 to deassert interrupt `(interrupt)`

Returns:

Carry Flag	Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vctovme vctovme

Purpose: Copies data from VX5530/VX5535 controller to VMEbus space.

Syntax:

```
DECLARE FUNCTION vctovme%(BYVAL width%, SEG src%, BYVAL
                          dest&, BYVAL count%)
DIM src%[ ... ]
ok% = vctovme%(width%, src%, dest&, count%)
```

Where

`width`	copy width
`src`	start source address in VX5530/VX5535 memory
`dest`	VXIbus address
`count`	byte count to transfer

Description: Efficiently copies a block of memory from VX5530/VX5535 memory to consecutive bus locations, using the current access mode. This is intended for communicating between two VX5530/VX5535s, or dealing with devices such as network cards that map their memory onto the bus.

The `(width)` parameter specifies how the data is moved: whether data is to be moved in 8-, 16-, or 32-bit blocks, and whether error checking is performed (see below). VXIbus transfers are always done on natural boundary alignment—16-bit quantities are written to the VXIbus only at even addresses, and 32-bit quantities are written to the VXIbus only at addresses divisible by 4. Transfers to non-aligned locations are done in a read-modify-write fashion, with the destination block read, appropriate bytes copied to the block, and the block replaced.

For example, with a 32-bit copy to a non-aligned address, the leading 32-bit word would be read from the destination, modified, and written back; then whole 32-bit values would be transferred; finally the trailing 32-bit word would be read from the destination, modified, and replaced.

NOTE

The "read-modify-write" sequence is done in software, and is not a RMW bus cycle.

NOTE

If a non-modified byte in a leading or trailing word contains a semaphore, and that semaphore is signaled while the copy is taking place, that signal may be lost.

If the Fast Copy (`BMFASTCOPY`) is not set in `(width)`, BERR is checked after every transfer. If there is an error, the copy is aborted and the BERR error handler is not called. Any errors are reflected by a non-zero return value of this function. This eliminates the need of requiring the program calling this function to coordinate with the BERR handler.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If the Fast Copy bit *is* set in `(width)`, only transfers to non-aligned locations (if any) are checked. This option takes advantage of the pipelining in the processor and the VXIbus interface hardware to achieve a five times improvement in transfer rate; however, errors may go undetected.

Note that `(count)` always contains the size of the transfer *in bytes* regardless of the width of the transfer. If `(count)` = 0, nothing will be transferred.

Any VXIbus mapping in place when this function is called is restored on return.

The following manifest constants are defined in **busmgr.bi** to make life easier:

```
' * Copy defines

CONST BMW8              = &H1       ' * 8-bit xfer width
CONST BMW16             = &H2       ' * 16-bit xfer width
CONST BMW32             = &H4       ' * 32-bit xfer width
CONST BMFASTCOPY        = &H80      ' * fast copy bit
```

Return Value: Returns `VCSUCCESS` if the copy is successful, otherwise it's the number of bytes remaining when a bus error occurs. This is the same as CX returned in the assembly language model.

Errors: Any errors are reflected by a non-zero return value.

Assembly: *Entry:*

AH	= BMCPOUT
AL	Copy width `(width)`
	1 8-bit wide copy
	2 16-bit wide copy
	4 32-bit wide copy
	Optionally ORed with
	80H Fast Copy bit
ES:DI	Starting source address in VC memory `(src)`
BX	High 16-bits of VXIbus destination address `(dest)`
SI	Low 16-bits of VXIbus destination address `(dest)`
CX	Byte count to transfer, maximum of 64K-1 bytes `(count)`

Returns:

CX	= `VCSUCCESS` if copy successful, otherwise number of bytes remaining on bus error
Carry Flag	Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcwaitintr
vcwaitintr2

vcwaitintr
vcwaitintr2

Purpose: Waits for interrupt before executing caller.

Syntax:

```
DECLARE FUNCTION vcwaitintr%(BYVAL mask%, SEG status&,
                             BYVAL ticks%)
ok% = vcwaitintr%(mask%, status&, ticks%)

DECLARE FUNCTION vcwaitintr2%(BYVAL mask%, SEG status&,
                              SEG memticks%)
ok% = vcwaitintr2%(mask%, status&, memticks%)
```

Where

`mask`	mask of interrupt
`status`	returned status/ID
`ticks`	number of 18.2 Hz clock ticks to wait for interrupt
`memticks`	address of clock ticks

Description: This function delays execution of the caller until either a bus interrupt arrives or a timer expires. The return code indicates which of these two happenstances transpired. In this implementation, only the highest priority interrupt is handled per call, where event/response pending is highest and the message interrupt is lowest. Other interrupts are left pending.

A 16-bit interrupt acknowledge cycle is performed when the interrupt arrives. It is up to the program to know whether the device returns an 8- or 16-bit Status/ID. For compatability with future products, this value is zero-extended to 32 bits.

`vcwaitintr` and `vcwaitintr2` return the mask of the highest priority interrupt that occurred (*e.g.*, `1 << intno`), zero if the timer expired, and ERRFAIL if some other error occurred. `vcwaitintr` and `vcwaitintr2` differ in that `vcwaitintr` takes clock ticks as a parameter, while `vcwaitintr2` takes a *pointer* to clock ticks as a parameter and returns the number of remaining clock ticks in that location.

The timer value is specified in DOS clock ticks (18.2 Hz) . If `(ticks) =0`, only one check will be made before returning. The appropriate interrupt acknowledge is done before returning to the caller. No bus state saving or restoring is done.

Interrupts are "sticky" in that whenever an interrupt occurs, that fact is remembered and will be returned by `vcwaitintr`. This eliminates the race condition that would otherwise exist between the device generating the interrupt and the program waiting for the interrupt; however, it can cause problems with the BusManager remembering stale interrupts. To avoid this problem a dummy wait with zero ticks should be done before using the device in order to clear out stale interrupts.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE

In order to use the clock for timing out, DOS interrupts are enabled.

Return Value: Mask of the interrupt that occurred or zero if timer expired.

Errors: Returns `ERRFAIL` on error.

Assembly: *Entry:*

AH = BMVXIWAIT
BX Mask of interrupts for which to wait `(mask)`
- 01H Message interrupt
- 02H VXIbus interrupt 1
- 04H VXIbus interrupt 2
- 08H VXIbus interrupt 3
- 10H VXIbus interrupt 4
- 20H VXIbus interrupt 5
- 40H VXIbus interrupt 6
- 80H VXIbus interrupt 7
- 100H Event/response pending

CX Number of 18.2 Hz clock ticks to wait for interrupt `(ticks)`

Returns:

AX Mask of the interrupt that occurred -or- zero if timer expired
BX Zero (0)
DX Status/ID of the highest priority interrupt `(status)`
CX Number of clock ticks remaining when the interrupt arrived
Carry Flag Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcwatchdog

vcwatchdog

Purpose: Specifies watchdog timer functions.

Syntax:

```
DECLARE FUNCTION vcwatchdog%(BYVAL code%)
ok% = vcwatchdog%(code%)
```

Where

`code` watchdog timer subfunction code

Description: The watchdog timer is designed to be used to protect against system lock-up in the case of error. If the timer is not reset or *tickled* within the specified interval, a watchdog error is signaled. Typically, the timer is tickled in frequently executed pieces of code, but never where there is the danger of infinite loops (*e.g.*, waiting for a device to become ready).

Two timer durations are available: 125 msec (suitable for hard realtime) and 8 sec.

Return Value: Returns `VCSUCCESS` if successful.

Errors: Returns `ERRFAIL` on error.

Assembly: *Entry:*

AH = BMWDT

AL Subfunction code `(code)`

0	Reset (tickle) timer
1	Disable timer
2	Enable timer
3	Set fast (125 msec) mode
4	Set slow (8 sec) mode

Returns:

Carry Flag Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcwscmd
vcwscmdnr

vcwscmd
vcwscmdnr

Purpose: Sends one word serial command.

Syntax:

```
DECLARE FUNCTION vcwscmd%(BYVAL ula%, BYVAL cmd%, SEG
                          result%, BYVAL ticks%)
ok% = vcwscmd%(ula%, cmd%, result%, ticks%)

DECLARE FUNCTION vcwscmdnr%(BYVAL ula%, BYVAL cmd%,
                          BYVAL ticks%)
ok% = vcwscmdnr%(ula%, cmd%, ticks%)
```

Where

`ula`	servant's unique logical address
`cmd`	command to send
`result`	result of the word serial command
`ticks`	timeout in 18.2 Hz clock ticks

Description: A command will be sent only when the servant device's WRDY bit is set. The `vcwscmdnr` function is similar to `vcwscmd` except that it will not wait for the response of the word serial command.

Return Value: Returns `VCSUCCESS` if successful or a negative number on failure.

Errors: Returns one of the following:

`ERRTIMEOUT`	TimeOut
`ERRBERR`	Bus Error
`ERRWS`	Word Serial Protocol Error
`ERRRTIMEOUT`	TimeOut on Receive
`ERRRBERR`	Bus Error on Receive

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Assembly: *Entry:*

AH	= BMWSCMD
AL	Servant's ULA (`ula`)
BX	Command to send (`cmd`)
ES:DI	Address at which to store result or 0 if no result expected (`result`)
CX	Number of 18.2 Hz clock ticks to send the command (`ticks`)

Returns:

AL	Return code. One of
	VCSUCCESS
	ERRTIMEOUT
	ERRBERR
	ERRWS
	ERRRTIMEOUT
	ERRRBERR
BX	Result value
CX	Number of clock ticks remaining
Carry Flag	Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Assembly: *Entry:*

AH	= BMWSRCVSTR
AL	Servant's ULA (`ula`)
BX:SI	Address of buffer to store received string (`msg`)
DX	Length of receive buffer
CX	Number of 18.2Hz clock ticks to send the string (`ticks`)

Returns:

AL	Return code. One of
	VCSUCCESS
	ERRBUFFERFULL
	ERRTIMEOUT
	ERRBERR
	ERRWS
	ERRRTIMEOUT
	ERRRBERR
DX	Number of bytes received (`bytecnt`)
CX	Number of clock ticks remaining, in assembly only
Carry Flag	Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcwsservarm vcwsservarm

Purpose: Arms the local VXIbus data register for the command receive function.

Syntax:

```
DECLARE FUNCTION vcwsservarm%(BYVAL code%)
ok% = vcwsservarm%(code%)
```

Where

`code` subfunction code

Description: The local VXIbus data register must be armed in order to receive a command. Arming for command receive causes a VXIbus-readable bit to be set indicating that a command will be accepted. In addition, the message interrupt may be enabled that will inform the program when the command arrives.

Subfunction code: `(code)`

0	Disarm command receive
1	Arm command receive
2	Arm command receive and enable message interrupt
4	Force disarm command receive
5	Force arm command receive
6	Force arm command receive and enable message interrupt

Subfunctions 0, 1, and 2 obey the VX5530/VX5535 locking protocol allowing multiple masters to communicate to the same device. This protocol requires that the VXI RESPONSE register not be touched by masters unless they are going to send a command. In environments where this rule may not be obeyed, the FORCE flavor of the various subfunctions may be used.

Return Value: Returns `VCSUCCESS` if successful.

Errors: Returns `ERRFAIL` on error.

Assembly: *Entry:*

AH = BMARM

AL Subfunction code `(code)`

0	Disarm command receive
1	Arm command receive
2	Arm command receive and enable message interrupt
4	Force disarm command receive
5	Force arm command receive
6	Force arm command receive and enable message interrupt

Returns:

Carry Flag Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcwsservpeek
vcwsservpeek2

vcwsservpeek
vcwsservpeek2

Purpose: Wait for a word serial command.

Syntax:

```
DECLARE FUNCTION vcwsservpeek%(msg&, BYVAL ticks%) )
ok% = vcwsservpeek%(msg&, ticks%)

DECLARE FUNCTION vcwsservpeek2%(msg&, SEG memticks%) )
ok% = vcwsservpeek2%(msg&, memticks%)
```

Where

`msg`	only first 2 bytes used in 16-bit command
`ticks`	number of 18.2 Hz clock ticks to wait for command
`memticks`	address of clock ticks

Description: `vcwsservpeek` does the same thing as `vcwsservrcv`, except that it does not remove the incoming command. The command stays available for subsequent `vcwsservpeek` or `vcwsservrcv` operations.

`vcwsservpeek` and `vcwsservpeek2` differ in that `vcwsservpeek` takes clock ticks as a parameter, while `vcwsservpeek2` takes a *pointer* to clock ticks as a parameter and returns the number of remaining clock ticks in that location.

NOTE

In order to use the clock for timing out, DOS interrupts are enabled.

Return Value: Returns the size of the command in bytes.

Errors: Returns 0 if timer expired, and
`ERRFAIL` if other error

Assembly: *Entry:*

AH	= BMMPEEK
CX	Number of 18.2 Hz clock ticks to wait for command `(ticks/memticks)`

Returns:

AL	Number of command bytes received (0, 2, or 4) (in assembly only)
DX, BX-DX	Received command (if any). 16-bit commands go in DX, 32-bit commands go in BX:DX with the most significant 16 bits in BX and the least significant 16 bits in DX
CX	Time remaining when command arrived (zero if timer expired) `(memticks)`
Carry Flag	Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcwsservrcv
vcwsservrcv2

vcwsservrcv
vcwsservrcv2

Purpose: The VX5530/VX5535 as a servant waits for a command to arrive and returns it.

Syntax:

```
DECLARE FUNCTION vcwsservrcv%(BYVAL code%, SEG msg&, BYVAL
                                        ticks%) )
ok% = vcwsservrcv%(code%, msg&, ticks%)

DECLARE FUNCTION vcwsservrcv2%(BYVAL code%, SEG msg&, SEG
                                        memticks%) )
ok% = vcwsservrcv2%(code%, msg&, memticks%)
```

Where

`code`	subfunction code
`msg`	only first 2 bytes used in 16-bit command
`ticks`	number of 18.2 Hz clock ticks to wait for command
`memticks`	address of clock ticks

Description: `(ticks)` or the data buffer of `(memticks)` contains the number of clock ticks for which to wait before giving up and returning an error. If `(ticks)` or the data buffer of `(memticks)` = 0, only one attempt will be made. The high 16-bits of the command are undefined if a 16-bit command is received.

Subfunction code: operation after command received `(code)`

0	Disarm command receive
1	Arm command receive
2	Arm command receive and enable command interrupt

`vcwsservrcv` and `vcwsservrcv2` differ in that `vcwsservrcv` takes clock ticks as a parameter, while `vcwsservrcv2` takes a *pointer* to clock ticks as a parameter and returns the number of remaining clock ticks in that location.

NOTE

In order to use the clock for timing out, DOS interrupts are enabled.

Return Value: The size of the command received.

Errors: Returns 0 if timer expired, and
`ERRFAIL` if other error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Assembly: *Entry:*

AH	= BMMRCV
AL	Subfunction code: operation after command received (`code`)
	0 Disarm command receive
	1 Arm command receive
	2 Arm command receive and enable command interrupt
CX	Number of 18.2 Hz clock ticks to wait for command (`ticks/memticks`)

Returns:

AL	Number of command bytes received (0, 2, or 4) (in assembly only)
DX, BX-DX	Received command (if any) . 16-bit commands go in DX, 32-bit command go in BX:DX with the most significant 16 bits in BX and the least significant 16 bits in DX
CX	Time remaining when command arrived (zero if timer expired)—result value of `memticks`
Carry Flag	Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcwsservsend
vcwsservsend2

vcwsservsend
vcwsservsend2

Purpose: Sends a word serial command from this servant to its commander.

Syntax:

```
DECLARE FUNCTION vcwsservsend%(BYVAL code%, BYVAL msg&,
                               BYVAL ticks%) )
ok% = vcwsservsend%(code%, msg&, ticks%)

DECLARE FUNCTION vcwsservsend2%(BYVAL code%, BYVAL msg&,
                                SEG memticks%) )
ok% = vcwsservsend2%(code%, msg&, memticks%)
```

Where

`code`	subfunction code
`msg`	only first 2 bytes used in 16-bit command
`ticks`	number of 18.2 Hz clock ticks to wait for command
`memticks`	address of clock ticks

Description: Sends a word serial command to this servant's commander by loading it to the VXIbus data registers and setting the outgoing data flag; however, before the command is sent the VXIbus data register must be clear. `(ticks)` or `(memticks)` contains the number of clock ticks to wait before giving up. If `(ticks)` or `(memticks)` = 0, only one check is made before returning an error. If there is no error and the low order bit of `(code)` is set, then the message interrupt is enabled. The high 16-bits of the command are not used if a 16-bit command is received; however, they *are* fetched by the BASIC interface routines.

Number of command bytes to put (2, or 4) `(code)`

0	Put no command, just wait for previous command to be read
1	Enable command interrupt and put no command
2	Put 16-bit command
3	Put 16-bit command and enable command interrupt
4	Put 32-bit command
5	Put 32-bit command and enable command interrupt

`vcwsservsend` and `vcwsservsend2` differ in that `vcwsservsend` takes clock ticks as a parameter, while `vcwsservsend2` takes a *pointer* to clock ticks as a parameter and returns the number of remaining clock ticks in that location.

NOTE

In order to use the clock for timing out, DOS interrupts are enabled.

Return Value: Returns `VCSUCCESS` if successful.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Errors: Returns `ERRFAIL` on error.

Assembly: *Entry:*

AH	= BMMPUT
AL	Number of command bytes to put (2, or 4)) `(code)`
0	Put no command, just wait for previous command to be read
1	Enable command interrupt and put no command
2	Put 16-bit command
3	Put 16-bit command and enable command interrupt
4	Put 32-bit command
5	Put 32-bit command and enable command interrupt
DX, BX-DX	Command to put. 16-bit commands go in DX, 32-bit commands go in BX:DX with the most significant 16 bits in BX and the least significant 16 bits in DX
CX	Number of 18.2 Hz clock ticks to wait for VXIbus data register to clear `(ticks/memticks)`

Returns:

CX	Time remaining when VXIbus data register cleared (zero if timer expired) `(memticks)`
Carry Flag	Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcwssndstr vcwssndstr

Purpose: Sends a series of bytes via the word serial BYTE AVAILABLE command.

Syntax:

```
DECLARE FUNCTION vcwssndstr%(BYVAL ula%, msg$, SEG
                                    bytecnt%, BYVAL ticks%)
ok% = vcwssndstr%(ula%, msg$, bytecnt%, ticks%)
```

Where

`ula`	servant's unique logical address
`msg`	address of string to send
`bytecnt`	number of bytes sent
`ticks`	timeout in 18.2 Hz clock ticks

Description: BYTE AVAILABLE commands are sent only when the device's DIR (Data Input Ready) and WRDY bits are set. This function sets the END bit in the last command of the series.

Return Value: Returns `VCSUCCESS` if successful or a negative number on failure.

Errors: Returns one of the following:

`ERRTIMEOUT`	TimeOut
`ERRBERR`	Bus Error
`ERRWS`	Word Serial Protocol Error

Assembly: *Entry:*

AH	= BMWSSNDSTR
AL	Servant's ULA (`ula`)
BX:SI	Address of the string to send (`msg`)
DX	Message length
CX	Number of 18.2 Hz clock ticks to send the string (`ticks`)

Returns:

AL	Return code. One of
	`VCSUCCESS`
	`ERRTIMEOUT`
	`ERRBERR`
	`ERRWS`

DX	Number of bytes sent (`bytecnt`)
CX	Number of clock ticks remaining
Carry Flag	Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcwsstat

vcwsstat

Purpose: Returns the word serial status of the designated servant.

Syntax:

```
DECLARE FUNCTION vcwsstat%(BYVAL ula%)
ok% = vcwsstat%(ula%)
```

Where

`ula` servant's unique logical address

Description: The status includes bits 14-8 of the servant's Bus Response Register returned as bit 14-8 of the result. This includes a reserved bit (14), DOR, DIR, ERR*, Read Ready, Write Ready, and FHS*.

Return Value: Returns status bits 14-8 of the designated servant.

Errors: Returns `ERRBERR` on error.

Assembly: *Entry:*

AH = BMWSSTAT
AL Servant's ULA `(ula)`

Returns:

AL Bits 14 through 8 of device's Bus Response Register
or ERRBERR
CX Number of clock ticks remaining, in assembly only
Carry Flag Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcvmectrl

vcvmectrl

Purpose: Allows reading, setting, and clearing of various control bits in the VMEbus subset of the VXIbus interface.

Syntax:

```
DECLARE FUNCTION vcvmectrl%(BYVAL code%, BYVAL flag%)
value% = vcvmectrl%(code%, flag%)
```

Where

`code`	subfunction code
`flag`	flag code

Description: The values are non-zero for asserted and zero for deasserted, without regard for the polarity of the actual hardware bits. For instance, asserting EXTENDED Self Test actually causes a hardware control bit to be set to zero, indicating that the VX5530/VX5535 is executing an extended self test because the EXTENDED Self Test control bit is negative true logic. This method allows software to be insulated from the actual flavor of the hardware controls.

See the **Assembly** heading for description of subfunction and flag codes.

Return Value: The return value of the function is defined only for subfunction 0 (read) . It should be checked only for equality to zero—individual bit settings depend on the particular flag code.

Errors: Returns `ERRFAIL` on error.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Assembly: *Entry:*

AH = BMVCTRL

AL Subfunction code (code)

- 0 Read flag
- 1 Assert flag
- 2 Deassert flag

DL Flag code (flag)

- 0 SYSFAIL out enable
- 1 SYSRESET in enable
- 2 (reserved)
- 3 SYSRESET out
- 4 Self test PASS
- 5 EXTENDED Self Test
- 6* Watchdog Timer Expired
- 7* VME ACFAIL asserted
- 8† VME BERR asserted
- 9* VME SYSFAIL asserted
- 10* VMEbus signal available
- 11* VMEbus access (read or write)
- 12* VMEbus write access
- 13* VMEbus pipeline busy
- 14 Sticky BERR. Set whenever a BERR occurs and cleared by software. Note: BusManager clears this bit when it detects a BERR.

Flags marked with * are read-only.

†BERR may be read and deasserted, but not asserted.

Returns:

AL Defined for subfunction 0 (read) only:

- = 0 if flag not asserted
- ≠ 0 if flag asserted

Carry Flag Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcvxictrl

vcvxictrl

Purpose: Allows the reading, setting and clearing of various control bits in the VXIbus interface.

Syntax:

```
DECLARE FUNCTION vcvxictrl%(BYVAL code%, BYVAL flag%)
value% = vcvxictrl%(code%, flag%)
```

Where

`code`	subfunction code
`flag`	flag code

Description: Values are non-zero for asserted and zero for deasserted, without regard for the polarity of the actual hardware bits. The flag code for triggers is the same as returned by `ORLMAllocate` (see section on the On Line Resource Manager). Thus, an application may call `ORLMAllocate` to allocate a trigger, and pass that same value to `vcvxictrl` to use the trigger.

See the **Assembly** heading for description of subfunction and flag codes.

Return Value: If the code passed into the function is READ (0), a 1 (one) will be returned if the flag is asserted, and a 0 (zero) will indicate that the flag is deasserted. If the code passed into the function is ASSERT (1) or DEASSERT (2), then a `VCSUCCESS` will be returned from the function if it is successful.

READing (0) a trigger returns the value that was last set via ASSERT or DEASSERT. SAMPLing (3) a trigger samples the actual value of the trigger line on the VXIbus. For a non-trigger flag (DIR, DOR, or ERR) READing and SAMPLing are equivalent.

Errors: Returns `ERRFAIL` if attempting to access a trigger line not defined on this hardware.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Assembly: *Entry:*

AH = BMVXCTRL

AL Subfunction code (`code`)

0	Read flag
1	Assert flag
2	Deassert flag
3	Sample trigger

DL Flag code (`flag`)

0	Data Input Ready (DIR)
1	Data Output Ready (DOR)
2	ERR flag
OLRMTTLTRG0	TTL Trigger Line 0
OLRMTTLTRG1	TTL Trigger Line 1
OLRMTTLTRG2	TTL Trigger Line 2
OLRMTTLTRG3	TTL Trigger Line 3
OLRMTTLTRG4	TTL Trigger Line 4
OLRMTTLTRG5	TTL Trigger Line 5
OLRMTTLTRG6	TTL Trigger Line 6
OLRMTTLTRG7	TTL Trigger Line 7
OLRMECLTRG0	ECL Trigger Line 0
OLRMECLTRG1	ECL Trigger Line 1
OLRMECLTRG2*	ECL Trigger Line 2
OLRMECLTRG3*	ECL Trigger Line 3
OLRMECLTRG4*	ECL Trigger Line 4
OLRMECLTRG5*	ECL Trigger Line 5

Trigger lines marked with an asterisk (*) are for future expansion and will not work on the standard VX5530/VX5535.

Returns:

AL Defined for subfunction 0 (read) only:
= 0 if flag not asserted
= 1 if flag asserted
= ERRFAIL if trigger line is not defined for this hardware

Carry Flag Set if error

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Section 5
TekVICS Online Resource Manager

Introduction

The Online Resource Manager (OLRM) gives application programs a high-level attribute-type interface to the devices in the VXIbus, and manages serially reusable resources, such as interrupt and trigger lines. The OLRM also allows VMEbus devices to be viewed in the same way as VXIbus devices.

The OLRM is attribute oriented, and allows devices to be addressed by either symbolic device name or logical address.

This section describes each command in detail. Commands are located in alphabetical order.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Calling the OLRM From MS BASIC

The following are examples of the usage of the OLRM functions.

```
i% = olrmgetnumattr%("Wavegen",0,OLRMSLOT)
```

Get the slot number of device Wavegen

```
i% = olrmgetnumattr%("globalmem",0,OLRMADDRESSBASE)
```

Get the memory base address of device globalmem

```
if olrmgetboolattr%("scanner1",0,OLRMSIGREG) <> 0 ...
```

Test scanner1 for a signal register

```
CALL olrmgetstrattr%("Wavegen",0, OLRMMANUFACTURER,
 manufname$)
```

Get symbolic manufacturer's name of Wavegen

```
retval% = olrmgetlist%("",0,OLRMDEVICES,256,lalist$)
```

Get list of logical addresses of all devices

```
ok% = olrmrename%("",25,"Mil1553")
```

Rename the device with logical address 25 as Mil1553

```
triggers% = olrmallocate%(OLRMTTLTRGANY2)
```

Allocate any two adjacent TTL trigger lines

vcmsb.bi and **busmgr.bi** are header files associated with the OLRM.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Alphabetical Command Listing

Following is an alphabetical listing and a brief purpose statement for each command. Commands are listed in detail in the remainder of this section.

olrmallocate	Allocates one, or a group of, trigger lines and interrupt lines
olrmdeallocate	Deallocates one, or a group of, trigger lines and interrupt lines
olrmgetboolattr	Returns a boolean attribute of a device
olrmgetlist	Returns an array of information about multiple devices
olrmgetmedia	Returns a value indicating which bridge a device uses
olrmgetnumattr	Returns a numerical attribute of a device
olrmgetslotnm	Returns an ASCII string for the name of the device in a particular slot
olrmgetstrattr	Returns an ASCII string attribute of a device
olrmrename	Renames a device

For most of these commands, the first two parameters are an ASCII device name and a numeric logical address. One or the other is used to refer to the device. In the BASIC interfaces, an empty string is used to indicate that the second parameter is to be used.

Unless otherwise noted, inappropriate parameters cause meaningless results to be returned (e.g., asking for the memory speed of a register-based VXIbus device).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

olrmallocate

olrmallocate

Purpose: Allocates trigger and interrupt line resources.

Syntax:

```
DECLARE FUNCTION olrmallocate%(BYVAL resource%)
ok% = olrmallocate%(resource%)
```

Where

`resource` type of resource

Description: Resources can be allocated specifically ("give me TTLTRG 4") and generically ("give me two TTL trigger lines").

The resource parameter may be one of the following:

OLRMTTLTRG0
OLRMTTLTRG1
OLRMTTLTRG2
OLRMTTLTRG3
OLRMTTLTRG4
OLRMTTLTRG5
OLRMTTLTRG6
OLRMTTLTRG7
OLRMTTLTRG01
OLRMTTLTRG23
OLRMTTLTRG45
OLRMTTLTRG67
OLRMTTLTRG0123
OLRMTTLTRG4567
OLRMTTLTRGANY
OLRMTTLTRGANY2
OLRMTTLTRGANY4
OLRMECLTRG0
OLRMECLTRG1
OLRMECLTRG2
OLRMECLTRG3
OLRMECLTRG4
OLRMECLTRG5
OLRMECLTRG01
OLRMECLTRG23
OLRMECLTRG45
OLRMECLTRGANY
OLRMECLTRGANY2
OLRMIRQ1
OLRMIRQ2
OLRMIRQ3
OLRMIRQ4
OLRMIRQ5
OLRMIRQ6
OLRMIRQ7
OLRMIRQANY

One can request the allocation of specific resources, groups of resources (e.g., TTL triggers 0 and 1), and "any" resources. To provide for D-size systems, resources include the extra four ECL triggers on the P3 connector.

To permit computation with resources, the encodings are numerically relative to the lowest-numbered resource of a class. Thus, for instance, OLRMTTLTRG1 is equal to OLRMTTLTRG0 + 1, and OLRMIRQ3 is equal to OLRMIRQ1 + 2.

Since the OLRMECLTRGANY and OLRMECLTRGANY2 cases could allocate, conceptually, nonexistent ECL triggers 2-5 in a C-size system, one should avoid using these in a C-size system.

NOTE

All resources are not necessarily available for allocation when the system is initialized. Specifically, the SURM allocates interrupt lines for VMEbus devices as described by use of the Configurator.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Return Value: If the resource was allocated, the returned value is nonzero, and is the resource number. In the case of multiple allocations (e.g., `olrmallocate (OLRM TTLTRGANY2)`), the value returned is that of the lowest-numbered of the resources. The returned value is 0 if the function is unsuccessful (e.g., specific resource already allocated, insufficient resources available, unknown resource).

Errors: None.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

olrmdeallocate

olrmdeallocate

Purpose: Places the specified resource(s) in the deallocated (available for allocation) state.

Syntax:

```
DECLARE SUB olrmdeallocate%(BYVAL resource%)
CALL olrmdeallocate%(resource%)
```

Where

`resource` type of resource

Description: The resource parameters can be any of those for **olrmallocate**, except for the "any" values.

Return Value: None.

Errors: None.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

olrmgetboolattr

olrmgetboolattr

Purpose: Returns requested information about a specific device.

Syntax:

```
DECLARE FUNCTION olrmgetboolattr%(devname$, BYVAL ula%,
                                  BYVAL attr%)
value% = olrmgetboolattr%(devname$, ula%, attr%)
```

Where

`devname`	device name
`ula`	unique logical address
`attr`	attribute (see chart below)

Description: The device can be addressed by symbolic name or logical address. The attribute may be one of the following. The comments indicate from where the information is obtained. Where a VXIbus register is indicated, a simulated register is used for VMEbus devices. The source "devtab" is the internal device table maintained by the SURM and OLRM.

Attribute	Source
OLRMREGISTERDEVICE	VXIbus ID register
OLRMMEMORYDEVICE	"
OLRMEXTENDEDDEVICE	"
OLRMMESSAGEDEVICE	devtab
OLRMA16ONLY	VXIbus ID register
OLRMA16A24	"
OLRMA16A32	"
OLRMA24A32ENABLED	VXIbus status register
OLRMMODID	"
OLRMEXTENDEDTEST	"
OLRMPASSED	"
OLRMSUPVSRONLY	VXIbus memory attribute register
OLRMBT	"
OLRMNP	"
OLRMD32	"
OLRMCMDR	VXIbus message protocol register
OLRMSIGREG	"
OLRMMASTER	"
OLRMINTERRUPTER	"
OLRMFHS	"
OLRMSHMEM	"
OLRMDOR	VXIbus message response register
OLRMDIR	"
OLRMERR	"

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Attribute	Source
OLRMRRDY	"
OLRMWRDY	"
OLRMFHSACTIVE	"
OLRMLOCKED	"
OLRMFAILED	devtab
OLRMNOTVXI	"
OLRMMEMALLOCATED	"
OLRMEXISTS	"

If the device is a VXIbus device, acquisition of most of the attributes causes a VXIbus access.

NOTE

The boolean value returned is always of positive logic,independent of the polarity of the actual VXIbus-defined bit. For instance, the attribute OLRMMODID returns TRUE if the device's MODID bit is 0; OLRMNP returns TRUE if a RAM device is nonvolatile or a ROM device electrically programmable.

Since most of the attributes are named the same way as in the VXIbus specification, familiarity with the naming in the VXIbus specification is assumed. The OLRM FAILED attribute denotes whether the SURM reported the device as failed and placed the device in the safe state. The OLRMNOTVXI attribute denotes whether the device is not a VXIbus device. The OLRMMEMALLOCATED attribute denotes whether address space for the device was reserved or allocated in the A24 or A32 address space. The OLRMEXISTS attribute denotes whether the referenced device (by symbolic name or logical address) is a known device.

Return Value: Information about the specified device.

Errors: If an error occurs (e.g., nonexistent device), the value returned is 0 (FALSE).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

olrmgetlist

olrmgetlist

Purpose: Returns a list of information as bytes in a character array, and a return value indicating the number of elements in the list.

Syntax:

```
DECLARE FUNCTION olrmgetlist%(devname$, BYVAL ula%, BYVAL
                                        attr%, value$)
retval% = olrmgetlist%(devname$, ula%, attr%, value$)
```

Where

`devname`	device name
`ula`	unique logical address
`attr`	attribute (see chart below)
`value`	the list of information

Description: The `value` argument returns the list of information.

Attribute	Source
OLRMDEVICES	devtab
OLRMSERVANTS	devtab

If the attribute is OLRMDEVICES, the `devname` and `ula` arguments are ignored. The logical addresses of all devices in the system (VXIbus and pseudo-VXIbus) are returned in the list.

If the attribute is OLRMSERVANTS, the logical addresses of the servants of the specified device are returned in the list. The device can be addressed by symbolic name (devname) or logical address.

Return Value: The number of elements in the list.

Errors: If an error occurs, the returned value is 0.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

olrmgetmedia

olrmgetmedia

Purpose: Finds the bridge that a device uses.

Syntax:

```
DECLARE FUNCTION olrmgetmedia&(devname$)
MEDIA& = olrmgetmedia&(devname$)
```

Description: This function returns the bridge media which the device is using. The constant definition for the bridge media is in the file **vcmsb.bi**.

Return Value: Returns the bridge media of the device specified in the `devname` parameter.

Errors: Returns `ERRILLEGALNAME` on error.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

olrmgetnumattr

olrmgetnumattr

Purpose: Returns requested numerical information about a specific device.

Syntax:

```
DECLARE FUNCTION olrmgetnumattr%(devname$, BYVAL ula%,
                                 BYVAL attr%)
value% = olrmgetnumattr%(devname$, ula%, attr%)
```

Where

`devname`	device name
`ula`	unique logical address
`attr`	attribute (see chart below)

Description: The device can be addressed by symbolic name or logical address.

Attribute	Source
OLRMCLASS	VXIbus ID register
OLRMADDRESSMODE	"
OLRMMANUFACTURER	"
OLRMREQMEMORY	VXIbus device-type register
OLRMMODEL	"
OLRMADDRESSBASE	VXIbus offset register
OLRMMEMORYTYPE	VXIbus memory attribute register
OLRMSPEED	"
OLRMLOGADDR	devtab
OLRMSLOT	"
OLRMCMDR	"
OLRMBID	VXIbus ID register
OLRMBDT	VXIbus device-type register
OLRMBSC	VXIbus status register
OLRMBSO	VXIbus offset register
OLRMBAT	VXIbus memory attribute register
OLRMBPR	VXIbus message protocol register
OLRMBRE	VXIbus message response register
OLRMBMH	VXIbus message data-high register
OLRMBML	VXIbus message data-low register

If the device is a VXIbus device, acquisition of most of the attributes causes a VXIbus access.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE

The available attributes cover both fields as well as whole registers. The encoding is the same as defined in the VXIbus specification (e.g., OLRMCLASS returns a value in the range 0..3).

The OLRMLOGADDR attribute denotes the logical address of the device. The OLRMSLOT attribute denotes the slot in which the device resides. The OLRM CMDR attribute denotes the logical address of the device's commander. Every device has a commander. The commander of the top level commander is itself. The OLRMBID, OLRMBDT, OLRMBSC, OLRMBSO, OLRMBAT, OLRMBPR, OLRMBRE, OLRMBMH, and OLRMBML attributes denote the value of the entire VXIbus register.

Return Value: As defined in the VXIbus specification.

Errors: The value returned is 0xFFFF.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

olrmgetslotname olrmgetslotname

Purpose: Finds the name of the device in a particular slot.

Syntax:

```
DECLARE FUNCTION olrmgetslotnm%(slot%, devname$)
ok% = olrmgetslotnm%(slot%, devname$)
```

where

`slot`	slot number
`devname`	device name in the slot

Description: The caller is responsible for allocating at least 13 bytes for the `devname` parameter.

Return Value: Returns `VCSUCCESS` if successful.

Errors: Returns `ERRFAIL` on error.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

olrmgetstringattr olrmgetstringattr

Purpose: Returns requested ASCII information about a specific device.

Syntax:

```
DECLARE SUB olrmgetstrattr%(devname$, BYVAL ula%, BYVAL
                                    attr%, value$)
CALL olrmgetstrattr%(devname$, ula%, attr%, value$)
```

Where

`devname`	device name
`ula`	unique logical address
`attr`	attribute (see chart below)
`value`	device information

Description: The device can be addressed by symbolic name or logical address.

Attribute	Source
OLRMDEVICENAME	devtab
OLRMMANUFACTURER	devtab
OLRMMODEL	devtab

These attributes are the symbolic values as reported by the SURM. The caller is responsible for allocating at least 13 bytes for the fourth parameter (the output string). The value is placed in this string and the address of this string is the returned value.

Return Value: The address of the output string.

Errors: If an error occurs, a NULL pointer is returned.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

olrmrename olrmrename

Purpose: Changes the symbolic name of a device.

Syntax:

```
DECLARE FUNCTION olrmrename%(devname$, BYVAL ula%, newname$)
ok% = olrmrename%(devname$, ula%, newname$)
```

Where

`devname`	device name
`ula`	unique logical address
`newname`	new name of device

Description: The device can be addressed by its current symbolic name or logical address. If the device is found, its name is changed to that of `newname` (or the first 12 characters of `newname` if larger than 12) and the returned value is identical to the newname parameter. If the device cannot be found, the returned value is NULL.

If the device being renamed is a VXIbus message-based device and MDS is installed, this function also calls the Message Delivery System's `MDSDelete Device` and `MDSCreateDevice` functions to notify MDS of the new name.

NOTE

`olrmrename` *does not change the hardware configuration file* **config.hwf**. *The name change is lost when the machine is shut down or rebooted.*

Return Value: The return value is identical to the `newname` parameter.

Errors: If an error occurs, a NULL pointer is returned.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Section 6
TekVICS Advanced Trigger Functions

Introduction

The Advanced Trigger Functions (ATF) provide advanced trigger, signal, and timing features within a VXIbus mainframe, as well as the necessary real time links between multiple VXIbus subsystems.

The user should be familiar with the functionality of the Advanced Trigger Functions. See Appendix A of the VX5530/VX5535 Users Manual for details.

Examples showing how to use the TekVICS Advanced Trigger Functions are provided at the end of this section.

NOTE

All of the ITB lines must be driven by a signal. None of the ITB lines can be left in an indeterminate state (neither high nor low), because resonance may occur in the other routed signals. One way to solve this is to find an unused STAR line and route it to all the unused ITB lines, then disable the receiver of the STAR line and enable the drivers of all the unused ITB lines.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATF Functions

Following is a functional listing of TekVICS Advanced Trigger Functions. Detailed descriptions of all functions follow later in this section.

ATF Functional Listing

Input/Output Unit	Configuration
ATFEnOutBNC( ) ATFEnInBNC( ) ATFOutBNCRt( ) ATFOutBNCBStat( )	ATFChkOption( ) ATFCkInstall( )
Delay Unit	**Logical Unit**
ATFSetDelayU( ) ATFEnDelayU( )	ATFSetLogicIn( ) ATFSetLogicOut( ) ATFEnLogicIn( ) ATFEnLogicOut( )
Clock Unit	**STARbus routing unit Unit**
ATFSelSysClk( )	ATFEnSRRec( ) ATFEnSRDrv( ) ATFSTARRoute( ) ATFSTARuproc( )
Power Temperature Monitor	**Protocol Unit**
ATFPowerTempStat( )	ATFSetProtoU( ) ATFEnProtoU( ) ATFSelTTLTRG( ) ATFSelECLTRG( ) ATFSendTrg( )

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Alphabetical Function Listing

Following is an alphabetical listing and a brief purpose statement for each function. Functions are listed in detail in the remainder of this section.

ATFCkInstall	Checks if ATF Device Driver is installed.
ATFChkOption	Checks current hardware configuration.
ATFEnDelayU	Enables/disables Delay Units.
ATFEnSRDrv	Enables/disables STARbus routing unit drivers.
ATFEnSRRec	Enables/disables STARbus routing unit receivers.
ATFEnInBNC	Enables/disables Input BNCs.
ATFEnLogicIn	Enables/disables Logical Unit input lines.
ATFEnLogicOut	Enables/disables Logical Unit output.
ATFEnOutBNC	Enables/disables Output BNCs.
ATFEnProtoU	Enables/disables Protocol Units.
ATFOutBNCRt	Routes signal from an ITB line to Output BNC.
ATFOutBNCBStat	Reads the current Output BNC B status.
ATFPowerTempStat	Measures power supply voltage and the board temperature inside the VX5530/VX5535.
ATFSelECLTRG	Selects ECL Trigger Line.
ATFSelSysClk	Selects internal/external clock.
ATFSelTTLTRG	Selects TTL Trigger Line.
ATFSendTrg	Sends internal μP trigger.
ATFSetDelayU	Sets up Delay Unit.
ATFSetLogicIn	Sets up Logical Unit input.
ATFSetLogicOut	Sets up Logical Unit output.
ATFSetProtoU	Sets up Protocol Unit.
ATFSTARRoute	Routes a signal through a STARbus routing unit.
ATFSTARuproc	Sets STARbus routing unit external trigger.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFCkInstall

ATFCkInstall

Purpose: Checks whether the Advanced Trigger Functions Device Driver is installed.

Syntax: `ATFCkInstall%( )`

Description: This function checks whether the Advanced Trigger Functions Device Driver (**atf.sys**) is installed.

Return Value: Returns `VCSUCCESS` if the driver is installed and `ERRFAIL` if the driver is not installed.

Errors: None.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFChkOption ATFChkOption

Purpose: Checks the current hardware configuration.

Syntax: `ATFChkOption%( )`

Description: Checks the current VX5530/VX5535 hardware configuration and saves it until the system is rebooted. This function can be used to check the number of STARbus routing units supported (1 or 3).

When the **atf.sys** device driver is installed, the driver parameters specify a default hardware configuration, and that default configuration is loaded internally on reboot. After **ATFChkOption** is executed, the current hardware configuration is saved internally. However, the default configuration will be restored each time the VX5530/VX5535 is rebooted.

NOTE

While this function is executing, it will output a signal to the Output BNC B.

Return Value: Returns one of the following constants which are defined in the **atf.h** include file.

`HWNOTAV`	Hardware not available
`ONESRCONFIG`	Only one STARbus routing unit is supported for STAR lines on VXIbus slots 2-5 (Option 01).
`FULLCONFIG`	All three STARbus routing units are supported.

Errors: None.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFEnDelayU

ATFEnDelayU

Purpose: Enables or disables signal routing through the Delay Unit.

Syntax: `ATFEnDelayU%(BYVAL delayunit%, BYVAL mode%)`

Where

`delayunit`	delay unit
`mode`	enable or disable

Description: Enables or disables signal routing through the Delay Unit.

The following constants are defined in the **atf.bi** include file.

Legal values for `delayunit` are:

`DELAY 1`	Delay Unit 1
`DELAY 2`	Delay Unit 2

Legal values for `mode` are:

`ENABLE`	Enable Delay Unit
`DISABLE`	Disable Delay Unit

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFEnSRDrv

ATFEnSRDrv

Purpose: Enables or disables STARbus routing unit drivers.

Syntax: `ATFEnSRDrv%(BYVAL unit%, BYVAL mode%)`

Where

`unit` STARbus routing unit
`mode` channels to be enabled

Description: Enables or disables the drivers of a STARbus routing unit. Enabling a driver causes the routed signal to be output on the corresponding channel.

`unit` specifies which STARbus routing unit is to be used. See legal values below. STARbus routing units 1 and 2 are not supported for Option 01 hardware.

`mode` specifies the channels to be enabled (up to 16 on each STARbus routing unit). By ORing the constants, multiple channels can be enabled at the same time.

The following constants are defined in the **atf.bi** include file.

Legal values for `unit` are:

SR0	STARbus routing unit 0
SR1	STARbus routing unit 1
SR2	STARbus routing unit 2

Legal values for `mode` are:

ITB0-7	ITB lines 0..7, available for all three Routing Units
STARX2-5	STARX 2..5 (STARbus routing unit 0)
STARY2-5	STARY 2..5 (STARbus routing unit 0)
STARX6-9	STARX 6..9 (STARbus routing unit 1)
STARY6-9	STARY 6..9 (STARbus routing unit 1)
STARX1,STARX10-12	STARX 1, STARX 10..12 (STARbus routing unit 2)
STARY1,STARY10-12	STARY 1, STARY 10..12 (STARbus routing unit 2)
0	Disables all drivers of the STARbus routing unit selected

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

ERRPARA	Parameter error
ERRHW	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFEnSRRec

ATFEnSRRec

Purpose: Enables or disables STARbus routing unit receivers.

Syntax: `ATFEnSRRec%(BYVAL unit%, BYVAL mode%)`

Where

`unit` STARbus routing unit

`mode` channels to be enabled

Description: Enables or disables the receivers of a STARbus routing unit. Enabling a receiver causes the input signal to be routed through the corresponding unit.

`unit` specifies which STARbus routing unit is to be used. See legal values below. STARbus routing units 1 and 2 are not supported for Option 01 hardware.

`mode` specifies the channels to be enabled (up to 16 on each STARbus routing unit). By ORing the constants, multiple channels can be enabled at the same time.

The following constants are defined in the **atf.bi** include file.

Legal values for `unit` are:

`SR0`	STARbus routing unit 0
`SR1`	STARbus routing unit 1
`SR2`	STARbus routing unit 2

Legal values for `mode` are:

`ITB0-7`	ITB lines 0..7, available for all three Routing Units
`STARX2-5`	STARX 2..5 (STARbus routing unit 0)
`STARY2-5`	STARY 2..5 (STARbus routing unit 0)
`STARX6-9`	STARX 6..9 (STARbus routing unit 1)
`STARY6-9`	STARY 6..9 (STARbus routing unit 1)
`STARX1,STARX10-12`	STARX 1, STARX 10..12 (STARbus routing unit 2)
`STARY1,STARY10-12`	STARY 1, STARY 10..12 (STARbus routing unit 2)
`0`	Disables all receivers of the STARbus routing unit selected

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFEnInBNC

ATFEnInBNC

Purpose: Enables or disables Input BNCs.

Syntax: `ATFEnInBNC%(BYVAL bnc%, BYVAL mode%)`

Where

`bnc`	Input BNC
`mode`	enable or disable

Description: Enables or disables Input BNCs. Enabling an input BNC causes the external signal to be connected to the appropriate Instrument Trigger Bus (ITB) line.

The following constants are defined in the **atf.bi** include file.

Legal values for `bnc` are:

`IBNCA`	Input BNC A
`IBNCB`	Input BNC B

Legal values for `mode` are:

`ENABLE`	Enable Input BNC
`DISABLE`	Disable Input BNC

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFEnLogicIn

ATFEnLogicIn

Purpose: Enables or disables Logic Unit input line.

Syntax: `ATFEnLogicIn%(BYVAL mode%)`

Where

`mode` input lines to be enabled

Description: Enables or disables Logic Unit input line. Enabling a Logic Unit input causes the selected signal to be routed through the Logic Unit.

The lower 7 bits of `mode` correspond to the Logic Unit's input lines 0-6. If bit x of `mode` is 1, input line x will be enabled. If bit x of `mode` is 0, input line x will be disabled.

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFEnLogicOut

ATFEnLogicOut

Purpose: Enables or disables Logic Unit output.

Syntax: `ATFEnLogicOut%(BYVAL mode%)`

Where

`mode` output mode

Description: Enables or disables Logic Unit output. The Logic Unit output is the logic OR of all the enabled inputs. The **ATFSetLogicIn** and **ATFSetLogicOut** functions are used to select other logic functions such as AND, NOR and NAND.

The following constants are defined in the **atf.bi** include file:

Legal values for `mode` are:

`ENABLE`	Enable output signal
`DISABLE`	Disable output signal

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFEnOutBNC

ATFEnOutBNC

Purpose: Enables or disables Output BNCs.

Syntax: `ATFEnOutBNC%(BYVAL bnc%, BYVAL mode%)`

Where

`bnc` Output BNC
`mode` enable or disable

Description: Enables or disables Output BNCs. Enabling an output BNC causes the connected signal to be output.

The following constants are defined in the **atf.bi** include file.

Legal values for `bnc` are:

OBNCA	Output BNC A
OBNCB	Output BNC B

Legal values for `mode` are:

ENABLE	Enable Output BNC
DISABLE	Disable Output BNC

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

ERRPARA	Parameter error
ERRHW	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFEnProtoU

ATFEnProtoU

Purpose: Enables or disables Protocol Unit.

Syntax: `ATFEnProtoU%(BYVAL unit%, BYVAL mode%)`

Where

`unit` Protocol Unit
`mode` enable or disable

Description: Enables or disables Protocol Unit. Enabling a Protocol Unit allows the selected protocol to be generated upon receipt of an input signal.

The following constants are defined in the **atf.bi** include file.

Legal values for `unit` are:

PROTO1	Protocol Unit 1
PROTO2	Protocol Unit 2

Legal values for `mode` are:

ENABLE	Enable Protocol Unit
DISABLE	Disable Protocol Unit

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

ERRPARA	Parameter error
ERRHW	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFOutBNCRt

ATFOutBNCRt

Purpose: Routes a signal from one of the ITB lines to Output BNC.

Syntax: `ATFOutBNCRt%(BYVAL bnc%, BYVAL itb%, BYVAL polarity%)`

Where

`bnc`	output BNC
`itb`	ITB line
`polarity`	signal polarity

Description: Routes a signal from one of the ITB lines to Output BNC and selects the polarity of the output signal.

The following constants are defined in the **atf.bi** include file.

Legal values for `bnc` are:

`OBNCA`	Output BNC A
`OBNCB`	Output BNC B

Legal values for `itb` are:

`ITB0...7`	ITB lines 0 - 7

Legal values for `polarity` are:

`POSI`	Non-inverted output signal
`NEGA`	Inverted output signal

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFOutBNCBStat

ATFOutBNCBStat

Purpose: Reads the current Output BNC B status.

Syntax: `ATFOutBNCBStat%( )`

Description: This function reads the current status of Output BNC B.

Return Value: If the output voltage is high, it returns a 1.
If the output voltage is low, it returns a 0.

Errors:

ERRHW	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFPowerTempStat ATFPowerTempStat

Purpose: Measures power supply voltage and the board temperature inside the VX5530/VX5535.

Syntax: `ATFPowerTempStat%(selection%, result!)`

Where

`selection`	supply or temperature selection
`result`	current reading

Description: Measures the current power supply voltages and current board temperature inside the VX5530/VX5535.

The power supply voltage reported in `result` can be one of the following:
+24V, +12V, +5V, –24V, –12V, –5.2V, –2V.

The following constants are defined in the **atf.bi** include file.

Legal values for `selection` are:

`P24V`	+24 V power supply
`P12V`	+12 V power supply
`P5V`	+5 V power supply
`N24V`	–24 V power supply
`N12V`	–12 V power supply
`N5PT2V`	–5.2 V power supply
`N2V`	–2 V power supply
`TEMPERATURE`	board temperature in Celsius

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFSelECLTRG

ATFSelECLTRG

Purpose: Selects ECL Trigger line.

Syntax: `ATFSelECLTRG%(BYVAL unit%, BYVAL ecltrg%)`

Where

`unit`	Protocol Unit
`ecltrg`	ECL Trigger Line

Description: Selects an ECL Trigger line for the Protocol Unit. If the Protocol Unit is set up for Extended Start Stop protocol or SYNC100 protocol, this function is used to select the ECL Trigger line for the protocol.

The following constants are defined in the **atf.bi** include file.

Legal values for `unit` are:

`PROTO1`	Protocol Unit 1
`PROTO2`	Protocol Unit 2

Legal values for `ecltrg` are:

`ECLTRG0`	ECL Trigger Line 0
`ECLTRG1`	ECL Trigger Line 1
`ECLTRG2`	ECL Trigger Line 2
`ECLTRG3`	ECL Trigger Line 3
`ECLTRG4`	ECL Trigger Line 4
`ECLTRG5`	ECL Trigger Line 5

NOTE

After executing this function, the Protocol Unit has to be enabled again before sending another trigger.

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFSelSysClk

ATFSelSysClk

Purpose: Selects internal/external clock.

Syntax: `ATFSelSysClk%(BYVAL clock%)`

Where

`clock` clock source

Description: Selects either internal or external clock as the CLK10 signal.

If internal clock is selected, it is generated by an internal clock in the VX5530/VX5535.

If external clock is selected, the CLK10 signal is phase-locked with the signal on ITB6 to maintain the clock rate at about 10 MHz.

The following constants are defined in the **atf.bi** include file:

Legal values for `clock` are:

`INTERNAL`	Internal clock
`EXTERNAL`	External clock

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFSelTTLTRG

ATFSelTTLTRG

Purpose: Selects TTL Trigger line.

Syntax: `ATFSelTTLTRG%(BYVAL unit%, BYVAL ttltrg%)`

Where

`unit`	Protocol Unit
`ttltrg`	TTL Trigger Line

Description: Selects a TTL Trigger line for the Protocol Unit. If the Protocol Unit is set up for Start Stop or Extended Start Stop protocol, this function is used to select the TTL Trigger line for the protocol.

The following constants are defined in the **atf.bi** include file.

Legal values for `unit` are:

`PROTO1`	Protocol Unit 1
`PROTO2`	Protocol Unit 2

Legal values for `ttltrg` are:

`TTLTRG0`	TTL Trigger Line 0
`TTLTRG1`	TTL Trigger Line 1
`TTLTRG2`	TTL Trigger Line 2
`TTLTRG3`	TTL Trigger Line 3
`TTLTRG4`	TTL Trigger Line 4
`TTLTRG5`	TTL Trigger Line 5
`TTLTRG6`	TTL Trigger Line 6
`TTLTRG7`	TTL Trigger Line 7

NOTE

After executing this function, the Protocol Unit has to be enabled again before sending another trigger.

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFSendTrg

ATFSendTrg

Purpose: Sends internal μP trigger.

Syntax: `ATFSendTrg%(BYVAL unit%, BYVAL command%)`

Where

`unit`	Protocol Unit
`command`	trigger command

Description: Sends internal μP trigger command to a Protocol Unit. Before executing this function, the Protocol Unit must be set up and enabled.

The following constants are defined in the **atf.bi** include file.

Legal values for `unit` are:

`PROTO1`	Protocol Unit 1
`PROTO2`	Protocol Unit 2
`PROTOBOTH`	Protocol Units 1 and 2

Legal values for `command` are:

`STARTTRG`	START command
`STOPTRG`	STOP command

NOTE

It is important to use trigger commands in pairs: if you issue a STOP command, it must be followed by a START command, and vice versa.

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFSetDelayU

ATFSetDelayU

Purpose: Sets up Delay Unit.

Syntax: `ATFSetDelayU%(BYVAL delayunit%, BYVAL delaytime%)`

Where

`delayunit`	delay unit
`delaytime`	delay time

Description: Sets up Delay Unit. The delay time ranges from 0 ns to 15 ns.

The following constants are defined in the **atf.bi** include file.

Legal values for `delayunit` are:

DELAY1	Delay Unit 1
DELAY2	Delay Unit 2

Legal values for `delaytime` are:

DELAY0NS	Delay 0 ns
DELAY1NS	Delay 1 ns
DELAY2NS	Delay 2 ns
DELAY3NS	Delay 3 ns
DELAY4NS	Delay 4 ns
DELAY5NS	Delay 5 ns
DELAY6NS	Delay 6 ns
DELAY7NS	Delay 7 ns
DELAY8NS	Delay 8 ns
DELAY9NS	Delay 9 ns
DELAY10NS	Delay 10 ns
DELAY11NS	Delay 11 ns
DELAY12NS	Delay 12 ns
DELAY13NS	Delay 13 ns
DELAY14NS	Delay 14 ns
DELAY15NS	Delay 15 ns

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

ERRPARA	Parameter error
ERRHW	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFSetLogicIn

ATFSetLogicIn

Purpose: Sets up Logic Unit input.

Syntax: `ATFSetLogicIn%(BYVAL inputline%, BYVAL mode%)`

Where

`inputline`	Logic Unit input line
`mode`	input line setup

Description: Sets up Logic Unit input line. The Logic Unit input lines correspond to ITB lines. The Logic Unit input can be set up for level or edge sensitive and for non-inverted or inverted signals. Combining **ATFSetLogicIn** with **ATFSetLogicOut** allows the Logic Unit to perform various logic functions such as AND, NOR and NAND.

The following constants are defined in the **atf.bi** include file.

Legal values for `inputline` are:

`LOGICI0`	Logic unit input 0—ITB0
`LOGICI1`	Logic unit input 1—ITB1
`LOGICI2`	Logic unit input 2—ITB2
`LOGICI3`	Logic unit input 3—ITB3
`LOGICI4`	Logic unit input 4—ITB4
`LOGICI5`	Logic unit input 5—ITB5
`LOGICI6`	Logic unit input 6—ITB6

Legal values for `mode` are:

`LEVELPOS`	Level sensitive and non-inverted
`LEVELNEG`	Level sensitive and inverted
`EDGEPOS`	Edge sensitive and non-inverted
`EDGENEG`	Edge sensitive and inverted

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFSetLogicOut

ATFSetLogicOut

Purpose: Sets up Logic Unit output.

Syntax: `ATFSetLogicOut%(BYVAL mode%)`

Where

`mode` output mode

Description: Sets up Logic Unit output mode. The output of the Logic Unit can be set to non-inverted or inverted mode. Combining **ATFSetLogicIn** with **ATFSetLogicOut** allows the Logic Unit to perform various logic functions such as AND, NOR and NAND.

The following constants are defined in the **atf.bi** include file.

Legal values for `mode` are:

`POSI`	Non-inverted output signal
`NEGA`	Inverted output signal

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFSetProtoU

ATFSetProtoU

Purpose: Sets up Protocol Unit.

Syntax: `ATFSetProtoU%(BYVAL unit%, BYVAL trgsrc%, BYVAL protocol%)`

Where

`unit`	protocol unit
`trgsrc`	trigger source
`protocol`	trigger protocol

Description: Sets up Protocol Unit. There are two Protocol Units in the VX5530/VX5535. Both of them have the same functionality. The trigger source for the Protocol Unit can be either from external or internal μP trigger.

For external trigger, ITB4 is used to trigger Protocol Unit 1 and ITB5 is used to trigger Protocol Unit 2.

For internal μP trigger, **ATFSendTrg** is used to send the trigger command.

Start Stop, Extended Start Stop or SYNC100 protocol can be selected for the Protocol Unit. In that case, **ATFSelECLTRG** or **ATFSelTTLTRG** functions are used to select the ECL Trigger line or the TTL Trigger line to be used in the protocol.

The following constants are defined in the **atf.bi** include file.

Legal values for `unit` are:

`PROTO1`	Protocol Unit 1
`PROTO2`	Protocol Unit 2

Legal values for `trgsrc` are:

`INTERNAL`	Internal μP trigger
`EXTERNAL`	External trigger

Legal values for `protocol` are:

`STARTSTOP`	Start Stop protocol
`ESTARTSTOP`	Extended Start Stop protocol
`SYNC100`	SYNC100 protocol

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFSTARRoute ATFSTARRoute

Purpose: Routes a signal through a STARbus routing unit.

Syntax:

```
ATFSTARRoute%(BYVAL unit%, BYVAL source%, BYVAL
                    destination%)
```

Where

`unit`	STARbus routing unit
`source`	STARbus routing unit source channel
`destination`	STARbus routing unit destination channel

Description: Routes a signal through a STARbus routing unit.

`unit` specifies the STARbus routing unit to use. See legal values below. STARbus routing units 1 and 2 are not supported for Option 01 hardware.

`source` specifies the STARbus routing unit input channel. See legal values below.

`destination` specifies the STARbus routing unit channels the signal is routed to. Multiple destination channels can be selected at the same time by ORing the channel constants.

The following constants are defined in the **atf.bi** include file.

Legal values for `unit` are:

`SR0`	STARbus routing unit 0
`SR1`	STARbus routing unit 1
`SR2`	STARbus routing unit 2

Legal values for `source` are:

`ITB0-7`	ITB lines 0..7, available for all three Routing Units
`STARX2-5`	STARX 2..5 (STARbus routing unit 0)
`STARY2-5`	STARY 2..5 (STARbus routing unit 0)
`STARX6-9`	STARX 6..9 (STARbus routing unit 1)
`STARY6-9`	STARY 6..9 (STARbus routing unit 1)
`STARX1,STARX10-12`	STARX 1, STARX 10..12 (STARbus routing unit 2)
`STARY1,STARY10-12`	STARY 1, STARY 10..12 (STARbus routing unit 2)
`SRUPROC`	STAR Route microprocessor control signal
`SRCLK10`	Clock 10 signal

The STAR Route microprocessor control signal can be routed to the destination channel only through STARbus routing unit 1 or STARbus routing 2. The `CLK10` signal can be routed to the destination channel only through STARbus routing unit 0.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Legal values for `destination` are:

`ITB0-7`	ITB lines 0..7, available for all three Routing Units
`STARX2-5`	STARX 2..5 (STARbus routing unit 0)
`STARY2-5`	STARY 2..5 (STARbus routing unit 0)
`STARX6-9`	STARX 6..9 (STARbus routing unit 1)
`STARY6-9`	STARY 6..9 (STARbus routing unit 1)
`STARX1,STARX10-12`	STARX 1, STARX 10..12 (STARbus routing unit 2)
`STARY1,STARY10-12`	STARY 1, STARY 10..12 (STARbus routing unit 2)

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ATFSTARuproc ATFSTARuproc

Purpose: Sets STARbus routing unit microprocessor control signal.

Syntax: `ATFSTARuproc%(BYVAL unit%, BYVAL mode%)`

Where

`unit`	STARbus routing unit
`mode`	microprocessor control signal

Description: This function sets up the microprocessor control signal to be routed by a STARbus routing unit. The STARbus routing unit can route an microprocessor control signal (controlled by the microprocessor) to any other channel by using the function **ATFSTARRoute** . The microprocessor control signal can be set to high or low value by the **ATFSTARuproc** function. This function is not supported for Option 01 hardware.

The following constants are defined in the **atf.bi** include file.

Legal values for `unit` are:

`SR1`	STARbus routing unit 1
`SR2`	STARbus routing unit 2

Legal values for `mode` are:

`POSI`	High voltage level
`NEGA`	Low voltage level

NOTE

This function may result in a different signal being routed to Output BNC B. Do not use Output BNC B and this function at the same time.

Return Value: Returns `VCSUCCESS` for success.

Errors: Returns one of the following:

`ERRPARA`	Parameter error
`ERRHW`	Hardware not available

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Advanced Trigger Function Examples

Example 1: Routes a signal from Input BNC A to Output BNC B, and inverts the output signal.

Input BNC A is connected to ITB6.

```
ATFEnInBNC(IBNCA, ENABLE)              ' enable input BNC A
ATFOutBNCRt(OBNCB, ITB6, NEGA)         ' route from ITB6 to Output BNC B
                                       and invert the output signal
ATFEnOutBNC(OBNCB, ENABLE)             ' enable output BNC B
```

Example 2: Do a logic OR with the Input BNC A signal and the Input BNC B signal, then output the resulting signal to Output BNC A.

Input BNC A is connected to ITB6 and Input BNC B is connected to ITB4.

```
ATFEnInBNC(IBNCA, ENABLE)              ' enable input BNC A
ATFEnInBNC(IBNCB, ENABLE)              ' enable input BNC B
ATFSetLogicIn(LOGICI4, LEVELPOS)       ' set up logic input 4 (Input   BNC B)
                                       for level sensitive and non-inverted
ATFSetLogicIn(LOGICI6, LEVELPOS)       ' set up logic input 6 (Input   BNC A)
                                       for level sensitive and non-inverted
ATFEnLogicIn(&H50)                     ' Logic Unit input lines 4 and 6
                                       are used
ATFEnLogicOut(ENABLE)                  ' enable logic output
ATFOutBNCRt(OBNCA, ITB7, POSI)         ' route from ITB7 to Output BNC A
ATFEnOutBNC(OBNCA, ENABLE)             ' enable Output BNC A
```

Example 3: Routes a signal from Input BNC A to STARX2 and STARY3.

Input BNC A is connected to ITB6.

```
ATFEnInBNC(IBNCA, ENABLE)                    ' enable input BNC A
ATFSTARRoute(SR0, ITB6, STARX2 OR STARY3)    ' In STARbus routing unit 0,
                                             signal from ITB6 is routed to
                                             STARY3 and STARX2
ATFEnSRRec(SR0, ITB6)                        ' enable ITB6 receiver
ATFEnSRDrv(SR0, STARX2 OR STARY3)            ' enable STARX2 + STARY3 driver
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example 4: Routes a signal from STARX7 to STARY3, STARX9, and STARY12.

ITB3 is selected to route the signal from STARbus Routing Unit 1 to STARbus Routing Units 0 and 2. Signal from STARX7 is routed to ITB3 and STARX9, then from ITB3 to STARY3 and STARY12.

```
ATFSTARRoute(SR1, STARX7, ITB3 OR STARX9)    ' route from STARX7 to ITB3
                                             and STARX9
ATFEnSRRec(SR1, STARX7)                      ' enable STARX7 receiver
ATFEnSRDrv(SR1, ITB3 OR STARX9)              ' enable ITB3 and STARX9 driver
ATFSTARRoute(SR0, ITB3, STARY3)              ' route from ITB3 to STARY3
ATFEnSRRec(SR0, ITB3)                        ' enable ITB3 receiver
ATFEnSRDrv(SR0, STARY3)                      ' enable STARY3 driver
ATFSTARRoute(SR2, ITB3, STARY12)             ' route from ITB3 to STARY12
ATFEnSRRec(SR2, ITB3)                        ' enable ITB3 receiver
ATFEnSRDrv(SR2, STARY12)                     ' enable STARY12 driver
```

Example 5: Routes a signal from Input BNC A to Delay Unit 1. Delay Unit 1 is set up to delay 10 ns and the output of Delay Unit 1 is routed to Output BNC B.

Since Input BNC A uses ITB6 as input and Delay Unit 1 uses ITB0 as input, the signal from ITB6 must route through one of the STARbus Routing Units to ITB0. In this example, STARbus Routing Unit 0 is chosen.

```
ATFEnInBNC(IBNCA, ENABLE)                    ' enable Input BNC A
ATFSetDelayU(DELAY1, DELAY10NS)              ' signal delay 10 ns through
                                             Delay Unit 1
ATFEnDelayU(DELAY1, ENABLE)                  ' enable Delay Unit 1
ATFSTARRoute(SR0, ITB6, ITB0)                ' route from ITB6 to ITB0 in
                                             STARbus routing unit 0
ATFEnSRRec(SR0, ITB6)                        ' enable ITB6 receiver
ATFEnSRDrv(SR0, ITB0)                        ' enable ITB0 driver
ATFOutBNCRt(OBNCB, ITB1, POSI)               ' signal from ITB1 to Output BNC B
ATFEnOutBNC(OBNCB, ENABLE)                   ' enable Output BNC B
```

Example 6: Routes the internal CLKL10 signal to Output BNC A.

ITB4 is chosen to route the signal.

```
ATFSelSysClk(INTERNAL)                       ' select internal CLK10 signal
ATFSTARRoute(SR0, SRCLK10, ITB4)             ' route from CLK10 to ITB4
ATFEnSRDrv(SR0, ITB4)                        ' enable ITB4 driver
ATFOutBNCRt(OBNCA, ITB4, POSI)               ' route from ITB4 to Output BNC A
ATFEnOutBNC(OBNCA, ENABLE)                   ' enable Output BNC A
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example 7: Do a logic OR with the STARbus Routing Unit 1 microprocessor control and STARX4, and route the resulting signal to STARX10.

ITB3 is chosen to route STARbus Routing Unit 1 microprocessor control signal. ITB4 is chosen to route the signal from STARX4 to the Logic Unit. The **ATFSTARuproc** function is used to set the microprocessor control signal to High.

```
ATFSTARRoute(SR1, SRUPROC, ITB3)      ' route from STARbus Routing Unit 1
                                      microprocessor control to ITB3
ATFEnSRDrv(SR1, ITB3)                 ' enable ITB3 driver
ATFSTARRoute(SR0, STARX4, ITB4)       ' route from STARX4 to ITB4
ATFEnSRRec(SR0, STARX4)               ' enable STARX4 receiver
ATFEnSRDrv(SR0, ITB4)                 ' enable ITB4 driver
ATFSetLogicIn(LOGICI3, LEVELPOS)      ' set up Logic Unit 3
ATFSetLogicIn(LOGICI4, LEVELPOS)      ' set up Logic Unit 4
ATFEnLogicIn(&H18)                    ' enable logic input 3 and 4
ATFEnLogicOut(ENABLE)                 ' enable logic output
ATFSTARRoute(SR2, ITB7, STARX10)      ' route from ITB7 to STARX10
ATFEnSRRec(SR2, ITB7)                 ' enable ITB7 receiver
ATFEnSRDrv(SR2, STARX10)              ' enable STARX10 driver
ATFSTARuproc(SR1, POSI)               ' set microprocessor control
                                      signal HIGH
```

Example 8: Send Extended Start Stop protocol from Protocol Unit 1 using ECL Trigger line 2 and TTL Trigger line 4 with the processor controlling the trigger command. After the setup, a START command and a STOP command are sent.

```
ATFSetProtoU(PROTO1, INTERNAL, ESTARTSTOP)  ' Protocol Unit 1 is set up
                                            for internal trigger and
                                            Extended Start Stop protocol
ATFSelECLTRG(PROTO1, ECLTRG2)               ' Protocol Unit 1 uses
                                            ECL Trigger line 2
ATFSelTTLTRG(PROTO1, TTLTRG4)               ' Protocol Unit 1 uses
                                            TTL Trigger line 4
ATFEnProtoU(PROTO1, ENABLE)                 ' enable Protocol Unit 1
ATFSendTrg(PROTO1, STARTTRG)                ' send START command
ATFSendTrg(PROTO1, STOPTRG)                 ' send STOP command
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example 9: Sends SYNC100 protocol from Protocol Unit 1 using ECL Trigger line 5. The trigger command comes from Input BNC B.

Both Input BNC B and Protocol Unit 1 are connected to ITB4.

```
ATFSetProtoU(PROTO1, EXTERNAL, SYNC100) ' Protocol Unit 1 is set up
                                        for external trigger and
                                        SYNC100 protocol
ATFSelECLTRG(PROTO1, ECLTRG5)           ' Protocol Unit 1 uses ECL
                                        Trigger line 5
ATFEnProtoU(PROTO1, ENABLE)             ' enable Protocol Unit 1
ATFEnInBNC(IBNCB, ENABLE)               ' enable Input BNC B
```

Example 10: Measures current +24 V, + 5 V power supply voltages and current board temperature.

```
ATFPowerTempStat(P24V, result!)         ' measure +24V voltage
ATFPowerTempStat(P5V, result!)          ' measure +5V voltage
ATFPowerTempStat(TEMPERATURE, result!)  ' measure temperature
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix A
I/O Control Commands

This Appendix contains a description of I/O Control commands used by the GPIB and VXIbus bridges. The define files containing these commands are **gpibrdg.bi** and **vxicmd.bi**, respectively.

GPIBRDG Control Commands

These commands allow the user to alter temporarily the default communication attributes for a device or to perform GPIB functions (such as Interface Clear). The commands are described in Table A-1.

Note that there is no restriction on the number of devices specified by the configurator. However, the bridge must keep track of changes that are programmably made to device attributes (such as timeout). The limit to the number of attribute default changes is fifty devices.

It is recommended that the users of these commands understand how to control the GPIB at a low level before attempting to use the Low Level Bus control functions.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1 GPIBRDG Control Commands

Command	Return Value	Description
GPIBINITIALIZE	0	Initializes the GPIB bridge. This removes any changes to default device characteristics that may have been changed by the GPIBRDG control commands (for example, TIMEOUT).
GPIBVERSION	Upper byte = major version, lower byte = minor version	Returns the software revision number of the GPIBRDG.
GPIBFUNCTIONSERIALPOLL	Device status byte	Performs a serial poll on the selected device and returns the device status byte.
GPIBFUNCTIONPARALLELPOLL	Result of parallel poll	Performs a parallel poll and returns the result. Use as follows: GPIBFUNCTIONPARALLELPOLL OR &H00YY0000; /* YY = hex byte code */
GPIBFUNCTIONPPOLLCONFIG	0	Allows the device to be configured for a parallel poll.
GPIBFUNCTIONGOTOLOCAL	0	Allows control to be temporarily returned to the device front panel.
GPIBFUNCTIONDEVICECLEAR	0	Allows the device to be cleared.
GPIBFUNCTIONTRIGGER	0	Sends the Group Execute Trigger (GET) command to the device. All devices may be triggered using the GPIBCOMMANDGET command.
GPIBFUNCTIONREADBYTE	0	Allows a single byte to be read from the device.
GPIBFUNCTIONWRITEBYTE	0	Allows a single byte to be written to the device.
GPIBMODEEOIWRITEON	0	Specifies that the EOI signal will be asserted at the end of the message.
GPIBMODEEOIWRITEOFF	0	Specifies that the EOI signal will NOT be asserted at the end of the message.
GPIBMODEEOSENDREAD	0	Specifies that a received message will be terminated when an EOS condition is detected.
GPIBMODEEOSREADMORE	0	Specifies that a received message will NOT be terminated when an EOS condition is detected.
GPIBMODEEOSEOION	0	Specifies that EOI will be asserted with EOS.
GPIBMODEEOSEOIOFF	0	Specifies that EOI will NOT be asserted with EOS.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1 GPIBRDG Control Commands (Cont)

Command	Return Value	Description
GPIBMODEEOS8BIT	0	Specifies that an eight bit compare will be used to detect EOS.
GPIBMODEEOS7BIT	0	Specifies that a seven bit compare will be used to detect EOS.
GPIBMODEEOSBYTEMASK	0	This command is the mask for specifying the EOS byte. Use as follows: GPIBMODEEOSBYTEMASK OR &H00YY0000; /* YY = hex byte code */
GPIBTIMEOUTOFF	0	Sets the interface timeout for current device to infinity.
GPIBTIMEOUT10US **GPIBTIMEOUT30US** **GPIBTIMEOUT100US** **GPIBTIMEOUT300US** **GPIBTIMEOUT1MS** **GPIBTIMEOUT3MS** **GPIBTIMEOUT10MS** **GPIBTIMEOUT30MS** **GPIBTIMEOUT100MS** **GPIBTIMEOUT300MS** **GPIBTIMEOUT1S** **GPIBTIMEOUT3S** **GPIBTIMEOUT10S** **GPIBTIMEOUT30S** **GPIBTIMEOUT100S** **GPIBTIMEOUT300S** **GPIBTIMEOUT1000S**	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Sets the timeout for current device to the time (approx) specified.
GPIBCOMMANDMASK	0	MASK is used to specify the low level GPIB message. Use as follows: GPIBCOMMANDMASK OR &H00YY0000; /* YY = low level interface message */

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1 GPIBRDG Control Commands (Cont)

Command	Return Value	Description
		This command is a pre-coded version of the GPIBCOMMANDMASK that sends the message specified as follows:
GPIBCOMMANDGTL	0	Go To Local
GPIBCOMMANDSDC	0	Selected Device Clear
GPIBCOMMANDPPC	0	Parallel Poll Configure
GPIBCOMMANDGET	0	Group Execute Trigger
GPIBCOMMANDTCT	0	Take ConTrol
GPIBCOMMANDLLO	0	Local LockOut
GPIBCOMMANDDCL	0	Device CLear
GPIBCOMMANDPPU	0	Parallel Poll Unconfigure
GPIBCOMMANDSPE	0	Serial Poll Enable
GPIBCOMMANDSPD	0	Serial Poll Disable
GPIBCOMMANDUNT	0	UNTalk
GPIBCOMMANDUNL	0	UNListen
GPIBCOMMANDLA0	0	Listen primary Address 0
GPIBCOMMANDTA0	0	Talk primary Address 0
GPIBCOMMANDSA0	0	Secondary Address 0
GPIBCONTROLIFC	0	Pulses the Interface Clear bus control signal.
GPIBCONTROLDMAON	0	Turns DMA on.
GPIBCONTROLDMAOFF	0	Turns DMA off.
GPIBRENON	0	Asserts the Remote Enable bus control line. The REN line tells all bus connected devices that they are under computer control, as opposed to front panel control. (Use GPIBFUNCTIONGOTOLOCAL to obtain temporary front panel control.)
GPIBRENOFF	0	Unasserts the Remote Enable bus control line. When the REN line is unasserted, front panel control is allowed in the device.
GPIBSTATUS	Interface status	Returns the status of the GPIB interface. For more information on the status word assignments, see Table A-2.
GPIBSTATUSSRQ	0 = SRQ unasserted, 1 = SRQ asserted	Returns the status of the GPIB interface signal called SRQ. The SRQ is an interrupt signal controlled by the instruments to tell the controller they need attention. The SRQ is usually unasserted when the device asserting SRQ has been serially polled.
GPIBSTATUSEND	0 = not at end, 1 = at end of message	Returns a value indicating whether the end of the message has been reached.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1 GPIBRDG Control Commands (Cont)

Command	Return Value	Description
GPIBSTATUSERROR	Error	Returns the last interface error. This must be used in conjunction with the GPIBSTATUS command which indicates validity. For more information, refer to Table A-3.
GPIBSTATUSCOUNT	Number-of-bytes-transferred	Returns the count in number-of-bytes-transferred, as a result of the last I/O function performed.
GPIBCURRENTTIMEOUT	Timeout code	Returns the current timeout code for the device. Current settings show attributes that may have been changed from the default settings via the control commands.
GPIBCURRENTEOSBYTE	EOS byte code	Returns the current EOS byte for the device. Current settings show attributes that may have been changed from the default settings via the control commands.
GPIBCURRENTEOSMODE	EOS mode code	Returns the current EOS mode for the device. Current settings show attributes that may have been changed from the default settings via the control commands. Mode code: Bit 0 = (1) terminate read on EOS Mode code: Bit 1 = (1) Set EOI with EOS on write Mode code: Bit 2 = (1) Eight bit compare (0) seven bit compare
GPIBCURRENTEOIMODE	0 = EOI off 1 = EOI on	Returns the current EOI mode for the device. Current settings show attributes that may have been changed from the default settings via the control commands.
GPIBCURRENTPRIMARY	Primary address	Returns the current primary address for the device. Current settings show attributes that may have been changed from the default settings via the control commands.
GPIBCURRENTSECONDARY	Secondary address	Returns the current secondary address for the device. Current settings show attributes that may have been changed from the default settings via the control commands.
GPIBDEFAULTTIMEOUT	Timeout code	Returns the default timeout code for the device. See Table A-4 for a list of default timeout codes. Default settings come from the GPIB Configurator.
GPIBDEFAULTEOSBYTE	EOS byte code	Returns the default EOS byte for the device.Default settings come from the GPIB Configurator.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1 GPIBRDG Control Commands (Cont)

Command	Return Value	Description
GPIBDEFAULTEOSMODE	EOS mode code	Returns the EOS mode settings for the device. Default settings come from the GPIB Configurator. Mode code: Bit 0 = (1) terminate read on EOS Mode code: Bit 1 = (1) Set EOI with EOS on write Mode code: Bit 2 = (1) Eight bit compare (0) seven bit compare
GPIBDEFAULTEOIMODE	0 = EOI off 1 = EOI on	Returns the default EOI mode for the device. Default settings come from the GPIB Configurator.
GPIBDEFAULTPRIMARY	Primary address	Returns the default primary address for the device. Default settings come from the GPIB Configurator.
GPIBDEFAULTSECONDARY	Secondary address	Returns the default secondary address for the device. Default settings come from the GPIB Configurator.

Table A-2 Status Word Layout

Bit Position	Hex Value	Description
15	8000	GPIB error.
14	4000	Time limit exceeded.
13	2000	GPIB-PC detected END or EOS.
12	1000	SRQ on.
11	800	Device requesting service.
8	100	I/O completed.
7	80	GPIB-PC is in lockout state.
6	40	GPIB-PC is in remote state.
5	20	GPIB-PC is controller-in-charge.
4	10	Attention is asserted.
3	8	GPIB-PC is talker.
2	4	GPIB-PC is listener.
1	2	GPIB-PC is in device trigger state.
0	1	GPIB-PC is in device clear state.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-3 GPIB Error Codes

Decimal Value	Explanation
0	DOS error.
1	Function requires GPIB-PC to be controller-in-charge.
2	No listener on write function.
3	GPIB-PC not addressed correctly.
4	Invalid argument to function call.
5	GPIB-PC not system controller as required.
6	I/O operation aborted.
7	Non-existent GPIB-PC board.
10	I/O started before previous operation completed.
11	No capability for operation.
12	File system error.
14	Command error during device call.
15	Serial poll status byte lost.
16	SRQ locked in ON position.

Table A-4 GPIB Timeout Codes

Actual Value	Minimum Timeout
0	disabled
1	10 μs
2	30 μs
3	100 μs
4	300 μs
5	1 ms
6	3 ms
7	10 ms
8	30 ms
9	100 ms
10	300 ms
11	1 s
12	3 s
13	10 s
14	30 s
15	100 s
16	300 s
17	1000 s

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

VXIbus Bridge Control Commands

NOTE

VXIbus communications can be changed between Word Serial protocol and Shared-Memory protocol with the IOUPGRADE and IODOWNGRADE commands.

VXIbus Word Serial Protocol

Table A-5 shows the I/O control commands for the VXIbus Word Serial protocol.

Table A-5 VXIbus Word Serial Control Commands

Command	Return Value	Description
TIMEOUT100MS **TIMEOUT1S** **TIMEOUT10S** **TIMEOUT100S** **TIMEOUT1000S**	No response	Sets timeout value to time specified.
BRIDGEVERSION	Lower byte = minor version, upper byte = major version	Returns the software version number.
DEVICECLEAR	No response	Sends Word Serial Clear command to device.
DEVICETRIGGER	No response	Sends Word Serial Trigger command to device.
DEVICESTATUS	Device status byte	Sends Word Serial Read STB command to device.
WSCOMMAND	Depends on command sent	Sends Word Serial command to device. The first word of the argument contains the actual Word Serial command.
LWSCOMMAND	Depends on command sent	Sends Long Word Serial command to device. The first long word of the argument contains the actual Long Word Serial command.
ELSCOMMAND	Depends on command sent	Sends Extended Long Word Serial command to device. The first two words of the argument contain the actual Extended Long Word Serial command.
IOUPGRADE	0 = success 1 = failure	Change the VXI communication protocol to Shared Memory protocol.

Since all the VXIbus Word Serial I/O control calls return a 16 bit value, the upper word of the response value should be ignored.

For example, to send the Word Serial command "Abort Normal Operation" to device "dvm1", the appropriate lines of code would be:

```
wscmd% = &HD8FF;
mpctrl%("dvm1", WSCOMMAND, wscmd%, response&);
```

To set the timeout value of device "dvm1" to 1 second, the code would be:

```
mpctrlnr("dvm1", TIMEOUT1S, dummy%());
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Shared-Memory Protocol

Table A-6 shows the five special control services associated with the VXIbus Shared-Memory Protocol.

Table A-6 VXIbus Shared-Memory Protocol Commands

Command	Return Value	Description
IOEOM	No response	Sets a binary switch inside the Shared-Memory bridge. If this switch is set, the bridge will transmit an End-Of-Message token after each bridge write operation. If the switch is cleared, no EOM will be transmitted. This feature allows a 488.2 message unit to be transmitted in several blocks if memory limitations are a consideration. By default, the IOEOM switch is cleared.
IOEOMQRY	Status of the IOEOM switch	Using this function, the programmer can determine if the IOEOM switch is set or cleared, indicating whether or not EOM tokens are being automatically appended by the Shared-Memory bridge.
IOFORMAT	No response	Selects the 32-bit value which will be placed in the Shared-Memory Protocol buffer header format fields. The default value used by the bridge indicates that 488.2 data is carried in the buffers. (Use of the buffer header format field is still undergoing standardization by the Consortium; this control service is intended to allow access to the format field should this become necessary as Shared-Memory Protocol evolves.)
IOTIMEOUT	No response	Specifies how long, in milliseconds, the bridge should wait to complete an I/O transaction. The bridge defaults to 2 seconds.
IOCLEAR	No response	Executes the Shared-Memory Protocol's channel clear protocol—used to clear a Shared-Memory channel that has become overloaded with data. (In normal operation, this control service isn't used.)
IODOWNGRADE	0 = success 1 = failure	Change the VXI communication protocol to Word Serial protocol.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix B
Writing BusManager Applications

Introduction

This section describes how to write drivers using the BusManager.

The Microsoft® BASIC and QuickBASIC TekVICS interface provides access to all BusManager functions. The reader is expected to have some experience writing drivers, and some familiarity with the Microsoft® BASIC version 6.0 (or later) compiler and linker, and (where necessary) QuickBASIC.

Two categories of VMEbus/VXIbus device drivers are:

1) Program-dedicated drivers. These are drivers that are a part of the program. Typically these drivers are a set of functions that are called by the main part of the code to perform various operations. Most devices fall into this category.

2) Resident drivers. These are drivers that are loaded at boot time. A resident driver is usually built as a DOS driver and specified for loading in **\config.sys**. A resident driver can also be built as a TSR (terminate-and-stay-resident program) and specified for loading in **\autoexec.bat**.

Program-dedicated drivers have a totally flexible applications interface—arbitrary calls may be easily added. They are also straightforward to implement; however, control of a device is limited to when that program is running.

Resident drivers can make a device accessible to all programs by designating it to be a DOS device, or by defining a service accessible via an interrupt. Writing a resident driver is much more complex than creating program-dedicated drivers. Resident drivers are typically implemented at least partially in assembly language, and often require the creation of an interface library to access device services from higher-level languages.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Using the BusManager Interface

The Microsoft® BASIC TekVICS interface consists of the **\tekvics\lib\vcmsb.lib** library, and these BASIC include files: **\vcmsb.bi, \busmgr.bi, \gpibrdg.bi,** **\vxicmd.bi,** and **\bdgioctl.bi**.

To use TekVICS functions in a driver, include the appropriate TekVICS header files in the modules in which the functions are used, and link the driver object files with the **vcmsb.lib** library. The header files contain the function and subroutine prototypes for the TekVICS interface, and define TekVICS constants. See Section 2 for more details on the contents of each header file.

NOTE

The TekVICS Microsoft® BASIC interface limits the length of strings to 512 characters.

The Bus Window

The primary means of accessing a device is via its control and status registers. These registers are addressable locations, usually in VMEbus A16 space. They are accessed through the VMEbus window. The VMEbus window is a 64KB region of memory within the first megabyte of memory on the VX5530/VX5535 (and therefore easily accessible) which can be mapped to arbitrary sections of the A16, A24, or A32 address spaces on 64KB boundaries. The bus window is only a *master* to the VMEbus—it has no slave address itself and cannot be used as the destination of accesses originated on other boards. This means, for instance, that you can't have a VMEbus device DMA into the bus window.

The mapping of the bus window into the VMEbus address space is controlled by the BusManager, which is installed at boot time and provides the necessary services to use the bus window. The main BusManager functions with regard to the bus window are:

- **MAP BUS.** Set the mapping of the bus window into VMEbus space
- **SET ACCESS MODE.** Set the address modifier (e.g., A16, A24, A32) and the *byte ordering*. The byte ordering may be either Intel-style (aka "little endian") or Motorola-style (aka "big endian").
- **SAVE STATE.** Quickly stash the internal representation of the bus window mapping, address modifier, and byte ordering (collectively known as the *state*) in a location specified by the program.
- **RESTORE STATE.** Quickly restore a previous state from the internal representation saved via SAVE STATE.

Since there may be several drivers accessing the bus window, each mapping it to a different location, care must be taken to save and restore the state used by your driver.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Interrupts

It is often desirable to use the VMEbus interrupts generated by a device to control its operation. There are 7 VMEbus interrupts, prioritized with 7 as the highest priority and 1 as the lowest priority. While it is possible for several devices to signal via the same interrupt, all the drivers responding to the interrupt must be in the same processor and coordinate amongst themselves—each VMEbus interrupt must be handled by exactly one processor.

When the BusManager detects an interrupt for which it is enabled, it issues a 16-bit Interrupt ACKnowledge (IACK) cycle on the VMEbus bus and gets back an 8- or 16-bit Status/ID response from the interrupting device. The BusManager can't tell the actual size of the response. It is up to the driver to know how many significant bits there are in the response. This Status/ID information is made available to the driver.

The primary BusManager functions for dealing with interrupts include:

- **WAIT FOR INTERRUPT.** Wait until one of a set of interrupts occurs or until the timer expires.
- **DISABLE VMEbus INTERRUPT.** Mask off an interrupt.
- **ENABLE VMEbus INTERRUPT.** Unmask an interrupt.

Polling for an interrupt is the easiest way to deal with device interrupts. The polling mechanism provided waits for one of a set of interrupts to occur, or for a timer to expire.

Polling Interrupts

The easiest way to deal with device interrupts is to use the WAIT FOR INTERRUPT function. The function waits for one of a set of interrupts to occur, or for a timer to expire.

The timer used for this and all other timed BusManager operations is the regular DOS 18.2 Hz clock. All functions that use the timer enable DOS interrupts internally to allow the clock to run.

Disabling and enabling an interrupt (via DISABLE VMEbus INTERRUPT and ENABLE VMEbus INTERRUPT) have no effect on WAIT FOR INTERRUPT. An "awaited" disabled interrupt will be detected as if it were enabled.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix C
BusManager QuickBASIC Applications

Using the Microsoft® QuickBASIC TekVICS Interface

The quick library **\tekvics\lib\vcmsb.qlb** is provided to be used when developing programs in the QuickBASIC programming environment, along with the BASIC include files. To use the quick library, invoke the QuickBASIC programming environment from DOS with a command line of the form:

```
QB myprog /l\tekvics\lib\vcmsb.qlb
```

which loads the QuickBASIC TekVICS functions.

The TekVICS definition modules found in the BASIC include file **vcmsb.bi** must be included by each QuickBASIC application that accesses TekVICS in order to properly access the library functions. The `$INCLUDE` metacommand is used to include the information in the specified file(s) into your application program. The following line must be inserted in a program to include TekVICS definitions, and must appear in the file before all TekVICS calls.

```
'$INCLUDE: '\tekvics\include\vcmsb.bi'
```

You may also need to insert the **busmgr.bi** include file in your program, if you intend to use BusManager constants defined only in that file:

```
'$INCLUDE: '\tekvics\include\busmgr.bi'
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vcwsrcvstr vcwsrcvstr

Purpose: Receives a series of bytes via the word serial BYTE REQUEST command.

Syntax:

```
DECLARE FUNCTION vcwsrcvstr%(ula%, msg$, bytecnt%,
                                        ticks%)
ok% = vcwsrcvstr%(ula%, msg$, bytecnt%, ticks%)
```

Where

`ula`	servant's unique logical address
`msg`	message buffer
`bytecnt`	number of bytes received
`ticks`	timeout in 18.2 Hz clock ticks

Description: BYTE REQUEST commands are sent only when the device's DOR (Data Output Ready) and WRDY bits are set.

The MS BASIC interface uses a fixed internal buffer of 512 bytes to construct strings, so received messages are limited to that size.

This function will terminate successfully when a byte with the END bit set is received. It will also terminate when the buffer is full, when a timeout occurs, when a VXIbus error occurs, or when a Word Serial Protocol error is detected.

Return Value: Returns `VCSUCCESS` if successful or a negative number on failure.

Errors: Returns one of the following:

`ERRTIMEOUT`	TimeOut
`ERRBERR`	Bus Error
`ERRWS`	Word Serial Protocol Error
`ERRBUFFERFULL`	Buffer Full
`ERRRTIMEOUT`	TimeOut on Receive
`ERRRBERR`	Bus Error on Receive

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

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

Gold-standard solutions

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

We buy equipment

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

Learn more!

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

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

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

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

ARTISAN® TECHNOLOGY GROUP