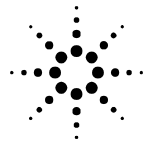




HP E1438A
100 MSample/second
ADC with Filters and FIFO

User's Guide



Agilent Technologies

Innovating the HP Way

HP Part Number E1438-90000

Printed in U.S.A.

Print Date: November 1999, First Edition

© Agilent Technologies, Inc. All rights reserved.
8600 Soper Hill Road, Everett, Washington 98205-1209 U.S.A.

Notices

The information contained in this manual is subject to change without notice.

Agilent Technologies makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Agilent Technologies shall not be liable for errors contained herein or direct, indirect, special, incidental, or consequential damages in connection with the furnishing, performance, or use of the material.

TRADEMARKS

UNIX[®] is a registered trademark in the United States and other countries.

Windows[®], MS Windows[®], Windows NT[®] are U.S. registered trademarks of Microsoft Corporation.

WARRANTY

A copy of the specific warranty terms applicable to your Agilent Technologies product and replacement parts can be obtained from your local Sales and Service Office.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Agilent Technologies, Inc.. This information contained in this document is subject to change without notice.

Use of this manual and CD-ROM supplied for this pack is restricted to this product only. Additional copies of the programs can be made for security and back-up purposes only.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013

Agilent Technologies, Inc.
395 Page Mill Road
Palo Alto, CA
94303-0870 USA

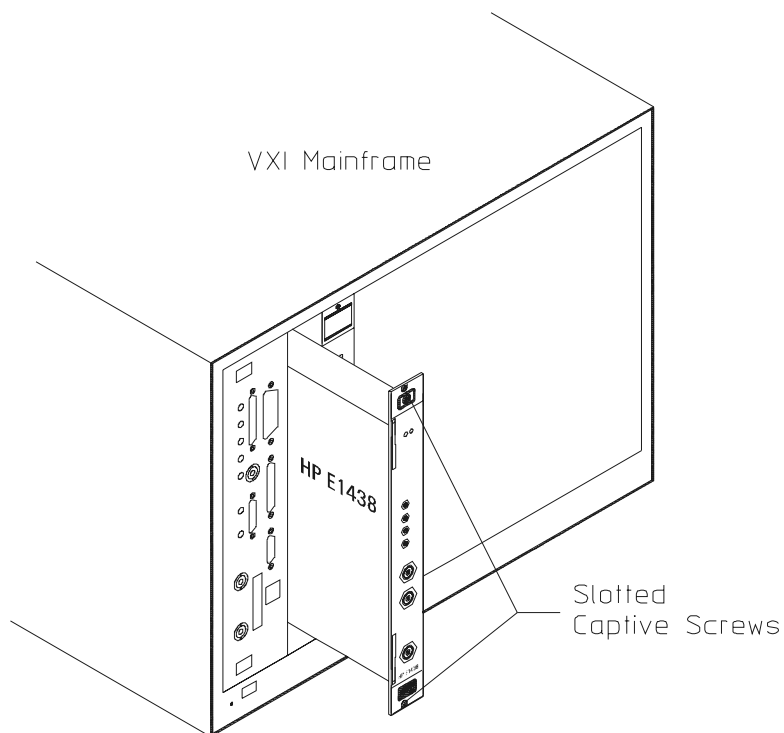
Rights for non-DOD U.S. Government Departments and Agencies are set forth in FAR 52.227-19(c)(1,2).

A Note about the Hewlett-Packard to Agilent Technologies Transition

This product, the HP E1438A, is being introduced at the time of transition from Hewlett-Packard to Agilent Technologies. While the name of the product and external aspects of the module reflect Hewlett-Packard, we have presented all the libraries with the *VXIplug&play* designation for Agilent Technologies. This means that you may develop any programs and not need to alter them for subsequent upgrades or for new, similar products which will be released as Agilent Technologies products.

The HP E1438A at a Glance

The HP E1438A 100 Msample/second Analog-to-Digital Converter with Filtering and Memory provides high precision digitizing for time and frequency domain applications along with signal conditioning, filtering, and memory. The module plugs into a single C-size slot in a VXI mainframe.



Number of Channels	1
Type of Input	50 ohm
Input Bandwidth	150 MHz, 40 MHz alias protected
Sample Rate	100 Msample/sec
Input Range	-21 to +27 dBm
Raw ADC resolution	12 bits
VXI Bus Support	VME and Local Bus
VXI Device Type	Register based
Size	C-sized, single slot

What You Get With the HP E1438A

The following items are included with your HP E1438A

Hardware:

- HP E1438A ADC, C-size VXI module
- CD-ROM for Windows and HP-UX setup

Software

- CD-ROM for installation

A Windows setup program which installs:

- The HP E1438A *VXIplug&play* libraries and drivers
- The HP E1438A HP-VEE header files and WinHelp
- Soft Front Panel program for the HP E1438A with source files, for Windows only
- Web-based help for the HP E1438A
- AGDSP function library and online help
- Example programs and source files
- Microsoft Visual C++ C-library and source files
- Microsoft Visual Basic header files

A depot file for HP-UX installation:

- Libraries and drivers
- Web-based help for the HP E1438A
- AGDSP function library and online help
- Example programs and source files

Documentation

- HP E1438A Installation and Service Guide
- Online documentation available after software installation:
 - HP E1438A User's Guide in PDF format (this document)
 - Web-based help files providing operational information and programmer's reference
 - WinHelp files for the HP E1438A Soft Front Panel
 - WinHelp for VEE

In This Book

This book documents the HP E1438A module. It provides

- hardware installation information
- software installation information
- getting started information
- operational information
- programmer's reference
- replaceable parts

Other Documentation

Installation and Service information is provided as a printed document as well as in this PDF document.

After running the setup program the following documentation is available:

- Web-based help files are available from the Start menu.
- Winhelp for the Soft Front Panel is available from the application.

Contents

1	Installing the HP E1438A	
	To inspect the HP E1438A - - - - -	2
	To install the HP E1438A- - - - -	3
	To store the module- - - - -	6
	To transport the module - - - - -	6
2	Getting Started with the HP E1438A	
	Introduction - - - - -	8
	System Requirements - - - - -	9
	To install the Windows <i>VXIplug&play</i> drivers for the HP E1438A in Windows - - - - -	10
	To install the HP-UX C-language drivers for the HP E1438A in HP-UX systems: - - - - -	11
	To use the Resource Manager - - - - -	12
	To use the program group (Windows) - - - - -	13
	To use the <i>VXIplug&play</i> Soft Front Panel (SPF)- - - - -	14
	To use the HP-UX libraries - - - - -	15
	To use the example programs - - - - -	16
3	Using the HP E1438A	
	HP E1438A overview - - - - -	20
	Programming the HP E1438A- - - - -	21
	The measurement loop- - - - -	23
	Frequency and filtering - - - - -	26
	Using clock and sync - - - - -	27
	Managing multiple modules - - - - -	28
	Transferring data- - - - -	34
4	HP E1438A Programmer's Reference	
	Introduction - - - - -	36

Functions listed by class	37
Functions listed by functional group	41
Functions listed alphabetically	47
Equivalent numeric values for variables	145
Commands which halt active measurements	149
Error messages	150
Default values	152
VXI<i>plug&play</i> Syntax Quick Reference	154

5 Module Description

Front Panel Description	160
VXI backplane connections	161
Block diagram and description	163

6 Replacing Assemblies

Replaceable parts	170
--------------------------------	------------

Technical Specifications

Glossary

INDEX

To inspect the HP E1438A

The HP E1438A single channel VXI ADC Module was carefully inspected both mechanically and electrically before shipment. It should be free of marks or scratches and it should meet its published specifications upon receipt.

If the module was damaged in transit, do the following:

- Save all packing materials.
- File a claim with the carrier
- Call your Hewlett-Packard sales and service office.

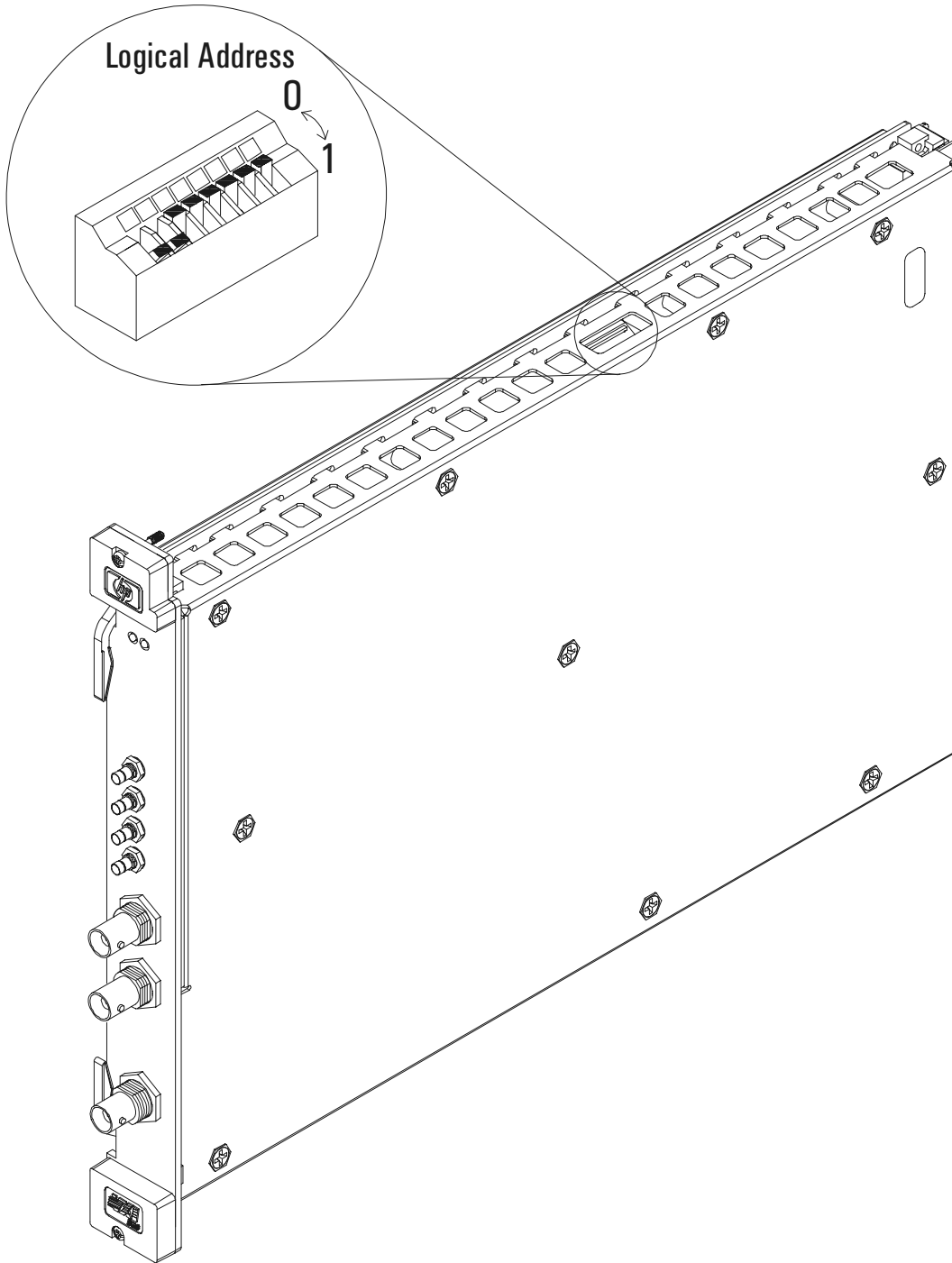
To install the HP E1438A

Caution

To protect circuits from static discharge, observe anti-static techniques whenever handling the HP E1438A VXI ADC Module

1. Set up your VXI mainframe. See the installation guide for your mainframe.
2. Select a slot in the VXI mainframe for the HP E1438A module.
The HP E1438A module's local bus receives ECL-level data from the module immediately to its left and outputs ECL-level data to the module immediately to its right. Every module using the local bus is keyed to prevent two modules from fitting next to each other unless they are compatible. If you will be using the local bus, select adjacent slots immediately to the left of the data-receiving module. If the VXI bus is used, maximum data rates will be reduced but the module can be placed in any available slot.
3. Using a small screwdriver or similar tool, set the logical address configuration switch on the HP E1438A. (See the illustration on the next page.) Each module in the system must have a unique logical address. The factory default setting is 1100 0000 (192).

Installing the HP E1438A
To install the HP E1438A

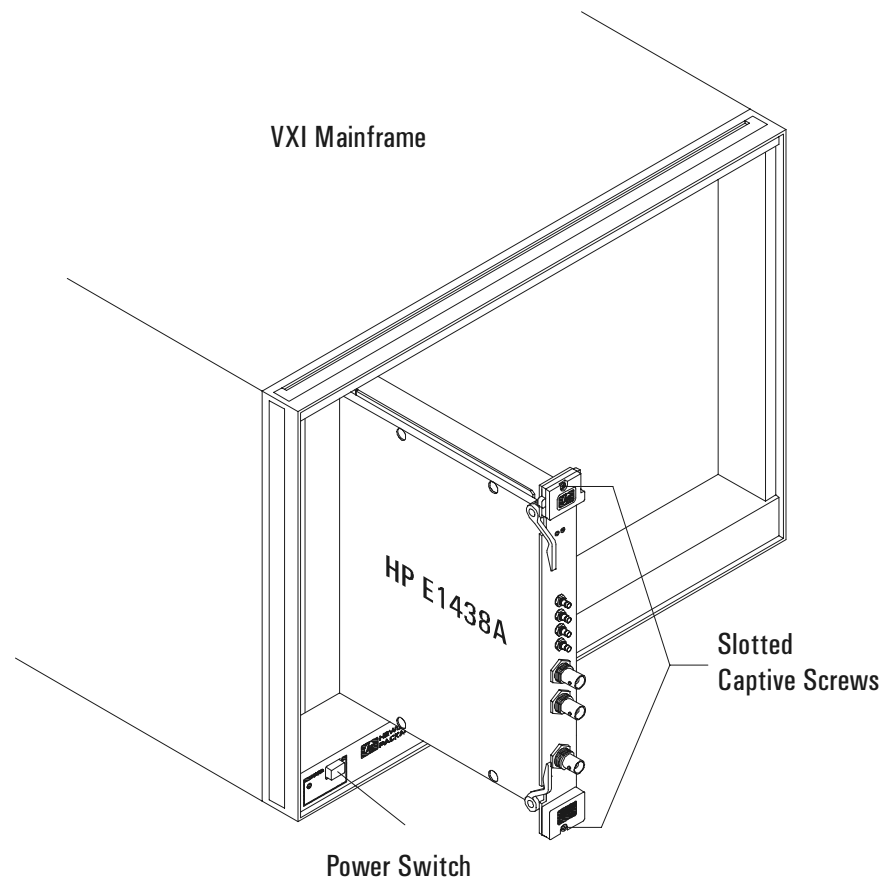


4. Set the mainframe's power switch to off (0).

Caution

Installing or removing the module with power on may damage components in the module.

5. Place the module's card edges (top and bottom) into the module guides in the slot.
6. Slide the module into the mainframe until the module connects firmly with the backplane connectors. Make sure the module slides in straight and that the insertion/extraction levers are pressed parallel to the front panel.
7. Attach the module's front panel to the mainframe chassis using the module's captive mounting screws.



To store the module

Store the module in a clean, dry, and static free environment.

For other requirements, see storage and transport restriction in “Technical Specifications”.

To transport the module

- Package the module using the original factory packaging or packaging identical to the factory packaging.
- If returning the module to Hewlett-Packard for service, attach a tag describing the following:
 - Type of service required
 - Return address
 - Model number
 - Full serial number

In any correspondence, refer to the module by model number and full serial number.

- Mark the container FRAGILE to ensure careful handling.
- If necessary to package the module in a container other than original packaging, observe the following (use of other packaging is not recommended):
 - Wrap the module in heavy paper or anti-static plastic.
 - Protect the front panel with cardboard.
 - Use a double-wall carton made of at least 200-pound test (32 ECT) material.
 - Cushion the module to prevent damage. For example, several layers of plastic bubble wrap is usually sufficient.

Caution

Do not use styrene pellets in any shape as packing material for the module. The pellets do not adequately cushion the module and do not prevent the module from shifting in the carton. In addition, the pellets create static electricity which can damage electronic components.

Introduction

This section helps you get your HP E1438A running and making simple measurements without programming. It shows you how to install the software libraries and how to run the Soft Front Panel program. It also introduces you to the example programs. Two versions of the Host Interface Library are available. One is the Windows Library which communicates with the hardware using VISA (Virtual Instrument Software Architecture). VISA is the input-output standard upon which all the *VXIplug&play* software components are based. The second version is the HP-UX 10.2 C-language Host Interface Library which also uses VISA.

This section assumes you have already installed the module in the VXI mainframe as shown in the previous chapter. It also assumes that you have installed a VXI interface according to the manufacturer's instructions.

Note

Be sure to read the readme file for important up-to-date software installation information.

System Requirements

System Requirements (Microsoft Windows)

- A Pentium-class personal computer:
- Microsoft Windows 95/98, or NT.
- One of the following interfaces:
 - HP FireWire – HP E8491B IEEE-1394 PC Link to VXI
 - National Instruments PCI MXI-2
 - Other VISA compliant VXI interface
- VISA (Virtual Instrument Software Architecture) library
- The computer must have a CD ROM drive for the installation media
- A Web browser

System Requirements (HP-UX)

- One of the following workstations:
 - An HP V743 VXI-embedded workstation
 - A stand-alone HP-UX workstation with a MXI interface
- The workstation must have a CD ROM drive for installation media
- VISA (Virtual Instrument Software Architecture) library
- HP-UX (version 10.2 or later)
- A Web browser

To install the Windows VXIplug&play drivers for the HP E1438A in Windows

This procedure assumes that you have already installed a VISA (Virtual Instrument Software Architecture) library. If not, you can still install these drivers but you will receive an error message reminding you to install the VISA library.

1. Insert the CD labeled: “HP E1438A 100 MSample/sec A-to-D Converter”
2. Run the program: *drive*:\windows\setup.exe
Where *drive* represents the drive containing the setup CD.
3. The setup program asks you to confirm or change the directory path. The default directory path is recommended.
4. A dialog box asks if you want to install startup shortcuts
This creates a program group called “AGE1438” within the Vxipnp directory which includes:
 - A shortcut to run the HP E1438A Soft Front Panel
 - A shortcut for the HP E1438A web-based online help file
 - A shortcut for the PDF version of the *HP E1438A User's Guide*
 - A shortcut for the AGDSP web-based online help file
 - Several shortcuts for example programs
 - A shortcut for a readme file
5. A readme file may be displayed. If so, be sure to read it and follow the instructions.

Note

Future upgrades will be distributed on the Web. To check your current revision run the Info Utility or check Help/About in the Soft Front Panel program.

To check for new revisions access the Agilent Technologies Web page <http://www.agilent.com/> and search for "E1438"

To install the HP-UX C-language drivers for the HP E1438A in HP-UX systems:

The e1438.depot file is in SD-UX format. To install the filesets:

1. Log in as root
2. Insert the CD labeled: "HP E1438A 100 MSample/sec A-to-D Converter" into the CD drive
3. Execute the swinstall (/usr/sbin/swinstall) utility as:
% swinstall
4. Use the interactive menus to install the file:
drive/hpux/e1439.dep
where *drive* represents the drive containing the setup CD.

Be sure to read the README file which contains important information on installation, viewing online help, and compiling example programs.

All files are installed in subdirectories under /opt/vxipnp.

Note

Future upgrades will be distributed on the Web. To check your current revision run the Info Utility.

To check for new revisions access the Agilent Technologies Web page <http://www.agilent.com/> and search for "E1438"

To use the Resource Manager

The Resource Manager is a program from your hardware interface manufacturer. It looks at the VXI mainframe to determine what modules are installed. You need to run it every time you power up. If you get the message: "VISUCCESS_DEVICE_NPRESENT" then run the Resource Manager.

Before running the HP E1438A software make sure that your hardware is configured correctly and that the Resource Manager runs successfully. Before using your measurement system, you must set up all of its devices, including setting their addresses and local bus locations. No two devices can have the same address. Usually addresses 0 and 1 are taken by the Resource Manager and are not available.

For more information about the Resource Manager, see the documentation with your hardware interface.

Note

Most Resource Managers will recognize the manufacturer and model number of the HP E1438A but if your interface requires that you enter this information manually, use the following:

Manufacturer number: 4095 (Hex FFF)

Model number: 622 (Hex 26E)

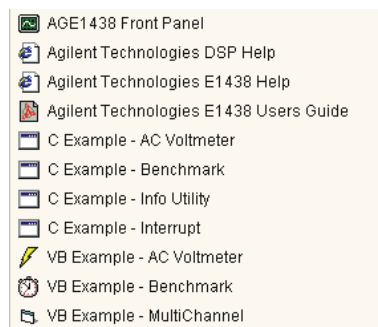
To use the program group (Windows)

If you chose to install the program group by the default method during the installation procedure you have a shortcut for a program group similar the one below. Access it through the Start button:

Programs \ Vxipnp \ age1438


This program group contains shortcuts which access the Soft Front Panel program, the User's Guide, online help, and example programs. The following pages provide an overview of these items.

If you did not choose to install the program group, executable files for each of the items represented by group shortcuts are available in the *drive:\vxipnp* directory and its subdirectories.



To use the VXIplug&play Soft Front Panel (SPF)

The the best place to start to explore the capabilities of the HP E1438A in a Windows environment is with the Soft Front Panel. The Soft Front Panel can be useful for checking your system to make sure that is installed correctly and that all of its parts are working. You can also use it to make actual measurements, since it accesses most of the HP E1438A's functionality.

Select the  AGE1438 Front Panel shortcut in your program group to start the program. This assumes you have already installed all required hardware and drivers (including VISA) and have run the configurator and Resource Manager required by your hardware interface.

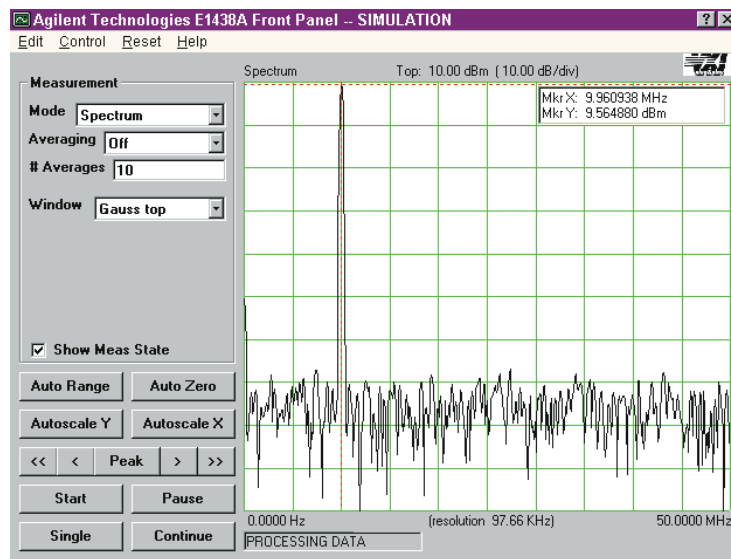
If prompted for the resource descriptor, use the default "VXI::192" unless the logical address of the HP E1438A has been changed from its default setting of 192. If it has been changed then type the appropriate logical address instead of 192. Press OK.

Note

You can also run the HP E1438 Front Panel in a simulation mode without an HP E1438A module, a hardware interface, or VISA libraries by typing "sim" in place of the resource descriptor.

The HP E1438 Front Panel Help, available from the Soft Front Panel Help menu, describes the capability of the Soft Front Panel and has links to functions which control and define many of the parameters.

The source files for this program are provided for you to use as sample code.



To use the HP-UX libraries

The README file is located at:

`/opt/vxipnp/hpux/age1438/README`

It contains the following information:

- overview of the directory structure
- how to access the online help
- how to run the example programs
- how to modify and compile the example programs.

To use the example programs

Several example programs are included to perform useful tasks for you and to serve as a basis for your own programs. When you installed your HP E1438A Windows or HP-UX libraries and drivers using the setup program or utility, you also installed executable and source code files for several useful example programs. The programs demonstrate programming the module with "C", Microsoft Visual Basic, and HP-VEE.

The executables for these examples require a HP E1438A and, for Windows, *VXIplug&play* support; in other words they will not run in simulation mode like the HP E1438A Soft Front Panel program.

shortcuts for the executables appear in the age1438 Windows program group if you chose to add it during setup.

In Windows environments executable files and source code for the Microsoft Visual Basic examples are installed in the *drive:\vxipnp\win[95|NT]\age1438\vb* directory. The VEE examples are in the *...\age1438\vee* directory, and "C" examples are in the *...\age1438\msc\examples* directory.

In the HP-UX environment executable files and source code for the C-language examples are installed in */opt/vxipnp/hpux/age1438*.

The group of programs described here may be supplemented with additional programs later which will be described in the online help or readme file.

acvolts_32.exe

This is about the simplest practical complete program using the HP E1438A and functions like an AC voltmeter. It is written in Visual Basic.

acvolts.exe

This is a console version of *acvolts_32.exe*, written in Microsoft Visual C++.

Benchmark_32.exe

This performance benchmark program is really more of a utility than an example, although source code is provided. It allows users to measure data transfer rates and command processing times on their system without having to write new code. The utility is written in Visual Basic.

bench.exe

This is a console version of *Benchmark_32.exe*, written in Microsoft Visual C++.

Multchan_32.exe

To use the example programs

This example shows how to synchronize two modules to achieve simultaneous sampling, filter decimation, and matched local oscillator phase. It is written in Visual Basic.

info.exe

This example shows how to retrieve option and revision information from an HP E1438, and it doubles as a handy utility. It is written as a console program in Microsoft Visual C++.

interrupt.exe

This example shows how to set up and trap a VXI interrupt to indicate an error condition in the HP E1438A. It is written as a console program in Microsoft Visual C++.

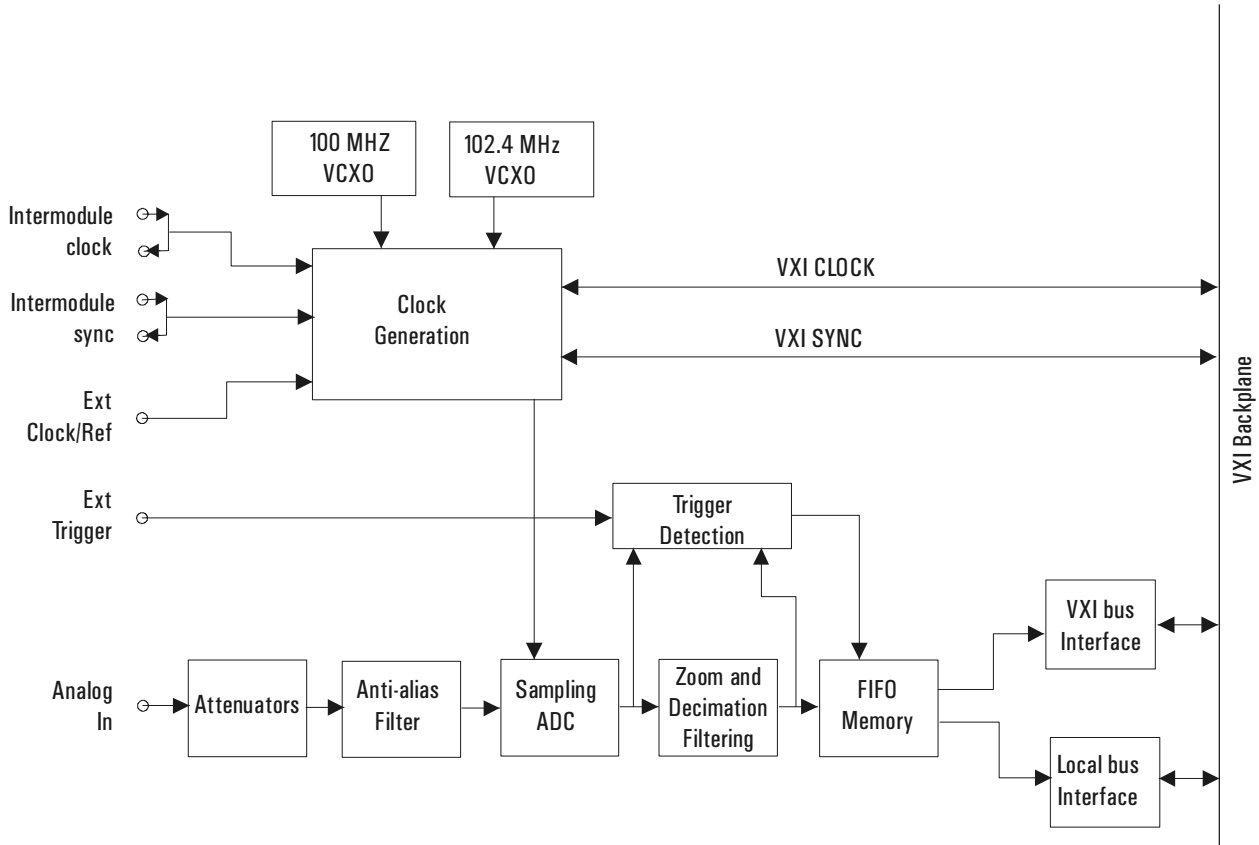
scope.vee

This is a simple one-channel example written in VEE. In order to view or execute it, the VEE programming environment must be installed on the system.

Getting Started with the HP E1438A

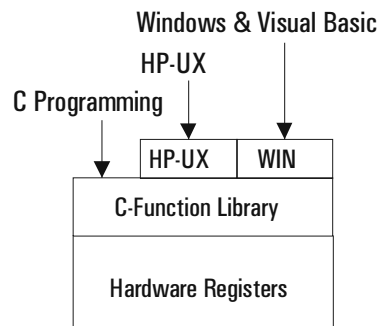
To use the example programs

HP E1438A overview



Programming the HP E1438A

The HP E1438A is shipped with software and documentation to support a broad set of choices of controllers, I/O interfaces, programming languages, and operating systems. By virtue of its compliance to the *VXIplug&play* standard, the HP E1438A is most easily controlled in an environment conforming to one of the supported *VXIplug&play* frameworks. However, support is also supplied for other common hardware and software environments. The relationship among the various levels of programming the HP E1438A is shown in the diagram below.



Windows framework

The primary development environment supported by the HP E1438A is the *VXIplug&play* WIN95/98, and WinNT framework specifications. It requires the following resources prior to the installation of the HP E1438A:

- An embedded or a stand-alone Pentium-class PC
- Microsoft Windows 95/98 or NT
- VISA interface library
- VISA compatible hardware interface
- Microsoft Visual C++ and/or Microsoft Visual Basic development system.

Additional details on the WIN framework can be found in the *VXIplug&play VPP-2 System Frameworks Specification, Revision 2.0*.

In addition to the C source code files, the HP E1438A includes compiled libraries, example programs, an interactive soft front panel program, online help files, and an installation program. The interactive soft front panel program allows the HP E1438A to be turned on, verified and used for simple tasks without writing any user programs.

Programming the HP E1438A

Compliance with the *VXIplug&play* WIN framework allows users of the HP-VEE graphical programming system to control the HP E1438A from that environment. This is accomplished by using the capability of HP-VEE to call functions in the C-library. Documentation and support for that capability is included with HP-VEE and is not addressed further in this document.

HP-UX, Series 700 environment

The HP-UX environment is supported for developers who prefer programming tools provided on the UNIX operating system. The system requirements include:

- HP series 700 workstation
- HP-UX operating system 10.2
- MXI interface
- C-language programming system.

In addition to the source code files, the HP E1438A includes compiled libraries, example programs, online help files, and an installation utility.

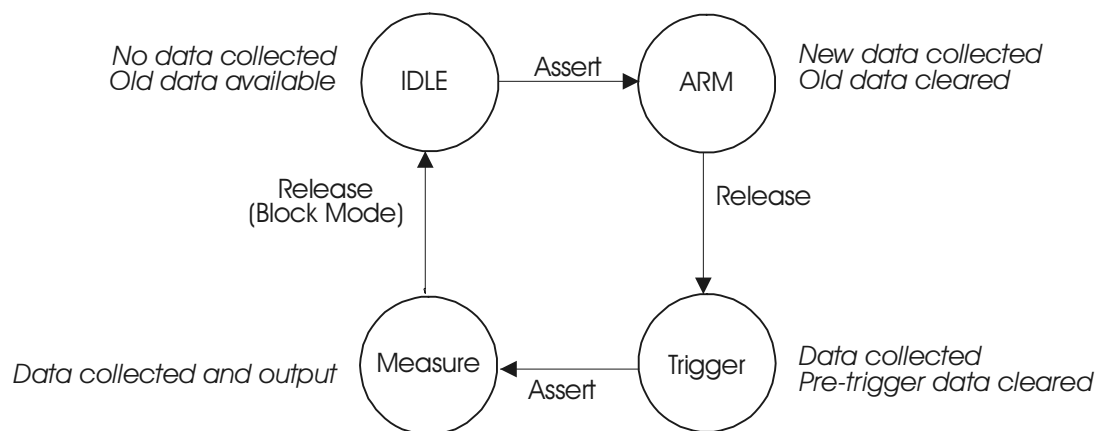
C programming

The HP E1438A is shipped with a source library of C-functions which can be called from user programs. This elevates the interface above the register level so the programmer does not have to be concerned with such things as register addresses and packing or splitting parameters into 16-bit register lengths. The library includes ANSI compliant source code files with all machine dependent code constrained to a single source file. By re-writing selected portions of the *machine.h* file, the programmer can create and compile an HP E1438A library which is compatible with virtually any development environment using the C language. The most common reason for re-writing *machine.h* is to accommodate I/O libraries other than VISA. In some cases the library may need merely to be re-compiled to target a different processor type for the host computer.

Porting the HP E1438A library to a different computer environment is likely to be a fairly straight forward task. However, some of the higher level tools shipped with the HP E1438A may not be as easily ported. The interactive soft front panel and some example programs include human interfaces which depend on certain display and keyboard support which may be system dependent. Although source code is included for these applications, porting them to a different environment may present a greater problem than porting the library itself. The installation utilities are specifically targeted to operate on the supported development environments and may not be available in other environments.

The measurement loop

The measurement loop progresses through four states. The transition from one state to the next is tied to the transition of the Sync signal. The effect of the Sync signal is summarized in the following diagram representing the four possible states of an HP E1438A module.



In the *Idle* state the HP E1438A places no new data into the FIFO output buffer memory although previously measured data is retained in the buffer memory and is available for output via the VME or local bus I/O ports. The module stays in the Idle state until the Sync line is asserted.

Upon entering the *Arm* state the HP E1438A clears old data. It remains in the Arm state until the Sync signal is released. If an HP E1438A is programmed with a pre-trigger delay, it collects enough data samples to satisfy this pre-trigger delay, and then releases the Sync line. If no pre-trigger delay has been programmed, the module releases the Sync line immediately. When all HP E1438As in a system have released the Sync line the module moves to the Trigger state.

Upon entering the *Trigger* state an HP E1438A, if it is programmed with a pre-trigger delay, continues collecting data into the FIFO, discarding any data prior to the pre-trigger delay. An HP E1438A remains in the Trigger state until the Sync line is asserted. The Sync line may be asserted by a direct command or by any HP E1438A which encounters a trigger condition and is programmed to assert the Sync line. When the Sync signal is asserted, all modules synchronously move to the Measure state.

In the *Measure* state the HP E1438A continues collecting data and sends the data saved in the FIFO memory to the selected I/O port, starting with the sample indicated by the trigger arrival, offset by the number of samples specified by the trigger delay. This data

The measurement loop

transfer continues until all data has been transferred or until the module meets the criteria for returning to the Idle state imposed by *block mode* or *continuous mode* operation constraints.

Modules programmed for *block mode* operation assert the Sync line until a complete block of data, including any pre-programmed pre- or post-trigger delay, has been collected and is available to the I/O port. The module then releases the Sync line. The module returns to the Idle state when the block of data has been collected.

In *continuous mode* a module releases sync immediately but moves to the Idle state only if explicitly programmed to do so or if the FIFO data buffer overflows because data cannot be read from the I/O port fast enough.

The measurement loop in multi-module systems

The following rules generally apply to transitions between states when multiple modules share a Sync signal:

- If any one module *asserts* the Sync line a synchronous state transition occurs for all modules in a system.
- All modules in a system must have *released* the Sync line in order to bring about a synchronous transition to Trigger state.
- In block mode each module releases the Sync line after its block of data has been collected. Immediately upon entering the Measure state in continuous mode each module releases the Sync line. It continues to collect and output data until it is programatically signaled to stop or until the FIFO overflows. With the Sync line released it is then possible to change the center frequency for one or multiple modules without interrupting the measurement. See “Synchronizing changes in multi-module systems” on page 32.
- A module may be programmed explicitly to inhibit its transition to the Arm state despite Sync transitions.
- In addition to controlling the progression through the four module states, the Sync signal is used to synchronize the decimation counters and local oscillators of multiple HP E1438A modules.

Delay and phase in triggered measurements

It is important to note that the trigger delay is specified in terms of output samples. When using the digital filters within the HP E1438A to reduce the sample rate, there are multiple ADC samples corresponding to each output sample. In order to determine the relationship between the first output sample of a block and the actual ADC sample where the trigger occurred, you must read the actual delay from the module using **age1438_trigger_delay_actual_get**.

This relationship varies from block to block, and is a function of the particular value of counters within the digital filters at the time the trigger occurs. Thus the actual delay from the trigger event is the delay from **age1438_trigger_delay_get** multiplied by 2^{sigBw} (from **age1438_filter_bw_get** if filter decimation is used, or $2^{(\text{sigBw}-1)}$ if filter decimation is off). This value is then added to the value returned by **age1438_trigger_delay_actual_get**. The result is in periods of the ADC sample clock. Special considerations apply in multi-module systems. See “Trigger and phase in multi-module systems” on page 33.

The measurement loop

When doing a zoomed measurement, it may also be helpful to know the phase of the digital LO at the point in time when the trigger occurred, since the LO is also running continuously and it will have an arbitrary phase relationship with the trigger event. **age1438_trigger_phase_actual_get** returns the phase of the LO at the trigger point. The LO phase could be used in time domain averaging of blocks, or other operations involving zoomed blocks of data, so that the varying phase of the LO can be removed from the calculation.

Frequency and filtering

The HP E1438A's center frequency is normally set at zero (baseband measurement). However, you may set the center frequency to a non-zero value in order to examine a narrower span away from baseband (zoom measurement). The frequency band of interest, represented by digitized time data samples from the ADC, is mixed with the HP E1438A digital LO, a complex exponential, at the desired center frequency. As a result the frequency band of interest in the input signal is shifted to a complex signal centered around DC. See "Synchronizing changes in multi-module systems" on page 32 for special considerations with respect to changing the center frequency in multi-module systems.

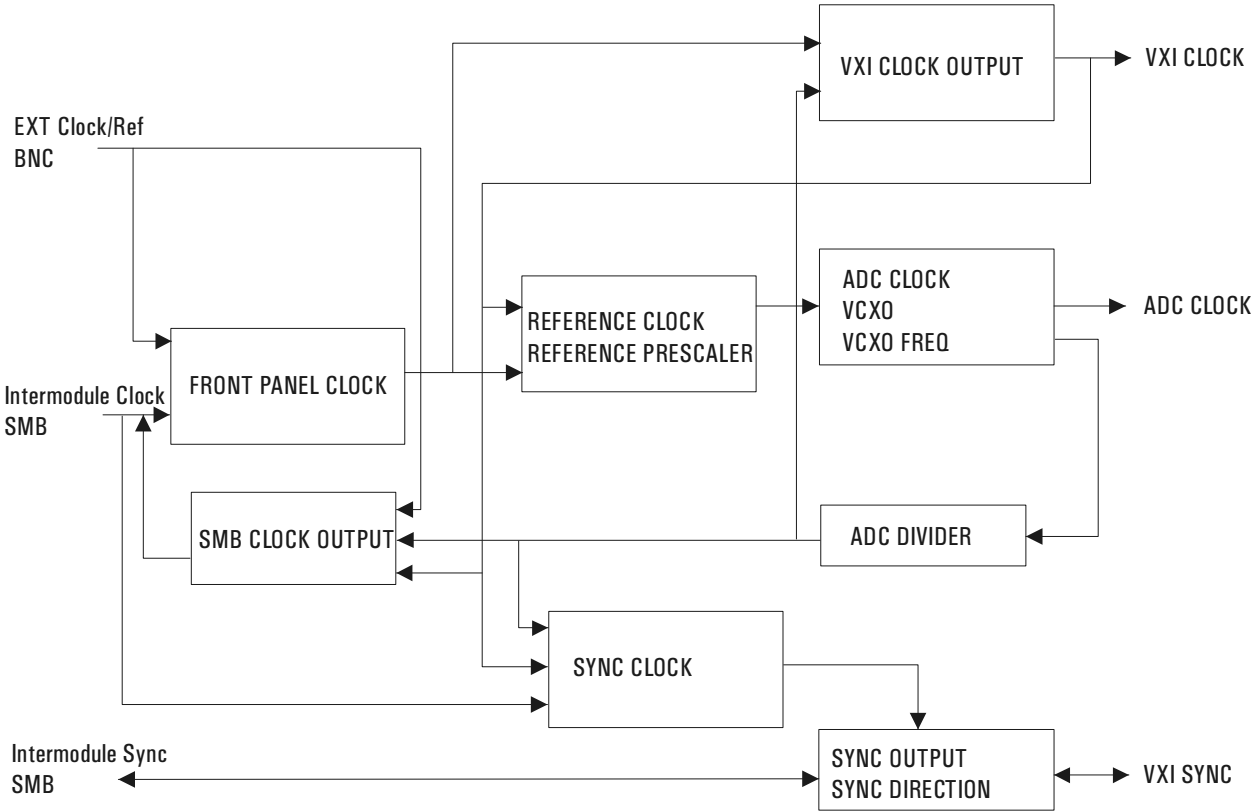
The default filter for HP E1438A measurements is an analog anti-alias filter. However, you may further isolate the frequency band of interest for more detailed analysis by using digital filtering. A decimating digital filter simultaneously decreases the bandwidth of the signal and decreases the sample rate. The built-in digital filters conform to the Nyquist sampling theorem which guarantees that the output sample rate may be reduced by the same factor as the signal bandwidth reduction while still maintaining a complete representation of the underlying bandlimited signal.

For each octave step in bandwidth reduction (except for the first octave) the HP E1438A digital filters automatically reduce the data rate by discarding alternate output samples. This process, called decimation, results in an output sample rate which is nominally four times the signal bandwidth whenever $sigBw > 0$. This is still double the theoretical rate necessary to fully characterize the band limited signal. However, because the digital filters do not have a perfectly abrupt cutoff, the sample rate cannot be reduced to the theoretical limit without some aliasing of signals in the transition frequency band of the filters. In many applications this limited aliasing potential is not important. For this reason you may optionally choose to apply a final factor-of-two decimation. See the Technical Specifications for detailed information on the digital filter shapes.

The decimation process used to reduce the output sample rate is driven from a "decimation counter" which keeps track of which samples to save and which ones to discard for each of the octave bandwidth reduction filter stages. In multi-module systems where synchronous sampling is required, the decimation counters in all the modules must be synchronous with each other. See "Synchronizing changes in multi-module systems" on page 32.

Using clock and sync

The following diagram shows the flow of clock and sync signals:



Managing multiple modules

Sharing Reference and Sync signals in multi-module systems

The HP E1438A supports synchronous operation among multiple HP E1438As by using a shared ADC clock and Sync signal to drive all the modules in a system. The shared Sync signal is used to synchronize critical operations including arming, triggering the beginning of data collection, setting a common phase of the local oscillators for zoom operation, and forcing concurrent output sample times when decimation is used. The Sync line transitions are constrained to not occur during the critical (setup and hold) regions of the external reference. The reference operates at 1/10 of the internal ADC clock, typically 10 or 10.24 MHz for a HP E1438A module. The reference can be either generated within the master module, or an external reference can be fed into the master module through a front panel BNC.

Clock distribution

When shared, the reference clock and sync lines are distributed among modules either on the VXI backplane using the ECL Trigger lines, or on the front panel using the SMB Clock/Ref extender connectors. When VXI backplane distribution is used with more than one VXI mainframe, the front panel Intermodule Clock and Sync connectors can be used to distribute clock and Sync lines from one mainframe to another.

Since the Sync transition timing relative to the reference input is critical, the module driving the Sync line should ideally be the same one identified as the master. However, when using backplane distribution, any HP E1438A in the same mainframe as the master can drive the Sync line.

When using the multi-sync mode of operation, the selection of front panel or backplane distribution of reference and Sync signals involves the following considerations:

- Backplane distribution requires the use of the ECL Trigger lines on the backplane, which are then unavailable to other modules.
- The overall time skew between the arrival of ADC clock edges is smaller when using backplane distribution, particularly if the master (or buffer) module is physically located in the center of the group of HP E1438A modules.
- Backplane distribution is more susceptible to pickup of jitter on the ADC clock from other digital activity on the VXI backplane. The extent of this pickup depends on the mainframe and on the other modules in the mainframe. One important step in reducing this pickup is to disable, whenever possible, the 10 MHz VXI clock generated by the slot-0 controller.
- For backplane distribution make sure that all modules conform to VXI specification 1.4 or later with regard to their attachment to the ECL Trigger lines. See the HP E1438A Technical Specifications for the clock jitter (phase noise) specification degradation using backplane distribution.

Managing multiple modules

- Front panel distribution requires the use of two short, equal length cables with SMB connectors between modules. In addition, unused SMB connectors on modules being used for front panel distribution must be terminated in 50 ohms.

Managing multiple modules

Managing multi-module systems

Note

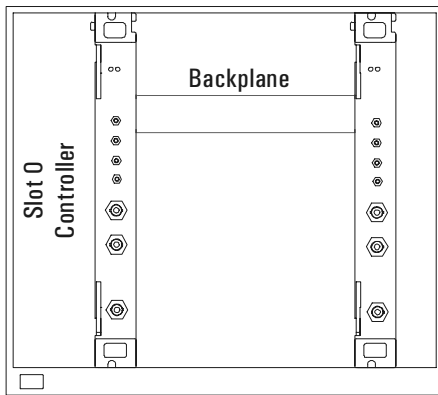
The ⊗ symbol indicates a 50 ohm terminator, which is required on unused SMB connectors in systems using front panel distribution

Module #1 – “Rear master, internal reference” on page 61

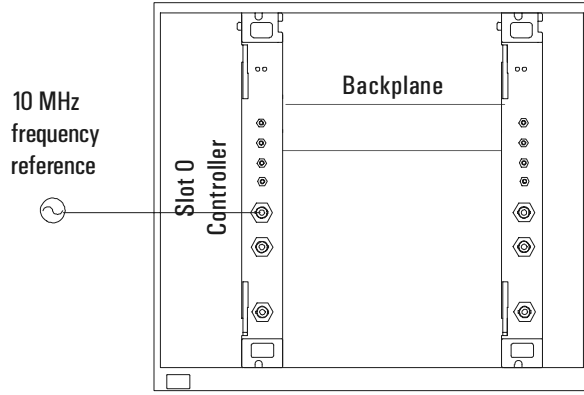
Module #2 – “Rear slave, phase locked to master” on page 62

Module #1 – “Front master, phase locked to external reference” on page 60

Module #2 – “Rear slave, phase locked to master” on page 62



Internal clock and SYNC distribution using VXI backplane ECL trigger lines.



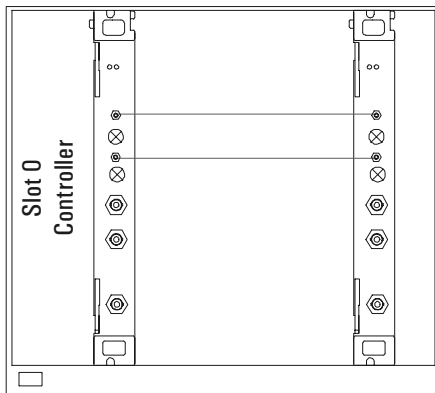
External reference and SYNC distribution using VXI backplane ECL trigger lines.

Module #1 – “Front master, internal reference” on page 59

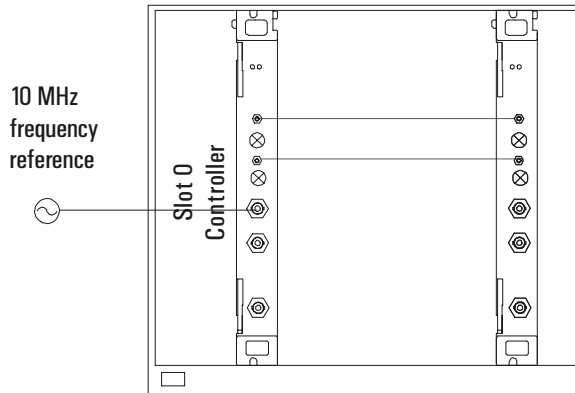
Module #2 – “Front slave, phase locked to master” on page 60

Module #1 – “Front master, phase locked to external reference” on page 60

Module #2 – “Front slave, phase locked to master” on page 60



Internal clock and SYNC distribution using front panel SMB clock and SYNC extender connections.



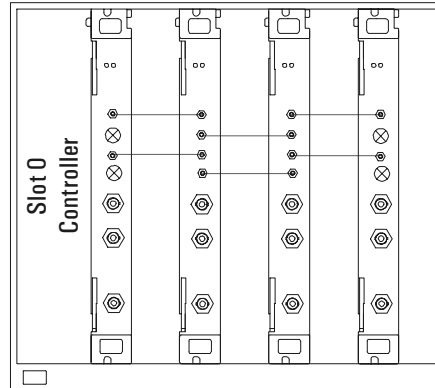
External reference and SYNC distribution using front panel SMB clock and SYNC extender connections.

Module #1 – “Front slave, phase locked to master” on page 60

Module #2 – “Front master, internal reference” on page 59

Module #3 – “Front slave, phase locked to master” on page 60

Module #4 – “Front slave, phase locked to master” on page 60



Sharing clock and SYNC among several modules via front panel distribution.

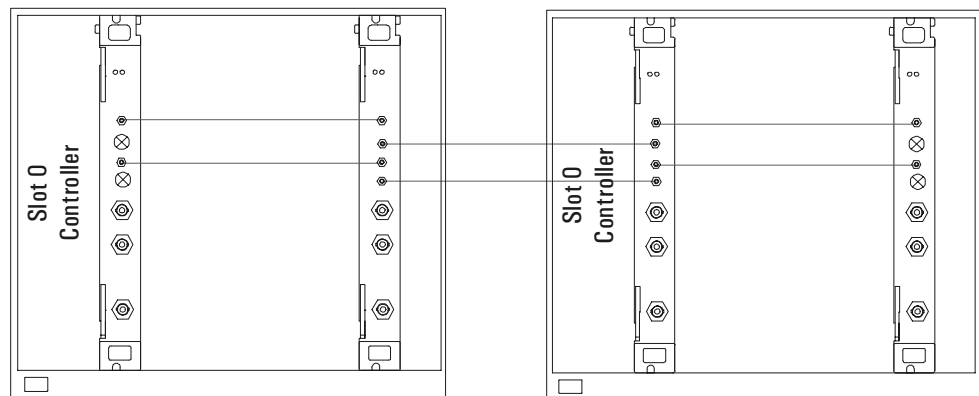
Managing multi-mainframe systems

Module #1 – “Front slave, phase locked to master” on page 60

Module #2 – “Front master, internal reference” on page 59

Module #3 – “Front slave, phase locked to master” on page 60

Module #4 – “Front slave, phase locked to master” on page 60



VXI Mainframe A

VXI Mainframe B

Clock and SYNC distribution using front panel extender connections within and between mainframes.

Managing multiple modules

Synchronizing changes in multi-module systems

Multi-module systems require special treatment with respect to timing of frequency and filter changes. Center frequency changes may involve synchronizing the local oscillators of all modules in a system. Digital filter changes in multi-module systems require that the decimation counters be synchronized.

Calling the following functions voids synchronized multi-module setups:

age1438_clock_setup and low-level clock setup functions
age1438_clock_recover
age1438_input_autozero
age1438_input_range_auto
age1438_self_test
age1438_state_recall

Special considerations apply to the measurement loop. See “The measurement loop in multi-module systems” on page 24.

Synchronous digital filter changes

In multi-module systems where synchronous sampling is required, the decimation counters in all the modules must be synchronous with each other. This condition can be forced by preparing each module in the system in advance. Any measurement in progress is terminated at this time and the module is placed in the Idle state. After each module is prepared, the next sync line transition causes the digital decimation counter to be reset and started at the same time. Once this is done the decimation counters stay synchronized as long as the same ADC clock is used.

If you also intend to change the center frequency along with the digital filters, you should synchronize the digital filters first. Otherwise the center frequency phase becomes unsynchronized when the digital filters are changed.

Synchronous center frequency changes

In multi-module systems you may prepare each module in advance of a frequency change, then perform the change synchronously by asserting the sync line. This preserves the phase relationship of the local oscillators for all modules in the system. Certain special considerations apply to multi-module frequency changes:

- If all modules in a system are in the Idle state when the Sync line transition occurs, the LO frequency is updated and the next measurement is armed.
- If all modules are in the measurement state in continuous mode when the Sync line transition occurs, the LO frequency is synchronously updated, and the measurement continues.
- In continuous mode care must be taken to assure that all modules are in the same state, either the Idle state or the Measure state, before the Sync line transition occurs, otherwise some modules re-arm while others continue the current measurement.
- In block mode the Sync line transition is ignored unless all modules are currently in the Idle state.

- If you also intend to change the digital filters along with the center frequency, you should synchronize the digital filters first. Otherwise the center frequency phase becomes non-synchronized when the digital filters are changed.

Trigger and phase in multi-module systems

When triggering is used in multiple modules, you do not need to measure phase differences between two or more channels *if* the channels are set up *identically* in terms of digital filtering and LO frequency, *and* the digital filters and LOs are correctly synchronized. Since the filters and LOs are synced together, their actual trigger delays and LO phases will be identical and will cancel out of relative phase measurements. Any remaining delay should be less than 10nS between two modules in the same mainframe.

Only the module which generates the trigger has knowledge of the delay between the trigger event and the start of data collection. Therefore, if you need the actual delay from the trigger, you should use the trigger delays from each module and the trigger delay correction from the module that generated the trigger. See “Delay and phase in triggered measurements” on page 24.

Transferring data

You can transfer data from the HP E1438A two different ways:

- The VMEbus is the universal data bus for VXI architecture. It provides flexibility and versatility in transferring data. Transfers over the VMEbus are 16 bits or 32 bits wide.
- The Local Bus supports faster transfer rates than the VMEbus. For example, if you are transferring data from the HP E1438A to the HP E1485A/B, the Local Bus provides a direct pipeline to the HP E1485's DSPs.

Using the Local Bus, you can transfer data in the background while processing data in a signal-processing module. All Local Bus data transfers originate in the HP E1438A and move towards a signal processing module to the right of the HP E1438A. If other modules generate data to the left of the input module, the HP E1438A passes the data to its right and inserts or appends its own data at the beginning or end of the frame.

Introduction

The programmer's reference is presented as a set of *VXIplug&play* functions since this is the primary targeted environment. However, when you performed the setup for the HP E1438A, drivers were installed to support various programming environments as described in "Programming the HP E1438A" in chapter 3.

The function descriptions in the programmer's reference are valid for all environments. Be sure to follow the instructions in "Getting Started with the HP E1438A," Chapter 2 to assure proper installation and to become familiar with the capabilities of your HP E1438A software in various programming environments. You should find the example programs particularly helpful for programming in various environments.

Many of the function descriptions in the programming reference include several related functions. You may use the primary function to set all related parameters or you may use the other functions within the group to set or query a single parameter.

Parameter variables are presented as alphanumeric values which are descriptive and easy to remember. However, for faster programming you may use the numeric equivalents for the parameter variables listed at the end of this section.

Functions listed by class

Component	Capability	Subclass	Function Name
INITIALIZE & CLOSE			age1438_init (on page 87)
			age1438_close (on page 63)
MEASURE	READ	INITIATE	age1438_meas_control (on page 105)
			age1438_meas_init (on page 108)
			age1438_meas_start (on page 109)
MEASURE	READ	FETCH	age1438_read (on page 112)
			age1438_read64 (on page 112)
			age1438_read_raw (on page 115)
MEASURE	CONFIGURE		age1438_clock_fs (on page 55)
			age1438_clock_fs_get (on page 55)
			age1438_clock_recover (on page 56)
			age1438_clock_setup (on page 57)
			age1438_clock_setup_get (on page 57)
			age1438_data_memsize_get (on page 64)
			age1438_data_scale_get (on page 65)
			age1438_data_setup (on page 66)
			age1438_filter_setup (on page 76)
			age1438_frequency_setup (on page 83)
			age1438_input_autozero (on page 89)
			age1438_input_range_auto (on page 92)
			age1438_input_setup (on page 95)
			age1438_trigger_setup (on page 136)
ROUTE	CONFIGURE		age1438_lbus_mode (on page 101)
			age1438_lbus_mode_get (on page 101)
			age1438_lbus_reset (on page 103)
			age1438_lbus_reset_get (on page 103)
UTILITY			age1438_error_message (on page 74)
			age1438_error_query (on page 75)
			age1438_input_range_convert (on page 93)
			age1438_interrupt_mask_get (on page 99)
			age1438_interrupt_priority_get (on page 99)
			age1438_interrupt_restore (on page 98)

Functions listed by class

Component	Capability	Subclass	Function Name
			age1438_interrupt_setup (on page 99)
			age1438_reset (on page 120)
			age1438_revision_query (on page 122)
			age1438_self_test (on page 123)
			age1438_state_save (on page 128)
			age1438_state_recall (on page 127)
MEASURE	CONFIGURE	LOW LEVEL	age1438_adc_clock (on page 51)
			age1438_adc_clock_get (on page 51)
			age1438_adc_divider (on page 52)
			age1438_adc_divider_get (on page 52)
			age1438_data_blocksize (on page 66)
			age1438_data_blocksize_get (on page 66)
			age1438_data_delay (on page 66)
			age1438_data_delay_get (on page 66)
			age1438_data_mode (on page 66)
			age1438_data_mode_get (on page 66)
			age1438_data_port (on page 66)
			age1438_data_port_get (on page 66)
			age1438_data_resolution (on page 66)
			age1438_data_resolution_get (on page 66)
			age1438_data_type (on page 66)
			age1438_data_type_get (on page 66)
			age1438_data_xfersize (on page 71)
			age1438_data_xfersize_get (on page 71)
			age1438_input_alias_filter (on page 95)
			age1438_input_alias_filter_get (on page 95)
			age1438_input_coupling (on page 95)
			age1438_input_coupling_get (on page 95)
			age1438_input_offset (on page 90)
			age1438_input_offset_get (on page 90)
			age1438_input_offset_save (on page 91)
			age1438_input_range (on page 95)
			age1438_input_range_get (on page 95)
			age1438_input_signal (on page 95)
			age1438_input_signal_get (on page 95)
			age1438_filter_decimate (on page 76)
			age1438_filter_decimate_get (on page 76)
			age1438_filter_bw (on page 76)
			age1438_filter_bw_get (on page 76)
			age1438_filter_sync (on page 79)

Component	Capability	Subclass	Function Name
			age1438_frequency_center (on page 83)
			age1438_frequency_center_get (on page 83)
			age1438_frequency_center_raw (on page 81)
			age1438_frequency_center_raw_get (on page 81)
			age1438_frequency_cmplxdc (on page 83)
			age1438_frequency_cmplxdc_get (on page 83)
			age1438_frequency_sync (on page 83)
			age1438_frequency_sync_get (on page 83)
			age1438_front_panel_clock_input (on page 86)
			age1438_front_panel_clock_input_get (on page 86)
			age1438_reference_clock (on page 118)
			age1438_reference_clock_get (on page 118)
			age1438_reference_prescaler (on page 119)
			age1438_reference_prescaler_get (on page 119)
			age1438_smb_clock_output (on page 126)
			age1438_smb_clock_output_get (on page 126)
			age1438_sync_clock (on page 131)
			age1438_sync_clock_get (on page 131)
			age1438_sync_direction (on page 132)
			age1438_sync_direction_get (on page 132)
			age1438_sync_output (on page 133)
			age1438_sync_output_get (on page 133)
			age1438_trigger_adclevel (on page 136)
			age1438_trigger_adclevel_get (on page 136)
			age1438_trigger_delay (on page 136)
			age1438_trigger_delay_get (on page 136)
			age1438_trigger_delay_actual_get (on page 134)
			age1438_trigger_gen (on page 136)
			age1438_trigger_gen_get (on page 136)
			age1438_trigger_maglevel (on page 136)
			age1438_trigger_maglevel (on page 136)
			age1438_trigger_phase_actual_get (on page 135)
			age1438_trigger_slope (on page 136)
			age1438_trigger_slope_get (on page 136)
			age1438_trigger_type (on page 136)
			age1438_trigger_type_get (on page 136)
			age1438_vcxo (on page 140)
			age1438_vcxo_freq (on page 141)
			age1438_vcxo_freq_get (on page 141)
			age1438_vcxo_freq_preset (on page 142)

Functions listed by class

Component	Capability	Subclass	Function Name
UTILITY		LOW LEVEL	age1438_vcxo_get (on page 140)
			age1438_vxi_clock_output (on page 143)
			age1438_vxi_clock_output_get (on page 143)
			age1438_attrib_get (on page 53)
			age1438_cal_get (on page 54)
			age1438_driver_debug_level (on page 72)
			age1438_driver_debug_level_get (on page 72)
			age1438_options_get (on page 110)
			age1438_product_id_get (on page 111)
			age1438_reset_hard (on page 121)
			age1438_serial_number (on page 125)
			age1438_serial_number_get (on page 125)
			age1438_status_get (on page 129)
			age1438_wait (on page 144)

Functions listed by functional group

This section lists the programming functions in groups of related functions. A brief description of each group follows:

“Initializing and closing” on page 42: You must initialize the I/O driver and set up each module before using any other functions.

“Identification” on page 43: These functions identify the module, serial number and options.

“Analog setup” on page 42: These functions determine how the analog input section is configured.

“Data format” on page 42: An HP E1438A can collect either real or complex data in 12-bit or 24-bit format. It can collect data into various block sizes or in a continuous mode. This data can be transferred either on the VXI backplane or over the Local Bus.

“Digital processing” on page 43: The decimation filter provides bandpass filtering and decimation capabilities. You may also select limited frequency spans away from baseband.

“Measurement control” on page 44: These functions initiate or terminate the measurement loop.

“Timing” on page 44: The clock signals for the ADC sample clock can be set in a variety of ways. One HP E1438A can be enabled to drive the sample clock line on the VXI backplane or front panel to enable synchronization of multiple HP E1438A modules.

“Trigger” on page 45: These functions set all parameters associated with triggering the beginning of data collection.

“Synchronization (controlling multiple modules)” on page 45: These functions support synchronous operation among multiple HP E1438As by using shared ADC clock and Sync signals to drive all the modules in a system.

“Reading data” on page 45: The HP E1438A reads data from either the VME or the Local Bus data port. This data can optionally be scaled and converted to floating point.

“Interrupts” on page 44: The HP E1438A can be programmed to interrupt via the VXI backplane whenever certain status conditions are present.

“Debugging” on page 43: Allows you to identify program and hardware problems.

Functions listed by functional group**Initializing and closing**

- age1438_init** (on page 87) – initializes the I/O driver for a module
- age1438_close** (on page 63) – closes the module's software connection

Analog setup

- age1438_input_setup** (on page 95) – sets all the analog input parameters
- age1438_input_alias_filter** (on page 95) – include/bypass the built-in analog anti-alias filter
- age1438_input_alias_filter_get** (on page 95) – gets the anti-alias filter state
- age1438_input_autozero** (on page 95) – nulls out the input dc offset in baseband mode
- age1438_input_coupling** (on page 95) – selects ac or dc input coupling
- age1438_input_coupling_get** (on page 95) – get the input coupling type
- age1438_input_offset** (on page 90) – sets the dc offset settings for the current range
- age1438_input_offset_get** (on page 90) – gets the dc offset settings
- age1438_input_offset_save** (on page 91) – saves the dc offset settings in NVRAM
- age1438_input_range** (on page 95) – sets the full scale input range
- age1438_input_range_auto** (on page 92) – performs auto-ranging
- age1438_input_range_convert** (on page 93) – converts input range to volts
- age1438_input_range_get** (on page 95) – gets the input range
- age1438_input_signal** (on page 95) – connect/disconnect the input signal to the input amplifier
- age1438_input_signal_get** (on page 95) – gets the input buffer amplifier state
- age1438_state_save** (on page 128) – saves the current module state
- age1438_state_recall** (on page 127) – recalls a previous module state

Data format

- age1438_data_setup** (on page 66) – sets all format and data output flow parameters
- age1438_data_blocksize** (on page 66) – determines the size of the output data block
- age1438_data_blocksize_get** (on page 66) – gets the output data block size
- age1438_data_delay** (on page 66) – determines FIFO delay in continuous mode
- age1438_data_delay_get** (on page 66) – gets FIFO delay
- age1438_data_memsize_get** (on page 64) – returns module's memory size
- age1438_data_mode** (on page 66) – selects block mode or continuous mode
- age1438_data_mode_get** (on page 66) – gets the data mode
- age1438_data_port** (on page 66) – selects VME bus or local bus output transmission
- age1438_data_port_get** (on page 66) – gets the output port designation
- age1438_data_resolution** (on page 66) – selects 12 or 24 bits data resolution
- age1438_data_resolution_get** (on page 66) – gets the data resolution
- age1438_data_scale_get** (on page 65) – gets the data scale factor used to convert raw data to volts
- age1438_data_type** (on page 66) – selects real or complex output data
- age1438_data_type_get** (on page 66) – gets output data type
- age1438_data_xfersize** (on page 71) – allows a specified amount of data to be read before an entire block has been acquired.
- age1438_data_xfersize_get** (on page 71) – gets the data transfer size
- age1438_lbus_mode** (on page 101) – sets the transmission mode of the local bus
- age1438_lbus_mode_get** (on page 101) – gets the local bus transmission mode
- age1438_lbus_reset** (on page 103) – resets the local bus

age1438_lbus_reset_get (on page 103) – gets the local bus reset state

Debugging

age1438_cal_get (on page 54) – gets last calibration date of specified board

age1438_clock_recover (on page 56) – allows recovery from an out-of-spec external sample clock

age1438_driver_debug_level (on page 72) – sets the debug level

age1438_driver_debug_level_get (on page 72) – gets the debug level

age1438_error_message (on page 74) – returns error information obtained from function calls

age1438_error_query (on page 75) – queries the module for the most recent error

age1438_status_get (on page 129) – retrieves the module's status register information

age1438_self_test (on page 123) – performs a self-test on the module and returns the result

Digital processing

age1438_filter_setup (on page 76) – sets the digital filter bandwidth and decimation filter parameters

age1438_filter_bw (on page 76) – selects a signal filter bandwidth

age1438_filter_bw_get (on page 76) – gets the signal filter bandwidth

age1438_filter_decimate (on page 76) – enables/disables an extra factor of 2 decimation

age1438_filter_decimate_get (on page 76) – gets current state of extra decimation

age1438_filter_sync (on page 79) – synchronizes the decimation filter counter

age1438_frequency_setup (on page 83) – sets all zoom center frequency parameters

age1438_frequency_center (on page 83) – sets the center frequency

age1438_frequency_center_get (on page 83) – gets the current center frequency

age1438_frequency_center_raw (on page 81) – quickly sets the center frequency

age1438_frequency_center_raw_get (on page 81) – gets the raw center frequency

age1438_frequency_cmplxdc (on page 83) – selects a complex baseband measurement

age1438_frequency_cmplxdc_get (on page 83) – gets the state of the baseband measurement mode

age1438_frequency_sync (on page 83) – prepares the module for a synchronous frequency change

age1438_frequency_sync_get (on page 83) – gets the state of the synchronous change mode

Identification

age1438_product_id_get (on page 111) – returns the module's product identification string

age1438_options_get (on page 110) – returns the module's options

age1438_serial_number (on page 110) – sets the module's serial number for product repair purposes

age1438_serial_number_get (on page 110) – returns the module's serial number

age1438_revision_query (on page 122) – returns strings that identify the date of the module's firmware revision

Functions listed by functional group**Interrupts**

- age1438_attrib_get** (on page 53) – gets low-level attributes of current I/O library session
- age1438_interrupt_setup** (on page 99) – sets all interrupt parameters
- age1438_interrupt_mask_get** (on page 99) – gets the interrupt event mask
- age1438_interrupt_priority_get** (on page 99) – gets the VME interrupt line
- age1438_interrupt_restore** (on page 98) – restores the interrupt masks to the most recent setting

Measurement control

- age1438_meas_control** (on page 105) – initiates and controls measurements in multi-module systems
- age1438_meas_init** (on page 108) – initiates a measurement without first checking for valid hardware setup
- age1438_meas_start** (on page 109) – checks for valid hardware setup and then initiates a measurement
- age1438_reset** (on page 120) – places the module in a known state
- age1438_reset_hard** (on page 121) – resets the module hardware

Timing

- age1438_clock_setup** (on page 57) – sets all timing parameters for commonly used measurement setups
- age1438_clock_setup_get** (on page 57) – gets the current clock setup
- age1438_clock_fs** (on page 55) – provides the frequency of an external sample clock
- age1438_clock_fs_get** (on page 55) – gets the current external sample clock frequency
- age1438_adc_clock** (on page 51) – specifies the ADC clock source
- age1438_adc_clock_get** (on page 51) – gets the ADC clock source
- age1438_adc_divider** (on page 52) – determines which divider is applied to the ADC clock source
- age1438_adc_divider_get** (on page 52) – gets the module's current clock divider state
- age1438_front_panel_clock_input** (on page 86) – specifies the source for the front panel clock
- age1438_front_panel_clock_input_get** (on page 86) – gets the front panel clock source
- age1438_reference_clock** (on page 118) – selects the source of the reference clock
- age1438_reference_clock_get** (on page 118) – gets the source of the reference clock
- age1438_reference_prescaler** (on page 119) – selects prescaling of the reference clock
- age1438_reference_prescaler_get** (on page 119) – gets prescaling of the reference clock
- age1438_smb_clock_output** (on page 126) – specifies which clock to output from the SMB clock connectors
- age1438_smb_clock_output_get** (on page 126) – gets which clock to output from the SMB clock connectors
- age1438_sync_clock** (on page 131) – selects the source of the sync signal
- age1438_sync_clock_get** (on page 131) – gets the source of the sync signal
- age1438_sync_direction** (on page 132) – selects front or rear panel availability of the

sync signal
age1438_sync_direction_get (on page 132) – gets the state of front or rear panel clock availability
age1438_sync_output (on page 133) – selects the output for the sync signal
age1438_sync_output_get (on page 133) – gets the output for the sync signal
age1438_vcxo (on page 140) – selects whether the module should use an internal clock source
age1438_vcxo_get (on page 140) – gets whether the internal clock source is on or off
age1438_vcxo_freq (on page 141) – selects which internal clock the module uses
age1438_vcxo_freq_get (on page 141) – gets which internal clock the module uses
age1438_vcxo_freq_preset (on page 142) – selects which internal clock source should be used as a default
age1438_vxi_clock_output (on page 143) – selects which clock drives the VXI clock
age1438_vxi_clock_output_get (on page 143) – gets which clock drives the VXI clock

Trigger

age1438_trigger_setup (on page 136) – sets all parameters associated with triggering the beginning of data collection
age1438_trigger_adclevel (on page 136) – specifies the threshold for the ADC trigger
age1438_trigger_adclevel_get (on page 136) – gets the trigger threshold
age1438_trigger_delay (on page 136) – specifies a pre- or post-trigger delay time
age1438_trigger_delay_get (on page 136) – gets the trigger delay time
age1438_trigger_delay_actual_get (on page 134) – gets the actual delay time from the most recent trigger event
age1438_trigger_gen (on page 136) – determines whether a module can generate a trigger
age1438_trigger_gen_get (on page 136) – gets the trigger generation status
age1438_trigger_maglevel (on page 136) – specifies the threshold for a magnitude trigger
age1438_trigger_maglevel_get (on page 136) – gets magnitude trigger threshold
age1438_trigger_phase_actual_get (on page 135) – returns a representation of the phase value of the LO at the most recent trigger point
age1438_trigger_slope (on page 136) – selects a positive or negative trigger
age1438_trigger_slope_get (on page 136) – gets trigger slope
age1438_trigger_type (on page 136) – specifies the trigger type
age1438_trigger_type_get (on page 136) – gets trigger type

Reading data

age1438_data_scale_get (on page 65) – gets data scale factor
age1438_read (on page 112) – reads scaled 32-bit float data from FIFO
age1438_read64 (on page 112) – reads scaled 64-bit float data from FIFO, specifically for VEE applications
age1438_read_raw (on page 115) – reads raw data from FIFO

Synchronization (controlling multiple modules)

age1438_clock_setup (on page 57) – supplies commonly used clock and sync configurations
See “Timing” on page 44 for low level clock and sync setup commands

Functions listed by functional group

- age1438_clock_setup_get** (on page 57) – gets the current clock and sync setup
- age1438_clock_fs** (on page 55) – provides a clock frequency for external sample clock configurations
- age1438_clock_fs_get** (on page 55) – gets the external clock frequency
- age1438_filter_sync** (on page 79) – synchronizes the decimation filter counter
- age1438_frequency_sync and age1438_frequency_center** (on page 83) – prepare the modules for frequency change
- age1438_meas_control** (on page 105) – synchronizes arming and triggering in multi-module systems
- age1438_trigger_gen** (on page 136) – determines whether a module can generate a trigger
- age1438_trigger_gen_get** (on page 136) – gets the trigger generation status
- age1438_wait** (on page 144) – facilitates the synchronization and control of multi-module systems

Functions listed alphabetically

age1438_adc_clock (on page 51) – determines the ADC clock source
age1438_adc_clock_get (on page 51) – gets the ADC clock source
age1438_adc_divider (on page 52) – determines which divider is applied to the ADC clock source
age1438_adc_divider_get (on page 52)– gets the module's current clock divider state
age1438_attr_get (on page 53) – gets low-level attributes of current I/O library session.
age1438_cal_get (on page 54) – gets last calibration date of specified board
age1438_clock_fs (on page 55) – provides the module with the frequency of an external sample clock
age1438_clock_fs_get (on page 55) – gets the current external sample clock frequency
age1438_clock_recover (on page 56) – allows recovery from an out-of-spec external sample clock
age1438_clock_setup (on page 57) – sets all timing parameters for commonly used measurement setups
age1438_clock_setup_get (on page 57) – gets the current clock setup
age1438_close (on page 63) – closes the module's software connection
age1438_data_blocksize (on page 66) – determines the size of the output data block
age1438_data_blocksize_get (on page 66) – gets the output data block size
age1438_data_delay (on page 66) – determines FIFO delay in continuous mode
age1438_data_delay_get (on page 66) – gets FIFO delay in continuous mode
age1438_data_memsize_get (on page 64) – returns module's memory size in megabytes
age1438_data_mode (on page 66) – selects block mode or continuous mode
age1438_data_mode_get (on page 66) – gets the data mode
age1438_data_port (on page 66) – selects VME bus or local bus output port transmission
age1438_data_port_get (on page 66) – gets the output port designation
age1438_data_resolution (on page 66) – selects 12 or 24 bits data resolution
age1438_data_resolution_get (on page 66) – gets the data resolution
age1438_data_scale_get (on page 65) – gets data scale factor used to convert raw data to volts
age1438_data_setup (on page 66) – sets all format and data output flow parameters
age1438_data_type (on page 66) – selects real or complex output data
age1438_data_type_get (on page 66) – gets output data type
age1438_data_xfersize (on page 71) – allows a specified amount of data to be read before an entire block has been acquired
age1438_data_xfersize_get (on page 71) – gets the data transfer size
age1438_driver_debug_level (on page 72) – sets the debug level

Functions listed alphabetically

age1438_driver_debug_level_get (on page 72) – gets the debug level

age1438_error_message (on page 74) – returns error information obtained from function calls

age1438_error_query (on page 75) – queries the module for the most recent error

age1438_filter_bw (on page 76) – selects a signal filter bandwidth

age1438_filter_bw_get (on page 76) – gets the signal filter bandwidth

age1438_filter_decimate (on page 76) – enables/disables and extra factor of 2 decimation

age1438_filter_decimate_get (on page 76) – gets current state of extra decimation

age1438_filter_setup (on page 76) – sets the digital filter bandwidth and decimation filter parameters

age1438_filter_sync (on page 79) – synchronizes the decimation filter counter

age1438_frequency_center (on page 83) – sets the center frequency

age1438_frequency_center_get (on page 83) – gets the current center frequency

age1438_frequency_center_raw (on page 81) – quickly sets the center frequency

age1438_frequency_center_raw_get (on page 81) – gets the raw center frequency

age1438_frequency_cmplxdc (on page 83) – selects a complex baseband measurement

age1438_frequency_cmplxdc_get (on page 83) – gets the state of the baseband measurement mode

age1438_frequency_setup (on page 83) – sets all the zoom center frequency parameters

age1438_frequency_sync (on page 83) – prepares the module for a synchronous frequency change

age1438_frequency_sync_get (on page 83) – gets the state of the synchronous change mode

age1438_front_panel_clock_input (on page 86) – specifies the source of the front panel clock

age1438_front_panel_clock_input_get (on page 86) – gets the front panel clock source

age1438_init (on page 87) – initializes the I/O driver for a module

age1438_input_alias_filter (on page 95) – include/bypass the built-in analog anti-alias filter

age1438_input_alias_filter_get (on page 95) – gets the anti-alias filter state

age1438_input_autozero (on page 89) – nulls out the input dc offset in baseband mode

age1438_input_coupling (on page 95) – selects ac or dc input coupling

age1438_input_coupling_get (on page 95) – get the input coupling type

age1438_input_offset (on page 90) – sets the dc offset settings for the current range

age1438_input_offset_get (on page 90) – gets the dc offset settings

age1438_input_offset_save (on page 91) – saves the dc offset settings in NVRAM

age1438_input_range (on page 95) – sets the full scale range

age1438_input_range_auto (on page 92) – performs auto-ranging in baseband mode

age1438_input_range_convert (on page 93) – converts the input range to volts

age1438_input_range_get (on page 95) – gets the input range

age1438_input_setup (on page 95) – sets all the analog input parameters

age1438_input_signal (on page 95) – connect/disconnect the input signal to the input amplifiers

age1438_input_signal_get (on page 95) – gets the input buffer amplifier state

age1438_interrupt_mask_get (on page 99) – gets the interrupt event mask

age1438_interrupt_priority_get (on page 99) – gets the VME interrupt line

age1438_interrupt_restore (on page 98) – restores the interrupt masks to the most recent setting

age1438_interrupt_setup (on page 99) – sets both interrupt parameters

age1438_lbus_mode (on page 101) – sets the local bus transmission mode

age1438_lbus_mode_get (on page 101) – gets the local bus mode

age1438_lbus_reset (on page 103) – resets local bus

age1438_lbus_reset_get (on page 103) – gets the local bus mode reset state

age1438_meas_control (on page 105) – initiates and controls measurements in multi-module systems

age1438_meas_init (on page 108) – initiates a measurement without first checking for valid hardware setup

age1438_meas_start (on page 109) – checks for valid hardware setup and then initiates a measurement

age1438_options_get (on page 110) – returns the module's options

age1438_product_id_get (on page 111) – returns the module's product identification string

age1438_read (on page 112) – reads scaled 32-bit float data from FIFO

age1438_read_raw (on page 115) – reads raw data from FIFO

age1438_read64 (on page 112) – reads scaled 64-bit float data from FIFO, specifically for VEE applications

age1438_reference_clock (on page 118) – selects the source of the reference clock

age1438_reference_clock_get (on page 118) – gets the source of the reference clock

age1438_reference_prescaler (on page 119) – selects prescaling of the reference clock

age1438_reference_prescaler_get (on page 119) – gets prescaling of the reference clock

age1438_reset (on page 120) – places the module in a known state

age1438_reset_hard (on page 121) – resets the module hardware

age1438_revision_query (on page 122) – returns strings that identify the date of the firmware revision.

age1438_self_test (on page 123) – performs a self-test on the module and returns the result

age1438_serial_number (on page 110) – sets the module's serial number for product repair purposes

age1438_serial_number_get (on page 110) – returns the module's serial number

age1438_smb_clock_output (on page 126) – specifies which clock to output from the SMB clock connectors

age1438_smb_clock_output_get (on page 126) – gets which clock to output from the SMB clock connectors

age1438_state_save (on page 128) – saves the current module state

age1438_state_recall (on page 127) – recalls a saved module state

age1438_status_get (on page 129) – retrieves module's status register information

age1438_sync_clock (on page 131) – selects the source of the sync signal

age1438_sync_clock_get (on page 131) – gets the source of the sync signal

Functions listed alphabetically

- age1438_sync_direction** (on page 132) – selects front or rear panel availability of the sync signal
- age1438_sync_direction_get** (on page 132) – gets the state of front or rear panel clock availability
- age1438_sync_output** (on page 133) – selects the output for the sync signal
- age1438_sync_output_get** (on page 133) – gets the output for the sync signal
- age1438_trigger_adclevel** (on page 136) – specifies the threshold for the ADC trigger
- age1438_trigger_adclevel_get** (on page 136) – gets the trigger threshold
- age1438_trigger_delay** (on page 136) – specifies a pre- or post-trigger delay time
- age1438_trigger_delay_actual_get** (on page 134) – gets the actual delay time from the most recent trigger event
- age1438_trigger_delay_get** (on page 136) – gets the trigger delay time
- age1438_trigger_gen** (on page 136) – determines whether a module can generate a trigger
- age1438_trigger_gen_get** (on page 136) – gets the trigger generation status
- age1438_trigger_maglevel** (on page 136) – specifies the threshold for a magnitude trigger
- age1438_trigger_maglevel_get** (on page 136) – gets magnitude trigger threshold
- age1438_trigger_phase_actual_get** (on page 135) – returns a representation of the phase value of the LO at the most recent trigger point
- age1438_trigger_setup** (on page 136) – sets all parameters associated with triggering the beginning of data collection
- age1438_trigger_slope** (on page 136) – selects a positive or negative trigger
- age1438_trigger_slope_get** (on page 136) – gets trigger slope
- age1438_trigger_type** (on page 136) – determines the trigger type
- age1438_trigger_type_get** (on page 136) – gets trigger type
- age1438_vcxo** (on page 140) – selects whether the module should use an internal clock source
- age1438_vcxo_freq** (on page 141) – selects which internal clock the module uses
- age1438_vcxo_freq_get** (on page 141) – gets which internal clock the module uses
- age1438_vcxo_get** (on page 140) – gets whether the internal clock source is on or off
- age1438_vcxo_freq_preset** (on page 142) – selects which internal clock source should be used as a default
- age1438_vxi_clock_output** (on page 143) – selects which clock drives the VXI clock
- age1438_vxi_clock_output_get** (on page 143) – gets which clock drives the VXI clock
- age1438_wait** (on page 144) – facilitates the synchronization and control of multi-module systems

age1438_adc_clock

Specifies the ADC clock source. This description also includes the query function:

age1438_adc_clock_get

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_adc_clock(ViSession id, ViInt16 adcClock);
```

```
ViStatus age1438_adc_clock_get(ViSession id, ViPInt16 adcClockPtr);
```

Note

This command should be used only for specialized custom clock requirements. Most useful clock setups can be supplied by age1438_clock_setup.

Parameters

id

is the VXI instrument session pointer returned by the **age1438_init** function.

adcClock

AGE1438_VCXO_INTERNAL selects an internal oscillator within the module. **age1438_vcxo_freq** determines which oscillator is used. **age1438_vcxo** determines whether the internal oscillator is turned on. You must use all three of the functions to provide the desired internal clock source.

AGE1438_VCXO_EXT_REF takes an external reference signal on the front panel and uses a phase-locked loop to convert it to the ADC clock of the module. The ADC clock can be either 100 MHz or 102.4 MHz. The external reference used by the phase lock loop to synthesize the ADC clock can be either a 10 MHz or 10.24 MHz signal.

AGE1438_EXT_SAMPLE_CLOCK uses an external sample clock selected by **age1438_reference_clock**.

adcClockPtr

points to the value of the current *adcClock*.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to "age1438_error_message" on page 74.

See Also

"Commands which halt active measurements" on page 149, "Default values" on page 152, "age1438_init" on page 87, "age1438_clock_setup" on page 57, "age1438_vcxo_freq" on page 141, "age1438_vcxo_freq_preset" on page 142, "age1438_vcxo" on page 140, "age1438_front_panel_clock_input" on page 86, "age1438_reference_clock" on page 118, "Using clock and sync" in chapter 3

age1438_adc_divider

Determines which divider is applied to the ADC clock source. This description also includes the query function:

age1438_adc_divider_get

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_adc_divider(ViSession id, ViInt16 adcDivider);
```

```
ViStatus age1438_adc_divider_get(ViSession id, ViPInt16 adcDividerPtr);
```

Note

This command should be used only for specialized custom clock requirements. Most useful clock setups can be supplied by age1438_clock_setup.

Description

This function should generally be left in the default mode. The alternate mode applies to a different model of the module.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

adcDivider AGE1438_DIVIDE_BY_10 divides the ADC clock by 10.

AGE1438_DIVIDE_BY_38 divides the ADC clock by 38.

adcDividerPtr points to the current value of *adcDivider*.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

Comments

The HP E1438A normally runs its sample clock at either 100 MHz or 102.4 MHz. The PLL divider divides the VCO by 10 to get either a 10 MHz or 10.24 MHz clock to compare to a 10 or 10.24 reference clock, which the user can supply through the front panel BNC. Alternatively the reference can come from a master module in the system via the back plane or front panel SMBs.

See Also

“Default values” on page 152, “age1438_init” on page 87, “age1438_clock_setup” on page 57, “Using clock and sync” in chapter 3

age1438_attrib_get

Gets low-level attributes of current I/O library session.

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_attrib_get(ViSession id, ViInt16 attribute, ViPint32 value);
```

Description

age1438_attrib_get is used primarily to manage the use of interrupts which requires making direct VISA function calls. Since interrupts are a shared resource across all modules using the VXI interface, it is not possible for the HP E1438A library, which governs single modules, to provide the functions to properly manage interrupts.

This function is used to access either the I/O library handle or the mapped I/O base address of the module. You should refer to the appropriate VISA documentation for descriptions of the I/O library functions.

Parameters

id	is the VXI instrument session pointer returned by the age1438_init function.
attribute	designates the type of attribute to return. AGE1438_IO_HANDLE accesses the I/O library handle. AGE1438_IO_ADDRESS points to the mapped I/O base address of the module. AGE1438_RM_HANDLE accesses the I/O library handle of the default resource manager. AGE1438_DATA_REGISTER points to the mapped address of the HP E1438A data register. One or both of these parameters are used when calling I/O library functions directly.
value	is the value of the requested attribute. For the VISA I/O library the value of the handle attribute corresponds to the vi parameter used by the majority of the I/O functions. The address attribute points to the base of the mapped I/O address space.

Example

See the interrupt.c example program.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“age1438_init” on page 87, “age1438_interrupt_setup” on page 99

age1438_cal_get

Gets last calibration date of specified board.

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_cal_get(ViSession id, ViInt16 board, ViPInt32 datestampPtr);
```

Description

age1438_cal_get is used to read the date stamp of the last calibration.

Parameters

id	is the VXI instrument session pointer returned by the age1438_init function.
board	AGE1438_01_BOARD returns calibration information for the 01 board. AGE1438_02_BOARD returns calibration information for the 02 board.
datestampPtr	points to the return location for the timestamp of the most recent saved calibrations. Format is MMDDYYYY, hh:mm in base 10 notation.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“age1438_init” on page 87

age1438_clock_fs

Provides the module with the frequency of an external sample clock. This description also includes the query:

age1438_clock_fs_get

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_clock_fs(ViSession id, ViReal64 fs);
```

```
ViStatus age1438_clock_fs_get(ViSession id, ViPReal64 fsPtr);
```

Description

This command is applicable only when an external sample clock is used. It is an order-dependent command and must be set after selecting the external sample clock.

When using an external sample clock or when a module is a non-master in a multi-module group, the frequency of the ADC clock is unknown by the module. It is the responsibility of the programmer to provide the correct frequency so that library functions dependent on *fs* operate properly. This value has no effect if the module is not set up to use the external sample clock.

Parameters

- id** is the VXI instrument session pointer returned by the **age1438_init** function.
- fs** provides the module with the frequency of an external sample clock (from 10,000,000 to 103,000,000) connected to the Ext Clk TTL connector.
- AGE1438_FS_MIN supplies the minimum external sample clock frequency.
- AGE1438_FS_MAX supplies the maximum external sample clock frequency.
- fsPtr** points to the current value of the external sample clock frequency. If the HP E1438A is set to the internal ADC clock, this query returns the value of that clock frequency. If the HP E1438A is set to the external clock, this query returns the last value entered via the **age1438_clock_fs** function.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Default values” on page 152, “age1438_init” on page 87, “age1438_clock_setup” on page 57, “age1438_front_panel_clock_input” on page 86, “Using clock and sync” in chapter 3

age1438_clock_recover

Allows recovery from an out-of-spec external sample clock.

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_clock_recover(ViSession id);
```

Description

This command is used to restore proper function if the module has received an out-of-spec external sample clock. An out-of-spec situation could occur if the external sample clock is removed or changed during operation, or if it has glitches which don't meet specs. In this case the module would cease functioning and this command must be issued in order to resume proper operation after restoring a valid clock.

Parameters**id****Return Value**

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to "age1438_error_message" on page 74.

See Also

"Commands which halt active measurements" on page 149, "age1438_init" on page 87, "age1438_clock_setup" on page 57

age1438_clock_setup

Sets all timing parameters for commonly used measurement setups. This description also includes a query:

age1438_clock_setup_get

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_clock_setup(ViSession id, ViInt16 clockSetup);
```

```
ViStatus age1438_clock_setup_get(ViSession id, ViPInt16 clockSetupPtr);
```

Description

age1438_clock_setup is used to select the source and distribution of clocking and synchronization signals used by the HP E1438A module. The primary clock signal used by the module is the ADC clock, for which the rising edges indicate the time for each sample of the analog-to-digital converter.

This function changes the settings controlled by the following lower-level functions:

```
age1438_adc_clock
age1438_adc_divider
age1438_front_panel_clock_input
age1438_reference_clock
age1438_reference_prescaler
age1438_smb_clock_output
age1438_sync_clock
age1438_sync_direction
age1438_sync_output
age1438_vcxo
age1438_vxi_clock_output
```

Note

This function does not alter settings made with `age1438_vcxo_freq`, but it does control whether the selected VCXO is actually running.

Setups using the external sample clock require the use of `age1438_clock_fs` to supply the clock frequency.

Parameters

id

is the VXI instrument session pointer returned by the **age1438_init** function.

clockSetup

This parameter provides a quick way to set up most of the timing parameters for several standard clock configurations. The following setups are available:

Functions listed alphabetically

Simple clock setups for stand-alone modules

Internal reference

AGE1438_SIMPLE_INT_REF

ADC_CLK	VCXO_INTERNAL
VCXO	VCXO_ON
ADC_DIVIDER	DIVIDE_BY_10
REFERENCE_PRESCALER	PRESCALE_BY_1
VXI_CLK_OUTPUT	CLOCK_OFF
REFERENCE_CLOCK	N/A
FRONT_PANEL_CLOCK	CLOCK_OFF
SMB_CLOCK_OUTPUT	CLOCK_OFF
SYNC_CLOCK	DIVIDED_ADC_CLOCK
SYNC_OUTPUT	SYNC_OUT_OFF
SYNC_DIRECTION	N/A

Phase locked to external reference

AGE1438_SIMPLE_EXT_REF

ADC_CLK	VCXO_EXT_REF
VCXO	VCXO_ON
ADC_DIVIDER	DIVIDE_BY_10
REFERENCE_PRESCALER	PRESCALE_BY_1
VXI_CLK_OUTPUT	CLOCK_OFF
REFERENCE_CLOCK	FRONT_PANEL_CLOCK
FRONT_PANEL_CLOCK	BNC_CLOCK
SMB_CLOCK_OUTPUT	CLOCK_OFF
SYNC_CLOCK	DIVIDED_ADC_CLOCK
SYNC_OUTPUT	SYNC_OUT_OFF
SYNC_DIRECTION	N/A

External sample clock**AGE1438_SIMPLE_EXT_SAMP**

ADC_CLK	EXT_SAMPLE_CLOCK
VCXO	VCXO_OFF
ADC_DIVIDER	DIVIDE_BY_10
REFERENCE_PRESCALER	PRESCALE_BY_1
VXI_CLK_OUTPUT	CLOCK_OFF
REFERENCE_CLOCK	FRONT_PANEL_CLOCK
FRONT_PANEL_CLOCK	BNC_CLOCK
SMB_CLOCK_OUTPUT	CLOCK_OFF
SYNC_CLOCK	DIVIDED_ADC_CLOCK
SYNC_OUTPUT	SYNC_OUT_OFF
SYNC_DIRECTION	N/A

Front panel master-slave setups, one master per mainframe**Front master, internal reference****AGE1438_FRNT_MSTR_INT_REF**

ADC_CLK	VCXO_INTERNAL
VCXO	VCXO_ON
ADC_DIVIDER	DIVIDE_BY_10
REFERENCE_PRESCALER	PRESCALE_BY_1
VXI_CLK_OUTPUT	CLOCK_OFF
REFERENCE_CLOCK	N/A
FRONT_PANEL_CLOCK	CLOCK_OFF
SMB_CLOCK_OUTPUT	DIVIDED_ADC_CLOCK
SYNC_CLOCK	DIVIDED_ADC_CLOCK
SYNC_OUTPUT	SYNC_OUT_SMB
SYNC_DIRECTION	FRNT_TO_REAR

Functions listed alphabetically**Front master, phase locked to external reference****AGE1438_FRNT_MSTR_EXT_REF**

ADC_CLK	VCXO_EXT_REF
VCXO	VCXO_ON
ADC_DIVIDER	DIVIDE_BY_10
REFERENCE_PRESCALER	PRESCALE_BY_1
VXI_CLK_OUTPUT	CLOCK_OFF
REFERENCE_CLOCK	FRONT_PANEL_CLOCK
FRONT_PANEL_CLOCK	BNC_CLOCK
SMB_CLOCK_OUTPUT	DIVIDED_ADC_CLOCK
SYNC_CLOCK	DIVIDED_ADC_CLOCK
SYNC_OUTPUT	SYNC_OUT_SMB
SYNC_DIRECTION	FRNT_TO_REAR

Front slave, phase locked to master**AGE1438_FRNT_SLAV_EXT_REF**

ADC_CLK	VCXO_EXT_REF
VCXO	VCXO_ON
ADC_DIVIDER	DIVIDE_BY_10
REFERENCE_PRESCALER	PRESCALE_BY_1
VXI_CLK_OUTPUT	CLOCK_OFF
REFERENCE_CLOCK	FRONT_PANEL_CLOCK
FRONT_PANEL_CLOCK	SMB_CLK
SMB_CLOCK_OUTPUT	CLOCK_OFF
SYNC_CLOCK	SMB_CLOCK
SYNC_OUTPUT	SYNC_OUT_SMB
SYNC_DIRECTION	FRNT_TO_REAR

Rear panel master-slave setups, one master per mainframe**Rear master, internal reference****AGE1438_REAR_MSTR_INT_REF**

ADC_CLK	VCXO_INTERNAL
VCXO	VCXO_ON
ADC_DIVIDER	DIVIDE_BY_10
REFERENCE_PRESCALER	PRESCALE_BY_1
VXI_CLK_OUTPUT	DIVIDED_ADC_CLOCK
REFERENCE_CLOCK	N/A
FRONT_PANEL_CLOCK	CLOCK_OFF
SMB_CLOCK_OUTPUT	CLOCK_OFF
SYNC_CLOCK	DIVIDED_ADC_CLOCK
SYNC_OUTPUT	SYNC_OUT_VXI
SYNC_DIRECTION	REAR_TO_FRNT

Rear master, phase locked to external reference**AGE1438_REAR_MSTR_EXT_REF**

ADC_CLK	VCXO_EXT_REF
VCXO	VCXO_ON
ADC_DIVIDER	DIVIDE_BY_10
REFERENCE_PRESCALER	PRESCALE_BY_1
VXI_CLK_OUTPUT	DIVIDED_ADC_CLOCK
REFERENCE_CLOCK	FRONT_PANEL_CLOCK
FRONT_PANEL_CLOCK	BNC_CLOCK
SMB_CLOCK_OUTPUT	CLOCK_OFF
SYNC_CLOCK	DIVIDED_ADC_CLOCK
SYNC_OUTPUT	SYNC_OUT_VXI
SYNC_DIRECTION	REAR_TO_FRNT

Functions listed alphabetically**Rear slave, phase locked to master**

AGE1438_REAR_SLAV_EXT_REF	
ADC_CLK	VCXO_EXT_REF
VCXO	VCXO_ON
ADC_DIVIDER	DIVIDE_BY_10
REFERENCE_PRESCALER	PRESCALE_BY_1
VXI_CLK_OUTPUT	CLOCK_OFF
REFERENCE_CLOCK	VXI_CLOCK
FRONT_PANEL_CLOCK	CLOCK_OFF
SMB_CLOCK_OUTPUT	CLOCK_OFF
SYNC_CLOCK	VXI_CLOCK
SYNC_OUTPUT	SYNC_OUT_VXI
SYNC_DIRECTION	REAR_TO_FRNT

clockSetupPtr points to the current value of *clockSetup*.

AGE1438_CUSTOM_CLOCK_SETUP is returned from **age1438_clock_setup_get** when low-level clock configuration functions are used to set up clocks to a non-standard configuration.

Example

The program multichan.exe example program provides an example of how to correctly set up a multi-module system with synchronous clocks.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

Effect on Active Measurement

age1438_setup aborts any measurement in progress.

See Also

“Commands which halt active measurements” on page 149, “Default values” on page 152, “age1438_init” on page 87, “age1438_clock_fs” on page 55, “age1438_vcxo_freq” on page 141, “age1438_vcxo_freq_preset” on page 142, “age1438_clock_recover” on page 56, “Using clock and sync” in chapter 3, “Managing multiple modules” in chapter 3

age1438_close

Closes the module's software connection.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_close(ViSession id);
```

Description

age1438_close terminates the software connection to the module, deallocates system resources, and places the module in the Idle state. After this function has been executed the specified *id* identifier is no longer a valid parameter for function calls.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“age1438_init” on page 87

age1438_data_memsize_get

Returns the module's memory size in megabytes.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_data_memsize_get(ViSession id, ViPInt16 memSizePtr);
```

Description

This command allows you to determine whether your module contains standard memory of 18 Mbytes or a larger memory option.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

memSizePtr points to the memory size in number of Megabytes.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to "age1438_error_message" on page 74.

See Also

"age1438_init" on page 87, "age1438_data_setup" on page 66

age1438_data_scale_get

Gets data scale factor.

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_data_scale_get(ViSession id, ViPReal64 scalePtr);
```

Description

age1438_data_scale_get calculates the correct scale factor for raw data using the current data resolution and input range. The factor returned by this function is used to multiply raw data to get data in volts.

When the module is providing only the real part of complex data, the data is doubled to provide consistent spectrum measurements. This occurs with either shift decimation or when the real part of a zoomed signal with a non-zero center frequency is taken.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

scalePtr points to the calculated scale factor with which to scale raw data to volts.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“age1438_init” on page 87, “age1438_data_setup” on page 66, “age1438_read_raw” on page 115, “age1438_input_range_auto” on page 92, “age1438_filter_setup” on page 76

age1438_data_setup

Sets all format and data output flow parameters. This description also includes information on the following functions which set or query the format and flow parameters individually:

age1438_data_blocksize determines the size of the output data block.
age1438_data_blocksize_get gets the output data block size.
age1438_data_delay determines the FIFO delay in continuous mode.
age1438_data_delay_get gets the FIFO delay in continuous mode.
age1438_data_mode selects block mode or continuous mode.
age1438_data_mode_get gets the data mode.
age1438_data_port selects VME bus or local bus output port.
age1438_data_port_get gets the output port designation.
age1438_data_resolution selects 12 or 24 bits data resolution.
age1438_data_resolution_get gets the data resolution.
age1438_data_type selects real or complex output data.
age1438_data_type_get gets output data type.

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_data_setup(ViSession id, ViInt16 dataType, ViInt16 resolution,
    ViInt16 mode, ViInt32 blocksize, ViInt32 dataDelay, ViInt16 reserved, ViInt16
    port);
ViStatus age1438_data_blocksize(ViSession id, ViInt32 blocksize);
ViStatus age1438_data_blocksize_get(ViSession id, ViPint32 blocksizePtr);
ViStatus age1438_data_delay(ViSession id, ViInt32 dataDelay);
ViStatus age1438_data_delay_get(ViSession id, ViPint32 dataDelayPtr);
ViStatus age1438_data_mode(ViSession id, ViInt16 mode);
ViStatus age1438_data_mode_get(ViSession id, ViPint16 modePtr);
ViStatus age1438_data_port(ViSession id, ViInt16 port);
ViStatus age1438_data_port_get(ViSession id, ViPint16 portPtr);
ViStatus age1438_data_resolution(ViSession id, ViInt16 resolution);
ViStatus age1438_data_resolution_get(ViSession id, ViPint16 resolutionPtr);
ViStatus age1438_data_type(ViSession id, ViInt16 dataType);
ViStatus age1438_data_type_get(ViSession id, ViPint16 dataTypePtr);
```

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

blocksize determines the number of sample points in each output data block.

AGE1438_BLOCKSIZE_MIN selects the minimum blocksize.

AGE1438_BLOCKSIZE_MAX selects the maximum blocksize.

AGE1438_BLOCKSIZE_DEF sets the default blocksize.

The range of available block sizes depends on the number of bytes required for each sample. The command accepts any number between 2 and memory size (in bytes) $\times 2/3$. If the requested block size falls outside the range shown in the table the previous valid value is used and a status register flag (bit 6) is set indicating a setup error. The blocksize is updated after the setup is changed to be valid.

For real data *blocksize* is the number of real data values per data block. For complex data *blocksize* is the number of complex data pairs per data block.

The following table summarizes the available block sizes for each setting of the *dataType*, and *resolution* parameters.

data type	resolution	min. block size	max block size in Msamples (2 M*72 memory)^a
real	12	6	12
real	24	3	6
complex	12	3	6
complex	24	2	3

a. Parity memory is used in non-parity mode, so 2M×72 bit memory yields 18 Mbytes of FIFO storage.

Note

Block size must be an even number. Considerably more samples may need to be taken in order to set the block available status bit.

blocksizePtr

points to the current value of the *blocksize* parameter. The returned value is the closest valid value to the requested block size.

dataDelay

is used to specify the minimum FIFO delay in number of samples. This parameter applies only in continuous mode.

AGE1438_DATA_DELAY_MAX sets the maximum allowable delay.

AGE1438_DATA_DELAY_MIN sets the minimum allowable delay.

dataDelayPtr

points to the current value of the *delay* parameter.

dataType

determines whether the HP E1438A collects and returns real or complex data.

Setting this parameter to AGE1438_REAL causes only the real part of the data to be returned for each sample

AGE1438_COMPLEX causes the real data followed by the imaginary data to be returned in each sample.

Normally, if the frequency set with the **age1438_frequency_setup** function is zero, the type should be set to **AGE1438_REAL** since the imaginary component of each sample is zero anyway. When non-zero center frequencies are used the type should normally be set to **AGE1438_COMPLEX**, otherwise the imaginary component of the signal is lost.

when *dataType* is set to **AGE1438_REAL** and there is a non-zero center frequency the data scale value is doubled for consistent spectrum measurements

dataTypePtr

points to the current value of the *dataType* parameter.

mode

selects whether the HP E1438A's data collection operates in block mode or continuous mode.

Functions listed alphabetically

AGE1438_BLOCK selects block transfer mode in which the measurement is halted after each block of data. To start collection of the next data block the module must be armed and triggered again. This mode is used whenever each block of data is to be associated with an individual trigger event.

AGE1438_CONTINUOUS means that a single arm and trigger event starts a measurement which runs continuously with no gaps between output data blocks. As long as the data is read out fast enough to prevent overflow in the output FIFO, the measurement continues. The continuous mode is useful for continuous signal processing applications where data gaps are unacceptable.

modePtr	points to the current value of the <i>mode</i> parameter.
reserved	is a short integer which is reserved for future use and should be set to 0.
port	determines which output port is used to take data from the HP E1438A module. Setting <i>port</i> to AGE1438_VME means the data is to be output using standard VME register reads. Setting <i>port</i> to AGE1438_LBUS means the data is to be output as a byte-serial data stream via the VXI local bus. When using the local bus port the module immediately to the right of the HP E1438A must be capable of receiving the local bus byte sequence.
portPtr	points to the current value of the <i>port</i> parameter.
resolution	selects data resolution of either 12 or 24 bits by using resolution values of AGE1438_12BIT or AGE1438_24BIT respectively. Choosing 12-bit precision allows for more samples in the FIFO memory. Choosing 24 bits allows more dynamic range. Because of the broadband white noise present on the input of the analog-to-digital converter, it is normally sufficient to use 12 bit resolution whenever the age1438_filter_setup function specifies a signal bandwidth greater than 10 MHz. For narrower bandwidths much of the broadband white noise is filtered out, resulting in lower noise in the output data. To take advantage of this lower noise, you should use the 24-bit data resolution.
resolutionPtr	points to the current value of the <i>resolution</i> parameter.

Comments

The following table summarizes the output word or byte sequence for each combination of *dataType*, *resolution*, and *port* parameters:

data type	data resolution	port	transfer width	pad bits^a	xfers^b	sequence^c
real	12 bit	VME	16 bit	4	1	R0[15:4],R1[15:4],...
complex	12 bit	VME	16 bit	4	2	R0[15:4],Q0[15:4], R1[15:4],Q1[15:4],...
real	24 bit	VME	16 bit	4	2	R0[31:16],R0[15:8], R1[31:16],R1[15:8],...
complex	24 bit	VME	16 bit	4	4	R0[31:16],R0[15:8], Q0[31:16],Q0[15:8], R1[31:16],R1[15:8],...

data type	data resolution	port	transfer width	pad bits ^a	xfers ^b	sequence ^c
real	12 bit	LBUS	32 bit	8	2	R0[15:8],R0[7:4], R1[15:8],R1[7:4],...
complex	12 bit	LBUS	32 bit	8	4	R0[15:8],R0[7:4], Q0[15:8],Q0[7:4], R1[15:8],...
real	24 bit	LBUS	32 bit	8	4	R0[31:24],R0[23:16], R0[15:8],R0[0:0], R1[31:24],...
complex	24 bit	LBUS	32 bit	8	8	R0[31:24],R0[23:16], R0[15:8],R0[7:0], Q0[31:24],Q0[23:16], Q0[15:8],Q0[0:0], R1[31:24],...

- a. Number of least significant bits set to zero.
- b. That is, transfers required per measurement
- c. Sequence Notation: R = real number transfer; Q = imaginary number transfer. Subscript denotes the measurement datum number. Bracketed indices show which measurement bits are contained in the transfer, MSB first. A 12-bit sample is padded to 16 bits with zero placed in the 4 LSBs whereas a 24-bit sample is padded up to 32 bits with zero placed in the 8 LSBs. Example: For a 12-bit sample, R0[15:8] indicates the 8 MSBs of the sample are transferred in the first nibble. Then R0[7:4] indicates the 4 LSBs of the sample are transferred in the 4 MSBs of second nibble (along with 4 zeros for padding).

The maximum rate at which data may be transferred to memory is determined by the ADC clock rate: $\text{MaxBytes/s} = 1.5 \times (\text{ADC clock rate})$. Divide MaxBytes/s by 1.5 to get the 12-bit sample rate, and by 3 to get the 24-bit sample rate.

A limitation also applies to 32-bit, complex data transfers. Because this type of transfer cannot be made at the full sample rate, a level of decimation must be added in order to reduce the sample rate.

The following table summarizes the relationship between data parameter combinations, decimation, filter bandwidth, precision, and whether the combination permits block or continuous measurements:

Note Continuous mode is only limited by the local bus transfer and read limitations of 50 MBytes/s.

decimations	filterBW	sample rate (Msamples/s)	BW $f_s=100$ MHz	12b real	24b real	12b complex	24b complex
n/a	0	100	40	b			
1	1	50	20	b		b	
0	2	50	10	b		b	
1	2	25	10	b,c	b	b	b

Functions listed alphabetically

decimations	filterBW	sample rate (Msamples/s)	BW $f_s=100$ MHz	12b real	24b real	12b complex	24b complex
0	3	25	5	bc	b	b	b
1	3	12.5	5	bc	bc	bc	b
0	4	12.5	2.5	bc	bc	bc	b
1	4	6.25	2.5	bc	bc	bc	bc
0	5	6.25	1.25	bc	bc	bc	bc

b = block mode

c = continuous mode to local bus

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Commands which halt active measurements” on page 149, “Default values” on page 152, “age1438_init” on page 87, “age1438_frequency_setup” on page 83, “age1438_filter_setup” on page 76, “age1438_meas_control” on page 105, “age1438_clock_setup” on page 57

age1438_data_xfersize

Allows data to be read before an entire block had been acquired.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_data_xfersize(ViSession id, ViInt32 xfersize);
```

```
ViStatus age1438_data_xfersize_get(ViSession id, ViPInt32 xfersizePtr);
```

Description

This command allows you to specify the allowable data transfer size in a situation where you want to read a large block of data in increments before an entire block has been acquired.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

xfersize specifies the data transfer size in bytes.

AGE1438_XFERSIZE_MIN selects the minimum allowable transfer size.

AGE1438_XFERSIZE_MAX selects the maximum allowable transfer size. *xfersize* must be a sub-multiple of *blocksize* or an error is generated.

AGE1438_XFERSIZE_DEF sets the default transfer size.

Note *xfersize* is reset by any subsequent change in the *blocksize* parameter and therefore must be specified after *blocksize*. See “age1438_data_setup” on page 66.

xfersizePtr points to the data transfer size in number of bytes.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than AGE1438_SUCCESS indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Commands which halt active measurements” on page 149, “age1438_init” on page 87, “age1438_data_setup” on page 66

age1438_driver_debug_level

Sets and gets the debug level.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_driver_debug_level(ViSession id, ViInt16 debugLevel);
```

```
ViStatus age1438_driver_debug_level_get(ViSession id, ViPInt16  
debugLevelPtr);
```

Description

This command allows you to set and get debug levels. Debug messages are sent to the application debugger using the Windows kernel function Output Debug String.

Note

This function only works under Windows.

This function only works with a debug build of the library.

Debug messages are received by the Microsoft Visual C++ debugger or can be received by the dbmon example program that comes with Microsoft Visual C++.

You can compile a DEBUG build by opening age1438_32.dsw, the Visual C++ project for the driver DLL, age1438_32.dll, and selecting the "age1438_32.dl-Win32 Debug" build configuration.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

debugLevel is the debug level.

debugLevelPtr points to the value of *debugLevel*.

Debug levels are defined as follows:

Debug Level	Description
AGE1438_DEBUG_LEVEL_0	Only output errors and algorithmic results
AGE1438_DEBUG_LEVEL_1	Add output of setup function calls
AGE1438_DEBUG_LEVEL_2	Add output of measurement function calls
AGE1438_DEBUG_LEVEL_3	Add output of status query function calls
AGE1438_DEBUG_LEVEL_4	Reserved
AGE1438_DEBUG_LEVEL_5	Add output of diagnostic function calls

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to "age1438_error_message" on page 74.

See Also

“age1438_init” on page 87

age1438_error_message

Returns error information obtained from function calls.

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_error_message(ViSession id, ViStatus statusCode, ViChar  
errorMessage[]);
```

Description

age1438_error_message takes an error return value generated by a function and translates it to a readable string. This function includes host errors as well as firmware errors.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

errorMessage represents the error message string up to 256 characters long.

Note

For this parameter you must allocate a character array of at least 256 characters AGE1438_STR_LEN_MIN, including the null byte, prior to calling this function in any programming language.

statusCode represents the instrument numeric error code.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“age1438_init” on page 87, “age1438_error_query” on page 75, “Error messages” on page 150

age1438_error_query

Queries the module for the first error in the queue.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_error_query(ViSession id, ViPint32 errorCode, ViChar  
errorMessage[]);
```

Description

age1438_error_query queries the module for the oldest error and returns the corresponding error message. This function does not report host errors that originate in the C library.

Parameters

id	is the VXI instrument session pointer returned by the age1438_init function.
errorCode	points to the instrument numeric error code.
errorMessage	points to the error message string up to 80 characters long. This message also indicates what function call generated the error.

Note

For this parameter you must allocate a character array of at least 256 characters AGE1438_STR_LEN_MIN, including the null byte, prior to calling this function in any programming language.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“age1438_init” on page 87, “age1438_error_message” on page 74

age1438_filter_setup

Sets the digital filter bandwidth and decimation filter parameters. This description also includes information on the following functions which set or query the decimation filter parameters individually:

age1438_filter_decimate selects an extra factor of 2 decimation.
age1438_filter_decimate_get gets current state of extra decimation
age1438_filter_bw selects a signal filter bandwidth.
age1438_filter_bw_get gets the signal filter bandwidth

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_filter_setup(ViSession id, ViInt16 sigBw, ViInt16 decimate);
ViStatus age1438_filter_decimate(ViSession id, ViInt16 decimate);
ViStatus age1438_filter_decimate_get(ViSession id, ViInt16 decimatePtr);
ViStatus age1438_filter_bw(ViSession id, ViInt16 sigBw);
ViStatus age1438_filter_bw_get(ViSession id, ViInt16 sigBwPtr);
```

Parameters

- id** is the VXI instrument session pointer returned by the **age1438_init** function.
- decimate** selects the data output sample rate. When this parameter is set to AGE1438_DECIMATE_OFF the output sample rate is:
- fs when $sigBw=0$, or
- $fs/2^{(sigBw-1)}$ when $sigBw>0$
- When *decimate* is set to AGE1438_DECIMATE_ON the output sample rate is reduced by an additional factor of two by discarding alternate samples.
- AGE1438_DECIMATE_SHIFT is like AGE1438_DECIMATE_ON but additional processing is performed that shifts the center frequency of zoomed data up by $fs/4$ and transforms the complex data stream into a real data stream without losing phase information. For consistent spectrum measurements the data scale value is doubled when using shift decimate.
- decimatePtr** points to the current value of the *decimate* parameter.
- sigBw** selects an alias protected signal filter bandwidth that is roughly $\pm fs/(2.56 \times 2^{(sigBw)})$ where fs is the ADC sample frequency. In zoom applications, where the center frequency is generally not zero, the zoom filter bandwidth is centered on the frequency programmed with the **age1438_frequency_setup** function. For baseband measurements the filter may equivalently be considered as a low pass filter of approximately bandwidth $fs/(2.56 \times 2^{(sigBw)})$ since the negative frequencies are generally of no interest. The valid range of sigBw is 0 through 18. When sigBw = 0, no digital filtering is applied to the signal and the module relies on the analog anti-alias filter to limit the signal bandwidth to $fs/2.56$.
- To more accurately calculate the bandwidth use the calculation $\pm fs \times k/2^{(sigBw)}$ where:
- k=.36 for .25 dB bandwidth
 - k=.44 for 3 dB bandwidth
 - k=.5 for 15 dB bandwidth

$k=.62$ for 110 dB bandwidth

AGE1438_SIG_BW_MAX sets *sigBw* to the maximum value and the filter bandwidth to the minimum.

AGE1438_SIG_BW_MIN sets *sigBw* to the minimum value and filter bandwidth to the maximum.

sigBwPtr points to the current value of the *sigBw* parameter.

Caution **Selecting AGE1438_DECIMATE_ON when *sigBw*=0 results in aliasing (garbage data) due to upper limit of the sampling frequency and, therefore, causes the SETUP_ERROR bit to be set.**

Selecting AGE1438_DECIMATE_SHIFT for non-zoomed data is not a useful configuration.

Comments

To ensure full alias-free operation the analog anti-alias filter (set by the **age1438_input_alias_filter** function) should be ON unless the application inherently bandlimits the input signal to less than $fs/2$. The analog anti-alias filter has a fixed bandwidth and thus is fully effective only when $fs \geq 100$ MHz. If a slower external ADC clock is used, an additional analog filter of the appropriate bandwidth may be required for full alias protection.

The decimation process used to reduce the output sample rate is driven from a "decimation counter" which keeps track of which samples to save and which ones to discard for each of the octave bandwidth reduction filter stages. In multi-module systems where synchronous sampling is required, the decimation counters in all the modules must be synchronous with each other. This condition can be forced by using the **age1438_filter_sync** function.

The following table lists parameter combinations (see also "age1438_data_setup" on page 66) which result in invalid measurement conditions:

Invalid parameter combinations

resolution (bits)	dataType	decimate	sigBw
12 or 24	REAL or COMPLEX	OFF or SHIFT	1
12 or 24	REAL or COMPLEX	ON or SHIFT	0
12 or 24	COMPLEX	any	0
24	REAL or COMPLEX	OFF	2
24	REAL or COMPLEX	any	0 or 1
12 or 24	COMPLEX	SHIFT	any

All other combinations are valid.

Example

Here are some bandwidth and sample rate results using the "k" calculation for bandwidth:
 $fs = 100$ MHz default internal ADC clock (all data in MHz)

Functions listed alphabetically

sigBw	Signal Bandwidth MHz		Sample Rate Msample/s	
	.25 dB	15 dB	Decimate OFF	Decimate ON
1	±18	±25	N/A	50
2	±9	±12.5	50	25
3	±4.5	±6.25	25	12.5
4	±2.25	±3.125	12.5	6.5
> 4	Continue to decimate by factors of two			

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Commands which halt active measurements” on page 149, “Default values” on page 152, “age1438_init” on page 87, “age1438_input_setup” on page 95, “age1438_clock_setup” on page 57, “age1438_frequency_setup” on page 83, “age1438_filter_sync” on page 79, “age1438_data_setup” on page 66, “Frequency and filtering” in chapter 3

age1438_filter_sync

Synchronizes the decimation counter for multi-module systems.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_filter_sync(ViSession id);
```

Description

This function causes the digital decimation counter to be reset by the next Sync line rising transition. By calling **age1438_filter_sync** for every HP E1438A module using a shared ADC clock, and then calling **age1438_meas_control** to cause a sync transition, the decimation counters are prepared to start at the same time. Once this is done the decimation counters stay synchronized as long as the same ADC clock is used. You do not need to resynchronize the decimation counters when the digital filter bandwidths are changed.

Note

Resetting the decimation counter causes a transient in the digital filters. The transient takes about 30 decimated output sample periods to decay 100 dB. See the step response graphs in the Technical Specifications for more detail.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

Comment

The correct procedure for using this command is:

1. Force all modules to idle using **age1438_meas_control**.
2. Call **age1438_filter_sync** for all modules.
3. Cause a sync transition with one module using **age1438_meas_control** without releasing force to idle.
4. Release force to idle on all modules.

If you also want to synchronize frequency or phase see **age1438_frequency_setup**. This procedure also applies to those commands for multi-module systems.

Example

The multichan.exe example program provides an example of how to correctly set up a multi-module system with synchronous filters.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to "age1438_error_message" on page 74.

Functions listed alphabetically

See Also

“age1438_init” on page 87, “age1438_filter_setup” on page 76, “age1438_frequency_setup” on page 83, “age1438_meas_control” on page 105, “Managing multiple modules” in chapter 3

age1438_frequency_center_raw

Provides a fast way to set the center frequency

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_frequency_center_raw(ViSession id, ViInt32 phase, ViInt32
interpolate);
```

```
ViStatus age1438_frequency_center_raw_get(ViSession id, ViInt32 phasePtr,
ViInt32 interpolatePtr);
```

Description

age1438_frequency_center_raw sets the center frequency without relying on the internal HP E1438A microprocessor to do floating point computations, since the internal microprocessor does not have a floating point co-processor.

Note

This command is not available in HP-VEE.

Parameters

id	is the VXI instrument session pointer returned by the age1438_init function.
phase	specifies the phase part of the frequency.
interpolate	specifies the interpolation part of the frequency.
phasePtr	points to the current actual value of <i>phase</i> .
interpolatePtr	points to the value of <i>interpolate</i> .

Comments

The following C code segment shows how to compute these parameters, where *freq* is (center frequency/sample rate):

```
static void rawFreq(double freq, long *phase, long *interpolate)
{
    long ph, in;
    freq *= -1048576.0;
    ph = (long)fabs(freq);
    in = (long)((fabs(freq) - (double)ph) * 48828125.0) + 0.5;
    if (freq < 0)
    {
        ph = -1 - ph;
        if (in != 0);
        in = 48828125 - in;
    }
    else;
    ph = ph + 1;
}
*phase = ph;
```

Functions listed alphabetically

```

    *interpolate = in;
    return;
}

```

The equivalent Visual Basic example follows:

```

Private Sub rawFreq(dblFreq as Double)
    Dim dblFx As Double
    Dim lngIn As Long
    Dim lngPh As Long

    dblFx = -1048576# * dblFreq
    lngPh = Fix(Abs(dblFx))
    lngIn = Fix((Abs(dblFx) - CDb1(lngPh)) * 48828125#) + 0.5)
    If (dblFx < 0) Then
        lngPh = (-1) - lngPh
        If (lngIn) Then
            lngIn = 48828125 - lngIn
        Else
            lngPh = lngPh + 1
        End If
    End If
    Call age1438_frequency_center_raw(lngId, lngPh, lngIn)

End Sub

```

Example

An example of this in VB is included in the Front Panel code and can be activated by changing the following declaration in frmMain of E1438.vbp.

```

Const constFreqCentRaw = False 'When TRUE, set center frequency with
                                'age1438_frequency_center_raw() instead of
                                'age1438_frequency_center()

```

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“age1438_init” on page 87, “age1438_frequency_setup” on page 83

age1438_frequency_setup

Sets all the zoom center frequency parameters. This description also includes information on the following functions which set or query frequency parameters individually:

age1438_frequency_center sets the center frequency
age1438_frequency_center_get gets the current center frequency
age1438_frequency_cmplxdc selects a complex baseband measurement
age1438_frequency_cmplxdc_get gets the state of the baseband measurement mode
age1438_frequency_sync prepares the module for a synchronous frequency change
age1438_frequency_sync_get gets the state of the synchronous change mode

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_frequency_setup(ViSession id, ViInt16 cmplxDC, ViInt16 sync,
ViReal64 centerFreq);
ViStatus age1438_frequency_center(ViSession id, ViReal64 centerFreq);
ViStatus age1438_frequency_center_get(ViSession id, ViReal64 centerFreqPtr);
ViStatus age1438_frequency_cmplxdc(ViSession id, ViInt16 cmplxDC);
ViStatus age1438_frequency_cmplxdc_get(ViSession id, ViInt16 cmplxDCPtr);
ViStatus age1438_frequency_sync(ViSession id, ViInt16 sync);
ViStatus age1438_frequency_sync_get(ViSession id, ViInt16 syncPtr);
```

Description

age1438_frequency_setup sets the center frequency of a zoomed measurement. The center of a frequency band of interest is converted to dc with this function. The frequency transition is phase continuous unless the center frequency is set to zero in which case the transition may be selected either to be phase continuous or phase reset. This function may also be used to synchronously change frequency in multiple-module systems.

Parameters

- | | |
|-------------------|--|
| id | is the VXI instrument session pointer returned by the age1438_init function. |
| cmplxDC | selects either a phase continuous or phase reset transition when <i>freq</i> =0.

AGE1438_CMPLXDC_OFF, combined with a frequency change to zero, causes phase to be reset to zero.

AGE1438_CMPLXDC_ON, combined with a frequency change to zero, does not reset the phase thereby generating a complex dc measurement at baseband. The state of this parameter does not affect any transition where <i>freq</i> is nonzero. Whether the real or complex data is saved and ultimately sent to the output port is determined by the age1438_data_type function |
| cmplxDCPtr | points to the current actual value of <i>cmplxDC</i> . |
| sync | when set to AGE1438_SYNC_OFF allows an immediate frequency change in single-module systems. |

Functions listed alphabetically

In multiple-module systems, setting this parameter to AGE1438_SYNC_ON prepares the modules for a frequency change, but does not actually bring about the change until the next ADC clock corresponding to the next assertion of the shared Sync signal. The Sync transition is generated by calling the **age1438_meas_control** function. Note that returning sync to OFF before the Sync signal transition has occurred forces an immediate asynchronous frequency change.

syncPtr

points to the value of *sync*.

centerFreq

supplies the center frequency normalized to the sample frequency. It is a number between -0.5 and +0.5, which is interpreted as a fraction of the sample frequency. *centerFreq* is the desired center frequency divided by the ADC sample frequency. For example, selecting 0.25 with a sample clock frequency of 100 MHz yields a center frequency of 25 MHz. The ADC sample frequency is returned by the **age1438_clock_fs_get** function. Negative frequencies select the negative image of the signal, which is spectrally inverted from the input signal.

AGE1438_CENT_FREQ_MIN selects the minimum allowable center frequency.

AGE1438_CENT_FREQ_MAX selects the maximum allowable center frequency.

AGE1438_CENT_FREQ_DEF sets the default center frequency.

centerFreqPtr

points to the current actual value of the center frequency (as a fraction of the sample clock frequency).

Comments

Although the *freq* parameter is a double precision floating point number, its effective resolution is $1/(2^{19} \times 5^{11})$. This allows exact specification of any multiple of 1 mHz when *fs*=100 or 102.4 MHz. The actual frequency is set to the nearest available value. This value is returned by the **age1438_frequency_center_get** function. In multi-module systems this value represents the pending value rather than the current value when a frequency change is incomplete due to a pending Sync signal transition.

In multiple-module systems it is often desirable to force the frequency change to occur synchronously in order to preserve the phase relationship of the LOs. You may accomplish this by setting the sync parameter to ON for all the modules which are to be changed.

In configurations involving synchronous operation of multiple HP E1438A modules, the **age1438_frequency_setup** function provides a mechanism to force all LOs to the same phase. You can do this by first setting the frequency to zero.

Example

The example program multichan.exe shows how to correctly perform synchronous frequency changes in a multi-module system.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to "age1438_error_message" on page 74.

See Also

“Default values” on page 152, “age1438_init” on page 87, “age1438_clock_setup” on page 57, “age1438_data_setup” on page 66, “age1438_clock_fs” on page 55, “age1438_meas_control” on page 105, “Frequency and filtering” in chapter 3

age1438_front_panel_clock_input

Specifies the source for the front panel clock. This description also includes the query function:

age1438_front_panel_clock_input_get

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_front_panel_clock_input(ViSession id, ViInt16 fpClock);
```

```
ViStatus age1438_front_panel_clock_input_get(ViSession id, ViPInt16  
fpClockPtr);
```

Note

This command should be used only for specialized custom clock requirements. Most useful clock setups can be supplied by age1438_clock_setup.

Description

This function selects a front panel clock source that is used to drive the analog to digital converter (ADC) for single module operation or when a module is used as the master ADC clock source for a multi-module system.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

fpClock AGE1438_CLOCK_OFF specifies no front panel source.

AGE1438_SMB_CLOCK specifies clock input from the front panel Intermodule Clock/SMB connectors.

AGE1438_BNC_CLOCK specifies clock input from the front panel Ext Clock/Ref BNC connector.

fpClockPtr returns a pointer to the current value of *fpClock*.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to "age1438_error_message" on page 74.

See Also

"Commands which halt active measurements" on page 149, "Default values" on page 152, "age1438_init" on page 87, "age1438_clock_setup" on page 57, "Using clock and sync" in chapter 3

age1438_init

Initializes the I/O driver for a module.

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_init(ViRsrc rsrcName, ViBoolean idQuery, ViBoolean  
resetInstr, ViPSession id);
```

Description

age1438_init must be the first routine called when you use the HP E1438A library. It establishes communication with the module and returns a module identification which is used with all subsequent functions involving this module. This function performs whatever initialization the I/O driver needs for the environment in which this library is running.

Parameters

- id** is a pointer to the VXI instrument Session identifier returned by this function for the module. This identifier is then used with all other functions which address this module. This value is not a VISA id and so cannot be used with VISA functions. Use `age1438_attrib_get` to get the VISA id.
- idQuery** set to `AGE1438_ON` verifies the identity of the instrument by checking the manufacturer ID and model number in the module's VXI register set.
If set to `AGE1438_OFF` the function does not verify the module's identity. It is helpful to disable the ID query if you want to use the driver with a similar module but do not need to modify the driver source code.
- resetInstr** places the module in the reset state when set to `AGE1438_ON`.
If set to `AGE1438_OFF`, the function disables the reset. Disabling the reset is useful for debugging in cases where resetting would take the instrument out of the state you want to test.
- rsrcName** specifies the interface and logical address. This descriptor varies depending on your I/O library.
An example of the descriptor form for the VISA I/O library is:
`VXI [Board] ::VXIlogical address [::INSTR]`

Comments

If you receive a resource descriptor error, see your I/O library documentation to determine the correct descriptor form.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Functions listed alphabetically

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Commands which halt active measurements” on page 149, “age1438_close” on page 63, “age1438_attrib_get” on page 53

age1438_input_autozero

Nulls out the input dc offset voltage (applies to baseband input configuration only).

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_input_autozero(ViSession id);
```

Description

age1438_input_autozero updates a table of dc offset corrections to be used with each input setup condition. The applicable correction from this table is automatically added to the input offset parameter to achieve the correct dc offset value. Because of the length of time needed to execute this function, it is not automatically called when the module is reset. Thus, the user program is responsible for explicitly initiating the auto zero. This function should be called at least once after the temperature of the module has stabilized. The interval between calls after that depends on the importance of dc accuracy in the user application. It is not necessary to call the auto zero function for every change of input setup parameters since the correction table maintains values for all setup conditions.

Note

Calling age1438_input_autozero aborts any measurement already in progress and eliminates LO phase coherence and filter synchronization in a synchronous multi-module system. See the age1438_filter_sync and age1438_frequency_sync functions for details on how to re-establish LO phase coherence and filter synchronization.

Calling this function deletes any saved state.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to "age1438_error_message" on page 74.

See Also

"Commands which halt active measurements" on page 149, "age1438_init" on page 87, "age1438_input_setup" on page 95, "age1438_filter_sync" on page 79, "age1438_frequency_setup" on page 83

age1438_input_offset

Sets the dc offset DAC setting for the current range. This description also includes the query:

age1438_input_offset_get

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_input_offset(ViSession id, ViInt16 coarseDac, ViInt16
    fineDac);
```

```
ViStatus age1438_input_offset_get(ViSession id, ViPInt16 coarseDacPtr,
    ViPInt16 fineDacPtr);
```

Description

These values are normally set by `age1438_input_autozero` so you generally would use this command only for special situations. The resultant values can be saved to non-volatile RAM with `age1438_input_offset_save`.

Each ac coupling range has a unique DAC setting. All dc coupling ranges use the same DAC setting as the highest range setting for ac coupling. The scaling between the coarse and fine DACs is approximately 100 to 1.

AGE1438_OFFS_DAC_MIN sets the minimum dc offset DAC setting.

AGE1438_OFFS_DAC_MAX sets the maximum dc offset DAC setting.

Parameters

id	is the VXI instrument session pointer returned by the age1438_init function.
coarseDac	sets values of 0 to 255.
fineDac	sets values of 0 to 255.
coarseDacPtr	returns a pointer to the current value of <i>coarseDac</i>
fineDacPtr	returns a pointer to the current value of <i>fineDac</i>

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “`age1438_error_message`” on page 74.

See Also

“Default values” on page 152, “`age1438_init`” on page 87, “`age1438_input_autozero`” on page 89, “`age1438_input_offset_save`” on page 91

age1438_input_offset_save

Saves all DAC offset settings to non-volatile RAM.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_input_offset_save(ViSession id);
```

Description

Use this command if you want DAC offset settings to persist past power-down.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“age1438_init” on page 87, “age1438_input_setup” on page 95, “age1438_input_offset” on page 90

age1438_input_range_auto

Performs auto-ranging (applies to baseband input configuration only).

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_input_range_auto(ViSession id, ViReal64 sec);
```

Description

age1438_input_range_auto sets the range of a HP E1438A to the lowest value that does not cause an ADC overload to occur. The algorithm starts at the lowest range and moves up until there is no ADC overload.

Note

Calling this function deletes any saved state.

Parameters

- id** is the VXI instrument session pointer returned by the **age1438_init** function.
- sec** is the time in seconds to take data at each range to insure that an overload is detected. Setting this parameter to 0.0 results in the time being set automatically according to an algorithm that depends on block size and filter bandwidth.
- AGE1438_RANGE_TIME_MIN selects the minimum autorange time.
- AGE1438_RANGE_TIME_MAX selects the maximum autorange time.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Commands which halt active measurements” on page 149, “age1438_init” on page 87, “age1438_input_setup” on page 95

age1438_input_range_convert

Converts the input range to volts.

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_input_range_convert(ViSession id, ViInt16 rangeIndex,  
ViPReal64 rangeVoltsPtr);
```

Description

age1438_input_range_convert converts the range of a HP E1438A

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

rangeIndex is the input range returned by **age1438_input_range_get**.

rangeVoltsPtr is the range in Volts.

Conversion values are as follows:

Variable	Range Index	Full Scale (dBm)	Full Scale Voltage (Vp)
AGE1438_RANGE_MAX	17		
AGE1438_RANGE_17	17	30	10.0
AGE1438_RANGE_16	16	27	7.08
AGE1438_RANGE_15	15	24	5.01
AGE1438_RANGE_14	14	21	3.55
AGE1438_RANGE_13	13	18	2.51
AGE1438_RANGE_12	12	15	1.78
AGE1438_RANGE_11	11	12	1.26
AGE1438_RANGE_10	10	9	.891
AGE1438_RANGE_9	9	6	.631
AGE1438_RANGE_8	8	3	.447
AGE1438_RANGE_7	7	0	.316
AGE1438_RANGE_6	6	-3	.224
AGE1438_RANGE_5	5	-6	.158
AGE1438_RANGE_4	4	-9	.112
AGE1438_RANGE_3	3	-12	.0794
AGE1438_RANGE_2	2	-15	.0562
AGE1438_RANGE_1	1	-18	.0398
AGE1438_RANGE_0	0	-21	.0282
AGE1438_RANGE_MIN	0		

Note

These values are approximate. For more accuracy use `age1438_data_scale_get`.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “`age1438_error_message`” on page 74.

See Also

“`age1438_init`” on page 87, “`age1438_input_setup`” on page 95, “`age1438_data_scale_get`” on page 65

age1438_input_setup

Sets all the analog input parameters. This description also includes information on the following functions which set or query the input parameters individually:

age1438_input_alias_filter selects or bypasses the built-in analog anti-alias filter
age1438_input_alias_filter_get gets the anti-alias filter state
age1438_input_coupling selects ac or dc input coupling
age1438_input_coupling_get get the input coupling type
age1438_input_range sets the full scale range
age1438_input_range_get gets the input range
age1438_input_signal connect/disconnect the input signal to the input amplifier
age1438_input_signal_get gets the input buffer amplifier state

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_input_setup(ViSession id, ViInt16 reserved, ViInt16 range,
    ViInt16 coupling, ViInt16 antiAlias, ViInt16 signal);
ViStatus age1438_input_alias_filter(ViSession id, ViInt16 antiAlias);
ViStatus age1438_input_alias_filter_get(ViSession id, ViInt16 antiAliasPtr);
ViStatus age1438_input_coupling(ViSession id, ViInt16 coupling);
ViStatus age1438_input_coupling_get(ViSession id, ViInt16 couplingPtr);
ViStatus age1438_input_range(ViSession id, ViInt16 range);
ViStatus age1438_input_range_get(ViSession id, ViInt16 rangePtr);
ViStatus age1438_input_signal(ViSession id, ViInt16 signal);
ViStatus age1438_input_signal_get(ViSession id, ViInt16 signalPtr);
```

Parameters

- id** is the VXI instrument session pointer returned by the **age1438_init** function.
- antiAlias** determines whether or not to use the built-in analog anti-alias filter. This parameter applies to the baseband input configuration only.
- AGE1438_ANTIALIAS_ON inserts a sharp-cutoff (9-pole) 40 MHz lowpass filter ahead of the analog-to-digital converter.
- AGE1438_ANTIALIAS_OFF replaces this filter with a soft-cutoff 3-pole low-pass filter. It is recommended that you leave the filter on at all times to insure band-limited, anti-aliased data.
- antiAliasPtr** points to the current value of the *antiAlias* parameter.
- coupling** specifies the ac or dc coupling mode of the input. This parameter applies to the baseband input configuration only.
- AGE1438_DC connects the input directly to the 50 Ohm buffer amplifier.
- AGE1438_AC inserts a 0.2 μ F capacitor between the input connector and the 50 Ohm buffer amplifier.
- couplingPtr** points to the current value of the *coupling* parameter for an HP E1438A or group of HP E1438As.
-

Functions listed alphabetically

range is a range index number between 0 and 17 which is transformed to a full scale voltage value.

AGE1438_RANGE_MAX sets the range to the maximum allowable.

AGE1438_RANGE_MIN sets the range to the minimum allowable.

Signal inputs with an absolute value larger than full scale generate an ADC overflow error.

Range values are as follows.

Variable	Range Index	Full Scale (dBm)	Full Scale Voltage (Vp)
AGE1438_RANGE_MAX	17		
AGE1438_RANGE_17	17	30	10.0
AGE1438_RANGE_16	16	27	7.08
AGE1438_RANGE_15	15	24	5.01
AGE1438_RANGE_14	14	21	3.55
AGE1438_RANGE_13	13	18	2.51
AGE1438_RANGE_12	12	15	1.78
AGE1438_RANGE_11	11	12	1.26
AGE1438_RANGE_10	10	9	.891
AGE1438_RANGE_9	9	6	.631
AGE1438_RANGE_8	8	3	.447
AGE1438_RANGE_7	7	0	.316
AGE1438_RANGE_6	6	-3	.224
AGE1438_RANGE_5	5	-6	.158
AGE1438_RANGE_4	4	-9	.112
AGE1438_RANGE_3	3	-12	.0794
AGE1438_RANGE_2	2	-15	.0562
AGE1438_RANGE_1	1	-18	.0398
AGE1438_RANGE_0	0	-21	.0282
AGE1438_RANGE_MIN	0		

These values are approximate. For more accuracy use `age1438_data_scale_get`.

rangePtr points to the current value of the *range* parameter.

signal determines whether or not the input signal is connected to the input amplifier.

AGE1438_SIGNAL_ON attaches the input signal to the 50 Ohm buffer amplifier.

AGE1438_SIGNAL_OFF redirects the input signal to a dummy 50 Ohm load, and feeds the buffer amplifier from an internally grounded 50 Ohm source resistance. The signal OFF setting is useful for making reference measurements without the signal applied. When using ac coupling the 0.2 μ F capacitor remains between the input connector and its 50 Ohm termination.

signalPtr points to the current value of the *signal* parameter.

Comments

To ensure full alias-free operation the analog anti-alias filter should be ON unless the application inherently bandlimits the input signal to less than $f_s/2$. The analog anti-alias filter has a fixed bandwidth and thus is fully effective only when $f_s \geq 100$ MHz. If a slower external ADC clock is used, an additional analog filter of the appropriate bandwidth may be required for full alias protection.

When using the analog anti-alias filter, you may need to set the range parameter higher than the actual range of the input signal. The reason for this is that step changes of input voltage cause an overshoot and ringing response at the output of the anti-alias filter. The peak overshoot actually exceeds the input voltage step by about 20%. The range setting must accommodate this overshoot to avoid an ADC overflow.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“age1438_init” on page 87, “age1438_input_autozero” on page 89, “age1438_input_range_auto” on page 92, “age1438_input_range_convert” on page 93, “age1438_data_scale_get” on page 65

age1438_interrupt_restore

Restores the interrupt masks to the setting last programmed with **age1438_interrupt_setup**.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_interrupt_restore(ViSession id);
```

Description

The interrupt masks set by the **age1438_interrupt_setup** function are cleared during the interrupt acknowledge cycle. This function restores the cleared interrupt masks.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“age1438_init” on page 87, “age1438_interrupt_setup” on page 99

age1438_interrupt_setup

Sets both interrupt parameters. This description also includes information on the following functions which query the interrupt parameters individually:

age1438_interrupt_mask_get gets the interrupt event mask
age1438_interrupt_priority_get gets the VME interrupt line

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_interrupt_setup(ViSession id, ViInt16 intrNum, ViInt16  
priority, ViInt16 mask);
```

```
ViStatus age1438_interrupt_mask_get(ViSession id, ViInt16 intrNum, ViPInt16  
maskPtr);
```

```
ViStatus age1438_interrupt_priority_get(ViSession id, ViInt16 intrNum,  
ViPInt16 priorityPtr);
```

Description

An HP E1438A has two independent interrupt generators, each capable of interrupting on one of the seven VME interrupt lines when a status condition specified by a mask occurs.

age1438_interrupt_setup sets the interrupt mask, priority and which of the two interrupt generators on the HP E1438A is to be used. The remaining **age1438_interrupt_** functions query the mask and priority individually.

Parameters

id	is the VXI instrument session pointer returned by the age1438_init function.
intrNum	is the number of the interrupt generator. The only values accepted are 0 and 1.
mask	specifies the mask of events on which to interrupt. This mask is created by ORing together the bits defined in bits 8 through 15 of the status register. The mask parameter format is 0xMM00 where MM represents the maskable upper 8 bits. The lower 8 bits cannot be used for generating interrupts, and therefore must be set to zero in this function call.
priority	specifies which of the seven VME interrupt lines to use. The only legal values are 0 through 7. Specifying 0 turns the interrupt off, while 7 is the highest priority.
maskPtr priorityPtr	contain the current value of the interrupt <i>mask</i> and <i>priority</i> parameters.

Comments

The mask is cleared during the interrupt acknowledge cycle. Therefore, the command must be sent again or restored with "age1438_interrupt_restore" on page 98 in order to generate further interrupts.

Example

The program interrupt.exe described in the example programs provides an example of how to use interrupts correctly.

Functions listed alphabetically

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Default values” on page 152, “age1438_init” on page 87, “age1438_status_get” on page 129, “age1438_attrib_get” on page 53, “age1438_interrupt_restore” on page 98

age1438_lbus_mode

Sets the local bus transmission mode. This description also includes the query:

age1438_lbus_mode_get gets the current local bus mode.

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_lbus_mode(ViSession id, ViInt16 lbusMode);
```

```
ViStatus age1438_lbus_mode_get(ViSession id, ViPInt16 lbusModePtr);
```

Description

age1438_lbus_mode sets the local bus to either generate, append, insert or pipeline data. The data port must be set to the local bus with the **age1438_data_port** function (See “age1438_data_setup” on page 66) before these modes take effect.

Parameters

id

is the VXI instrument session pointer returned by the **age1438_init** function.

lbusMode

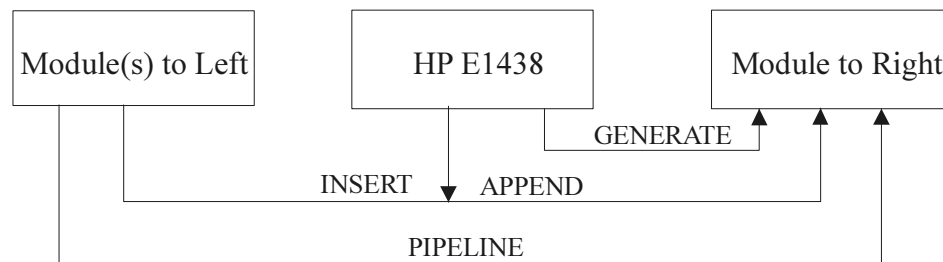
selects the transmission mode of the local bus when it is enabled by the **age1438_data_port** function.

AGE1438_GENERATE forces the module at *id* to generate data only, not passing through data from other modules on the local bus.

AGE1438_APPEND causes the HP E1438A to pass data through from modules on its left and append its data to the end.

AGE1438_INSERT causes the HP E1438A to place its data on the local bus and then pass data through from modules on its left.

AGE1438_PIPELINE causes the HP E1438A to pipe data through from modules on its left without appending or inserting its own data. The state of this parameter is unaffected by switching back and forth between the local bus and the VME backplane with the **age1438_data_port** function.



lbusModePtr

points to the current value of the *lbusMode* parameter.

Return Value

Functions listed alphabetically

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Default values” on page 152, “age1438_init” on page 87, “age1438_data_setup” on page 66

age1438_lbus_reset

Resets the local bus. This description also includes the query:

age1438_lbus_reset_get gets the current local bus reset state

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_lbus_reset(ViSession id, ViInt16 lbusReset);
```

```
ViStatus age1438_lbus_reset_get(ViSession id, ViPInt16 lbusResetPtr);
```

Description

In order to avoid glitches in the local bus data, the local bus interface has strict requirements as to the order in which modules in a VXI mainframe have their local bus interface reset. Upon power-up or whenever any single module in the mainframe is put into a reset state, all modules should be placed into the reset state from left to right. Then all modules can be take out of reset from left to right.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

lbusReset puts the HP E1438A's local bus into reset or takes it out of reset.

AGE1438_LBUS_RESET_ON puts the HP E1438A's local bus into reset while AGE1438_LBUS_RESET_OFF takes the HP E1438A out of reset.

lbusResetPtr points to the current value of the *lbusReset* parameter.

Example

When HP E1438As are used with the E1485 measurement controller, the E1485 must be reset while all of the HP E1438As are being held in reset to avoid initial glitches in the local bus data. The HP E1438As should be taken out of reset only after the first **age1438_meas_control** release is issued. The correct way to reset the local bus is as follows:

```
lbus_control(LBUS_CTL_RESET, 0); /* reset the E1485 lbus */
for all id{
    age1438_lbus_reset(id, AGE1438_ON); /* hold HP E1438As in reset */
}

/*Set LBUS mode for all modules */
for all id{
    age1438_meas_control(id, AGE1438_RELEASE, AGE1438_ASSERT);
    /* first arming */
    age1438_lbus_reset(id, AGE1438_OFF);
    /* remove reset from HP E1438As, has no effect after first time */
}
lbus_control(LBUS_CTL_RESET, 1); /* unreset the E1485 lbus */
```

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Functions listed alphabetically

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Default values” on page 152, “age1438_init” on page 87

age1438_meas_control

Initiates and controls measurements in multi-module systems.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_meas_control(ViSession id, ViInt16 idle, ViInt16 sync);
```

Description

age1438_meas_control explicitly controls the measurement state.

Parameters

id	is the VXI instrument session pointer returned by the age1438_init function.
idle	selects the condition of the Idle state. AGE1438_ASSERT holds the module in the Idle state. AGE1438_RELEASE reverses a previous AGE1438_ASSERT or ensures that no forced Idle is active.
sync	age1438_meas_control also changes the state of the Sync signal, which is used to arm or trigger an HP E1438A module. In systems containing multiple HP E1438A modules the Sync signal is used to arm or trigger all modules simultaneously, and also to synchronize decimation counters and local oscillators among the HP E1438A modules. selects the state of the sync signal. AGE1438_ASSERT causes the module to assert the Sync signal. AGE1438_RELEASE causes the module to release the Sync signal. When parameters of the age1438_clock_setup function which enable sync output are selected the module shares the sync signal with other HP E1438A modules. If any one of these modules asserts this shared Sync signal it then becomes asserted for all of them. All modules must release it before the shared Sync signal is released. Asserting then releasing the Sync line is used to start a measurement, load local oscillator values, or take a digital filter out of reset. These situations require a Sync line transition but do not require that the Sync line be held in a asserted state.

Note

When the Sync line is asserted, it remains asserted for an adequate number of ADC clock cycles to ensure that the signal effect propagates to all the modules in the system. You can determine when the command is completed by looking as the Sync/Idle Complete bit in the Status Register.

Comments

See “The measurement loop” in chapter 3 for details on how a measurement progresses through the four states.

This function performs the following sequence:

1. **Waits for both the AGE1438_STATUS_HARDWARE_SET and AGE1438_STATUS_SYNC_COMPLETE bits to be set.**

Functions listed alphabetically

2. Returns **AGE1438_STATUS_WAIT_TIMEOUT** if more than three seconds elapses in step 1.
3. Returns **AGE1438_SETUP_ERROR** if **AGE1438_STATUS_SETUP_ERROR** was detected in step 1.
4. Writes data to the control register as prescribed by arguments to the function.
5. Clears the overload count maintained by the API. See “Comments on Overload” on page 113
6. Waits for **AGE1438_STATUS_SYNC_COMPLETE**.
7. Returns **AGE1438_SYNC_NOT_COMPLETE** if more than three seconds elapse in step 6, otherwise it returns **AGE1438_SUCCESS**.

Special conditions prevail during the Measure state. If programmed for block mode operation in the Measure state, the module asserts the Sync signal (regardless of the **age1438_meas_control** sync parameter setting) until a complete block of data has been collected and is available to the I/O port. When the shared Sync signal is released, indicating that all block mode data collection is finished, all block mode modules move synchronously to the idle state. In continuous mode the module releases the Sync signal immediately after moving into the measure state. This allows the **age1438_meas_control** function to manipulate the Sync signal to cause synchronous changes to LO frequency while a continuous measurement is in progress. In continuous mode a module moves to the idle state only if explicitly programmed to do so or whenever the FIFO data buffer overflows.

In addition to controlling the progression through the four module states, the Sync signal is used to allow for synchronizing the decimation counters and local oscillators of multiple HP E1438A modules and synchronizing the $f_s/10$ clock during external sampling. This is done by calling **age1438_filter_sync** and/or **age1438_frequency_sync** prior to asserting Sync with **age1438_meas_control**. This is normally done with the module in the Idle state; however, the center frequency can also be changed in the Measure state with **age1438_frequency_sync** if the modules are all programmed for continuous (non-block mode) data collection.

If all modules in a multi-module system are in the Idle state when the **age1438_meas_control** sync parameter is asserted, the LO frequency is updated and the next measurement is armed. If all modules are in the measurement state in continuous mode, the LO frequency is synchronously updated, and the measurement continues. In continuous mode you should ensure that all modules are in the same state, either the Idle state or the Measure state, before using **age1438_meas_control** to assert Sync. Otherwise some modules re-arm while others continue the current measurement. In block mode the sync assertion is ignored unless all modules are in the Idle state.

The **age1438_meas_control** function assures that a single module is in a valid state by checking that the *hardware complete* and *sync valid* bits in the status register are both true. In synchronous multi-module systems you should use the **age1438_wait** function for each module to assure a valid state in non-master modules within a synchronous group.

In the case of systems made up of multiple mainframes you must be aware that only modules in the mainframe containing the master module, as defined by **age1438_clock_setup**, may assert sync. Any sync asserted in other mainframes is ignored by modules in all mainframes. This is true only for rear panel sync. Front panel sync is not sensitive to master mainframe designation.

Example

The program multichan.exe described in the example programs provides an example of how to correctly set up a multi-module measurement using **age1438_meas_control** to initiate state transitions.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Commands which halt active measurements” on page 149, “Default values” on page 152, “age1438_init” on page 87, “age1438_status_get” on page 129, “age1438_data_setup” on page 66, “age1438_filter_sync” on page 79, “age1438_frequency_setup” on page 83, “age1438_clock_setup” on page 57, “age1438_wait” on page 144, “age1438_read” on page 112, “Managing multiple modules” in chapter 3, “The measurement loop” in chapter 3

age1438_meas_init

Initiates a measurement without first checking for valid hardware setup.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_meas_init(ViSession id);
```

Description

age1438_meas_init provides an easy way to initiate a measurement in a single module.

Note

This command is slightly faster and slightly less robust than `age1438_meas_start`.

Parameters

id

is the VXI instrument session pointer returned by the **age1438_init** function.

Comments

See “The measurement loop” in chapter 3 for details on how a measurement progresses through the four states.

This function performs the following sequence:

- 1. Clears the overload count maintained by the API. See “Comments on Overload” on page 113**
- 2. Moves the module to Idle state.**
- 3. Generates a Sync transition which moves the module to the Arm state.**
- 4. Always returns AGE1438_SUCCESS (no error conditions can be detected by this function).**

Return Value

This function always returns **AGE1438_SUCCESS** and does not return any error conditions.

See Also

“Commands which halt active measurements” on page 149, “`age1438_init`” on page 87, “`age1438_meas_start`” on page 109, “`age1438_meas_control`” on page 105, “`age1438_status_get`” on page 129

age1438_meas_start

Checks for valid hardware setup and then initiates a measurement.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_meas_start(ViSession id);
```

Description

age1438_meas_start provides an easy way to initiate a measurement in a single module system. This command waits for a valid hardware setup, then, if the instrument is in a valid state, performs the equivalent of **age1438_meas_init**.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

Comments

See “The measurement loop” in chapter 3 for details on how a measurement progresses through the four states.

This function performs the following sequence:

1. **Waits for AGE1438_STATUS_HARDWARE_SET bit to be set.**
2. **Returns AGE1438_START_ERROR if more than three seconds elapses in step 1.**
3. **Returns AGE1438_SETUP_ERROR if AGE1438_STATUS_SETUP_ERROR was detected in step 1.**
4. **Performs age1438_meas_init and returns AGE1438_SUCCESS.**

Example

The program `acvolts.exe` described in the example programs provides an example of how to initiate a very simple measurement using **age1438_meas_start**.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “`age1438_error_message`” on page 74.

See Also

“Commands which halt active measurements” on page 149, “`age1438_meas_control`” on page 105, “`age1438_meas_init`” on page 108, “`age1438_status_get`” on page 129, “`age1438_read`” on page 112, “The measurement loop” in chapter 3

age1438_options_get

Identifies module options.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_options_get(ViSession id, ViChar options[]);
```

Description

Returns a list of options separated by commas.

Parameters

id	is the VXI instrument session pointer returned by the age1438_init function.
options	returns a string of up to 256 characters. For example "144" indicates option 144 (memory) is installed.

Note

For this parameter you must allocate a character array of at least 256 characters AGE1438_STR_LEN_MIN, including the null byte, prior to calling this function in any programming language.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“age1438_init” on page 87

age1438_product_id_get

Gets the module's product identification string.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_product_id_get(ViSession id, ViChar productId[]);
```

Description

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

productId returns the module ID such as "HP E1438A".

Note

For this parameter you must allocate a character array of at least 256 characters AGE1438_STR_LEN_MIN, including the null byte, prior to calling this function in any programming language.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to "age1438_error_message" on page 74.

See Also

"age1438_init" on page 87

age1438_read

Reads scaled 32-bit float data from the VME backplane register. This description also includes the following function:

age1438_read64 reads scaled 64-bit float data, implemented specifically for VEE applications.

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_read(ViSession id, ViReal32 data[], ViInt32 sampleCount,
ViPInt16 overloadPtr);
```

```
ViStatus age1438_read64(ViSession id, ViReal64 data[], ViInt32 sampleCount,
ViPInt16 overloadPtr);
```

Description

age1438_read returns a block of floating point data from the HP E1438A that has been scaled to be in volts. The function waits for a block of data to be ready before attempting to read the block.

These functions can only read data from the VME backplane register. The data port of the HP E1438A must be set to AGE1438_VME by the **age1438_data_port** function for these functions to be effective.

This function performs the following sequence:

1. Checks for AGE1438_STATUS_READ_BLOCK and AGE1438_STATUS_OVERLOAD.
2. If there is an overload then the overload count maintained by the API is incremented.
3. If a block of data is NOT ready:
 - a. the function immediately returns the current measurement state and
 - b. the value of the overload argument is set to AGE1438_OFF.
4. If a block of date IS ready:
 - a. data is read from the module,
 - b. converted to a floating point number and scaled,
 - c. the function returns any errors that were encountered when reading the data,
 - d. the value of the overload argument is set to AGE1438_ON, and
 - e. the overload count maintained by the API is set to zero.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

- data** is a pointer to the array into which the floating point data is to be placed. Be sure to allocate sufficient storage space at this location to hold the full data record as determined by the *sampleCount* parameter. Note that when the module is set to complex data type, the output data record contains $2 \times \text{sampleCount}$ floating point values. For real data the record contains *sampleCount* floating point values.
- sampleCount** for **age1438_read** *sampleCount* is the number real or complex data values to read. Real data is one 32-bit floating point value. Complex data is made up of two 32-bit floating point values comprising the real and imaginary values.
- for **age1438_read64** *sampleCount* is the number real or complex data values to read. Real data is one 64-bit floating point value. Complex data is made up of two 64-bit floating point values comprising the real and imaginary values.
- This should never be set larger than the blocksize parameter set in the **age1438_data_blocksize** function. In continuous data collection mode, *sampleCount* should be set equal to blocksize to ensure that the entire data block is read out.
- overloadPtr** returns an overload indicator. The way to properly use the overload argument for the **age1438_read** or **age1438_read64** function is this:
1. Set up the hardware.
 2. Call **age1438_meas_start**.
 3. Call **age1438_read** or **age1438_read64**.
If data is not available, the read function returns immediately with one of the following return values, and the overload indication is AGE1438_OFF:
 - AGE1438_NO_DATA_MEASUREMENT_IN_PROGRESS**
 - AGE1438_NO_DATA_MEASUREMENT_PAUSED**
 - AGE1438_NO_DATA_WAITING_FOR_TRIGGER**
 - AGE1438_NO_DATA_WAITING_FOR_ARM**
 When data is available, AGE1438_SUCCESS is returned and the overload value reflects whether an overload was encountered for the given data block.
 4. In continuous mode, subsequent data blocks can be read by calling a **age1438_read** or **age1438_read64** function again (**age1438_meas_start** should not be called again).
 5. When used in this way, an overload indication is given for each and every data block read in block mode. In continuous mode the overload indicator only means there was an overload sometime after calling **age1438_meas_start**.

Comments on Overload

Since reading the status register clears the overload bit, overloads are tracked at the API level.

In block mode, you receive the overload indication on a block-by-block basis by calling **age1438_meas_start** and **age1438_read** in sequence.

In continuous mode, depending on the effective sample rate of the instrument and how often data is read, an overload indication returned by **age1438_read** may or may not correspond to the data returned. The overload indication only means that an overload has occurred since the most recent call to **age1438_meas_init**, **age1438_meas_init**, or

Functions listed alphabetically

age1438_read, whichever was issued last. You should be aware that it is likely that the reported overload occurred in data which has been acquired in the module, is waiting in the FIFO, but has not yet been read.

Return Value

AGE1438_SUCCESS
AGE1438_NO_DATA_MEASUREMENT_IN_PROGRESS
AGE1438_NO_DATA_MEASUREMENT_PAUSED
AGE1438_NO_DATA_WAITING_FOR_TRIGGER
AGE1438_NO_DATA_WAITING_FOR_ARM

See Also

“age1438_init” on page 87, “age1438_data_setup” on page 66, “age1438_meas_start” on page 109, “age1438_meas_init” on page 108, “age1438_meas_control” on page 105, “age1438_status_get” on page 129, “The measurement loop” in chapter 3

age1438_read_raw

Reads raw, unscaled data from the VME backplane register.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_read_raw(ViSession id, ViPInt16 data[], ViInt32 wordCount,  
ViPInt16 overloadPtr);
```

Description

age1438_read_raw returns a block of raw, unscaled integer data from the FIFO.

This function can only read data from the VME backplane register. The data port of the HP E1438A must be set to AGE1438_VME by the **age1438_data_port** function for this function to be effective.

This function performs the following sequence:

1. Checks for AGE1438_STATUS_READ_BLOCK and AGE1438_STATUS_OVERLOAD.
2. If there is an overload then the overload count maintained by the API is incremented.
3. If a block of data is NOT ready:
 - a. the function immediately returns the current measurement state and
 - b. the value of the overload argument is set to AGE1438_OFF.
4. If a block of data IS ready:
 - a. data is read from the module,
 - b. the function returns any errors that were encountered when reading the data,
 - c. the value of the overload argument is set to AGE1438_ON, and
 - d. the overload count maintained by the API is set to zero.

Parameters

id	is the VXI instrument session pointer returned by the age1438_init function.
data	is a pointer to the array into which the raw data record is to be placed. Be sure to allocate sufficient storage space to hold the full data record as determined by the <i>wordCount</i> parameter.
wordCount	<i>wordCount</i> is the total number of data values to read into the data array from the HP E1438A output FIFO. The maximum <i>wordCount</i> depends on the blocksize, data type, and data resolution parameter settings.

Functions listed alphabetically

Data type	Resolution (bits)	Words per sample
REAL	12	2
REAL	24	4
COMPLEX	12	4
COMPLEX	24	8

In continuous data collection mode, *wordCount* should be set equal to the maximum possible *wordCount* to ensure that the entire data block is read out.

overloadPtr

returns an overload indicator. See “Comments on Overload” on page 113. The way to properly use the overload argument for the **age1438_read_raw** function is this:

1. Set up the hardware.
2. Call **age1438_meas_start**.
3. Call **age1438_read_raw**.

If data is not available, the read function returns immediately with one of the following return values, and the overload indication is AGE1438_OFF:

AGE1438_NO_DATA_MEASUREMENT_IN_PROGRESS
AGE1438_NO_DATA_MEASUREMENT_PAUSED
AGE1438_NO_DATA_WAITING_FOR_TRIGGER
AGE1438_NO_DATA_WAITING_FOR_ARM

When data is available, AGE1438_SUCCESS is returned and the overload value reflects whether an overload was encountered for the given data block.

4. In continuous mode, subsequent data blocks can be read by calling the **age1438_read_raw** function again (**age1438_meas_start** should not be called again).
5. When used in this way, an overload indication is given for each and every data block read in block mode. In continuous mode the overload indicator only means there was an overload sometime after calling **age1438_meas_start**.

Note

The primary purpose of the `age1438_read_raw` function is to provide the fastest possible way to read blocks of data from the module. Since this command does not perform data scaling after reading data it may save 10-20% of the overall `age1438_read` time, depending on the host computer in use. The resulting data ordering is dependent on the data type and resolution. The array may be cast as a long before reading the data to provide whole words.

Example

A declaration in the Front Panel example program can be changed to exercise `age1438_read_raw()` in `frmMain` of `e1438.vbp`:

```
Const constFreqCentRaw = False 'when TRUE, use age1438_read_raw()
                                'instead of age1438_read
```

Return Value

AGE1438_SUCCESS
AGE1438_NO_DATA_MEASUREMENT_IN_PROGRESS
AGE1438_NO_DATA_MEASUREMENT_PAUSED
AGE1438_NO_DATA_WAITING_FOR_TRIGGER
AGE1438_NO_DATA_WAITING_FOR_ARM

See Also

“age1438_init” on page 87, “age1438_read” on page 112, “age1438_status_get” on page 129,
“age1438_data_setup” on page 66, “The measurement loop” in chapter 3

age1438_reference_clock

Selects the source of the reference clock. This description also includes the query function:

age1438_reference_clock_get

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_reference_clock(ViSession id, ViInt16 refClock);
```

```
ViStatus age1438_reference_clock_get(ViSession id, ViPInt16 refClockPtr);
```

Note

This command should be used only for specialized custom clock requirements. Most useful clock setups can be supplied by age1438_clock_setup.

Parameters

- id** is the VXI instrument session pointer returned by the **age1438_init** function.
- refClock** AGE1438_FRONT_PANEL_CLOCK specifies the front panel clock be uses as the reference clock.
AGE1438_VXI_CLOCK specifies that the VXI (rear panel) clock be used as the reference clock.
- refClockPtr** Returns a pointer to the current value of *refClock*.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Default values” on page 152, “age1438_init” on page 87, “age1438_clock_setup” on page 57, “age1438_vxi_clock_output” on page 143, “age1438_front_panel_clock_input” on page 86, “Using clock and sync” in chapter 3

age1438_reference_prescaler

Selects prescaling of the reference clock. This description also includes the query function:

age1438_reference_prescaler_get

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_reference_prescaler(ViSession id, ViInt16 refPrescaler);
```

```
ViStatus age1438_reference_prescaler_get(ViSession id, ViInt16  
refPrescalerPtr);
```

Note

This command should be used only for specialized custom clock requirements. Most useful clock setups can be supplied by age1438_clock_setup.

Description

This function should generally be left in the default mode. The alternate mode applies to a different model of the module.

Parameters

- id** is the VXI instrument session pointer returned by the **age1438_init** function.
- refPrescaler** AGE1438_PRESCALE_BY_1 divides the reference clock by one.
AGE1438_PRESCALE_BY_4 divides the reference clock by four.
- refPrescalerPtr** Returns a pointer to the current value of *refPrescalerPtr*.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Default values” on page 152, “age1438_init” on page 87, “age1438_clock_setup” on page 57, “age1438_front_panel_clock_input” on page 86, “age1438_vxi_clock_output” on page 143, “Using clock and sync” in chapter 3

age1438_reset

Places the module in a known state.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_reset(ViSession id);
```

Description

age1438_reset returns the module's internal data structures to the power-up state but does not reset the hardware. This function can be called separately by this function, or may be selected in conjunction with the **age1438_init** function.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

Comments

The reset values are listed in "Default values" on page 152.

This command takes about 100 ms to complete.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to "age1438_error_message" on page 74.

See Also

"Commands which halt active measurements" on page 149, "age1438_init" on page 87, "age1438_reset_hard" on page 121

age1438_reset_hard

Resets the module to the power-up state.

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_reset_hard(ViSession id);
```

Description

age1438_reset_hard resets the module's firmware and hardware including the processor.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

Comments

The reset values are listed in "Default values" on page 152. In addition, the hardware registers, including the save register, are reset to the power-up state.

This command takes about 5 seconds to complete.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to "age1438_error_message" on page 74.

See Also

"Commands which halt active measurements" on page 149, "age1438_init" on page 87, "age1438_reset" on page 120

age1438_revision_query

Returns strings that identify the date of the firmware revision.

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_revision_query(ViSession id, ViChar driverRev[], ViChar
instrRev[]);
```

Parameters

- id** is the VXI instrument session pointer returned by the **age1438_init** function.
- driverRev** returns the date and time of the module's driver revision in the form:
a.dd.dd OPERS Ddd Mmm Date hh:mm:ss YYYY where Ddd is the abbreviated day of the week and Date is an integer from 1 to 31
- instRev** returns the date, time, and board number of the module's firmware revision in the form:
mm-dd-yyyy hh:mm 01Bd: xxxx; 02Bd:xxxx where xxxx is a manufacturer's date code used for service purposes.

Note

For this parameter you must allocate a character array of at least 256 characters AGE1438_STR_LEN_MIN, including the null byte, prior to calling this function in any programming language.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to "age1438_error_message" on page 74.

See Also

"age1438_init" on page 87

age1438_self_test

Performs a self-test and returns the result of that self test.

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_self_test(ViSession id, ViPInt16 testResult, ViChar
    testMessage[]);
```

Description

The HP E1438A self test includes the following tests:

- Digital: verifies the integrity of paths from LO chip through the filters to the memory controller.
- Serial: verifies the integrity of serial setup path for each board.
- Memory: fills the entire DRAM then verifies that all the data is correct.
- Analog: verifies that auto zero adjust is working and that the input is triggering.
- Clock: verifies that 100 MHz and 102.4 MHz oscillators are working properly.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

testMessage points to the self test status message string up to 256 characters long.

Note **For this parameter you must allocate a character array of at least 256 characters AGE1438_STR_LEN_MIN, including the null byte, prior to calling this function in any programming language.**

testResult points to the instrument numeric error code.

Possible test result values are:

Error Message	Error Code (hex)	Self Test Status Message
AGE1438_ST_SUCCESS	0x000	self test successful
AGE1438_ST_HARDWARE_FAIL	0x001	hardware failure
AGE1438_ST_SERIAL1_FAIL	0x002	serial 1 test failed
AGE1438_ST_SERIAL2_FAIL	0x004	serial 2 test failed
AGE1438_ST_CLOCK1_FAIL	0x008	100 MHz clock test failed
AGE1438_ST_CLOCK2_FAIL	0x010	102.4 MHz clock test failed
AGE1438_ST_MEMORY_FAIL	0x020	memory test failed
AGE1438_ST_DIGITAL1_FAIL	0x040	real data path failed

Functions listed alphabetically

Error Message	Error Code (hex)	Self Test Status Message
AGE1438_ST_DIGITAL2_FAIL	0x080	complex data path failed
AGE1438_ST_ANALOG_FAIL	0x100	analog test failed
AGE1438_ST_EXECUTION_ERR	0x400	self-test execution error

Note

The required completion time for self-test is up to 25 seconds depending on the amount of memory in the module.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Commands which halt active measurements” on page 149, “age1438_init” on page 87

age1438_serial_number

Sets the serial number of the module. This description also includes the query function:

age1438_serial_number_get

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_serial_number(ViSession id, ViChar serialNum[]);
```

```
ViStatus age1438_serial_number_get(ViSession id, ViChar serialNum[]);
```

Caution

This command is to be used for repair purposes only.

Description

This command is used to reassign a serial number after a module has been serviced.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

serialNum sends or gets a serial number of less than 16 characters

Note

For this parameter you must allocate a character array of at least 256 characters AGE1438_STR_LEN_MIN, including the null byte, prior to calling this function in any programming language.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to "age1438_error_message" on page 74.

See Also

"age1438_init" on page 87

age1438_smb_clock_output

Specifies which clock to output from the SMB clock connectors. This description also includes the query function:

age1438_smb_clock_output_get

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_smb_clock_output(ViSession id, ViInt16 smbClock);
```

```
ViStatus age1438_smb_clock_output_get(ViSession id, ViInt16 smbClockPtr);
```

Note

This command should be used only for specialized custom clock requirements. Most useful clock setups can be supplied by age1438_clock_setup.

Description

This function selects the source of the output for the front panel SMB clock connectors.

Parameters

id	is the VXI instrument session pointer returned by the age1438_init function.
smbClock	AGE1438_CLOCK_OFF specifies no output from the SMB clock connectors. AGE1438_DIVIDED_ADC_CLOCK specifies that the divided ADC clock be output from the SMB clock connectors. AGE1438_BNC_CLOCK specifies that the BNC input be output from the SMB clock connectors. AGE1438_VXI_CLOCK specifies the VXI clock be output from the SMB clock connectors.
smbClockPtr	Returns a pointer to the current value of <i>smbClock</i> .

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Default values” on page 152, “age1438_init” on page 87, “age1438_clock_setup” on page 57, “age1438_front_panel_clock_input” on page 86, “Using clock and sync” in chapter 3

age1438_state_recall

Recalls a module's previous instrument state.

age1438_state_recall

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_state_recall(ViSession id);
```

Description

This function aborts any active measurement and recalls the instrument state previously saved by `age1438_state_save`. This function requires >100 ms to complete.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “`age1438_error_message`” on page 74.

See Also

“Commands which halt active measurements” on page 149, “`age1438_init`” on page 87, “`age1438_state_save`” on page 128

age1438_state_save

Saves the module's current instrument state.

```
age1438_state_save
```

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_state_save(ViSession id);
```

Description

This function may be used to save a state to which you want to return later. `age1438_reset` does not change a saved state. The state is not saved to non-volatile RAM.

Note

The saved state is lost by issuing the following commands: `age1438_input_range_auto`, `age1438_input_autozero`, `age1438_self_test`, and `age1438_reset_hard`.

Parameters

id is the VXI instrument session pointer returned by the `age1438_init` function.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “`age1438_error_message`” on page 74.

See Also

“`age1438_init`” on page 87, “`age1438_state_recall`” on page 127

age1438_status_get

Reads status register information for the module.

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_status_get(ViSession id, ViPInt16 statusPtr);
```

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

statusPtr points to the status word. The bits are defined below:

Status Bit	Definition	Description
0-1	MEAS_STATUS	These two bits indicate the current state of the measurement loop as shown in the table below. See “The measurement loop” in chapter 3 for more information about these states: 00 IDLE 01 ARM 10 MEASURE 11 TRIGGER
2	STATUS_PASSED	Passed: This bit is always set to 1
3	STATUS_READY	This bit is set when the module is ready after power-on. See the VXIbus Specifications for more information.
4	STATUS_UNUSED1	Reserved for future use
5	STATUS_UNUSED2	Reserved for future use
6	STATUS_SETUP_ERROR	Setup error: An invalid parameter value was requested. If an invalid block size was requested, the closest valid block size is used until a change to an interrelated parameter makes the requested block size valid. If a data resolution, data type, filter bandwidth, trigger delay, or filter decimation parameter was requested which would result in an inability to make a measurement, the previous valid parameter is used until a change to an interrelated parameter makes the requested parameter valid
7	STATUS_SYNC_COMPLETE	Sync/Idle Complete: This bit is set when the most recent user-initiated Sync or Idle change has propagated through to all modules in a system. The change is a result of asserting SYNC or forcing Idle via the Control Register or issuing a meas_control command or function
8	STATUS_READ_VALID	This flag is set whenever there is at least one valid 16-bit data word available to be read via the VME data register. Not valid when using the local bus data port.

Functions listed alphabetically

Status Bit	Definition	Description
9	STATUS_BLOCK_READY	This bit is set in continuous mode whenever the size of the data in the FIFO is equal to or greater than the block size register. Check this bit before reading data to insure that a block of data may be transferred without fear of running out of data, thereby holding up the Local bus or VME bus. This bit is set in block mode whenever the module has successfully taken a block size number of samples since the most recent trigger and is cleared when the block is read out, when force to Idle is asserted, or when the module is armed for another measurement.
10	STATUS_ARMED	This bit is set whenever the module is in the Trigger state, or is in the Arm state and has satisfied its pre-trigger requirements. When this bit is set, the module releases the VXI Sync line. Once all modules release the Sync line, then all modules go to the Trigger state.
11	STATUS_FIFO_OVERFLOW	FIFO Overflow: This bit set when the FIFO buffer overflows in continuous mode
12	STATUS_OVERLOAD	This bit is set whenever the ADC converts a sample that exceeds the range of the ADC. The bit is cleared when the Status register is read.
13	STATUS_ERROR_QUEUE	This bit is set whenever there is an error in the error queue. It is cleared when the error queue is empty
14	STATUS_MODID	A (1) in this field indicates that the module is not selected via the P2 MODID line. A (0) indicates that the module is selected by a high state on the P2 MODID line
15	STATUS_HARDWARE_SET	This bit is set when all commands are complete and the hardware has been set

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“age1438_init” on page 87

age1438_sync_clock

Selects the source of the sync clock. This description also includes the query function:

age1438_sync_clock_get

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_sync_clock(ViSession id, ViInt16 syncClock);
```

```
ViStatus age1438_sync_clock_get(ViSession id, ViPInt16 syncClockPtr);
```

Note

This command should be used only for specialized custom clock requirements. Most useful clock setups can be supplied by age1438_clock_setup.

Parameters

- id** is the VXI instrument session pointer returned by the **age1438_init** function.
- syncClock** AGE1438_SMB_CLOCK specifies using the front panel clock on the SMB connectors as the sync clock.
AGE1438_VXI_CLOCK specifies using the VXI (rear panel) clock as the sync clock.
AGE1438_DIVIDED_ADC_CLOCK specifies using the divided ADC clock as the sync clock.
- syncClockPtr** Returns a pointer to the current value of *syncClock*.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Default values” on page 152, “age1438_init” on page 87, “age1438_clock_setup” on page 57, “age1438_sync_direction” on page 132, “age1438_sync_output” on page 133, “Using clock and sync” in chapter 3

age1438_sync_direction

Selects front or rear panel availability of the sync signal. This description also includes the query function:

age1438_sync_direction_get

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_sync_direction(ViSession id, ViInt16 syncDirection);
```

```
ViStatus age1438_sync_direction_get(ViSession id, ViPInt16 syncDirectionPtr);
```

Note

This command should be used only for specialized custom clock requirements. Most useful clock setups can be supplied by age1438_clock_setup.

Description

This function determines whether the front or rear panel sync signal is available to the other panel.

Parameters

- id** is the VXI instrument session pointer returned by the **age1438_init** function.
- syncDirection** AGE1438_SYNC_FRNT_TO_REAR specifies that front panel sync signal be available on the VXI backplane (rear panel).
AGE1438_SYNC_REAR_TO_FRNT specifies that the VXI backplane sync signal be available on the front panel SMB sync connectors.
- syncDirectionPtr** Returns a pointer to the current value of *syncDirection*.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Default values” on page 152, “age1438_init” on page 87, “age1438_sync_output” on page 133, “age1438_sync_clock” on page 131, “Using clock and sync” in chapter 3

age1438_sync_output

Selects the output for the sync signal. This description also includes the query function:

age1438_sync_output_get

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_sync_output(ViSession id, ViInt16 syncOutput);
```

```
ViStatus age1438_sync_output_get(ViSession id, ViPInt16 syncOutputPtr);
```

Note

This command should be used only for specialized custom clock requirements. Most useful clock setups can be supplied by age1438_clock_setup.

Description

This function selects which output the module should use for its sync signal.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

syncOutput AGE1438_SYNC_OUT_OFF specifies no sync signal output.

AGE1438_SYNC_OUT_BOTH specifies that the sync signal be output to both the front panel SMB sync connectors and the VXI backplane.

AGE1438_SYNC_OUT_SMB specifies that the sync signal be output to the front panel SMB sync connectors.

AGE1438_SYNC_OUT_VXI specifies that the sync signal be output to the VXI backplane.

syncOutputPtr Returns a pointer to the current value of *syncOutput*.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Default values” on page 152, “age1438_init” on page 87, “age1438_clock_setup” on page 57, “age1438_sync_clock” on page 131, “Using clock and sync” in chapter 3

age1438_trigger_delay_actual_get

Returns the actual trigger delay from the most recent trigger event.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_trigger_delay_actual_get(ViSession id, ViPInt32  
actualDelayPtr);
```

Description

This delay value provides more accuracy than the trigger delay parameter alone since it includes a measurement of the fractional part of the output sample period between the actual trigger event and the next available output sample. The trigger delay accuracy improves the delay value to one ADC sample clock period rather than one output sample period. This can result in a substantial improvement in accuracy when narrow bandwidth decimation filtering is used.

age1438_trigger_delay_actual_get must be called for each new trigger event that requires precise delay measurement. The actual delay is still expressed in ADC sample periods.

In multiple module systems, the actual delay of the triggering module should be used to correct data from other modules in the system.

Parameters

id	is the VXI instrument session pointer returned by the age1438_init function.
actualDelayPtr	points to the returned actual delay from the most recent trigger event and the resulting first output sample time.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“age1438_init” on page 87, “age1438_trigger_setup” on page 136, “age1438_trigger_phase_actual_get” on page 135, “Delay and phase in triggered measurements” in chapter 3, “Trigger and phase in multi-module systems” in chapter 3

age1438_trigger_phase_actual_get

Returns a representation of the phase value of the LO at the most recent trigger point.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_trigger_phase_actual_get(ViSession id, ViPInt16  
actualPhasePtr);
```

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

actualPhasePtr points to the returned value which is an integer from -32769 to 32767 and should be interpreted as follows:

```
AGE1438_TRIG_PHASE_0 represents 0 degrees (or 0)  
AGE1438_TRIG_PHASE_90 represents 90 degrees (or 16384)  
AGE1438_TRIG_PHASE_180 represents ±180 degrees (or -32768)  
AGE1438_TRIG_PHASE_270 represents +270 (-90) degrees (or -16384)
```

This phase value is not corrected for the actual delay as returned by the `age1438_trigger_delay_actual_get` command. If a more accurate value of the LO phase is desired, it should be corrected:

```
accurate LO phase = age1438_trigger_phase_actual_get - age1438_  
trigger_delay_actual_get * age1438_frequency_center_get * 65536
```

The LO phase could be used in time domain averaging of blocks, or other operations involving zoomed blocks of data, so that the varying phase of the LO can be removed from the calculation.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “`age1438_error_message`” on page 74.

See Also

“`age1438_init`” on page 87, “`age1438_trigger_setup`” on page 136, “`age1438_trigger_delay_actual_get`” on page 134, “`age1438_frequency_setup`” on page 83, “Delay and phase in triggered measurements” in chapter 3, “Trigger and phase in multi-module systems” in chapter 3

age1438_trigger_setup

Sets all triggering parameters. This description also includes information on the following functions which set or query the trigger parameters individually:

age1438_trigger_adclevel specifies the trigger threshold for an ADC trigger
age1438_trigger_adclevel_get gets the ADC trigger threshold
age1438_trigger_delay specifies a pre- or post-trigger delay time
age1438_trigger_delay_get gets the trigger delay time
age1438_trigger_gen determines whether a module can generate a trigger
age1438_trigger_gen_get gets the trigger generation status
age1438_trigger_maglevel specifies the trigger threshold for a magnitude trigger
age1438_trigger_maglevel_get gets magnitude trigger threshold
age1438_trigger_slope selects a positive or negative trigger
age1438_trigger_slope_get gets trigger slope
age1438_trigger_type determines the trigger type
age1438_trigger_type_get gets trigger type

VXIplug&play Syntax

```
#include "age1438".h

ViStatus age1438_trigger_setup(ViSession id, ViInt16 trigType, ViInt32
    trigDelay, ViInt16 adcLevel, ViInt16 magLevel, ViInt16 slope, ViInt16 genTrig);
ViStatus age1438_trigger_adclevel(ViSession id, ViInt16 adcLevel);
ViStatus age1438_trigger_adclevel_get(ViSession id, ViInt16 adcLevelPtr);
ViStatus age1438_trigger_delay(ViSession id, ViInt32 trigDelay);
ViStatus age1438_trigger_delay_get(ViSession id, ViInt32 trigDelayPtr);
ViStatus age1438_trigger_gen(ViSession id, ViInt16 genTrig);
ViStatus age1438_trigger_gen_get(ViSession id, ViInt16 genTrigPtr);
ViStatus age1438_trigger_maglevel(ViSession id, ViInt16 magLevel);
ViStatus age1438_trigger_maglevel_get(ViSession id, ViInt16 magLevelPtr);
ViStatus age1438_trigger_slope(ViSession id, ViInt16 slope);
ViStatus age1438_trigger_slope_get(ViSession id, ViInt16 slopePtr);
ViStatus age1438_trigger_type(ViSession id, ViInt16 trigType);
ViStatus age1438_trigger_type_get(ViSession id, ViInt16 trigTypePtr);
```

Description

An HP E1438A can be triggered to collect data in a variety of ways. The trigger can be internally generated or can come from an external source. Multiple modules can be triggered synchronously. A variable pre- and post-trigger delay can be programmed for data collection. The slope and level of the trigger point on a signal can be selected. The source of the internal trigger can be either the output of the ADC or the magnitude of the complex output of the decimation filter.

age1438_trigger_setup is the function that sets all trigger parameters at once. An HP E1438A generates a trigger only when it is in the TRIGGER state and the Sync line on the VXI backplane is released. When a trigger is generated, the HP E1438A asserts the Sync line.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

- adcLevel** is used to set the triggering signal threshold when using the ADC trigger source. This threshold is (full scale \times `adclevel`/2048), where $-2048 \leq \text{adclevel} \leq 2047$. There is hysteresis around the threshold in order to prevent multiple triggers from a single threshold crossing. Hysteresis is 20 ADC counts, or about 1% full scale.
- Use `AGE1438_ADC_LEVEL_MAX` to set the maximum allowable level.
- Use `AGE1438_ADC_LEVEL_MIN` to set the minimum allowable level.
- Use `AGE1438_ADC_LEVEL_DEF` to set the default ADC trigger threshold.
- adcLevelPtr** points to the current value of the *adclevel* parameter.
- trigDelay** is the time delay, in units of output samples, between when a trigger is received and the first data point in the output data.
- `AGE1438_TRIG_DELAY_MIN` selects the minimum allowable trigger delay.
- `AGE1438_TRIG_DELAY_MAX` selects the maximum allowable trigger delay.
- `AGE1438_TRIG_DELAY_DEF` sets the default trigger delay.
- Negative values indicate a pre-trigger condition where samples prior to the trigger event are included in the output data. The amount of pre-trigger delay is limited to the number of samples which can be saved in the buffer memory. See the **age1438_data_setup** function description for the number of bytes used per sample. The delay limits depend on the data type as follows:
- Trigger delay in output samples (DRAMsize in bytes)**
- | | 24 bit complex | 24 bit real
12 bit complex | 12 bit real |
|---------------------|--------------------------|-------------------------------|----------------------------|
| Post trigger | $2^{31}-1$ | $2^{31}-1$ | $2^{31}-1$ |
| Pre-trigger | $48-(\text{DRAMsize}/6)$ | $48-(\text{DRAMsize}/3)$ | $48-(\text{DRAMsize}/1.5)$ |
- If *trigDelay* is $<48-(\text{DRAMsize}/1.5)$ a bad parameter error is set.**
- trigDelayPtr** points to the current value of the of *delay*.
- genTrig** determines whether a module may generate a trigger.
- `AGE1438_GENERATE_ON` enables triggering.
- `AGE1438_GENERATE_OFF` disables triggering. This is useful in multi-module systems with the same trigger type where you want only certain module(s) to generate a trigger.
- genTrigPtr** points to the current value of the *gen* parameter.
- magLevel** is used to set the triggering to detect when the envelope of a signal rises above the threshold while using the magnitude trigger source. It requires a positive trigger slope.
- `AGE1438_MAG_LEVEL_MAX` sets the maximum allowable level and `AGE1438_MAG_LEVEL_MIN` sets the minimum allowable level.
- `AGE1438_MAG_LEVEL_FS` sets the full scale magnitude trigger threshold.
- `AGE1438_MAG_LEVEL_DEF` sets the default magnitude trigger threshold.

Functions listed alphabetically

The threshold is set to $(AGE1438_MAG_LEVEL_SCALE \times magLevel)$ dB relative to full scale signal, where $-337 \leq magLevel \leq 40$.

Note

When *magLevel* and *sigBw* (see “age1438_filter_setup” on page 76) are both set to zero an illegal state results, causing the *setup_error* status bit to be set.

Note

Magnitude triggering is performed on the log magnitude of the digital filter output. Magnitude triggering occurs when the log magnitude of the digital filter output rises from BELOW the specified magnitude trigger threshold level to ABOVE that level. Because of these facts magnitude trigger operation will not always be intuitive, and there are two distinct cases that can be misinterpreted as improper operation:

Case 1: Magnitude triggering may not occur when the magnitude trigger threshold level is set below the known maximum amplitude of the input signal. The problem in such a case is that the trigger threshold level is actually set too low, so that few, if any, filtered signal samples fall below that level. A transition from below the magnitude trigger threshold to above may never be detected if a sample is not taken while the signal is above the trigger threshold. The solution is to INCREASE the magnitude trigger level to the level at which there are frequent filter samples occurring both above and below the magnitude trigger threshold.

Case 2: Due to the fact that triggering is performed on the absolute (logged) magnitude of the filtered signal, trigger slope has no effect when magnitude triggering.

magLevelPtr

points to the current value of the *magLevel* parameter.

slope

selects the edge of the trigger source on which a trigger occurs for ADC and external triggers. AGE1438_POSITIVE sets triggering on the positive slope and AGE1438_NEGATIVE on the negative slope. Only a positive slope is applicable to *magLevel* triggering.

slopePtr

points to the current value of the of the trigger *slope* parameter.

trigType

determines the trigger source.

AGE1438_ADC generates a trigger based on the raw data samples from the ADC.

AGE1438_MAG generates a trigger based on the log magnitude of the signal after it has been filtered to a selectable bandwidth around the center frequency established by the **age1438_frequency_setup** function.

AGE1438_EXTERNAL uses transitions on the signal applied to the BNC external trigger connector on the front panel.

AGE1438_USER disables the module from any event-driven trigger generation though it is still possible to force the module to trigger a measurement by pulling the Sync line once the module is in the trigger state. You may do this by calling the **age1438_meas_start** function, waiting for the module to reach the trigger state, then triggering the measurement by using **age1438_meas_control** to pull the Sync line.

AGE1438_IMMEDIATE triggers a measurement immediately upon entering the trigger state.

Note

In multi-module systems all modules should be use the same trigger type in order to have the same actual delay.

trigTypePtr

points to the current value of *trigType*.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Commands which halt active measurements” on page 149, “Default values” on page 152, “age1438_init” on page 87, “age1438_frequency_setup” on page 83, “age1438_data_setup” on page 66, “age1438_filter_setup” on page 76, “age1438_meas_start” on page 109, “age1438_meas_control” on page 105, “age1438_trigger_phase_actual_get” on page 135, “age1438_trigger_delay_actual_get” on page 134, “Managing multiple modules” in chapter 3, “Delay and phase in triggered measurements” in chapter 3

age1438_vcxo

Selects whether the internal clock source in the module is turned on or off. This description also includes the query function:

age1438_vcxo_get

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_vcxo(ViSession id, ViInt16 vcxoState);
```

```
ViStatus age1438_vcxo_get(ViSession id, ViPInt16 vcxoStatePtr);
```

Note

This command should be used only for specialized custom clock requirements. Most useful clock setups can be supplied by age1438_clock_setup.

Description

This function selects whether the internal clock source is turned on or off. If an internal source is used the age1438_vcxo_freq function selects which internal source to use.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

vcxoState AGE1438_VCXO_OFF specifies that both internal clock sources are turned off. AGE1438_VCXO_ON that an internal source is turned on.

vcxoStatePtr Returns a pointer to the current value of *vcxoState*.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to "age1438_error_message" on page 74.

See Also

"Commands which halt active measurements" on page 149, "Default values" on page 152, "age1438_init" on page 87, "age1438_clock_setup" on page 57, "age1438_vcxo_freq" on page 141, "Using clock and sync" in chapter 3

age1438_vcxo_freq

Selects which internal clock source the module should use. This description also includes the query function:

age1438_vcxo_freq_get

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_vcxo_freq(ViSession id, ViInt16 vcxoFreq);
```

```
ViStatus age1438_vcxo_freq_get(ViSession id, ViInt16 vcxoFreqPtr);
```

Description

This function selects which internal clock source to use. The `age1438_vcxo` function determines whether the internal source is activated.

Parameters

id	is the VXI instrument session pointer returned by the age1438_init function.
vcxoFreq	AGE1438_VC XO_100000KHZ selects the 100 MHz internal VCXO source. AGE1438_VC XO_102400KHZ selects the 102.4 MHz internal VCXO source.
vcxoFreqPtr	Returns a pointer to the current value of <i>vcxo_freq</i> .

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Commands which halt active measurements” on page 149, “Default values” on page 152, “age1438_init” on page 87, “age1438_clock_setup” on page 57, “age1438_vcxo_freq_preset” on page 142, “age1438_vcxo” on page 140, “Using clock and sync” in chapter 3

age1438_vcxo_freq_preset

Selects which internal clock source should used as a default.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_vcxo_freq_preset(ViSession id, ViInt16 vcxoFreq);
```

Description

This function selects which internal clock frequency should be set in NVRAM for use upon power-up and reset.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

vcxoFreq AGE1438_VC XO_100000KHZ uses the 100 MHz internal source.

AGE1438_VC XO_102400KHZ uses the 102.4 MHz internal source.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“Default values” on page 152, “age1438_init” on page 87, “age1438_clock_setup” on page 57, “age1438_vcxo” on page 140, “age1438_vcxo_freq” on page 141, “Using clock and sync” in chapter 3

age1438_vxi_clock_output

Selects which clock drives the VXI clock. This description also includes the query function:

age1438_vxi_clock_output_get

VXIplug&play Syntax

```
#include "age1438".h
```

```
ViStatus age1438_vxi_clock_output(ViSession id, ViInt16 vxiClock);
```

```
ViStatus age1438_vxi_clock_output_get(ViSession id, ViPInt16 vxiClockPtr);
```

Note

This command should be used only for specialized custom clock requirements. Most useful clock setups can be supplied by age1438_clock_setup.

Description

This function selects which clock the module should use to drive its VXI clock.

Parameters

- id** is the VXI instrument session pointer returned by the **age1438_init** function.
- vxiClock** AGE1438_FRONT_PANEL_CLOCK specifies that the specified front panel clock drive the VXI clock.
AGE1438_CLOCK_OFF specifies not driving vxi clock on the backplane.
AGE1438_DIVIDED_ADC_CLOCK specifies using the divided ADC clock to drive the vxi clock.
- vxiClockPtr** Returns a pointer to the current value of *vxiClock*.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to "age1438_error_message" on page 74.

See Also

"Default values" on page 152, "age1438_init" on page 87, "age1438_clock_setup" on page 57, "Using clock and sync" in chapter 3

age1438_wait

Facilitates the synchronization and control of multi-module systems.

VXI*plug&play* Syntax

```
#include "age1438".h
```

```
ViStatus age1438_wait(ViSession id);
```

Description

This function assures that all slave modules are completely set up before issuing measurement control commands to the master module. Prior to calling **age1438_meas_control** for the master module in multi-module systems, you should call **age1438_wait** for each other module within the related synchronous group to which you have previously sent commands.

This function polls the status register of the indicated module until the **AGE1438_STATUS_HARDWARE_SET** and **AGE1438_STATUS_SYNC_COMPLETE** bits are both true, or until approximately three seconds have elapsed. The function returns **AGE1438_SUCCESS** immediately after the status bits are set, or, if the time-out limit is reached, **AGE1438_STATUS_WAIT_TIMEOUT** is returned.

Parameters

id is the VXI instrument session pointer returned by the **age1438_init** function.

Return Value

AGE1438_SUCCESS indicates that a function was successful.

Values other than **AGE1438_SUCCESS** indicate an error condition or other important status condition. To determine the error message, pass the return value to “age1438_error_message” on page 74.

See Also

“age1438_init” on page 87, “age1438_meas_start” on page 109, “age1438_meas_control” on page 105

Equivalent numeric values for variables

Variable Name	Numeric Value
AGE1438_01_BOARD	0
AGE1438_02_BOARD	1
AGE1438_12BIT	1
AGE1438_24BIT	0
AGE1438_AC	1
AGE1438_ADC	1
AGE1438_ADC_LEVEL_DEF	0
AGE1438_ADC_LEVEL_MAX	2047
AGE1438_ADC_LEVEL_MIN	-2048
AGE1438_ANTIALIAS_OFF	0
AGE1438_ANTIALIAS_ON	1
AGE1438_APPEND	2
AGE1438_ASSERT	1
AGE1438_BLOCK	0
AGE1438_BLOCKSIZE_DEF	1024
AGE1438_BLOCKSIZE_MAX	201326544
AGE1438_BLOCKSIZE_MIN	2
AGE1438_BNC_CLOCK	1
AGE1438_CENT_FREQ_DEF	0.0
AGE1438_CENT_FREQ_MAX	+ .5
AGE1438_CENT_FREQ_MIN	- .5
AGE1438_CLOCK_OFF	0
AGE1438_COMPLEX	1
AGE1438_CONTINUOUS	1
AGE1438_CMPLXDC_OFF	0
AGE1438_CMPLXDC_ON	1
AGE1438_CUSTOM_CLOCK_SETUP	-1
AGE1438_DATA_DELAY_MAX	201326544
AGE1438_DATA_DELAY_MIN	0
AGE1438_DATA_REGISTER	3
AGE1438_DC	0
AGE1438_DEBUG_LEVEL_0	0

Equivalent numeric values for variables

Variable Name	Numeric Value
AGE1438_DEBUG_LEVEL_1	1
AGE1438_DEBUG_LEVEL_2	2
AGE1438_DEBUG_LEVEL_3	3
AGE1438_DEBUG_LEVEL_4	4
AGE1438_DEBUG_LEVEL_5	5
AGE1438_DECIMATE_OFF	0
AGE1438_DECIMATE_ON	1
AGE1438_DECIMATE_SHIFT	2
AGE1438_DIVIDE_BY_10	0
AGE1438_DIVIDE_BY_38	1
AGE1438_DIVIDED_ADC_CLOCK	2
AGE1438_EXTERNAL	2
AGE1438_EXT_SAMPLE_CLOCK	2
AGE1438_FRNT_MSTR_EXT_REF	8
AGE1438_FRNT_MSTR_INT_REF	7
AGE1438_FRNT_SLAV_EXT_REF	9
AGE1438_FRNT_SYNC_EXT_SAMP	21
AGE1438_FRONT_PANEL_CLOCK	3
AGE1438_FS_MAX	103e6
AGE1438_FS_MIN	10e6
AGE1438_GENERATE	1
AGE1438_GENERATE_ON	1
AGE1438_GENERATE_OFF	0
AGE1438_IMMEDIATE	4
AGE1438_INSERT	3
AGE1438_ID_ADDRESS	1
AGE1438_ID_HANDLE	0
AGE1438_LBUS	1
AGE1438_LBUS_RESET_OFF	0
AGE1438_LBUS_RESET_ON	1
AGE1438_MAG	3
AGE1438_MAG_LEVEL_DEF	-128
AGE1438_MAG_LEVEL_FS	0
AGE1438_MAG_LEVEL_MAX	40
AGE1438_MAG_LEVEL_MIN	-337
AGE1438_MAG_LEVEL_SCALE	0.37628749457997662
AGE1438_NEGATIVE	1
AGE1438_OFF	VI_OFF
AGE1438_OFFS_DAC_MAX	255
AGE1438_OFFS_DAC_MIN	0

Variable Name	Numeric Value
AGE1438_ON	VI_ON
AGE1438_PIPELINE	0
AGE1438_POSITIVE	0
AGE1438_PRESCALE_BY_1	0
AGE1438_PRESCALE_BY_4	1
AGE1438_RANGE_0	0
AGE1438_RANGE_1	1
AGE1438_RANGE_2	2
AGE1438_RANGE_3	3
AGE1438_RANGE_4	4
AGE1438_RANGE_5	5
AGE1438_RANGE_6	6
AGE1438_RANGE_7	7
AGE1438_RANGE_8	8
AGE1438_RANGE_9	9
AGE1438_RANGE_10	10
AGE1438_RANGE_11	11
AGE1438_RANGE_12	12
AGE1438_RANGE_13	13
AGE1438_RANGE_14	14
AGE1438_RANGE_15	15
AGE1438_RANGE_16	16
AGE1438_RANGE_17	17
AGE1438_RANGE_MAX	17
AGE1438_RANGE_MIN	0
AGE1438_RANGE_TIME_MAX	20
AGE1438_RANGE_TIME_MIN	0
AGE1438_REAL	0
AGE1438_RELEASE	0
AGE1438_REAR_MSTR_EXT_REF	15
AGE1438_REAR_MSTR_INT_REF	14
AGE1438_REAR_SLAV_EXT_REF	16
AGE1438_REAR_SYNC_EXT_SAMP	22
AGE1438_RM_HANDLE	2
AGE1438_SIG_BW_MAX	18
AGE1438_SIG_BW_MIN	0
AGE1438_SIGNAL_OFF	0
AGE1438_SIGNAL_ON	1
AGE1438_SIMPLE_EXT_REF	1
AGE1438_SIMPLE_EXT_SAMP	2

Equivalent numeric values for variables

Variable Name	Numeric Value
AGE1438_SIMPLE_INT_REF	0
AGE1438_SMB_CLOCK	4
AGE1438_STR_LEN_MIN	256
AGE1438_SYNC_FRNT_TO_REAR	0
AGE1438_SYNC_OFF	0
AGE1438_SYNC_ON	1
AGE1438_SYNC_OUT_BOTH	3
AGE1438_SYNC_OUT_OFF	0
AGE1438_SYNC_OUT_SMB	2
AGE1438_SYNC_OUT_VXI	1
AGE1438_SYNC_REAR_TO_FRNT	1
AGE1438_TRIG_DELAY_DEF	0
AGE1438_TRIG_DELAY_MAX	(2e+31-1)
AGE1438_TRIG_DELAY_MIN	-191999952
AGE1438_TRIG_PHASE_0	0
AGE1438_TRIG_PHASE_90	16384
AGE1438_TRIG_PHASE_180	-32768
AGE1438_TRIG_PHASE_270	-16384
AGE1438_USER	0
AGE1438_VCXD_10000KHZ	0
AGE1438_VCXD_102400KHZ	1
AGE1438_VCXD_EXT_REF	1
AGE1438_VCXD_INTERNAL	0
AGE1438_VCXD_OFF	0
AGE1438_VCXD_ON	1
AGE1438_VME	0
AGE1438_VXI_CLOCK	5
AGE1438_XFERSIZE_DEF	1024
AGE1438_XFERSIZE_MAX	96000000
AGE1438_XFERSIZE_MIN	2

Commands which halt active measurements

age1438_adc_clock
age1438_clock_recover
age1438_clock_setup
age1438_data_blocksize
age1438_data_delay
age1438_data_resolution
age1438_data_type
age1438_data_xfersize
age1438_filter_bw
age1438_filter_decimate
age1438_filter_setup
age1438_front_panel_clock_input
age1438_init
age1438_input_autozero
age1438_input_range_auto
age1438_meas_control
age1438_meas_init
age1438_meas_start
age1438_reset
age1438_reset_hard
age1438_self_test
age1438_state_recall
age1438_trigger_delay
age1438_trigger_setup
age1438_vcxo
age1438_vcxo_freq

Commands which void synchronized multi-module setups:

age1438_clock_setup and low-level clock setup functions
age1438_clock_recover
age1438_input_autozero
age1438_input_range_auto
age1438_self_test
age1438_state_recall

Error messages

Warnings and errors are based on the value VI_ERROR

Error Number	Parameter	Description
0x0000	AGE1438_SUCCESS	No error, command succeeded
0x80000000	AGE1438_ERR_BASE	Base number for error values
AGE1438_ERR_BASE + 0x0001	AGE1438_BAD_COMMAND	Invalid command code
AGE1438_ERR_BASE + 0x0002	AGE1438_INVALID_HW_CONFIG	The hardware configuration is not supported
AGE1438_ERR_BASE + 0x0003	AGE1438_PARM_ERROR	Invalid command parameter
AGE1438_ERR_BASE + 0x0004	AGE1438_NV_SAVE_ERROR	Error while saving to non-volatile memory
AGE1438_ERR_BASE + 0x0005	AGE1438_DOWNLOAD_ERROR	Error while downloading new firmware
AGE1438_ERR_BASE + 0x0006	AGE1438_SERIAL_TIMEOUT	Serial bus time-out; hardware error
AGE1438_ERR_BASE + 0x0007	AGE1438_BYTE_SWAP_ERROR	Incorrect byte-order setting
AGE1438_ERR_BASE + 0x0008	AGE1438_START_ERROR	Start error
AGE1438_ERR_BASE + 0x0009	AGE1438_HARDWARE_FAILURE	Hardware failure
AGE1438_ERR_BASE + 0x000a	AGE1438_WATCHDOG_RESET_ERROR	Watchdog timer caused a hard reset, possibly due to a hardware problem
AGE1438_ERR_BASE + 0x0011	AGE1438_NO_DATA_MEASUREMENT_IN_PROGRESS	No data available, a measurement is in progress.
AGE1438_ERR_BASE + 0x00102	AGE1438_NO_DATA_MEASUREMENT_PAUSED	No data available, the measurement is paused
AGE1438_ERR_BASE + 0x0013	AGE1438_NO_DATA_WAITING_FOR_TRIGGER	No data available, trigger has not occurred
AGE1438_ERR_BASE + 0x0014	AGE1438_NO_DATA_WAITING_FOR_ARM	No data available, acquiring pre-trigger data
AGE1438_ERR_BASE + 0x0016	AGE1438_NO_E1438_FOUND	No AGE1438 found at specified logical address
AGE1438_ERR_BASE + 0x0017	AGE1438_PROC_READY_TIMEOUT	Time-out is waiting for AGE1438 command processor
AGE1438_ERR_BASE + 0x0018	AGE1438_MEMORY_ALLOCATION_ERROR	Memory allocation error

Error Number	Parameter	Description
AGE1438_ERR_BASE + 0x001b	AGE1438_INTERFACE_HARDWARE_INCOMPATIBILE	Interface hardware incompatible with instrument drivers
AGE1438_ERR_BASE + 0x001d	AGE1438_NULL_ID	ID parameter is zero, function aborted
AGE1438_ERR_BASE + 0x0001e	AGE1438_STATUS_WAIT_TIMEOUT	Time-out waiting for desired status
AGE1438_ERR_BASE + 0x00067	AGE1438_AUTOZERO_ERROR	Autozero error
AGE1438_ERR_BASE + 0x006c	AGE1438_AUTORANGE_ERROR	Autorange error
AGE1438_ERR_BASE + 0x0080	AGE1438_SETUP_ERROR	Hardware setup error
AGE1438_ERR_BASE + 0x0081	AGE1438_SYNC_NOT_COMPLETE	Command or Idle assertion did not complete

Errors required for SICL/SPII when using HP E1485

Error Number	Parameter	Description
AGE1438_ERR_BASE + 0x0082	AGE1438_UNKNOWN_STATUS	Unknown error
AGE1438_ERR_BASE + 0x0083	AGE1438_SHARED_MEMORY_MAP_ERROR	Conflict in memory mapping
AGE1438_ERR_BASE + 0x0084	AGE1438_SPII_ERROR	Unexpected SPII error

Default values

Function	Parameter	Default Value
"age1438_adc_clock" on page 51	adcClock	VCXO_INTERNAL
"age1438_adc_divider" on page 52	adcDivider	DIVIDE_BY_10
"age1438_clock_setup" on page 57	clockSetup	SIMPLE_INT_REF
"age1438_data_setup" on page 66	blocksize	1024
	dataDelay	DATA_DELAY_MIN
	dataType	REAL
	mode	BLOCK
	port	VME
	resolution	12BIT
"age1438_data_xfersize" on page 71	xfersize	1024
"age1438_filter_setup" on page 76	decimate	DECIMATE_OFF
	sigBw	0
"age1438_frequency_setup" on page 83	cmplxDC	CMPLXDC_OFF
	centerFreq	0
	sync	SYNC_OFF
"age1438_front_panel_clock_input" on page 86	fpClock	CLOCK_OFF
"age1438_input_setup" on page 95	antialias	ANTIALIAS_ON
	coupling	DC
	range	RANGE_MAX
	signal	SIGNAL_ON
"age1438_interrupt_setup" on page 99	mask	0
	priority	0
"age1438_lbus_mode" on page 101	lbusMode	PIPELINE
"age1438_lbus_reset" on page 103	lbusReset	ON
"age1438_meas_control" on page 105	idle	RELEASE
	sync	RELEASE
"age1438_reference_clock" on page 118	refClock	VXI_CLOCK
"age1438_reference_prescaler" on page 119	refPrescaler	PRESCALE_BY_1
"age1438_smb_clock_output" on page 126	smbClock	CLOCK OFF
"age1438_sync_clock" on page 131	syncClock	DIVIDED_ADC_CLOCK
"age1438_sync_direction" on page 132	syncDirection	FRNT_TO_REAR
"age1438_sync_output" on page 133	syncOutput	SYNC_OUT_OFF

Function	Parameter	Default Value
"age1438_trigger_setup" on page 136	adcLevel	0
	genTrig	ON
	magLevel	-128
	slope	POSITIVE
	trigDelay	0
	trigType	IMMEDIATE
"age1438_vcxo" on page 140	vcxoState	VCXO_ON
"age1438_vcxo_freq" on page 141	vcxoFreq	VCXO_100000KHZ
"age1438_vcxo_freq_preset" on page 142	vcxoFreq	VCXO_100000KHZ
"age1438_vxi_clock_output" on page 143	vxiClock	CLOCK_OFF

VXIplug&play Syntax Quick Reference

ViStatus age1438_adc_clock(ViSession *id*, ViInt16 *adcClock*)
ViStatus age1438_adc_clock_get(ViSession *id*, ViPInt16 *adcClockPtr*)
ViStatus age1438_adc_divider(ViSession *id*, ViInt16 *adcDivider*)
ViStatus age1438_adc_divider_get(ViSession *id*, ViPInt16 *adcDividerPtr*)
ViStatus age1438_attr_get(ViSession *id*, ViInt16 *attribute*, ViPInt32 *value*)
ViStatus age1438_cal_get(ViSession *id*, ViInt16 *board*, ViPInt32 *timestampPtr*)
ViStatus age1438_clock_fs(ViSession *id*, ViReal64 *fs*)
ViStatus age1438_clock_fs_get(ViSession *id*, ViPReal64 *fsPtr*)
ViStatus age1438_clock_recover(ViSession *id*)
ViStatus age1438_clock_setup(ViSession *id*, ViInt16 *clockSetup*)
ViStatus age1438_clock_setup_get(ViSession *id*, ViPInt16 *clockSetupPtr*)
ViStatus age1438_close(ViSession *id*)
ViStatus age1438_data_memsize_get(ViSession *id*, ViPInt16 *memSizePtr*)
ViStatus age1438_data_scale_get(ViSession *id*, ViPReal64 *scalePtr*)
ViStatus age1438_data_setup(ViSession *id*, ViInt16 *dataType*, ViInt16 *resolution*, ViInt16 *mode*, ViInt32 *blocksize*, ViInt32 *dataDelay*, ViInt16 *reserved*, ViInt16 *port*)
ViStatus age1438_data_blocksize(ViSession *id*, ViInt32 *blocksize*)
ViStatus age1438_data_blocksize_get(ViSession *id*, ViPInt32 *blocksizePtr*)
ViStatus age1438_data_delay(ViSession *id*, ViInt32 *dataDelay*)
ViStatus age1438_data_delay_get(ViSession *id*, ViPInt32 *dataDelayPtr*)
ViStatus age1438_data_mode(ViSession *id*, ViInt16 *mode*)
ViStatus age1438_data_mode_get(ViSession *id*, ViPInt16 *modePtr*)
ViStatus age1438_data_port(ViSession *id*, ViInt16 *port*)
ViStatus age1438_data_port_get(ViSession *id*, ViPInt16 *portPtr*)
ViStatus age1438_data_resolution(ViSession *id*, ViInt16 *resolution*)
ViStatus age1438_data_resolution_get(ViSession *id*, ViPInt16 *resolutionPtr*)
ViStatus age1438_data_type(ViSession *id*, ViInt16 *dataType*)
ViStatus age1438_data_type_get(ViSession *id*, ViPInt16 *dataTypePtr*)
ViStatus age1438_data_xfersize(ViSession *id*, ViInt32 *xfersize*)
ViStatus age1438_data_xfersize_get(ViSession *id*, ViPInt32 *xfersizePtr*)
ViStatus age1438_driver_debug_level(ViSession *id*, ViInt16 *debugLevel*)
ViStatus age1438_driver_debug_level_get(ViSession *id*, ViPInt16 *debugLevelPtr*)
ViStatus age1438_error_message(ViSession *id*, ViStatus *statusCode*, ViChar *errorMessage*[])
ViStatus age1438_error_query(ViSession *id*, ViPInt32 *errorCode*, ViChar *errorMessage*[])
ViStatus age1438_filter_setup(ViSession *id*, ViInt16 *sigBw*, ViInt16 *decimate*)
ViStatus age1438_filter_decimate(ViSession *id*, ViInt16 *decimate*)
ViStatus age1438_filter_decimate_get(ViSession *id*, ViPInt16 *decimatePtr*)

ViStatus age1438_filter_bw(ViSession id, ViInt16 sigBw)
ViStatus age1438_filter_bw_get(ViSession id, ViPInt16 sigBwPtr)
ViStatus age1438_filter_sync(ViSession id)
ViStatus age1438_frequency_center_raw(ViSession id, ViInt32 phase, ViInt32 interpolate)
ViStatus age1438_frequency_center_raw_get(ViSession id, ViPInt32 phasePtr, ViPInt32 interpolatePtr)
ViStatus age1438_frequency_setup(ViSession id, ViInt16 cmplxDC, ViInt16 sync, ViReal64 centerFreq)
ViStatus age1438_frequency_center(ViSession id, ViReal64 centerFreq)
ViStatus age1438_frequency_center_get(ViSession id, ViPReal64 centerFreqPtr)
ViStatus age1438_frequency_cmplxdc(ViSession id, ViInt16 cmplxDC)
ViStatus age1438_frequency_cmplxdc_get(ViSession id, ViPInt16 cmplxDCPtr)
ViStatus age1438_frequency_sync(ViSession id, ViInt16 sync)
ViStatus age1438_frequency_sync_get(ViSession id, ViPInt16 syncPtr)
ViStatus age1438_front_panel_clock_input(ViSession id, ViInt16 fpClock)
ViStatus age1438_front_panel_clock_input_get(ViSession id, ViPInt16 fpClockPtr)
ViStatus age1438_init(ViRsrc rsrcName, ViBoolean idQuery, ViBoolean resetInstr, ViPSession id)
ViStatus age1438_input_autozero(ViSession id)
ViStatus age1438_input_offset(ViSession id, ViInt16 coarseDac, ViInt16 fineDac)
ViStatus age1438_input_offset_get(ViSession id, ViPInt16 coarseDacPtr, ViPInt16 fineDacPtr)
ViStatus age1438_input_offset_save(ViSession id)
ViStatus age1438_input_range_auto(ViSession id, ViReal64 sec)
ViStatus age1438_input_range_convert(ViSession id, ViInt16 rangeIndex, ViPReal64 rangeVoltsPtr)
ViStatus age1438_input_setup(ViSession id, ViInt16 reserved, ViInt16 range, ViInt16 coupling, ViInt16 antiAlias, ViInt16 signal)
ViStatus age1438_input_alias_filter(ViSession id, ViInt16 antiAlias)
ViStatus age1438_input_alias_filter_get(ViSession id, ViPInt16 antiAliasPtr)
ViStatus age1438_input_coupling(ViSession id, ViInt16 coupling)
ViStatus age1438_input_coupling_get(ViSession id, ViPInt16 couplingPtr)
ViStatus age1438_input_range(ViSession id, ViInt16 range)
ViStatus age1438_input_range_get(ViSession id, ViPInt16 rangePtr)
ViStatus age1438_input_signal(ViSession id, ViInt16 signal)
ViStatus age1438_input_signal_get(ViSession id, ViPInt16 signalPtr)
ViStatus age1438_interrupt_restore(ViSession id)
ViStatus age1438_interrupt_setup(ViSession id, ViInt16 intrNum, ViInt16 priority, ViInt16 mask)
ViStatus age1438_interrupt_mask_get(ViSession id, ViInt16 intrNum, ViPInt16 maskPtr)
ViStatus age1438_interrupt_priority_get(ViSession id, ViInt16 intrNum, ViPInt16 priorityPtr)
ViStatus age1438_lbus_mode(ViSession id, ViInt16 lbusMode)
ViStatus age1438_lbus_mode_get(ViSession id, ViPInt16 lbusModePtr)
ViStatus age1438_lbus_reset(ViSession id, ViInt16 lbusReset)

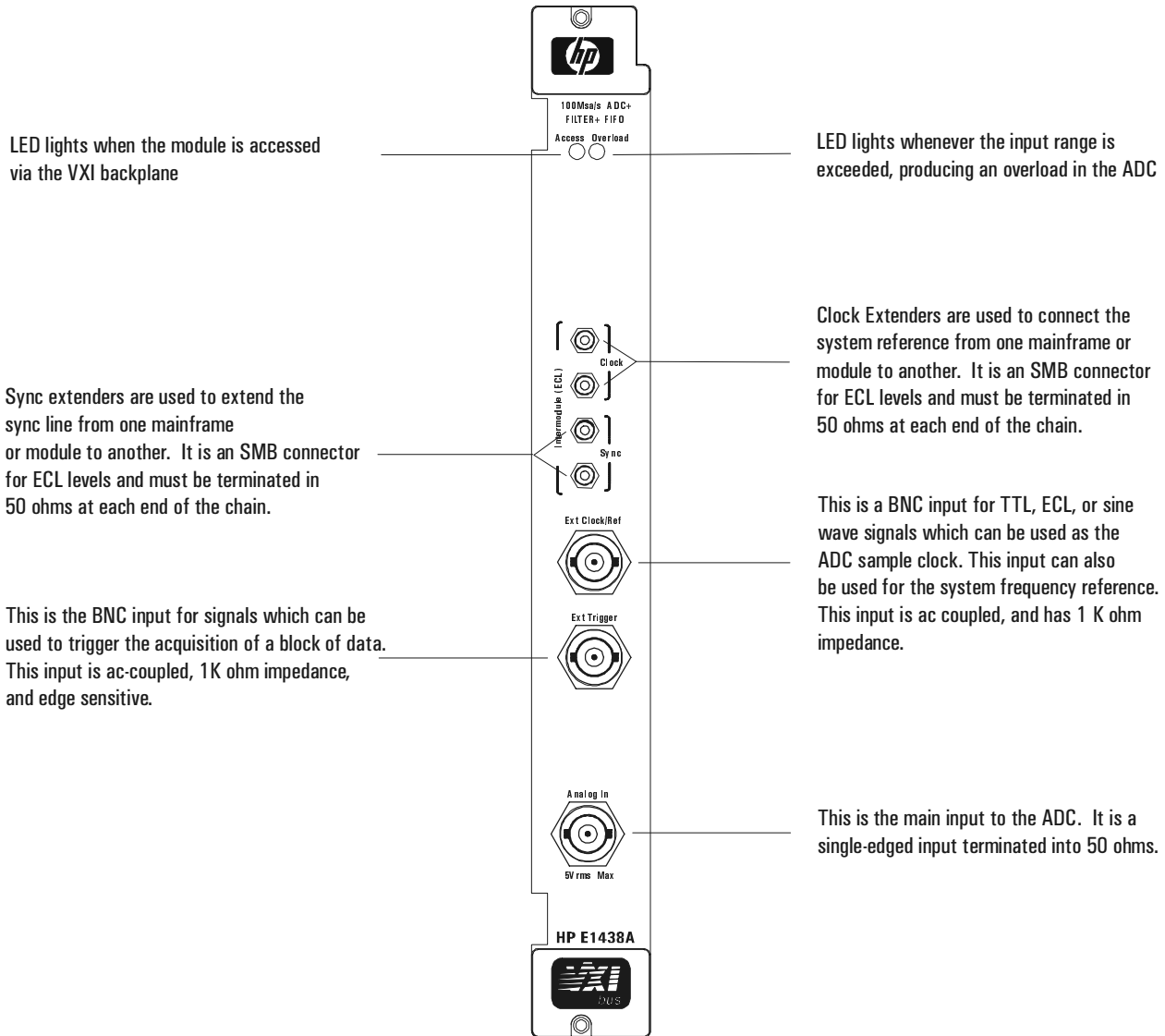
VXIplug&play Syntax Quick Reference

ViStatus age1438_lbus_reset_get(**ViSession** *id*, **ViPInt16** *lbusResetPtr*)
ViStatus age1438_meas_control(**ViSession** *id*, **ViInt16** *idle*, **ViInt16** *sync*)
ViStatus age1438_meas_init(**ViSession** *id*)
ViStatus age1438_meas_start(**ViSession** *id*)
ViStatus age1438_options_get(**ViSession** *id*, **ViChar** *options[]*)
ViStatus age1438_product_id_get(**ViSession** *id*, **ViChar** *productId[]*)
ViStatus age1438_read(**ViSession** *id*, **ViReal32** *data[]*, **ViInt32** *sampleCount*,
ViPInt16 *overloadPtr*)
ViStatus age1438_read64(**ViSession** *id*, **ViReal64** *data[]*, **ViInt32**
sampleCount, **ViPInt16** *overloadPtr*)
ViStatus age1438_read_raw(**ViSession** *id*, **ViPInt16** *data[]*, **ViInt32**
wordCount, **ViPInt16** *overloadPtr*)
ViStatus age1438_reference_clock(**ViSession** *id*, **ViInt16** *refClock*)
ViStatus age1438_reference_clock_get(**ViSession** *id*, **ViPInt16** *refClockPtr*)
ViStatus age1438_reference_prescaler(**ViSession** *id*, **ViInt16** *refPrescaler*)
ViStatus age1438_reference_prescaler_get(**ViSession** *id*, **ViPInt16**
refPrescalerPtr)
ViStatus age1438_reset(**ViSession** *id*)
ViStatus age1438_reset_hard(**ViSession** *id*)
ViStatus age1438_revision_query(**ViSession** *id*, **ViChar** *driverRev[]*, **ViChar**
instrRev[])
ViStatus age1438_self_test(**ViSession** *id*, **ViPInt16** *testResult*, **ViChar**
testMessage[])
ViStatus age1438_serial_number(**ViSession** *id*, **ViChar** *serialNum[]*)
ViStatus age1438_serial_number_get(**ViSession** *id*, **ViChar** *serialNum[]*)
ViStatus age1438_smb_clock_output(**ViSession** *id*, **ViInt16** *smbClock*)
ViStatus age1438_smb_clock_output_get(**ViSession** *id*, **ViPInt16**
smbclockPtr)
ViStatus age1438_state_recall(**ViSession** *id*)
ViStatus age1438_state_save(**ViSession** *id*)
ViStatus age1438_status_get(**ViSession** *id*, **ViPInt16** *statusPtr*)
ViStatus age1438_sync_clock(**ViSession** *id*, **ViInt16** *syncClock*)
ViStatus age1438_sync_clock_get(**ViSession** *id*, **ViPInt16** *syncClockPtr*)
ViStatus age1438_sync_direction(**ViSession** *id*, **ViInt16** *syncDirection*)
ViStatus age1438_sync_direction_get(**ViSession** *id*, **ViPInt16**
syncDirectionPtr)
ViStatus age1438_sync_output(**ViSession** *id*, **ViInt16** *syncOutput*)
ViStatus age1438_sync_output_get(**ViSession** *id*, **ViPInt16** *syncOutputPtr*)
ViStatus age1438_trigger_delay_actual_get(**ViSession** *id*, **ViPInt32**
actualDelayPtr)
ViStatus age1438_trigger_phase_actual_get(**ViSession** *id*, **ViPInt16**
actualPhasePtr)
ViStatus age1438_trigger_setup(**ViSession** *id*, **ViInt16** *trigType*, **ViInt32**
trigDelay, **ViInt16** *adcLevel*, **ViInt16** *magLevel*, **ViInt16** *slope*, **ViInt16** *genTrig*)
ViStatus age1438_trigger_adclevelevel(**ViSession** *id*, **ViInt16** *adcLevel*)
ViStatus age1438_trigger_adclevelevel_get(**ViSession** *id*, **ViPInt16** *adcLevelPtr*)
ViStatus age1438_trigger_delay(**ViSession** *id*, **ViInt32** *trigDelay*)
ViStatus age1438_trigger_delay_get(**ViSession** *id*, **ViPInt32** *trigDelayPtr*)
ViStatus age1438_trigger_gen(**ViSession** *id*, **ViInt16** *genTrig*)
ViStatus age1438_trigger_gen_get(**ViSession** *id*, **ViPInt16** *genTrigPtr*)
ViStatus age1438_trigger_maglevel(**ViSession** *id*, **ViInt16** *magLevel*)
ViStatus age1438_trigger_maglevel_get(**ViSession** *id*, **ViPInt16** *magLevelPtr*)

ViStatus age1438_trigger_slope(ViSession *id*, ViInt16 *slope*)
ViStatus age1438_trigger_slope_get(ViSession *id*, ViPInt16 *slopePtr*)
ViStatus age1438_trigger_type(ViSession *id*, ViInt16 *trigType*)
ViStatus age1438_trigger_type_get(ViSession *id*, ViPInt16 *trigTypePtr*)
ViStatus age1438_vcxo(ViSession *id*, ViInt16 *vcxoState*)
ViStatus age1438_vcxo_get(ViSession *id*, ViPInt16 *vcxoStatePtr*)
ViStatus age1438_vcxo_freq(ViSession *id*, ViInt16 *vcxoFreq*)
ViStatus age1438_vcxo_freq_get(ViSession *id*, ViPInt16 *vcxoFreqPtr*)
ViStatus age1438_vcxo_freq_preset(ViSession *id*, ViInt16 *vcxoFreq*)
ViStatus age1438_vxi_clock_output(ViSession *id*, ViInt16 *vxiClock*)
ViStatus age1438_vxi_clock_output_get(ViSession *id*, ViPInt16 *vxiClockPtr*)
ViStatus age1438_wait(ViSession *id*)

Module Description

Front Panel Description



VXI backplane connections

Power Supplies and Ground

The HP E1438A conforms to the VME and VXI specifications for pin assignment. The current drawn from each supply is given in Technical Specifications.

Data Transfer Bus

The HP E1438A conforms to the VME and VXI specifications for pin assignment and protocol. Only A16/D16/D32 data transfers are supported. Thus the upper addresses are ignored.

DTB Arbitration Bus

The HP E1438A module is not capable of requesting bus control, thus it does not use the Arbitration bus. To conform to the VME and VXI specifications, it passes the bus lines through.

Priority Interrupt Bus

The HP E1438A generates interrupts by applying a programmable mask to its status bits. The priority of the interrupt is determined by the interrupt priority setting in the control register.

Utility Bus

The VME specification provides a set of lines collectively called the utility bus. Of these lines, the HP E1438A only uses the SYSRESET* line.

Pulling the SYSRESET* line low (a hardware reset) has the same effect as setting the reset bit in the Control Register (a software reset), with two exceptions. The exceptions are:

- The Control Register is also reset.
- All logic arrays are reloaded.

Reloading the logic arrays enables the hardware reset to recover from power dropouts which may invalidate the logic setup.

Local Bus

The VXI specification includes a 12-wire local bus between adjacent module slots. Using the local bus, Hewlett-Packard has defined a standard byte-wide ECL protocol that transfers data from left to right at up to 100 Mbyte/s. The HP E1438A can be programmed to output its data using this high speed port instead of the VME data output register. The Data Port Control register determines which output port is used.

VXI backplane connections

Trigger Lines

The VXI specification provides 8 TTL and 2 ECL trigger lines which can be used for module-specific signaling. When programmed in a multi-input configuration, the HP E1438A uses the ECL trigger lines, designating ECLTRG0 as the Sync line and ECLTRG1 as the 10MHz Reference Clock (CLOCK). These lines can be extended to other mainframes using the SMB connectors on the front panel. The SMB connectors can also be used for intermodule synchronization within a mainframe, leaving the ECL trigger lines free for other purposes.

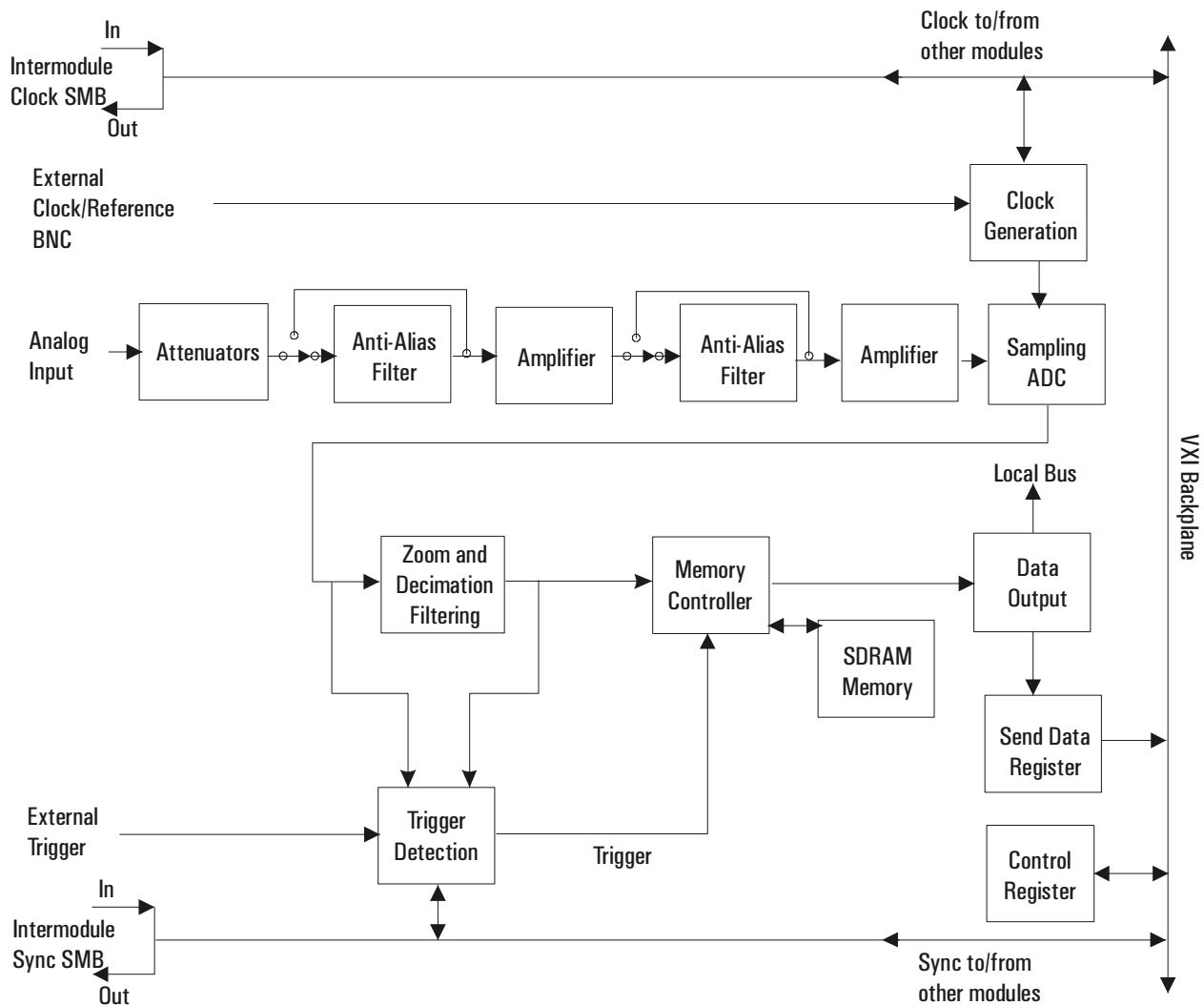
The CLOCK line is the master reference clock for a synchronous system of multiple HP E1438A modules. Only one HP E1438A module in each mainframe is allowed to drive this line.

The Sync line is used to send timing signals among HP E1438A modules in a multi-input system. Any module which drives this line must do so synchronously with CLOCK so that transitions on Sync do not occur near the rising edge of CLOCK. This ensures that all modules with a synchronous state machine clocked on CLOCK interprets Sync in a consistent manner for each cycle of the state machine. Sync is used for synchronizing, arming, and triggering signals between HP E1438A modules. The interpretation of the Sync line is dependent on the states of the module described in “The measurement loop” in chapter 3. The HP E1438A module is also capable of controlling the Sync line synchronously via the control register.

For more information on multi-module operation see “Managing multiple modules” in chapter 3.

Block diagram and description

More detailed descriptions of selected elements in the diagram below appear further on in this section.

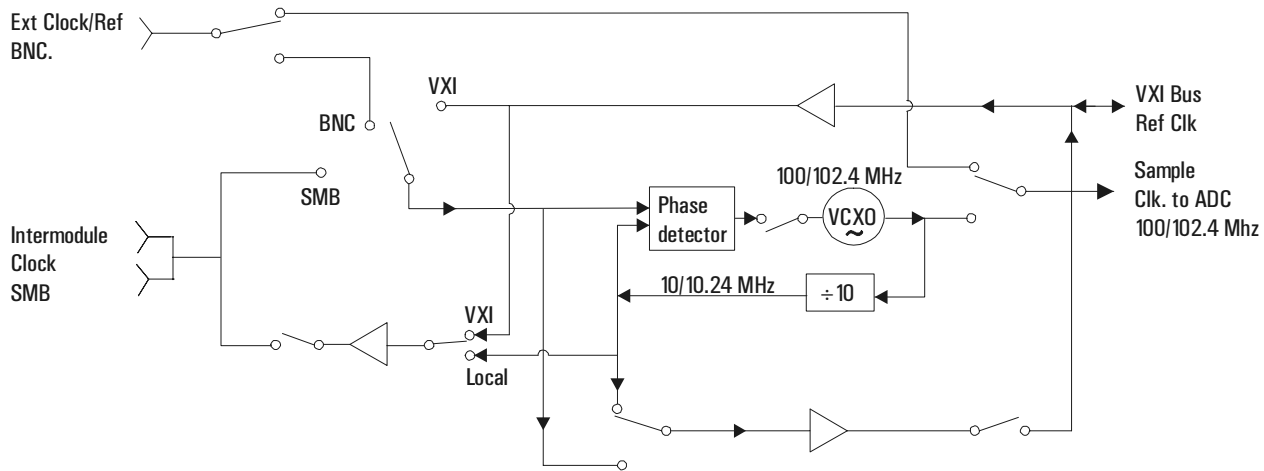


Clock Generation

The usual source for a clock signal is the 100 MHz or the 102.4 MHz crystal oscillator inside the HP E1438A. However, the HP E1438A can also accept an external clock signal through a front-panel BNC "Ext Clock/Ref". This signal can be TTL, ECL, or sine wave.

Block diagram and description

In a system using more than one HP E1438A, the ADCs can be synchronized by programming them to use a common ECL line on the backplane as a reference. One of the modules can be the clock master that drives this line. This master clock can be extended to other mainframes by connecting a "Intermodule Clock" SMB connector to a "Intermodule Clock" SMB connector on an HP E1438A in the second mainframe.

**Input**

The input is terminated by the input amplifier which follows the first half of the anti-alias filter. The bandwidth of the input is 40 MHz. The attenuation of the input is programmable.

Under program control, the input signal can be ac coupled. This allows the system to measure low level ac signals in the presence of a large dc offset.

Anti-alias Filter

Since the normal ADC sample rate is 100 MHz, a complete representation of the input signal can be achieved only for bandwidths up to 50 MHz. Frequency components above 50 MHz can cause ambiguous results (aliasing).

The anti-alias filter attenuates these high frequency components to reduce aliasing. The anti-alias filter in the HP E1438A is flat to 40 MHz and rejects signals above 60 MHz by at least 90 dB. Thus the 0-40 MHz frequency range of the sampled signal is -90 dB alias free. The filter's transition band from 40 MHz to 60 MHz affects flatness and allow some aliasing in the sampled signal frequency range of 40 MHz to 50 MHz.

In cases where alias filtering is not necessary the HP E1438A can be programmed to bypass the anti-alias filter. To avoid incorrect results, the alias filter bypass mode should be used with caution; it is not recommended for normal operation.

Sampling ADC

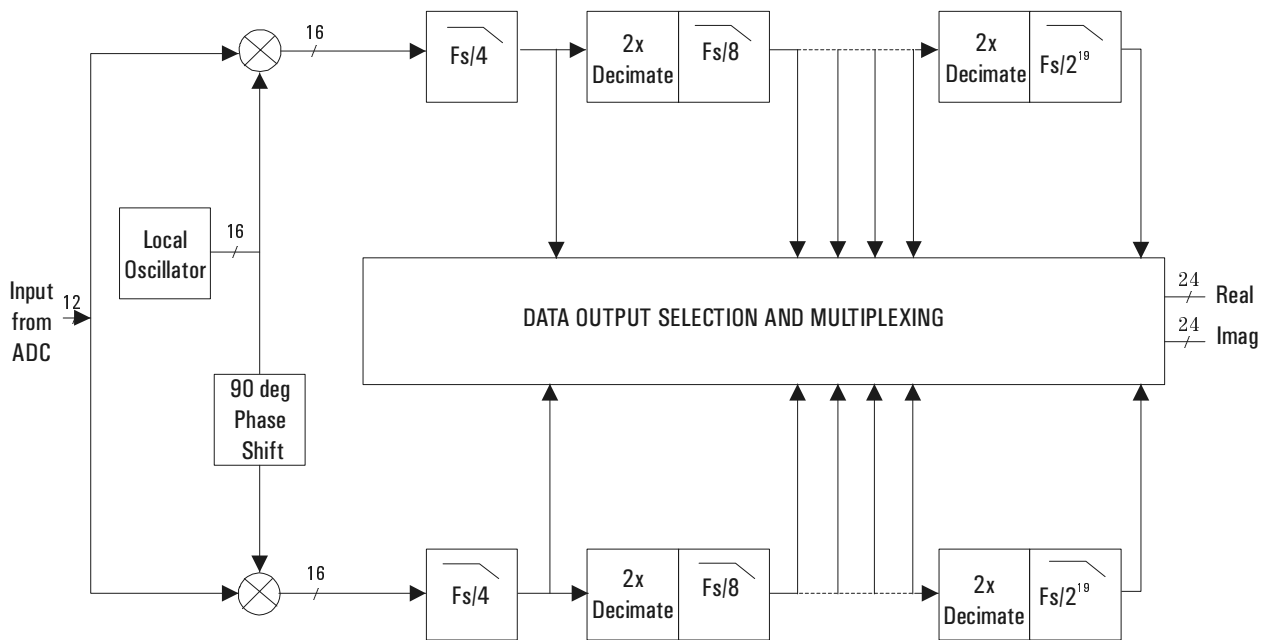
The heart of the HP E1438A is a precision analog-to-digital converter (ADC). The ADC generates 12 bit outputs at a sample rate up to 102.4 MHz.

Zoom and Decimation Filtering

This section uses digital circuitry to allow programmable changes in the center frequency and signal bandwidth of the HP E1438A (zoom). This is done at high speed for real-time operation.

Bandwidth is controlled by a chain of digital low-pass filters (see the diagram below). Each of the filters reduces the bandwidth by a factor of two (decimation). With the ADC sample rate (f_s) set to the standard internal 100.0 MHz rate, the bandwidth choices are 40 MHz, 20 MHz, 10 MHz, ... 76 Hz around the programmed local-oscillator (LO) frequency.

Real and imaginary components of the signal are each computed to 24-bit precision, so the complex output of the decimation filtering block contains 48 bits. Whether or not all of these bits are stored in memory is programmable.



Memory Controller and SDRAM Memory

The HP E1438A can be programmed to save the real component of the signal or to save the complete complex signal. The data precision can be set to 12 bits or 24 bits. Thus, each sample occupies from 1.5 to 6 bytes of memory in the SDRAM. The memory controller block packs the selected data into 72-bit words which are stored in the SDRAM memory. Since the standard SDRAM depth is $2M \times 72$ bits, it is possible to hold up to 12-Msamples in memory at one time.

The memory may be configured either in block mode or in continuous mode. In block mode, data collection initiated by a trigger proceeds until a specified block length is captured. The measurement is then paused so that the data can be read out. This mode is useful in capturing single transient events or whenever the output data rate is too high to be read and processed in real time.

Block diagram and description

In continuous mode, data collection is initiated by a trigger and continues as long as the SDRAM memory does not overflow. Data may be read out of the memory while the measurement is in progress. If the reading of data is sufficiently fast, the SDRAM memory never overflows and the measurement continues indefinitely. If the SDRAM memory should ever overflow then the measurement stops and waits for data to be read out, the measurement to be re-armed, and a new trigger to be initiated. This mode of operation is useful for real-time applications that employ a high speed signal processor to continuously read and operate on each sample of data. Data can be read from the SDRAM memory in bursts to accommodate pauses for such things as disk access times or block mode computations.

The effective trigger time may be offset from the actual trigger event by programming a trigger timing offset. See the Technical Specifications for the limits of the pre-trigger and post-trigger offset.

Data Output

There are two ways to output data from the HP E1438A: by way of the VXI backplane or by way of the local bus.

To use the VXI backplane, the HP E1438A can be programmed so that the output of the memory controller is sent to the Send Data register. The 12- or 24-bit sample data is zero-padded out to 16 or 32 bits. The register can then be read by any controller compatible with the VME standard. Maximum data flow is about 2 MB/s.

The local bus allows data transfers over a high speed 8-bit ECL bus to an adjacent module (to the right) in the VXI mainframe. Multiple adjacent HP E1438A modules can send data to one signal processor module. The signal processor must be one which supports the Hewlett-Packard ECL local bus protocol, such as the HP E1485A/B. In addition to higher speed (up to 66 MB/s), the local bus has the advantage that data can be output at the same time that control signals are being sent over the VXI backplane.

In both of the data output modes, the samples must be read out sequentially, offset by the trigger delay.

Trigger Detection

The trigger event used to start a measurement can be generated in five different ways:

- Software
- External
- ADC threshold
- Log-magnitude
- Immediate

External and ADC threshold triggering modes support slope selection. In ADC or log-magnitude mode the trigger threshold has hysteresis (20 ADC sample counts for the ADC trigger, and 1.5 dB for the magnitude trigger) to prevent noise-generated triggers of the wrong slope. Log magnitude triggering is based on the magnitude of the complex signal after zooming and filtering and only supports positive slope trigger detection.

For external mode, a trigger signal must be supplied at the "Ext Trigger" connector on the front panel. This input is AC coupled with an impedance of 1 K ohm so any signal with a sharp rising or falling transition greater than 100 mV (i.e. TTL or ECL) can be used as an

Block diagram and description

external trigger source. Minimum pulse width is 300 ns. Because the "Ext Trigger" input is an ac-coupled comparator with hysteresis, its initial state is unknown. Before using it, a trigger pulse should be applied to initialize it to a known state.

Any HP E1438A module can trigger other HP E1438A modules using a shared sync line on the VXI backplane. This Sync line can be extended to other mainframes by connecting a "Sync" SMB connector in one mainframe to a "Sync" SMB connector on a HP E1438A in the second mainframe. All modules in a synchronous system are triggered on the same ADC sample.

The HP E1438A hardware samples the trigger source once every sample clock, so the trigger condition must be present for at least one sample clock in order to be recognized.

Control Registers

The HP E1438A module is controlled by firmware using registers mapped into the 16-bit VXI address space.

Module Description

Block diagram and description

Replaceable parts

The HP E1438A must be returned to Agilent Technologies for service or calibration. This section shows you how to add or replace memory modules.

For information on upgrading your module or replacing parts, contact your local Agilent Technologies sales and service office. See the Technical Specifications for a list of office locations and addresses.

Ordering Information

To order HP parts in the U.S., call HP Parts Direct Ordering at (800) 798-5487. Outside the U.S., please contact your local HP parts center.

Code Numbers

The following table provides the name and location for the manufacturers' code numbers (Mfr. Code) listed in the replaceable parts tables.

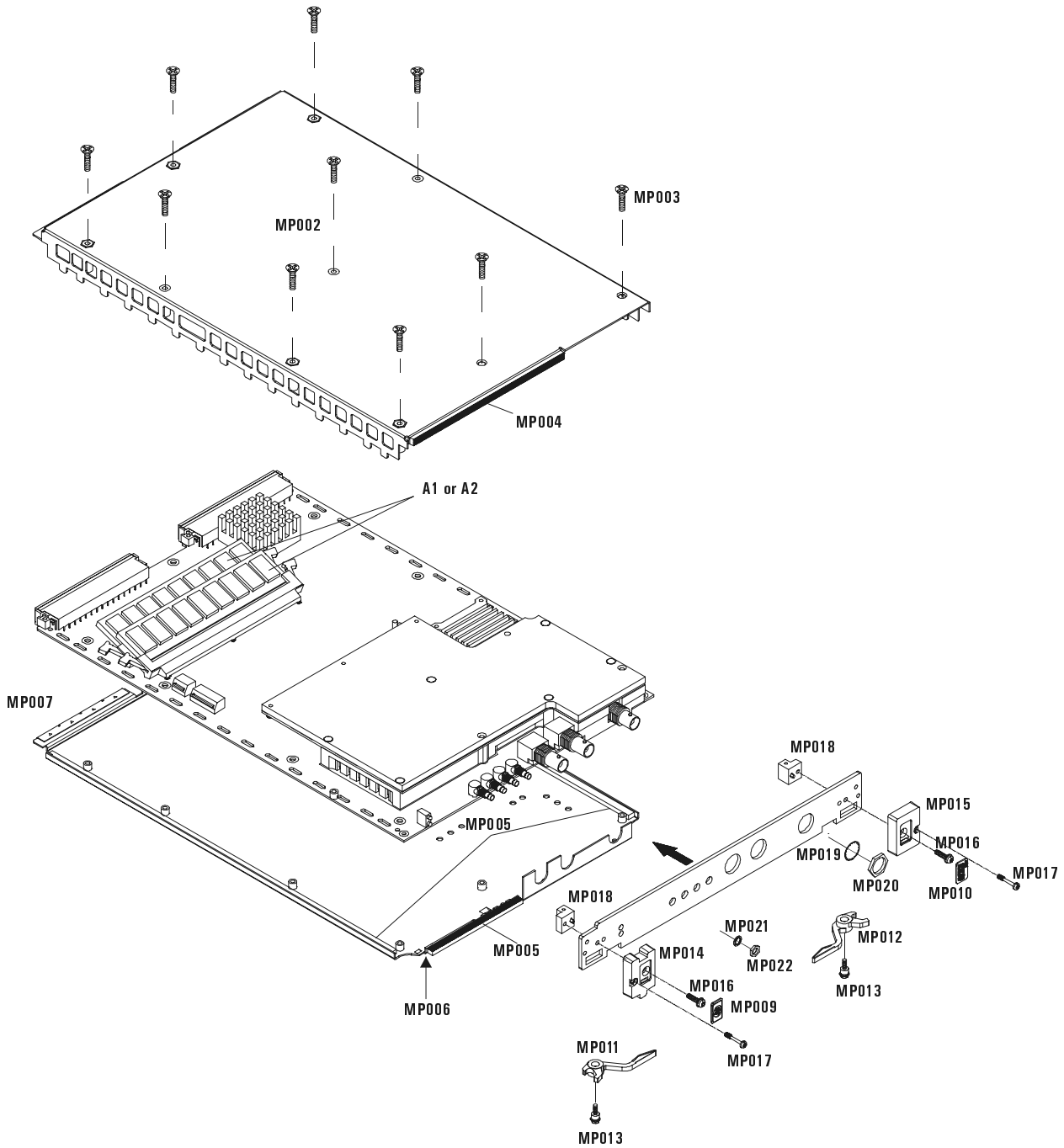
Mfr. No.	Mfr. Name	Location
28480	Agilent Technologies Company	Palo Alto, CA U.S.A.
03647	Instrument Specialties Co. Inc.	Delaware Water Gap, PA U.S.A.
04637	Phelps Dodge Corp.	New York, NY U.S.A.
16044	Kingston Technology Corp.	Fountain Valley, CA U.S.A.
07606	ITW Inc. / Medalist	Glenview, IL U.S.A.
04605	Fischer Special Mfg. Co	Cincinnati, OH U.S.A.
05610	Textron, Inc.	Providence, RI U.S.A.

Replaceable parts

Assemblies

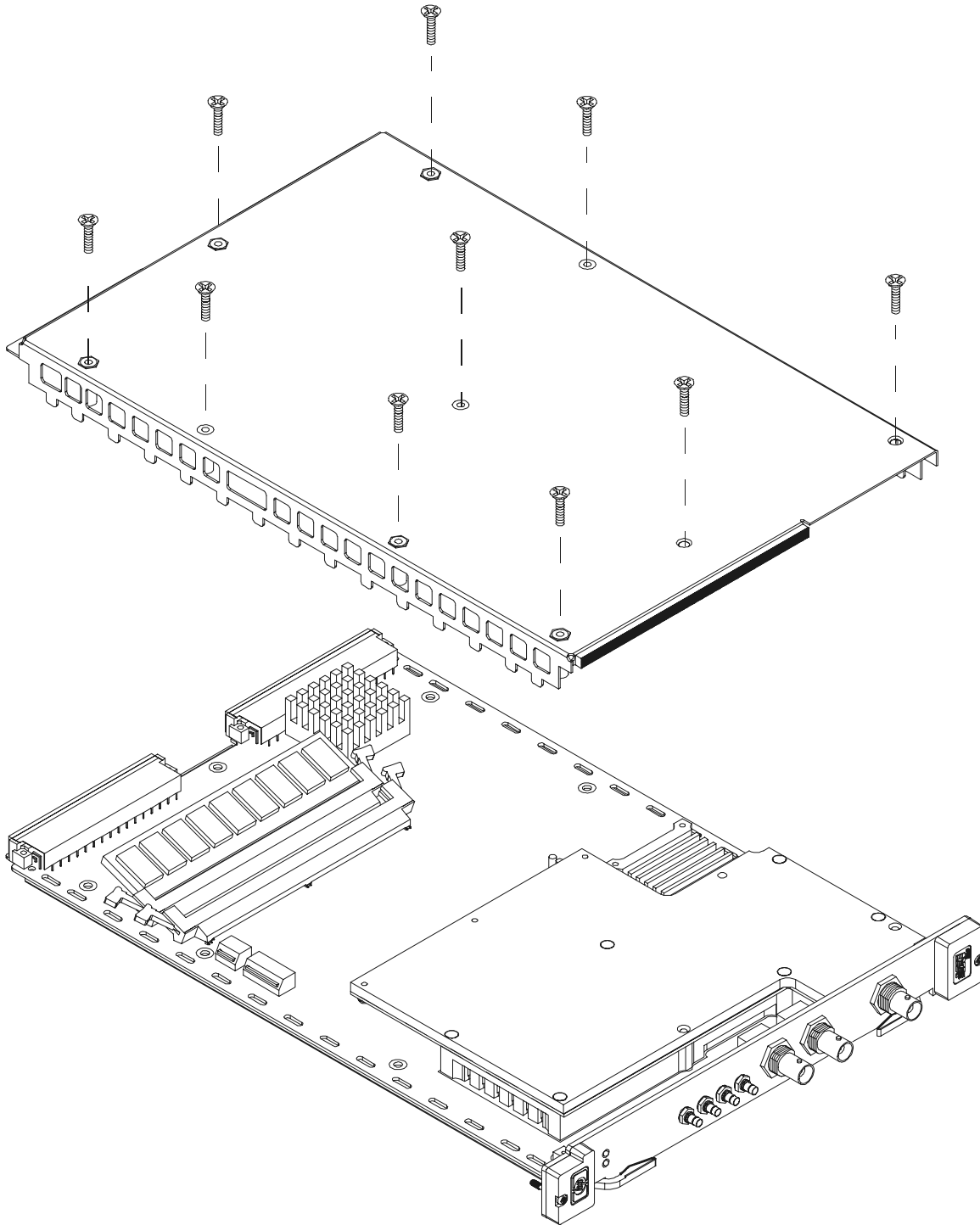
Caution

The module is static sensitive. Use the appropriate precautions when removing, handling, and installing to avoid damage.

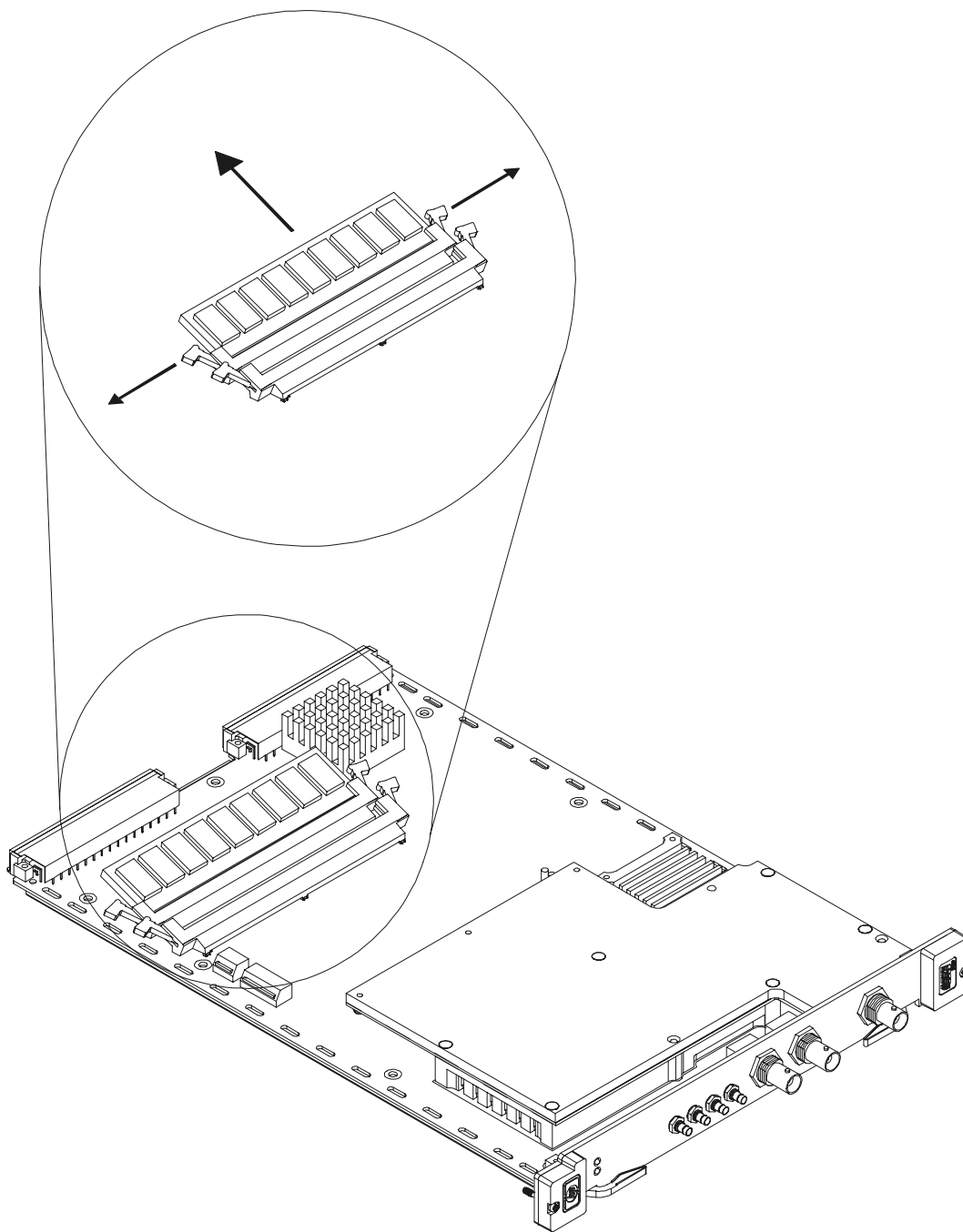


Ref Des	HP Part Number	Qty	Description	MfrCode	Part Number
	E1438-69201	1	EXCHANGE MODULE	28480	E1438-69201
A1	1818-7889	1	SYNC DIMM 16MB 2X72 66MHZ - 16 M mem	16044	KTM66X72/16
A2	1818-7901	2	SYNC-DIMM 16MX72 PC100 168-DIMM - 128 M mem	16044	KGM100X72C3/ 128
MP001	E1438-00203	1	SHTF-BOTTOM COVER	28480	E1438-00203
MP002	E1438-00202	1	SHTF-TOP COVER	28480	E1438-00202
MP003	0515-1135	10	SCREW-MACH M3 x 0.5 25MM-LG	05610	0515-1135
MP004	E1438-40601	1	GSKTRFI-FRT PNL	28480	E1438-40601
MP005	E1485-40601	2	GSKTRFI-BTTM CVR	28480	E1485-40601
MP006	8160-0686	2	RFI STRIP-FINGERS	03647	00786-185
MP007	8160-0634	0.4	RFI STRIP-FINGERS	03647	0097-0611
MP008	E1438-00204	1	FRONT PANEL 'E1438' VXI	28480	E1438-00204
MP009	E1400-84308	1	PLFNAME 'HP' LOGO	28480	E1400-84308
MP010	E1400-84307	1	PLFNAME VXI BUS	28480	E1400-84307
MP011	E1400-45101	1	MOLD - TOP	28480	E140045101
MP012	E1400-45102	1	MOLD - BOTTOM	28480	E140045102
MP013	E1400-00610	2	SCR-ASM SHLDR	28480	E1400-00610
MP014	E1400-45011	1	MOLD LBUS-ECL/ECL	28480	E1400-45011
MP015	E1400-45009	1	MOLD BTTM LOGO BASE	28480	E1400-45009
MP016	0515-0664	2	SCREW MACHINE ASSEMBLY M3 X 0.5 12MM-LG	07606	0515-0064
MP017	0515-2733	2	SCREW SPCL M2.5 X 0.45 17MM-LG PAN-HD	13962	0515-2733
MP018	E1400-40104	2	CAST	28480	E1400-40104
MP019	2190-0068	3	WASHER-LK INTL T 1/2 IN .505-IN-ID	07606	1924-02NP
MP020	2950-0154	3	NUTHEX-DBL-CHAM 1/2-28-THD .078-IN-THK	04605	2950-0154
MP021	2190-0124	4	WASHER-LK INTL T NO. 10 .195-IN-ID	04637	500222
MP022	2950-0078	4	NUTHEX-DBL-CHAN 10-32-THD .067-IN-THK	04637	500220

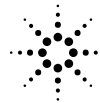
To remove the top cover



To remove the A1, A2 assemblies

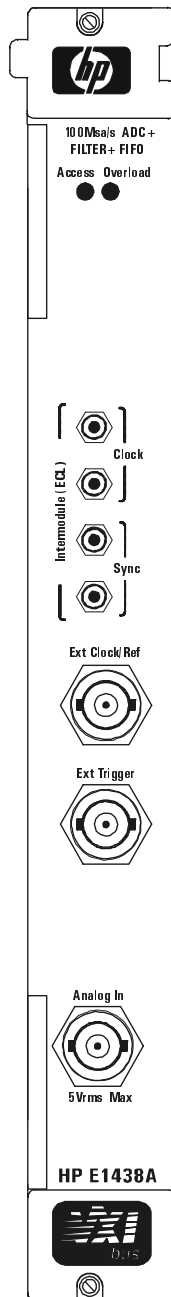


Replaceable parts



HP E1438A

Technical Specifications



100 MSa/s Digitizer

November 1999

The HP E1438A is ideal for application in signal acquisition and analysis, high resolution ATE and radar testing. This single-channel 100 MSa/s digitizer combines exceptional spurious-free dynamic range with alias-protected signal conditioning, center frequency tunable digital filtering, and a large signal capture memory, in a single-wide C-size VXI module.

Abbreviations

F_s = sample rate of DAC.

F_c = cut off frequency of high pass or low pass filters.

dBfs = dB relative to full scale amplitude range.

dBc = dB relative to carrier amplitude.

Typical = typical, non-warranted, performance specification included to provide general product information.

Specification Note

Specifications describe warranted performance over a temperature range of 0° to 55° C, after a 15 minute warm up from ambient conditions. Supplemental characteristics identified as “typical” and “characteristic” provide useful information by giving non-warranted performance parameters. Typical performance is applicable from 20° to 30° C.

Specifications

Input Specification

Input Characteristics

BNC connector, shell grounded to chassis.

50Ω impedance.

dc coupled or ac coupled through 0.2 μF capacitor.

Input signal can be switched to ground.

40 MHz anti-alias filter with bypass switch.

Input Ranges +30 to -21 dBm in 3 dB steps

dBm 50 Ohms	Volts peak
30 dBm	10.0 Vp
27 dBm	7.08 Vp
24 dBm	5.01 Vp
21 dBm	3.55 Vp
18 dBm	2.51 Vp
15 dBm	1.78 Vp
12 dBm	1.26 Vp
9 dBm	891 mVp
6 dBm	631 mVp
3 dBm	447 mVp
0 dBm	316 mVp
-3 dBm	224 mVp
-6 dBm	158 mVp
-9 dBm	112 mVp
-12 dBm	79.4 mVp
-15 dBm	56.2 mVp
-18 dBm	39.8 mVp
-21 dBm	28.2 mVp

ADC Overload Level 0 dBfs typical

Return Loss of 50 Ohm Input Impedance

0.1 to 40 MHz: > 18 dB (1.3 VSWR)

Amplitude Accuracy

(Power measurement, at 10 MHz, 0 to -40 dBfs)

Alias filter on: ± 0.7 dB

Flatness

(dB relative to 10 MHz, excluding digital filter response)

Alias filter on, freq < 40 MHz: ± 1.0 dB
Alias filter off, freq < 40 MHz: ± 2.0 dB
Alias filter off, at 100 MHz: -18 dB (typical)

DC Offset

Auto-zero accuracy: ± 2 %fs (typical)
Temperature drift: < ± 0.1 mV / °C (typical)

Input Bias Current < 50 μA (typical)

Anti Alias Filter Stopband Rejection > 90 dB

(60 MHz to 200 MHz, typical value for +27 and +30 dBm ranges)

Signal-to-Noise Ratio

(full scale input, full bandwidth, excluding distortion. See noise, distortion and spur specs)

Alias filter on: > 60 dB (typical)
Alias filter off: > 55 dB (typical)

Input Noise Density

(Alias filter on, internal sample clock)

100 kHz to 40 MHz:	< -133 dBfs/Hz
10 kHz to 100 kHz:	< -131 dBfs/Hz
1 kHz to 10 kHz:	< -125 dBfs/Hz
100 Hz to 1 kHz:	< (-95 -10 LOG(f)) dBfs/Hz
Sensitivity:	< -155 dBm/Hz (typical)

Residual Responses

(with 50Ω termination at input connector, 2 kHz to 40 MHz)

Harmonic Distortion, Aliased Harmonic Distortion, and Spurious Responses.

Input signals > -10 dBfs:	< -65 dBc
Input signals -10 to -20 dBfs:	< -70 dBc
Input signals < -20 dBfs:	< -70 dBc or < -90 dBfs

Intermodulation Distortion

(Two in-band signals 1 MHz apart. Measured in dBc, relative to one signal.)

0 to 30 MHz input signals:	
each signal -6 to -14 dBfs:	< -65 dBc
each signal -14 to -20 dBfs:	< -70 dBc
each signal < -20 dBfs:	< -70 dBc or < -90 dBfs
30 to 40 MHz input signals:	
each signal -6 to -14 dBfs:	< -62 dBc
each signal -14 to -23 dBfs:	< -67 dBc
each signal < -23 dBfs:	< -67 dBc or < -90 dBfs
3 rd Order products, each input -16 dBfs:	-85 dBc (typical)

Phase Noise Density

(single sideband power density of 10 MHz signal, <0.05 G vibration, absolute or relative. Block data transfer mode, see Note 1)

$\Delta f = 10$ kHz:	< -128 dBc/Hz (typical)
$\Delta f = 1$ kHz:	< -120 dBc/Hz (typical)
$\Delta f = 100$ Hz:	< -110 dBc/Hz (typical)

Discrete Sidebands

(5 Hz to 100 kHz Δf , see Notes 1 and 2)

$\Delta f > 20$ kHz:	< -90 dBc
$\Delta f < 20$ kHz:	< -90 dBc (typical, Note 1)
Inter-module clock via VXI lines:	< -80 dBc (typical)

Note 1. Phase noise and sidebands performance at frequency offsets of less than 20 kHz may be degraded by noise and ripple on the VXI power supplies.

Note 2. Specifications for Dynamic Range, Spurious Responses and Sidebands require the mainframe containing the HP E1438 to have Option 918 (connector shields E1400-80920) installed. In addition, all modules in the mainframe must comply with the VXI 1.4 specification for ECL trigger lines, the 10 MHz VXI system clock must be turned off, and the HP E1438 External Clock input must be disconnected when not being used. Dynamic range specifications require 24-bit data resolution.

Sample Clock and DSP Specifications

Clock Sources

Internal sample clock frequency:	100 MSa/s or 102.4 MSa/s (program control)
External reference for internal clock:	10 MHz for 100 MSa/s, 10.24 MHz for 102.4 MSa/s
External sample clock frequency range:	10 MHz to 102.4 MHz

Internal Clock Specifications

Frequency accuracy, 0 to 40° C:	± 7 ppm
Frequency accuracy, 40 to 55° C:	± 10 ppm
External reference lock range:	± 6 ppm (typical)

Clock Input/Output Characteristics

External sample clock/reference input:	BNC connector. ac-coupled comparator with 1 K Ω impedance. Accepts TTL, ECL, or > -6 dBm sine waves
Trigger input:	BNC connector. ac-coupled comparator with 1 K Ω impedance. Detects pulses > 300 ns with edges > 100 mV
Inter-module front panel clock/sync: Inter-module VXI backplane clock/Sync: 10 MHz reference output :	SMB connector, ECL-10K compatible. VXI backplane ECLTRG lines. SMB connector +8 dBm

Multi-module Sampling Skew

Within mainframe, uncorrected:	< 10 ns (typical).
Between mainframes, 1m cable, uncorrected:	< 25 ns (typical)
Resolution of correction:	5 ps (nominal)

Digital Decimation Filters	17 octave steps (40 MHz to 305 Hz), < 0.215 dB ripple, software correctable
-----------------------------------	--

Digital Local Oscillator	< 0.01 Hz tuning resolution
---------------------------------	-----------------------------

Regulatory Compliance

Safety Standards	Designed for compliance to EN 61010-1(1993)
-------------------------	---

Radiated Emissions and Immunity	EN 61326-1 (see Note 2, page 3)
--	------------------------------------

Environmental

Operating Restrictions

Maximum altitude	4600m, above 2285m derate operating temperature by -3.6° C per 1000m
Ambient Temperature	0 to 55° C
Humidity	10% to 90% at 40° C, non-condensing

Typical Performance Charts

The following charts are included as supplemental, non-warranted characteristics)

Performance Benchmarks

(Benchmarks are included as supplemental, non-warranted characteristics)

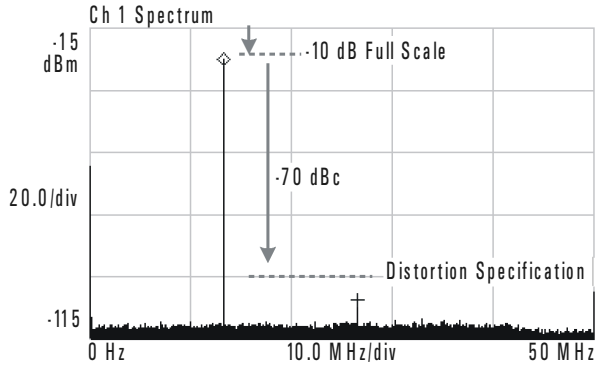
VXI/VME continuous data transfer rate (From E1438A to MXI-II VXI controller, D32 VME word size) 2.2 MBytes/s

Local bus data transfer rate (From E1438A to ideal consumer) 66 MBytes/s

Library function control of module (MXI-II VXI controller)

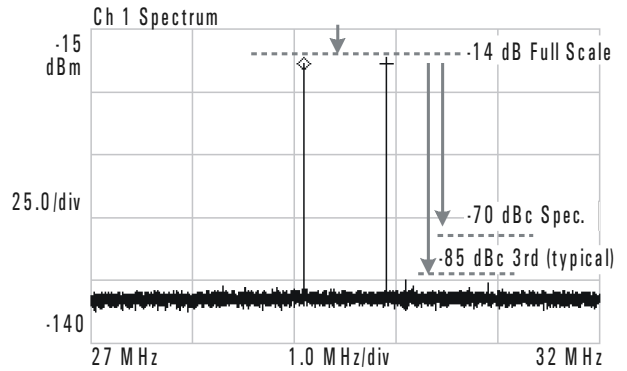
Measurement start: 8.5 μ s
center frequency change (raw): 600 μ s

Harmonic Distortion



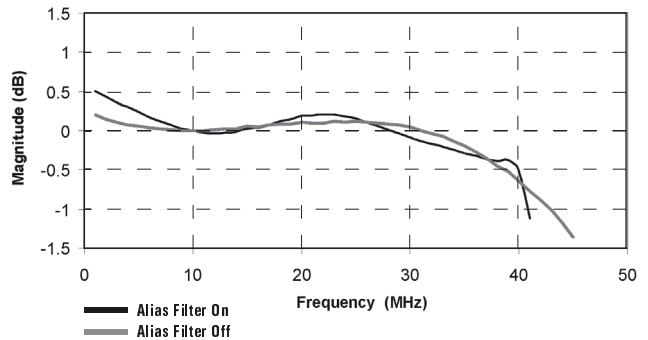
Harmonic Distortion performance with a -25 dBm 13 MHz signal on the -15 dBm range

Intermodulation Distortion

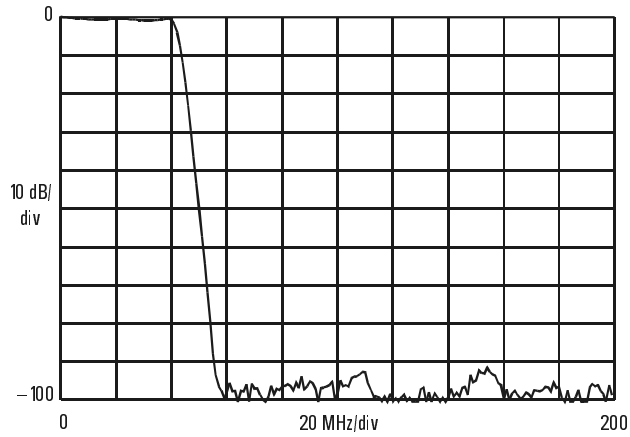


Intermodulation Distortion performance with two -14 dBfs tones near 30 MHz on the -15 dBm range.

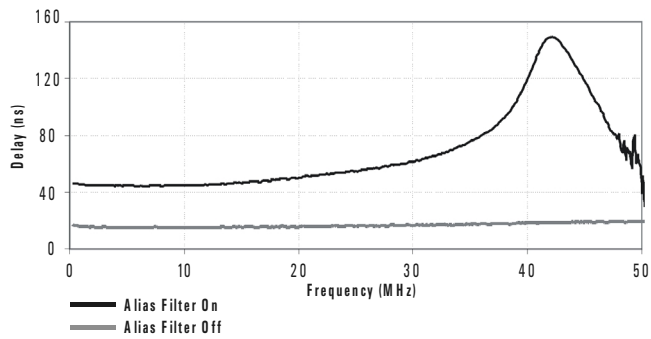
Response vs. Frequency - Pass Band



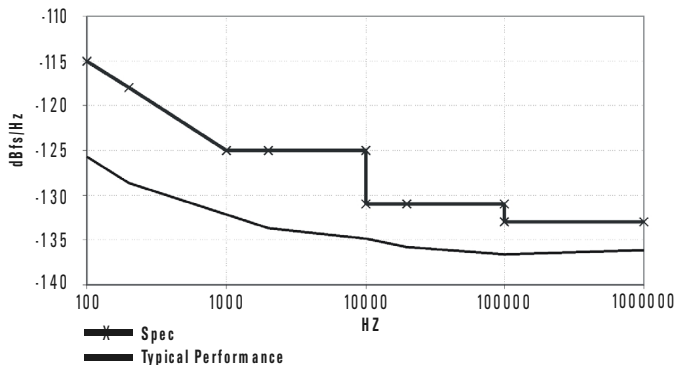
Filter Characteristics for Analog Anti Alias Filter, Magnitude (dB) vs. Frequency (MHz)



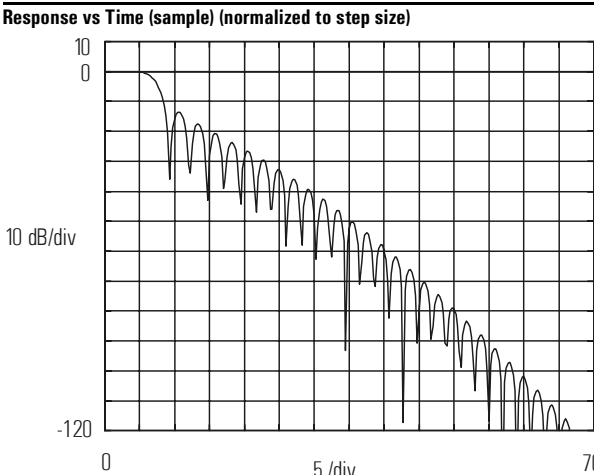
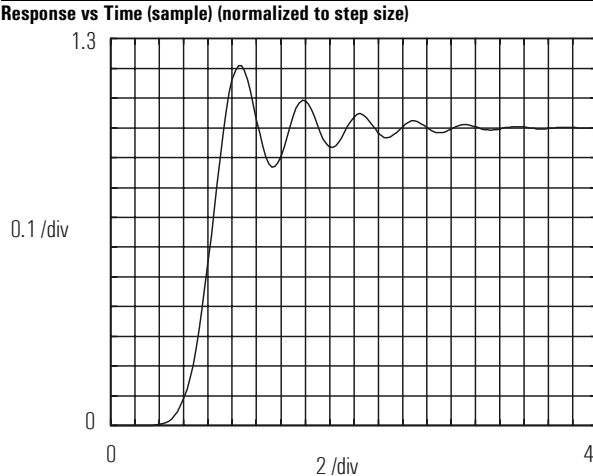
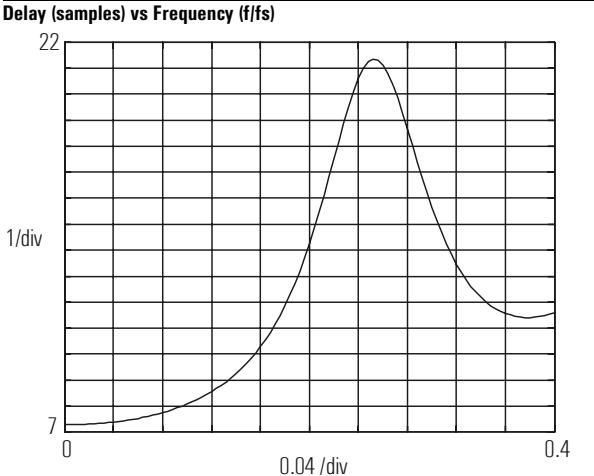
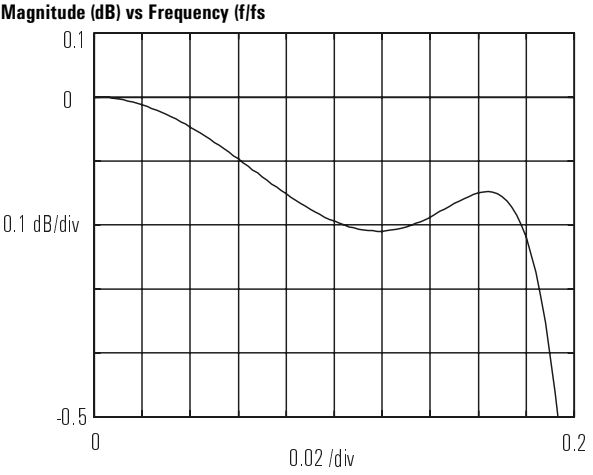
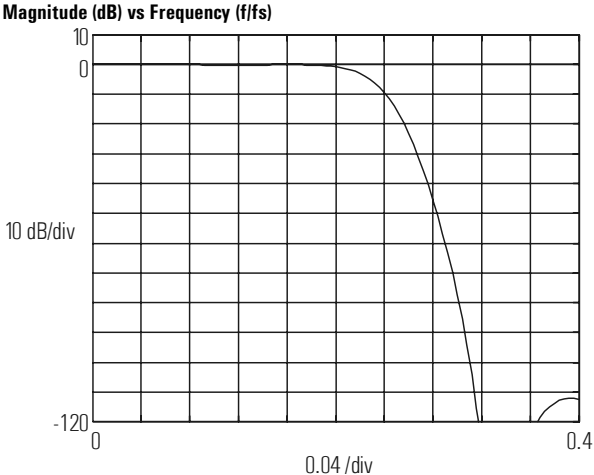
Analog Anti Alias Filter Group Delay vs. Frequency



Input Noise Performance



**Filter Characteristics for Low-pass
Digital Filter Without Decimation,
sigBw=3**



fs = output sample rate



General

VXI Standard Information

Conforms to VXI revision 1.4. See Note 1, page 3 concerning section B.8.6, Conducted Susceptibility.

C-size, single slot width.

Register based programming.

"Slave" Data Transfer Bus functionality.

A16 address capability.

D16/D32 data capability.

Local Bus capability

Requires ECLTRG0 and ECLTRG1 lines for module synchronization.

VXI Power Requirements	dc Current	Dynamic Current
+5 V:	4.3 A	0.3 A
-5.2 V:	2.9 A	0.1 A
-2 V:	0.7 A	0.1 A
+12 V:	0.6 A	0.3 A
-12 V:	0.3 A	0.02 A
+24 V:	0.04 A	0.02 A
-24 V:	0.04 A	0.02 A
+5 V Standby:	0.0 A	0.0 A

VXI Cooling Requirements

For 10° C rise:	3.3 liters/second, 0.67 mm H ₂ O
For 15° C rise:	2.2 liters/second, 0.30 mm H ₂ O

Warm-up Time	15 Minutes
---------------------	------------

Calibration Interval	1 Year (no field adjustments)
-----------------------------	-------------------------------

Warranty

This product is distributed, warranted, and supported by Agilent Technologies.

The HP E1438A comes with a 3 year warranty. During that period, the unit will either be replaced or repaired, at Agilent Technologies option, and returned to the customer without charge.

Related Literature

HP Test System and VXI Products Catalog
p/n 5968-3698EUS
p/n 5968-3698EN
p/n 5968-5609E

For more information on Agilent Technologies test & measurement products, applications, services, and for a current sales office listing, visit our web site, <http://www.agilent.com/find/tmdir>. You can also contact one of the following centers and ask for a test and measurement representative.

United States:

Agilent Technologies
Test and Measurement Call Center
P.O. Box 4026
Englewood, CO 80155-4026
(tel) 1 800 452 4844

Canada:

Agilent Technologies Canada Inc.
5150 Spectrum Way
Mississauga, Ontario
L4W 5G1
(tel) 1 877 894 4414

Europe:

Agilent Technologies
Test & Measurement
European Marketing Organization
P.O. Box 999
1180 AZ Amstelveen
The Netherlands
(tel)(31 20) 547-9999

Japan:

Agilent Technologies Japan Ltd.
Measurement Assistance Center
9-1, Takakura-Cho, Hachioji-Shi,
Tokyo 192,-8510 Japan
(tel) (81) 426 56-7832
(fax) (81) 426 56-7840

Latin America:

Agilent Technologies
Latin American Region Headquarters
5200 Blue Lagoon Drive, Suite #950
Miami, Florida 33126 U.S.A.
(tel) (305) 267 4245
(fax) (305) 267 4286

Australia/New Zealand:

Agilent Technologies Australia Pty Ltd..
347 Burwood Highway
Forest Hill, Victoria 3131 Australia
(tel) 1 800 629 485 (Australia)
(fax) (61 3) 9272 0749
(tel) 0 800 738 378 (New Zealand)
(fax) (64 4) 802 6881

Asia Pacific:

Agilent Technologies
24/F, Cityplaza One, 1111 King's Road
Taikoo Shing, Hong Kong
(tel) (852) 3197-7777
(fax) (852) 2506 9284

**Technical data is subject to change.
Copyright © 1999 Agilent Technologies, Inc.
Printed in USA 11/99
5968-8233E**

Glossary

anti-alias filter	An analog low pass filter inserted the signal path to eliminate undesirable frequency components which appear under the alias of another (baseband) frequency. For more information, see <i>Spectrum and Network Measurements</i> available through your Hewlett-Packard Sales Office.
baseband	A band in the frequency spectrum that begins at zero. In contrast a zoomed band is centered on a specific center frequency.
block mode	A mode in which the HP E1438A stops taking data as soon as a block of data has been collected.
block size	The number of sample points in a block of data. For complex data, block size is the number of complex data pairs per data block.
continuous mode	A mode in which the HP E1438A collects data continuously. It does not stop taking data unless the FIFO overflows.
decimation filter	A digital filter that simultaneously decreases the bandwidth of the signal and decreases the sample rate. The digital filter provides alias protection and increases frequency resolution. For more information, see <i>Spectrum and Network Measurements</i> available through your Hewlett-Packard Sales Office.
FIFO	A First In, First Out buffer and controller used to transmit data.
LO	Local oscillator
VCXO	Voltage controlled crystal oscillator
zoom	Selects a frequency span around a specified center frequency. This is also known as band selectable operation.

INDEX

A

ac coupling, selecting 95
ADC, circuit description 164
address, module
 See logical address
age1438_adc_clock 51
age1438_adc_clock_get 51
age1438_adc_divider 52
age1438_adc_divider_get 52
age1438_attrib_get 53
age1438_cal_get 54
age1438_clock_fs 55
age1438_clock_fs_get 55
age1438_clock_recover 56
age1438_clock_setup 57
age1438_clock_setup_get 57
age1438_close 63
age1438_data_blocksize 66
age1438_data_blocksize_get 66
age1438_data_delay 66
age1438_data_delay_get 66
age1438_data_memsize_get 64
age1438_data_mode 66
age1438_data_mode_get 66
age1438_data_port 66
age1438_data_port_get 66
age1438_data_resolution 66
age1438_data_resolution_get 66
age1438_data_scale_get 65
age1438_data_setup 66
age1438_data_type 66
age1438_data_type_get 66
age1438_data_xfersize 71
age1438_data_xfersize_get 71
age1438_driver_debug_level 72
age1438_driver_debug_level_get 72
age1438_error_message 74
age1438_error_query 75
age1438_filter_bw 76
age1438_filter_bw_get 76
age1438_filter_decimate 76
age1438_filter_decimate_get 76
age1438_filter_setup 76
age1438_filter_sync 79
age1438_frequency_center 83
age1438_frequency_center_get 83
age1438_frequency_center_raw 81
age1438_frequency_center_raw_get 81
age1438_frequency_emplxdc 83
age1438_frequency_emplxdc_get 83
age1438_frequency_setup 83
age1438_frequency_sync 83
age1438_frequency_sync_get 83
age1438_front_panel_clock_input 86
age1438_front_panel_clock_input_get 86
age1438_init 87
age1438_input_alias_filter 95
age1438_input_alias_filter_get 95
age1438_input_autozero 89
age1438_input_coupling 95
age1438_input_coupling_get 95
age1438_input_offset 90
age1438_input_offset_get 90
age1438_input_offset_save 91
age1438_input_range 95
age1438_input_range_auto 92
age1438_input_range_convert 93
age1438_input_range_get 95
age1438_input_setup 95
age1438_input_signal 95
age1438_input_signal_get 95
age1438_interrupt_mask_get 99
age1438_interrupt_priority_get 99
age1438_interrupt_restore 98
age1438_interrupt_setup 99
age1438_lbus_mode 101
age1438_lbus_mode_get 101
age1438_lbus_reset 103
age1438_lbus_reset_get 103
age1438_meas_control 105
age1438_meas_init 108
age1438_meas_start 109
age1438_options_get 110
age1438_product_id_get 111
age1438_read 112
age1438_read_raw 115
age1438_read64 112
age1438_reference_clock 118
age1438_reference_clock_get 118
age1438_reference_prescaler 119
age1438_reference_prescaler_get 119
age1438_reset 120
age1438_reset_hard 121
age1438_revision_query 122
age1438_self_test 123
age1438_serial_number 125
age1438_serial_number_get 125
age1438_smb_clock_output 126
age1438_smb_clock_output_get 126
age1438_state_recall 127

age1438_state_save 128
 age1438_status_get 129
 age1438_sync_clock 131
 age1438_sync_clock_get 131
 age1438_sync_direction 132
 age1438_sync_direction_get 132
 age1438_sync_output 133
 age1438_sync_output_get 133
 age1438_trigger_adclevel 136
 age1438_trigger_adclevel_get 136
 age1438_trigger_delay 136
 age1438_trigger_delay_actual_get 134
 age1438_trigger_delay_get 136
 age1438_trigger_gen 136
 age1438_trigger_gen_get 136
 age1438_trigger_maglevel 136
 age1438_trigger_maglevel_get 136
 age1438_trigger_phase_actual_get 135
 age1438_trigger_setup 136
 age1438_trigger_slope 136
 age1438_trigger_slope_get 136
 age1438_trigger_type 136
 age1438_trigger_type_get 136
 age1438_vcxo 140
 age1438_vcxo_freq 141
 age1438_vcxo_freq_get 141
 age1438_vcxo_freq_preset 142
 age1438_vcxo_get 140
 age1438_vxi_clock_output 143
 age1438_vxi_clock_output_get 143
 age1438_wait 144
 Agilent Technologies name use iii
 alias filter
 See anti-alias filter
 alias protection
 See anti-alias filter
 analog filter
 See anti-alias filter
 anti-alias filter
 circuit description 164
 default 26
 described 26
 selecting 76 , 95
 using 26
 appending data on local bus 101
 arbitration bus, DTB 161
 arm state, described 23
 auto-ranging 92
 autozero 89

B

backplane connections 161
 bandwidth
 control circuit description 165
 filter selection 76
 baseband measurements
 complex 83
 overview 26
 block

 mode, explained 23
 size, determining 67
 block diagram
 circuit description 163
 clock and sync 27
 functional overview 20
 buffer amplifier, selecting 96
 bus transfers, data 34

C

C programming
 overview 21
 source library 22
 calibration data, reading 54
 center frequency
 See Also frequency
 setting 83
 circuit description 163
 clock
 circuit description 163
 distribution 28
 easy setup 57
 external reference 30
 external sample frequency 55
 front panel, selecting 86
 generation 163
 resetting 56
 setup 27
 sharing 28 , 57 , 163
 source, specifying 51
 sync source 131
 synchronization 57
 closing an instrument session 63
 complex data output, specifying 67
 configuring a VXI system 12
 continuous mode, explained 23
 control registers, circuit description 167
 conversion, range 93
 corrections, dc offset 89
 coupling, input 95

D

data
 on local bus 101
 output, circuit description 166
 port, selecting 68
 data formatting
 circuit description 165
 specifying 66
 data transfer bus 161
 dc coupling, selecting 95
 dc offset correction 89
 decimation counters
 synchronizing 105
 decimation filter
 and triggering 24
 changes 32
 circuit description 165
 described 26
 selecting 76

DEVICE NPRESNT 12**digital filter**

See decimation filter

drivers

installing HP-UX 11

installing Windows 10

DTB arbitration bus 161**E****ending an instrument session 63****error messages**

listed 150

reading 74

reading firmware 75

example programs

C 16

HP-VEE 17

using 16

Visual Basic 16

Windows 14

external

clock frequency 55

reference clock 30

trigger, selecting 138

F**filter bandwidth**

See Also decimation filter

setting 76

filter decimation

See decimation filter

filtering

overview 26

See Also anti-alias filter

See Also decimation filter

span, See zoom measurements

firmware revision, determining 122**floating input, selecting 96****formatting data**

See data formatting

frequency

center, changing 32

center, overview 26

center, setting 83

external sample clock 55

synchronizing changes 83

front panel

clock output 126

connectors 160

hardware 160

signal distribution 29

software 14

G**generating**

data on local bus 101

interrupts 99

grounding 161**H****hardware interface 12****hardware reset 121****Hewlett-Packard name use iii****HP E1485, using with 34****HP product name iii****HP-UX**

installing libraries 11

online help 11

programming environment 22

programming overview 21

using libraries 15

HP-VEE

example program 17

I**id, module 87, 111****idle state**

described 23

forcing 63, 105

initializing the I/O driver 87**initiating**

an instrument session 87

measurements 105, 109

input

circuit description 164

coupling 95

setup 95

inserting data on local bus 101**installing**

hardware 3

libraries, HP-UX 11

memory 175

module 3

software 10

Windows libraries 10

instrument state

recalling 127

saving 128

interface, hardware 12**interrupt**

generation 99

managing 53

mask, setting 99

priority, setting 99

invalid measurement conditions 77**L****local bus**

backplane connections 161

described 161

resetting 103

selecting 68

setting mode 101

transfers 34, 166

local oscillators

phase and triggering 135

See Also clock 163

synchronizing 105

logical address

- default 3
 - selecting 3
- M**
- measurement
 - initiating 105
 - initiating single module 108 , 109
 - invalid conditions 77
 - states, described 23
 - measurement loop 23
 - memory
 - circuit description 165
 - installing 175
 - size, determining 64
 - mode
 - measurement 23
 - output 66
 - model number, viewing 111
 - module model number 111
 - multiple modules
 - managing 24 , 28 , 32 , 79 , 83 , 105 , 136 , 144 , 162
 - triggering 134
- N**
- numeric variable values 145
- O**
- offset correction, dc 89
 - offset, input 90 , 91
 - online help
 - HP-UX 11 , 11
 - Windows 13
 - options, identifying 110
 - output formatting 66
 - output mode 67
 - overview
 - clock and sync 27
 - data transfer 34
 - frequency and filtering 26
 - measurement state sequence 23
 - programming 21
 - synchronization 32
- P**
- packaging the module 6
 - parameter variable values 145
 - parts, ordering or replacing 170
 - phase
 - and delay in triggering 24
 - and trigger 135
 - pipelining data on local bus 101
 - port selection, data 68
 - power supplies 161
 - power-up state, forcing 120
 - prescaling clock reference 119
 - priority interrupt bus 161
 - programming overview 21
- R**
- range
 - auto 92
 - conversion 93
 - input 96
 - raw data, scaling 65
 - reading data 112 , 115
 - real data output, specifying 67
 - recalling instrument state 127
 - resetting
 - bad clock 56
 - the local bus 103
 - the module 87 , 120 , 121
 - resolution selection, data 68
 - return values listed 150
 - revision, firmware 122
 - revisions, driver 10 , 11
- S**
- sample rate
 - and decimation 76
 - sample clock frequency 55
 - sample output rate, selecting 77
 - sample rate
 - determining 69
 - saving instrument state 128
 - scale factor 65
 - scaled data, reading 112
 - scaling raw data 65
 - SDRAM memory 165
 - self test, performing 123
 - serial number, getting 125
 - setting the range automatically 92
 - sharing clock and sync 28
 - shipping the module 6
 - smb
 - clock output 126
 - connectors 160
 - connectors, terminating 29
 - state
 - recalling 127
 - saving 128
 - states, measurement 23
 - status register
 - and interrupts 99
 - bits defined 129
 - storing the module 6
 - sync
 - and frequency change 83
 - and measurement state 23
 - and trigger 136
 - clock source 131
 - decimation filter 79
 - direction 132
 - output, selecting 133
 - setup 27
 - sharing 28 , 166
 - signal, asserting and releasing 105
 - synchronizing
 - decimation counters 105
-

- filter decimation 79
- local oscillators 105
- synchronizing measurements 79 , 83 , 105 , 136 , 144
- system requirements 9 , 21 , 22

T

- terminating an instrument session 63
- theory of operation 163
- timing
 - See Also clock
 - See Also trigger
 - setup 27
 - signals 162
- transfer size, determining and specifying 71
- transmission mode, local bus 101
- transporting the module 6
- trigger
 - and decimation filtering 24
 - backplane lines 161
 - delay and phase 24
 - delay setting 137
 - delay, actual 134
 - detection, circuit description 166
 - generation, selecting 137
 - in multiple modules 134
 - level setting 137
 - lines, extending 162
 - phase, actual 135
 - slope, selecting 138
 - state 136
 - state, described 23
 - type, selecting 138

U

- UNIX, See HP-UX
- unscaled data, reading 115
- upgrades 10 , 11
- utility bus 161

V

- variable values 145
- VEE
 - see HP-VEE 17
- verifying operation 14
- Visual Basic
 - example program 16

VME

- bus transfers 34
- port, selecting 68
- reading data on 112

VXI

- backplane connection 161
- bus transfers 34 , 166
- interface, configuring 12

W

- Windows
 - example program 14
 - installing libraries 10

- programming overview 21

Z

- zoom measurements
 - and phase 25
 - and triggering 25
 - circuit description 165
 - overview 26
 - selecting 83
 - setting center frequency 83