VT1432A

16 Channel 51.2 kSamples/s Digitizer plus DSP

User's Guide

Part Number 82-0067-000
Printed in U.S.A.
Print Date: July 30, 2004

2031 Main Street, Irvine, CA 92614-6509 U.S.A.

NOTICE

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

VXI TECHNOLOGY 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. VXI Technology 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 this material.

WARRANTY

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

Use of this manual and flexible disk(s) or tape cartridge(s) 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 government is subject to restrictions as set forth in subdivision (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

VXI Technology, Inc.
2031 Main Street
Irvine, CA 92614-6509

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

In This Book

The VT1432A 16 Channel 51.2 kSamples/s Digitizer plus DSP is a C-size VXI module. “51.2 kSamples/s” refers to the maximum sample rate of 51,200 samples per second. The VT1432A may contain up to four 4-channel input assemblies so that the module may have a total of up to sixteen inputs. The module plugs into a single C-size slot in a VXI mainframe.

This book documents the VT1432A module, including information on how to use it. It provides:

- Installation information.
- Examples to expedite getting started, with information on how to use the VXI*plug&play* Host Interface Library functions. There is also a chapter about the C-Language version of the Host Interface Library. There are instructions for printing the Function Reference for the Host Interface Library if desired. The Function Reference can be accessed by way of online manual pages and online help.
- Information on how to use the VT1432A.
- A descriptions of the module.
- Descriptions of the Arbitrary Source and Tachometer options.
- Descriptions of the Break Out Boxes which can be used with the module.
- Service information (troubleshooting and replacing assemblies).
- Details about the module’s VXI registers (as an appendix).

Safety Summary

The following general safety precautions must be observed during all phases of operation of this instrument. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture and intended use of the instrument. VXI Technology, Inc. assumes no liability for the customer's failure to comply with these requirements.

GENERAL

This product is a Safety Class 1 instrument (provided with a protective earth terminal). The protective features of this product may be impaired if it is used in a manner not specified in the operation instructions.

All Light Emitting Diodes (LEDs) used in this product are Class 1 LEDs as per IEC 60825-1.

ENVIRONMENTAL CONDITIONS

This instrument is intended for indoor use in an installation category II, pollution degree 2 environment. It is designed to operate at a maximum relative humidity of 95% and at altitudes of up to 2000 meters. Refer to the specifications tables for the ac mains voltage requirements and ambient operating temperature range.

BEFORE APPLYING POWER

Verify that the product is set to match the available line voltage, the correct fuse is installed and all safety precautions are taken. Note the instrument's external markings described under Safety Symbols.

GROUND THE INSTRUMENT

To minimize shock hazard, the instrument chassis and cover must be connected to an electrical protective earth ground. The instrument must be connected to the ac power mains through a grounded power cable, with the ground wire firmly connected to an electrical ground (safety ground) at the power outlet. Any interruption of the protective (grounding) conductor or disconnection of the protective earth terminal will cause a potential shock hazard that could result in personal injury.

FUSES

Only fuses with the required rated current, voltage and specified type (normal blow, time delay, etc.) should be used. Do not use repaired fuses or short-circuited fuse holders. To do so could cause a shock or fire hazard.

DO NOT OPERATE IN AN EXPLOSIVE ATMOSPHERE

Do not operate the instrument in the presence of flammable gases or fumes.

DO NOT REMOVE THE INSTRUMENT COVER

Operating personnel must not remove instrument covers. Component replacement and internal adjustments must be made only by qualified service personnel.

Instruments that appear damaged or defective should be made inoperative and secured against unintended operation until they can be repaired by qualified service personnel.

WARNING

The WARNING sign denotes a hazard. It calls attention to a procedure, practice or the like, which, if not correctly performed or adhered to, could result in personal injury. Do not proceed beyond a WARNING sign until the indicated conditions are fully understood and met.

Caution

The CAUTION sign denotes a hazard. It calls attention to an operating procedure or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the product. Do not proceed beyond a CAUTION sign until the indicated conditions are fully understood and met.

Safety Symbols

 Warning, risk of electric shock

 Caution, refer to accompanying documents

 Alternating current

 Both direct and alternating current

 Earth (ground) terminal

 Protective earth (ground) terminal

 Frame or chassis terminal

 Terminal is at earth potential.

 Standby (supply). Units with this symbol are not completely disconnected from ac mains when this switch is off

Table of Contents

Chapter 5. Module Description

Chapter 6. The Arbitrary Source Option (VT1432A-1D4)

Chapter 7. The Tachometer Option (VT1432A-AYF)

Chapter 8. Break Out Boxes

Support Resources

Support resources for this product are available on the Internet and at VXI Technology customer support centers.

VXI Technology
World Headquarters

VXI Technology, Inc.
2031 Main Street
Irvine, CA 92614-6509

Phone: (949) 955-1894
Fax: (949) 955-3041

VXI Technology
Cleveland Instrument Division

VXI Technology, Inc.
7525 Granger Road, Unit 7
Valley View, OH 44125

Phone: (216) 447-8950
Fax: (216) 447-8951

VXI Technology
Lake Stevens Instrument Division

VXI Technology, Inc.
1924 - 203 Bickford
Snohomish, WA 98290

Phone: (425) 212-2285
Fax: (425) 212-2289

Technical Support

Phone: (949) 955-1894
Fax: (949) 955-3041
E-mail: support@vxitech.com

Visit http://www.vxitech.com *for worldwide support sites and service plan information.*

1

Installing the VT1432A

Installing the VT1432A

This chapter contains instructions for installing the VT1432A 16-Channel 51.2 kSamples/s Digitizer plus DSP Module and its drivers. This chapter also includes instructions for transporting and storing the module.

To inspect the VT1432A

The VT1432A 16-Channel 51.2 kSamples/s Digitizer plus DSP 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 a VXI Technology sales and service office.

To install the VT1432A

Caution To protect circuits from static discharge, observe anti-static techniques whenever handling the VT1432A 16-Channel 51.2 kSamples/s Digitizer plus DSP Module.

1 Set up the VXI mainframe. See the mainframe's installation guide for assistance.

2 Select a slot in the VXI mainframe for the VT1432A module.

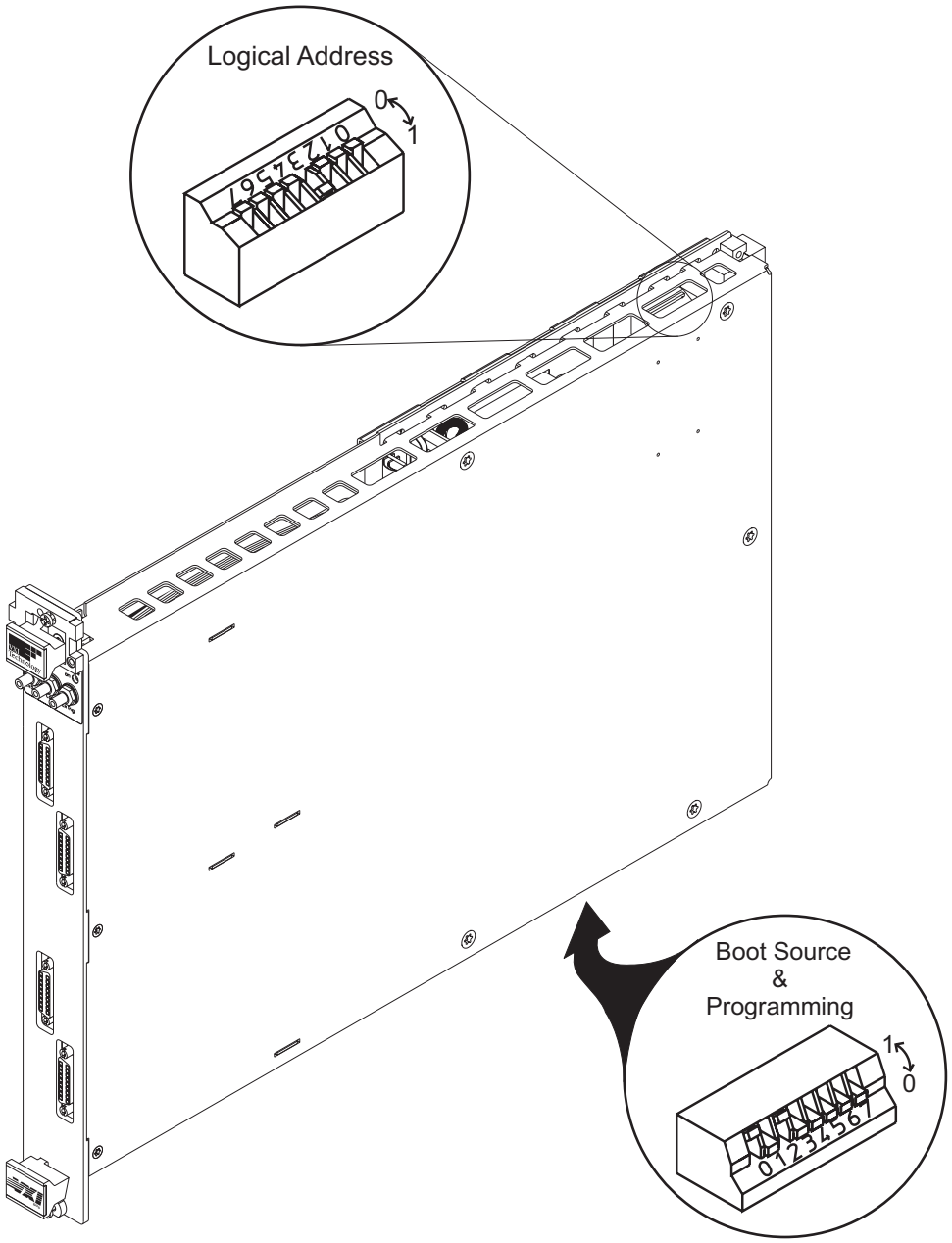
The VT1432A module's local bus receives ECL-level data from the module immediately to its left and outputs ECL-level data to the module immediately to its right. Every module using the local bus is keyed to prevent two modules from fitting next to each other unless they are compatible. If using the local bus, select adjacent slots immediately to the left of the data-receiving module. The local bus can support up to nine VT1432A modules at full span at real time data rates. 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 VT1432A.

(See the illustration on the next page.) Each module in the system must have a unique logical address. The factory default setting is 0000 1000 (8). If a GPIB command module will be controlling the VT1432A module, select an address that is a multiple of 8. If the VXI system dynamically configures logical addresses, set the switch to 255.

4 Check the settings of the Boot Source and ROM Programming switches on the bottom of the module.

Set switches 1 and 3 (BS1 and BS3) up and all the other switches down.



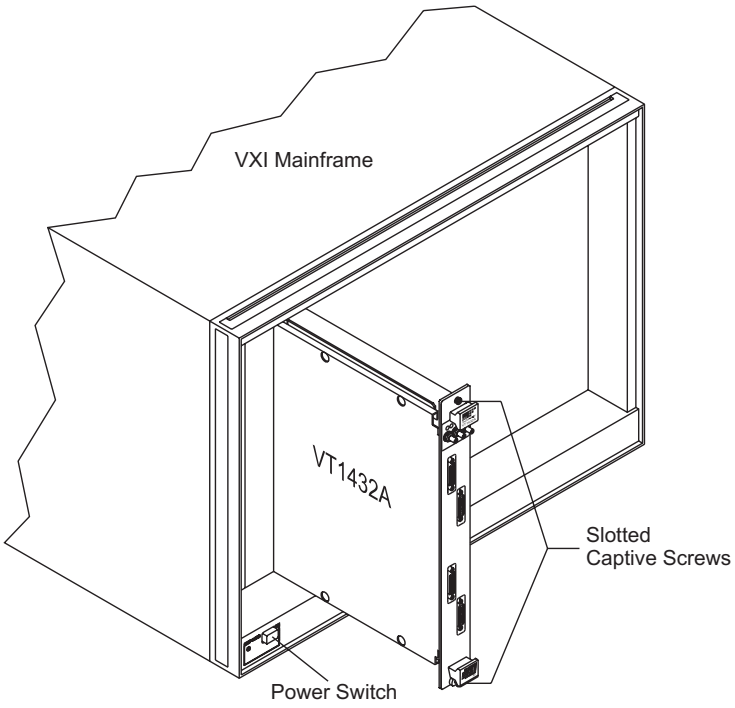
5 Set the mainframe's power switch to standby (⏻).

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

6 Place the module's card edges (top and bottom) into the module guides in the slot.

7 Slide the module into the mainframe until the module connects firmly with the backplane connectors. Make sure the module slides in straight.

8 Attach the module's front panel to the mainframe chassis using the module's captive mounting screws.



Install the host interface libraries

After the hardware has been assembled, the next step in installing the VT1432A is to install the host interface libraries. Refer to the chapter titled “Getting Started With the VT1432A” to continue the installation process.

To store the module

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

For other requirements, see storage and transport restrictions in the chapter titled: “Specifications.”

To transport the module

- Package the module using the original factory packaging or packaging identical to the factory packaging.
  Containers and materials identical to those used in factory packaging are available through VXI Technology offices.
- If returning the module to VXI Technology 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 350-pound test material.
  - ❑ Cushion the module to prevent damage.

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.

2

Getting Started With the VT1432A

Introduction

This chapter provides assistance in getting the VT1432A running and making simple measurements. It shows how to install the software libraries and how to run some of the example programs that are included.

For more information see the other chapters in this book and the on-line function reference. (See “Where to get more information” in the chapter titled “Using the VT1432A).”

Two versions of the Host Interface Library are available. One is HP-UX C-Language Host Interface Library which uses SICL (the Standard Instrument Interface Library) to communicate to the VT1432A hardware. The other is the HP-UX, Windows 95 or later and Windows NT VXI*plug&play* Library which communicates with the hardware using VISA (Virtual Instrument Software Architecture.) VISA is the input/output standard upon which all the VXI*plug&play* software components are based.

This chapter mainly covers the VXI*plug&play* version and it also includes some examples using the C-Language version. If using the C-Language version, please refer to the chapter titled "The Host Interface Library."

NOTE

The C-Language Host Interface Library has been provided for the purpose of backward compatibility and is no longer supported. New users should use the VXI*plug&play* Library while older users are encouraged to migrate their applications to the VXI*plug&play* library.

To install the VXI*plug&play* libraries

System Requirements (Microsoft Windows 95 or later and NT)

- An IBM compatible personal computer with either Microsoft Windows 95 or later or Microsoft Windows NT. (With either Windows OS, use the VXI*plug&play* library).
- Additional hardware and software to connect the IBM compatible computer to a VXI mainframe.
- Software is supplied on CD-ROM.

System Requirements (HP-UX 10.20)

- One of the following workstations:
  - ❑ An HP V743 VXI-embedded workstation.
  - ❑ A stand-alone HP Series 700 workstation with an Agilent/HP E1489I EISA-to-MXIbus card and an Agilent/HP E1482B VXI-MXI Bus Extender.
- Software is supplied on CD-ROM, so a CD-ROM drive is needed.
- HP-UX Version 10.20. This version of HP-UX can use either the C-Language library or the VXI*plug&play* library.
- SICL/VISA (Agilent/HP product number E2091E, version E.01.01 or later).

VT1432A Software Distribution

The VT1432A distribution (software) is shipped on CD-ROM with the VT1432A module. This distribution includes the VT1432A C-Language Host Interface library for HP-UX, the VT1432A VXI*plug&play* Host Interface library for HP-UX, Windows 95 or later and Windows NT with associated examples and manual pages.

Getting Updates (Windows)

The latest version of the VT1432A instrument drivers can be found on-line at www.vxitech.com.

Getting Updates (HP-UX)

For the latest HP-UX instrument drivers, please contact VXI Technology Customer Support Services. Contact information can be found in the *Support* section of this manual on page xiii.

To install the Windows VXI*plug&play* drivers for the VT1432A (for Windows 95 or later and Windows NT)

1 Insert the VXI*plug&play* Drivers and Product Manuals CD into the CD-ROM drive.

2 Run the program: d:\drivers\DAQ Drivers\driver_vxipnp_e1432_a_06_13.exe. (If the disc is in a driver other than "drive d:," replace "d:\" with the letter of the drive containing the driver disc.) Note that the "a_06_13" references the software revision and will vary. Follow the on-screen instructions to continue.

3 The *VISA Installation Information* dialogue box will appear. This indicates window will indicate whether or not the VISA library has been correctly installed previously. If not installed, an error message will appear as a reminder to install the library. Click Next to continue.

4 The *Choose Program Folder Items* dialogue box provides options that can be included in the Start Menu Program Folder for the VT1432A. Press Next when finished with selections to continue.

5 The *Select Program Folder* dialogue box appears providing the opportunity to change the name of the program folder that will be created in the Start Menu. The default name is "VXIPNP." Click Next to continue.

6 Setup creates a program group called “Hpe1432” (located typically in c:\VXIPNP\WINNT). It includes:

An icon to run the Soft Front Panel
An icon for HELP text
An icon for UNISTALL the README text can be included optionally.

These icons can also appear in the Startup Menu Program Files (see step 6).

7 After the program files load, the *Setup Complete* dialogue box will appear. It provides the opportunity to view the Readme file and to run the Soft Front Panel upon completion of set up. Click Next after the desired selections are made.

To install the HP-UX VXI*plug&play* drivers for the VT1432A (for HP-UX systems)

1. Log in as root.

2. Insert the VT1432A CD_ROM into the CD-ROM drive or obtain the latest VT1432A distribution.

3. Type `swinstall`.

   See the HP-UX Reference manual for information on the swinstall command.

   The VT1432A distribution is normally installed in the /opt/vxipnp/hpux/hpe1432/ directory. The files have extensions such as .h, .fp, .sl and .hlp.

The Resource Manager

The Resource Manager is a program from the hardware interface manufacturer. It looks at the VXI mainframe to determine what modules are installed. It should be run every time the system is powered up. If a "No VT1432A can be found in this system" message appear, run the Resource Manager.

Before running the VT1432A software make sure that the hardware is configured correctly and that the Resource Manager runs successfully. Before using the measurement system, all of its devices must be set up, including setting the 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 hardware interface documentation.

The VXI*plug&play* Soft Front Panel (SFP)

Using the soft front panel

If the VT1432A software is run in a Microsoft Windows OS (95 or later) or Windows NT, the Soft Front Panel (SFP) program can be used to interface with the VT1432A.

The Soft Front Panel can be useful for checking the system to make sure that it is installed correctly and that all of its parts are working. However, it is not very useful for making measurements. It cannot be controlled from a program and it does not access all of the VT1432A's functionality.

Note

The software examples provided in this manual contain "E" references (e.g. "E1432") used by the previous manufacturer. The VXI Technology equivalent can be derived at by exchanging the "E" references with "VT" references (e.g. "VT1432").

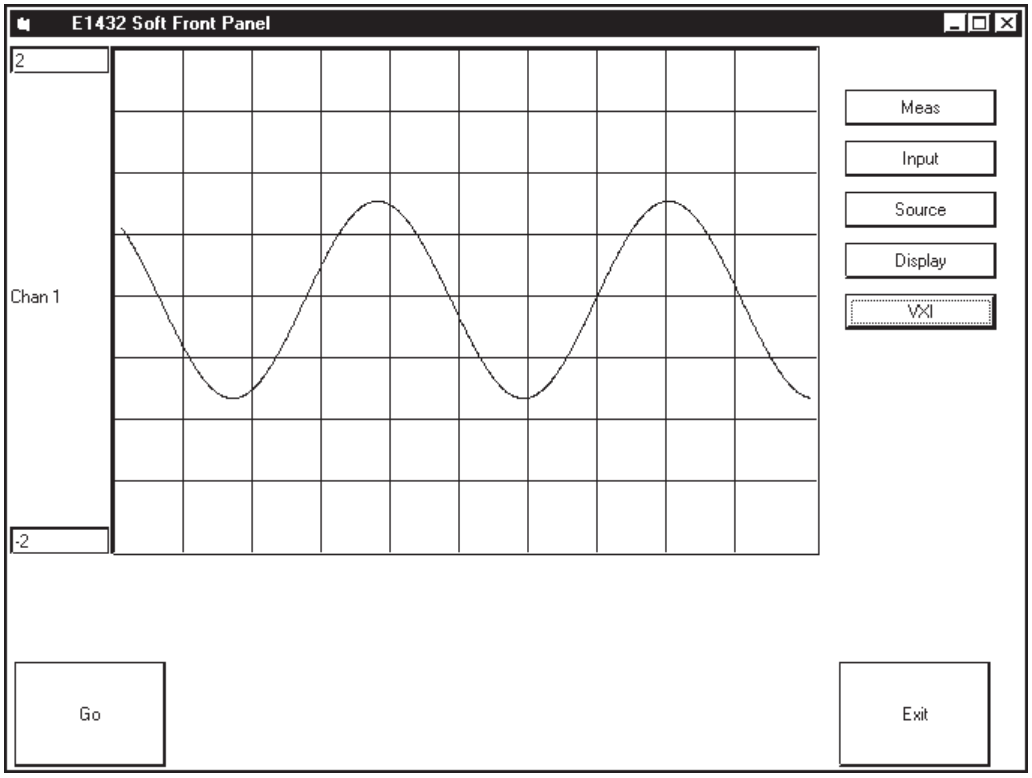


Figure 2-1: The Soft Front Panel interface

The buttons on the right side of the SFP display are defined as follows:

Meas

This button opens the Measurement Control dialog box which sets:

- Measurement single/repeat
- Mode block/continuous
- Trigger auto/manual/input
- Frequency span
- Block size

Input

This button opens a dialog box which sets up the VT1432A's inputs and configures:

- Channel number
- Range
- AC or DC coupling
- Grounding method
- Digital anti-alias filter
- Analog anti-alias filter
- Trigger on/off
- Trigger mode level/bound
- Trigger level
- Hysteresis
- Trigger Slope

There is a checkbox to make all channels identical.

Source

This opens a dialog box for controlling the source output of the VT1432A's source. This is only available for VT1432A's that have the Arbitrary Source Option VT1432A-1D4. This sets:

- Channel number
- Active on/off
- Mode sine/burst sine/random/burst random
- Ramp rate
- Sine frequency
- Sine phase
- Output normal/grounded/open/cal/multi
- Cola (Constant Output Level Amplifier) off/on
- Duty Cycle
- Sum off/on
- Seed
- Range

Display

This button opens a dialog box that specifies how the data will be displayed. For each trace, an input channel (or OFF) and an output file can be specified.

VXI

This button opens a dialog box showing the modules installed in the VXI mainframe and indicating which are active and inactive. The “resource name” for each module is the interface card name that has been assigned to it.

Go

Use the Go button to start the measurement.

Exit

Use the Exit button to exit the Soft Front Panel.

VEE example programs

scope.vee

This program displays four channels with time record and FFT for each channel. Agilent VEE (Virtual Engineering Environment) is used for the following examples.

The scope.vee program is located at \Hpe1432\examples\vee\ on a Microsoft Windows system or at /usr/e1432/vee-examples on an HP-UX system.

To run scope.vee, first type:

veetest

To begin using Agilent VEE.

In Agilent VEE click on File, then Open. In the Open File dialog box select scope.vee from the list of files. Then click OK.

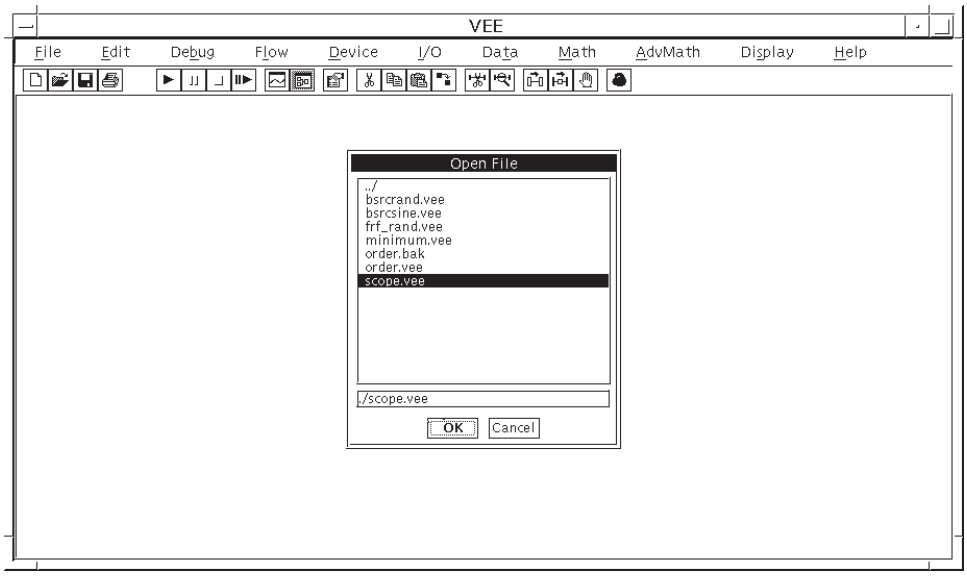


Figure 2-2: Agilent VEE - Open File dialog box

The program scope.vee starts, showing four channels, with time record and FFT for each channel.

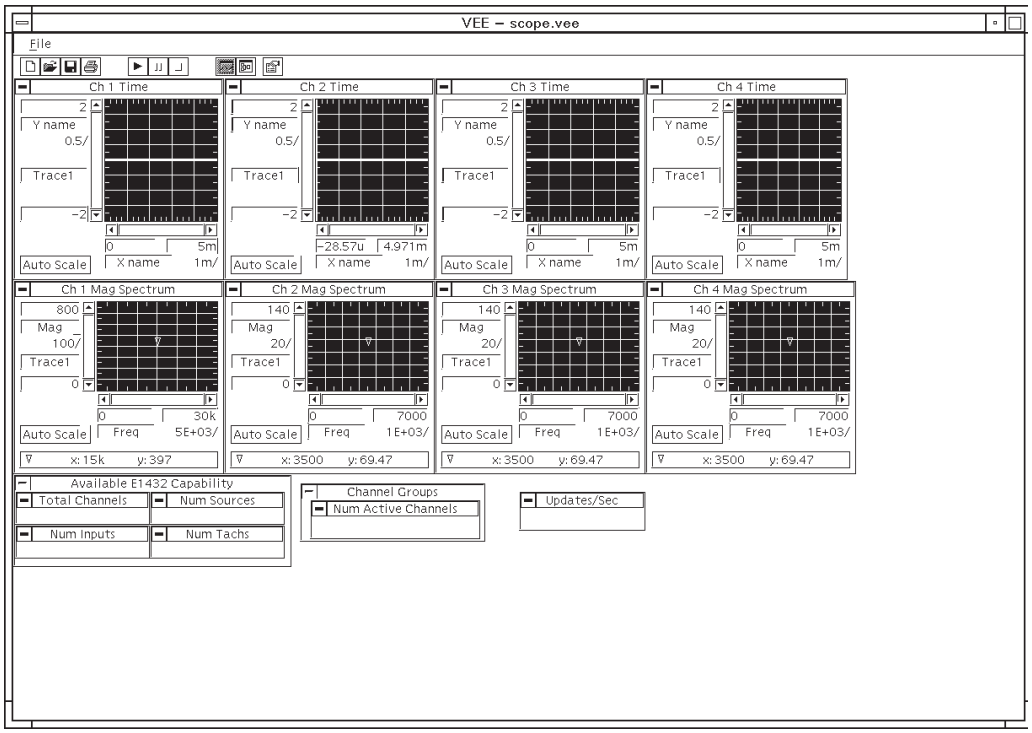


Figure 2-3: scope.vee - panel view

To start a measurement, click the Run button on the toolbar (triangle symbol). To pause, click on the Pause button (two vertical bars, next to the Run button). To stop the measurement, click the Stop button (square symbol).

This screen is VEE's panel view, an interface which allows for as much interaction with the system as with the actual front panel of a standalone instrument. VEE's detail view screen allows for configuration of the system and the view panel for making measurements.

To look at the scope.vee program "behind the scenes," click on the View Detail button on the toolbar (chart symbol). To return to the original (panel) view, click on the View Panel button (sine wave symbol).

Click on the View Detail button again to look at the detail view screen

To use and modify scope.vee, it is necessary to be familiar with using the Agilent VEE program. Refer to Agilent VEE documentation if necessary. In View Detail mode, click on the Help on the menu bar for assistance in using Agilent VEE.

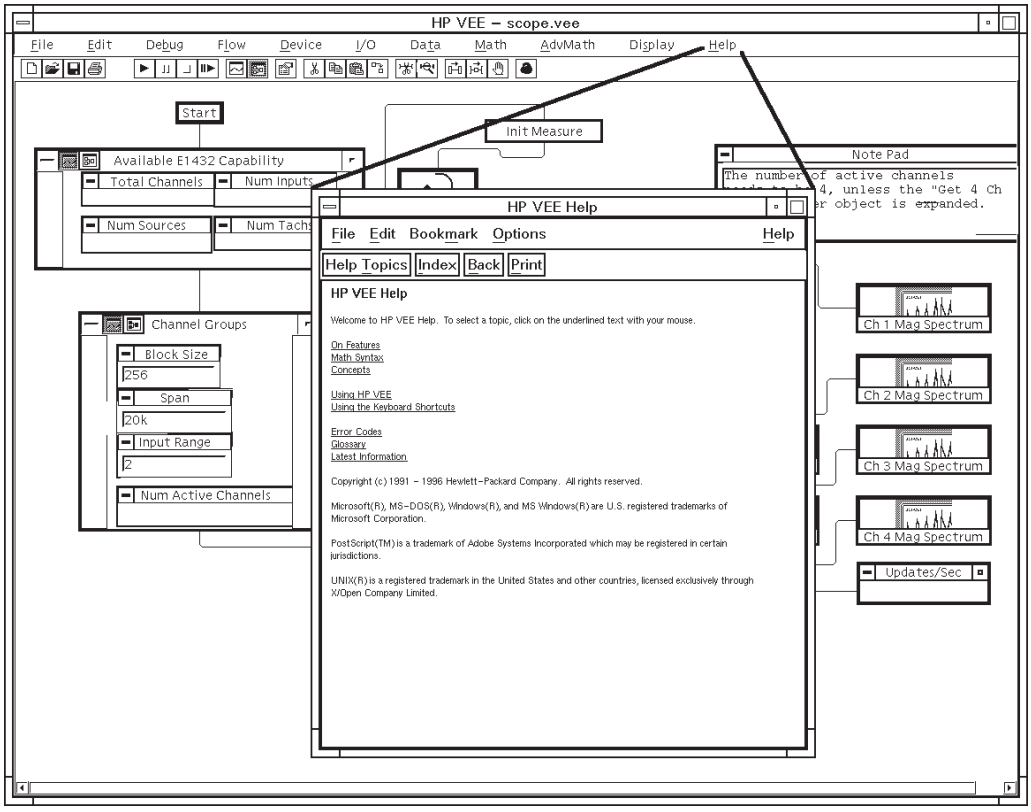


Figure 2-4: Agilent VEE help text

In detail view there are boxes representing parts of the scope.vee program. For programs that are too large to be viewed all at one time, use the scroll bars at the bottom and left side of the screen to scroll the display. To see more detail, double-click on a box or click on the View Detail (chart symbol) button on the top bar of the box. Some of the boxes contain a function. Clicking on a function displays the parameters associated with it.

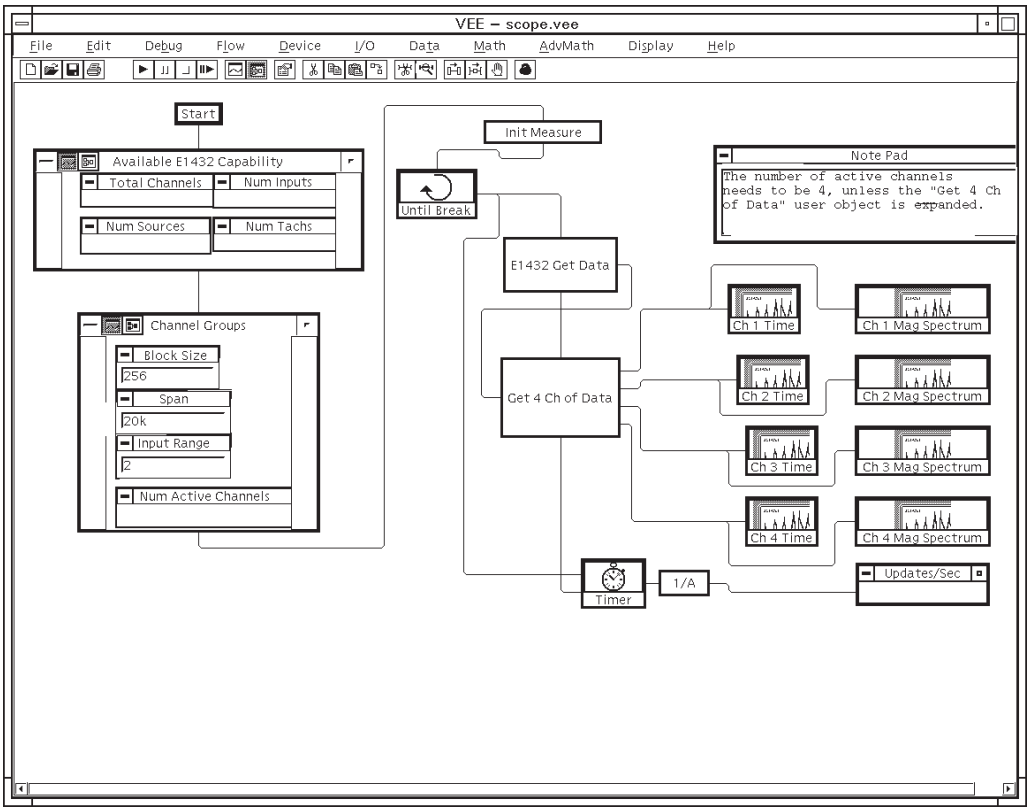


Figure 2-5: scope.vee - detail view

To specify a new function, click on the blank space in the box where the function is to be. A dialog box appears with a list of functions. After selecting a function, choose Panel to “hard code” constants that the function will use or choose Parameters to allow a parameter to be input from elsewhere (from the user or another function). The input appears as a “pin” on the chart diagram. In the scope.vee program the user can select the block size, span and range.

Click Add To Panel in the Edit menu to make a box in the detail view visible on the panel view. This gives the user access to enter parameters or view results.

Clicking on Alphnumeric in the Display menu sets up a box to specify how to display the output of a function.

Use Agilent VEE to look at the functions that make up the simple “scope.vee” program. This is an example of how the VT1432A can be programmed using Agilent VEE.

Click on the Panel View button (sine wave symbol) to go to panel view. Set up the system to provide input signals to some of the input connectors of the VT1432A. Then use the scope.vee to interface to view the time records and FFTs of the input signals.

When exiting Agilent VEE, the program will ask to save any changes made to scope.vee. Click No or, if the changes are to be saved, click Cancel and then use File/Save As to save the changes as a different filename.

minimum.vee

This program provides a simple example to assist in the learning and use of the VT1432A library, although it is not intended to be a finished “user-friendly” program. It contains the minimum number of functions needed (nine functions) to get data from the VT1432A module. It does not even include a “panel” user interface, so the first screen seen is the VEE View Detail screen. Use the scroll bar at the bottom of the screen to scroll the display and see all of the detail view.

The minimum.vee program simply takes data for one channel and then stops. It may be useful to examine this program and use it as a starting point for learning how to write VEE programs for the VT1432A.

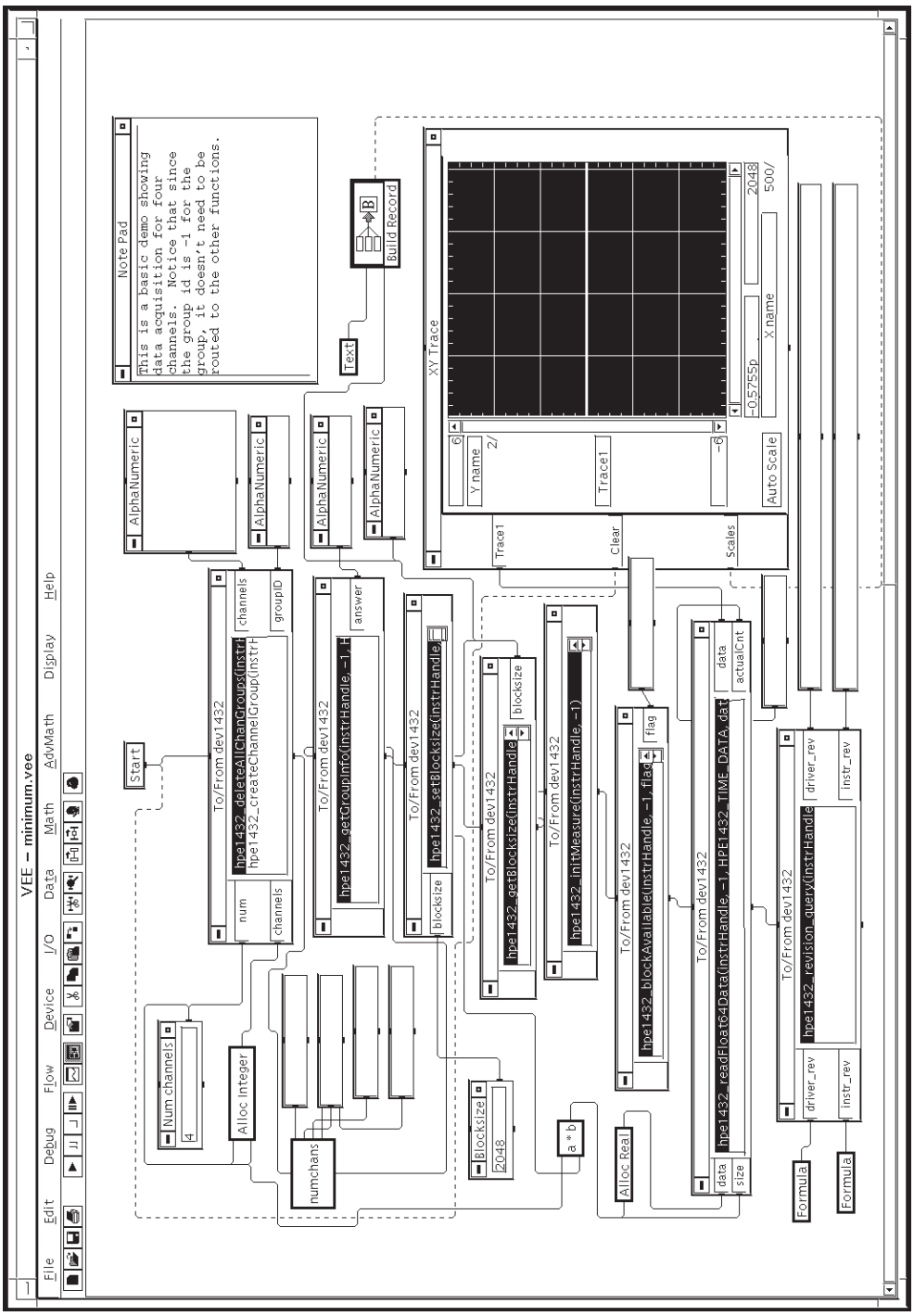


Figure 2-6: minimum.vee (scroll to see entire display)

Other VEE example programs

There are several other example programs that can be examined in the same way as scope.vee. These programs are in the path \Hpe1432\examples\vee\ on a Microsoft Windows system or /usr/e1432/vee-examples on an HP-UX system.

bsrcsine.vee (Burst SouRCe SINE)

This program is similar to scope.vee. It displays eight (rather than four) channels with time record and FFT for each channel. It also turns on the source in burst sine mode and ramps up the source output. The user can specify the duty cycle, ramp rate, level of the source and frequency of the source. This program works with VT1432A's which are equipped with the source option ID4.

bsrcrand.vee (Burst SouRCe RANDom)

This program is like bsrcsine.vee except the source is turned on in burst random mode.

frf_rand.vee. (Frequency Response Function RANDom)

This program displays the frequency response of four channels. One way to set up this example is to connect a cable between the channel 1 and channel 2 inputs. Then connect channel 3 to channel 1 through a "black box" containing the circuit to be tested (using a "T" on channel 1). Channel 4 remains unconnected. The display will show a response for channel 2 over channel 1 (a flat response for the bare cable) and a response for channel 3 over channel 1 (representing the frequency response of the "unknown" circuit). Channel 4 will show a random signal since it has no input.

order.vee

This program can be used only with a VT1432A with the tachometer option. It takes four channels of data and displays two channels. It shows raw time domain data and resampled data for each rpm value. The raw data can then be processed with a program such as Matlab to make order ratio maps.

C-Language Host Interface Library example programs

The VT1432A C-Language Host Interface Library comes with several example programs, which help demonstrate how to use the library. These example programs are found in the "VXIPNP\winXX\Hpe1432\examples\hpux" directory or the "/opt/e1432/examples" directory. The programs in this directory are all very small, so that they will be easily understood and easy to copy into a real application.

The files in the examples directory are:

README	A file containing the information given here.
detect.c	Shows how to use SICL calls to find the logical addresses of the VT1432A modules in a system.
example.c	Shows the basics of setting up a VT1432A, starting a measurement and reading a block of data.
intr.c	Shows how to set up SICL and a VT1432A to use interrupts for data collection.
src_intr.c	Shows how to set up SICL and a VT1432A to use interrupts with a VT1432A-1D4 Source board, for overload shutdown and overread.
tachmon.c	Shows how to monitor a tach channel signal using the other inputs in the VT1432A module.
Makefile	A UNIX Makefile which can be used to compile all of the programs in the examples directory.

Demo Programs

In addition to example programs, the VT1432A Host Interface library also comes with demo programs. These programs are found in the"/opt/e1432/demo" directory.

One of these demo programs, called "semascope," demonstrates that the VT1432A hardware and software are working properly. When run, it identifies the VT1432A modules in the VXI system, runs a measurement using the VT1432A modules that it finds and plots the results in X11 windows. This program is not meant to be an example of how to use the VT1432A library, although the source code is not provided.

Other demo programs include "rpmtrig" and "rpmtrig2" and "semascope3."

Running a demo program: semascope.c

To run this program, type:/opt/e1432/demo/semascope. This program displays the time records for 32 channels (when hooked up to two VT1432A modules with 16 channels each). The channel that is active for changing the display is highlighted. To exit, double-click the horizontal bar symbol in the upper left corner of the window.

To see a list of parameters for semascope, type:

semascope -u

To specify a parameter, type its letter code after “semascope” on the command line.

The source code for this program is at:

/opt/e1432/demo/semascope.c

Use a text viewer or editor (such as the “more” utility in UNIX) to list the source code for semaphore.c. Examine the code to learn more about how this example program works.

Visual Basic example programs

VEE and the C Host Interface Library can be used on both UNIX and PC systems. In addition, the PC can use Visual Basic. Visual Basic example programs are at VXIPNP\winXX\Hpe1432\examples\vb\ on a Microsoft Windows system.

3

Using the VT1432A

Introduction

This chapter shows how to use the VT1432A using the VXI*plug&play* Host Interface Library.

The Host Interface Library for the VT1432A is a set of functions that allow the user to program the register-based VT1432A at a higher level than register reads and writes. The library allows groups of VT1432As to be set up and programmed as if they were one entity.

Two versions of the Host Interface Library are included. One is the HP-UX C-Language Host Interface Library which uses SICL (the Standard Instrument Interface Library) to communicate to the VT1432A hardware. It works for HP-UX 10.20. The other is the VXI*plug&play* Library for Windows 95 or later, Windows NT and HP-UX 10.20 which communicates with the hardware using VISA (Virtual Instrument Software Architecture). VISA is the input/output standard upon which all the VXI*plug&play* software components are based.

This chapter covers the VXI*plug&play* version, but it will also be useful to users of the C-Language version. If using the C-Language version, refer to the chapter titled “The Host Interface Library” for more information.

The library includes routines to set up and query parameters, start and stop measurements, read and write data and control interrupts. Routines to aid debugging and perform low-level I/O are also included.

For information on diagnostics see the chapter titled “Troubleshooting the VT1432A.”

What is VXI*plug&play*?

VXI Technology uses VXI*plug&play* technology in the VT1432A. This section outlines some of the details of VXI*plug&play* technology.

Overview

The fundamental idea behind VXI*plug&play* is to provide VXI users with a level of standardization across different vendors well beyond what the VXI standard specifications spell out. The VXI*plug&play* Alliance specifies a set of core technologies centering on a standard instrument driver technology.

VXI Technology offers VXI*plug&play* drivers for VEE-Windows. The VXI*plug&play* instrument drivers exist relative to so-called “frameworks”. A framework defines the environment in which a VXI*plug&play* driver can operate. The VT1432A has VXI*plug&play* drivers for the following frameworks: Windows 95 or later, Windows NT, and HP-UX.

VXI*plug&play* drivers

The VT1432A VXI*plug&play* driver is based on the following architecture:

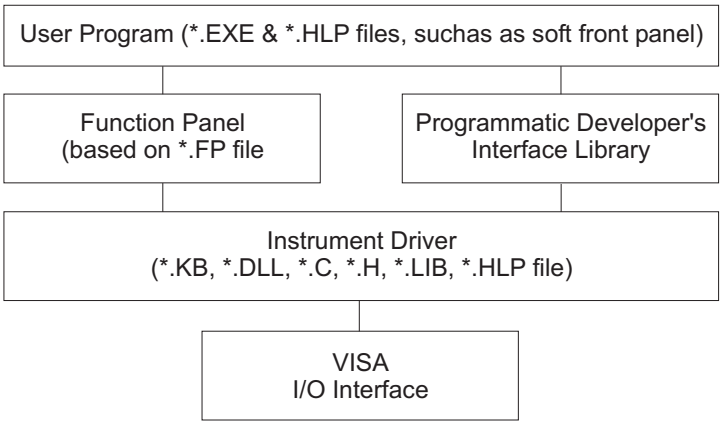


Figure 3-7: VXI Plug&Play driver architecture

It is most useful to discuss this architecture from the bottom up.

The VISA/VTL I/O interface allows interoperability of the VXI*plug&play* driver technology across interfaces.

The actual instrument driver itself is a DLL (Dynamic Linked Library) created from:

- A set of source (.C) files.
- A set of header (.H) files, used for compiling the file as well as to describe the driver's calls to any program using the driver.
- A standard driver library (.LIB) file, to provide the standard functionality all the drivers would require.

This DLL is a set of calls to perform instrument actions — at heart, that's all a VXI*plug&play* driver is — a library of instrument calls.

This driver is accessed by Windows applications programs written in languages such as Visual C++ or Visual BASIC, using programming environments such as VEE or NI LabView.

A Windows Help (.HLP) file is included which provides descriptive information and code samples for the functions in the VXI*plug&play* DLL. This help file can be viewed in the standard Windows Help viewer. A viewer for HP-UX is provided in /opt/hyperhelp - see the READ.ME file.

The Soft Front Panel (SFP)

The Soft Front Panel is a stand-alone Windows application, built on top of the VXI*plug&play* driver DLL; it is used for instrument evaluation and debugging and as a demo. It is not a programmable interface to the instrument, nor can it be used to generate code.

The soft front panel also accesses the same Windows Help file as provided with the DLL.

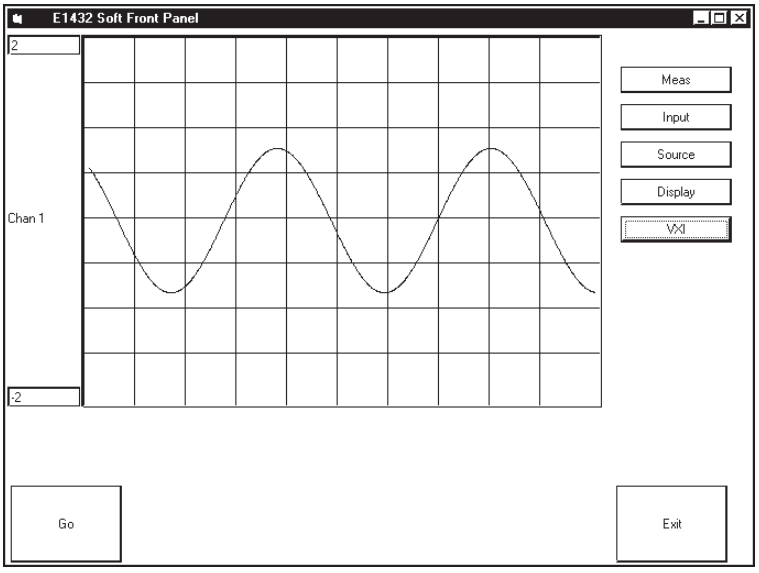


Figure 3-8: An example of a soft front panel (SFP)

Header and Library Files

In a Windows environment, the following files are in the directory \Vxipnp:

KBASE \ hpe1432.kb	Knowledge base file
winXX\ Bin \ hpe1432_32.dll	The VXI*plug&play* driver
winXX\ Include\ hpe1432.h	Header for linking to the VXI*plug&play* driver
winXX \ Lib \ Msc \ hpe1432_32.lib	Lib for linking C programs to VXI*plug&play*

The following files are in the directory \Vxipnp\WinXX\Hpe1432:

hpe1432.exe	Soft front panel program
hpe1432.bas	header for Visual Basic
hpe1432.fp	The “FP” file used by VEE and CVI
hpe1432.hlp	Windows help file
Read.me	The latest information for the product
examples \ vb *	Visual Basic example programs
examples \ c *	C example programs
examples \ vee *	Agilent VEE example programs
lib \ sema.bin	Firmware program for the VT1432A
lib \ sfp.ico	Icon for help file
lib \ sinewave.ico	Icon for Soft Front Panel
source*	Source files for hpe1432_32.dll

In the HP-UX environment, the following files are in the directory /opt/vxipnp/hpux/hpe1432:

hpe1432.fp	The “FP” file used by VEE
.h	Header file
.hlp	Hyperhelp file (see /opt/hyperhelp/README for information on how to view hpe1432.hpl In the HP-UX environment.)
.sl (lower-case “SL”)	The VXI*plug&play* shared library

Channels and groups

This section gives some information about using channels and groups. For more detailed information see the VT1432A help text.

Channel Groups

In the VT1432A VXI*plug&play* driver, a channel group is the basic unit of hardware control. To control any channel, it must first be assigned to a group with the hpe1432_createChannelGroup function. In addition to creating the group, this function returns a "handle" that uniquely identifies the group. This handle can then be used to direct functions to all channels in the group.

When a channel group is created, all input and tach channels in the group are automatically activated and all source channels are inactivated. But, when deleted, the input and tach channels are not automatically inactivated. Any input or tach channel that remains active after its group is deleted will continue to supply data to its module's FIFO buffer during a measurement—consuming module resources. For this reason, channels should always explicitly inactivated in a group before deleting them. Channels can be inactivated with hpe1432_setActive. Delete channel groups with hpe1432_deleteChannelGroup and hpe1432_deleteAllChanGroups.

Also when creating a channel group, channels which are not mentioned in the new group are not turned off. Any channels that are not to be active must be explicitly inactivated. (An exception is a power-up, when only the channels in the initial channel group are active.)

Initialization

The command used to initialize the system is hpe1432_init. This function initializes the VXI*plug&play* library and registers all VT1432A modules. It also checks the existence of a VT1432 module at each of the logical addresses given in the resource list and allocates logical channel identifiers for each channel in all of the VT1432s. Input channels, source channels and tach/trigger channels are kept logically separated.

Most other functions cannot be used until after hpe1432_init, but there are two functions which can be used before initialization to get information needed by hpe1432_init. These are hpe1432_find and hpe1432_getHWConfig. hpe1432_find searches the VXI mainframe and returns the VXI Logical Address for every VT1432A found. hpe1432_getHWConfig returns additional information about the hardware.

After hpe1432_init is run, use hpe1432_getNumChans to get the total count of inputs, sources and tachs for all VT1432A modules named in the hpe1432_init call.

Creating a Channel Group

The function hpe1432_createChannelGroup creates and initializes a channel group. A channel group allows commands to be issued to several VT1432A channels at once, simplifying system setup. Channel groups can overlap. The state of an individual VT1432A channel that is in more than one channel group is determined by the most recent operation performed on any group to which this channel belongs.

As a side effect, this function makes all input and tach channels in the channel group active and all source channels in the channel group inactive. This function does not inactivate other channels within the modules that the channels are in and does not preset the channels in the new group.

After a channel group has been created, use hpe1432_getGroupInfo to get selected information about the group. It is possible for hpe1432_getGroupInfo to be set up to return the number of modules, channels, inputs, sources or tachs in the group. It can also return a list of the modules, channels, inputs, sources or tachs.

Input, Source and Tach Channels

Channel numbers must fall in particular ranges for different types of channels. Input channel numbers range from 1 to 4095. Source channel numbers range from 4097 to 8191. Tach channel numbers range from 8193 to 12287.

A mixture of input, source and tach channels may exist in one group. However it is also important for many functions to be sent only to the appropriate type of channel. For example, asking for a blocksize from a tach channel can cause an error. It may be useful to set up several channel groups at the beginning of the program: one for input channels, one for source channels, one for tach channels and one that combines all three channel types. The input handle could be used for input-only functions, the source for source-only functions and the tach handle for tach-only functions. The “all-channels” handle could then be used for all other functions.

Multiple-module/mainframe Measurements

Grouping of Channels/Modules

The interface library for the VT1432A is designed to allow programming of several channels from one or several distinct modules, as if they were one entity. Each VT1432A module has up to 16 channels. The library may control up to a maximum of 255 VT1432A modules (8160 channels).

The function hpe1432_createChannelGroup can be used to declare any number of groups of channels, possibly overlapping. Each group can be uniquely identified by a group ID.

The 'target' of a library function is either a channel, a group or (rarely) a module, depending on the nature of the call. When the same library function may be called with either a channel or a group identifier, its 'target' is shown by a parameter named ID.

Multiple-module Measurements

A channel group that spans more than one module will need to be configured to use the TTL trigger lines on the VXI Bus for inter-module communications. This configuration is automatically performed in the hpe1432_initMeasure call unless defeated using hpe1432_setAutoGroupMeas.

The following discussion outlines what hpe1432_initMeasure does automatically. This must be done by the user if hpe1432_setAutoGroupMeas has been used to defeat auto configuration.

There are eight VXI TTL trigger lines that can be used for multi-module synchronization. Often, these lines are used in pairs, one for sample clock and one for Sync/Trigger. The hpe1432_setTtltrgLines function selects which TTL trigger lines to use; this function always uses the TTL trigger lines in pairs. Calling hpe1432_setClockSource with the group ID will set all modules to the same pair.

All modules need to be set to use the shared sync line rather than the default setting of internal sync. This can be done with the hpe1432_setMultiSync function, using the group ID.

One module of the set of modules needs to be set to output the sync pulse. The module with the lowest VXI logical address is called the "system module" and is assigned this duty. This can be set with the hpe1432_setMultiSync function call, using the lowest channel ID in the group (NOT the group ID).

All modules except the “system module” need to be set to use the VXI TTL trigger lines as the clock source. Use hpe1432_setClockSource for this.

Set the “system module” to output the clock. Use hpe1432_setClockMaster for this. After this is done, all system sync pulses come from the “system module” and drive the measurement state machines on all boards in the group.

Possible Trigger Line Conflict

The following describes a scenario where VT1432A modules might conflict and prevent a proper measurement. The conditions allowing the conflict are complex but must be understood by the user.

After a measurement has completed, the modules are left set up. If a module (call it module ‘A’) is driving the TTL trigger lines and a different group is started which also drives the TTL trigger lines (and that different group does not include module ‘A’), then module ‘A’ will conflict and prevent the other group from functioning. In this case make a call to hpe1432_finishMeasure (using the old group ID which includes ‘A’) to turn off module ‘A’ and allow the new group to function.

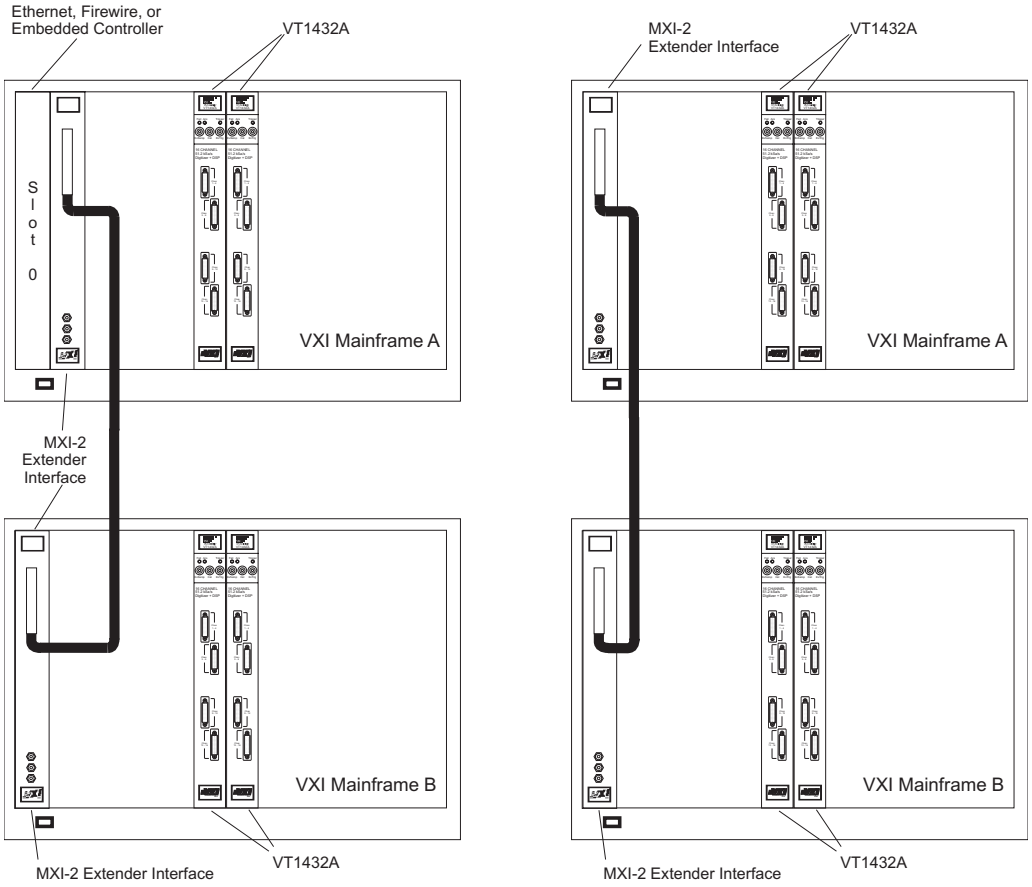
Note that if the new group includes all modules of the old group, the conflict will not occur since hpe1432_initMeasure will reset all modules as needed. Also note that single-module groups do not drive the TTL trigger lines, so single-module groups are immune from causing or receiving this conflict.

Managing Multiple-mainframe Measurements

In a single-mainframe measurement, the VT1432A communicates with other VT1432As through the TTLTRG lines. However, when using the VXI-MXI bus extender modules, the TTLTRG lines, which carry the group synchronization pulse and sample clock, are extended only in one direction. This unidirectional signal connection restricts the types of measurements that can be made in a multiple mainframe environment.

The following types of multiple mainframe measurements cannot be performed:

- ❑ Unequal pre-trigger delay settings between mainframes
- ❑ Channel triggering by channels in Mainframe B
- ❑ Lower spans or longer blocksizes in Mainframe B
- ❑ Different digital filter settling times between VT1432A modules



Example 1: Slot 0 Controller **Example 2: MXI-2 Daisy Chain**

Figure 3-9: Multiple mainframes - two mainframes

In the example above, Mainframe A contains the Slot 0 Controller for a multiple mainframe system. Mainframe A is connected to Mainframe B with a VXI-MXI interface, Agilent/HP E1482B. To successfully manage this multiple mainframe environment, use the following guidelines.

- ❑ Locate modules with logical addresses less than 128 in Mainframe A.
- ❑ Locate modules with logical addresses greater than 127 in Mainframe B.
- ❑ Locate the highest-numbered channels in Mainframe A.
- ❑ Locate the last module in the module list specified in the call to hpe1432_init in Mainframe A.
- ❑ Locate the module that generates the group synchronization pulse in Mainframe A.
- ❑ Locate the channels performing channel triggering in Mainframe A.
- ❑ Locate the module with the shared sample clock in Mainframe A.
- ❑ If a groupID is not used with the call hpe1432_readRawData or hpe1432_readFloat64Data, empty the VT1432As' FIFOs in Mainframe B before Mainframe A. In other words, do not empty the FIFOs in Mainframe A unless the FIFOs in Mainframe B have been emptied. For more information about groupID see "Grouping of Channels/Modules" in this chapter.
- ❑ If more than two mainframes are needed, daisy-chain them together. Treat each mainframe after the first as a Mainframe B. See the example on the next page.

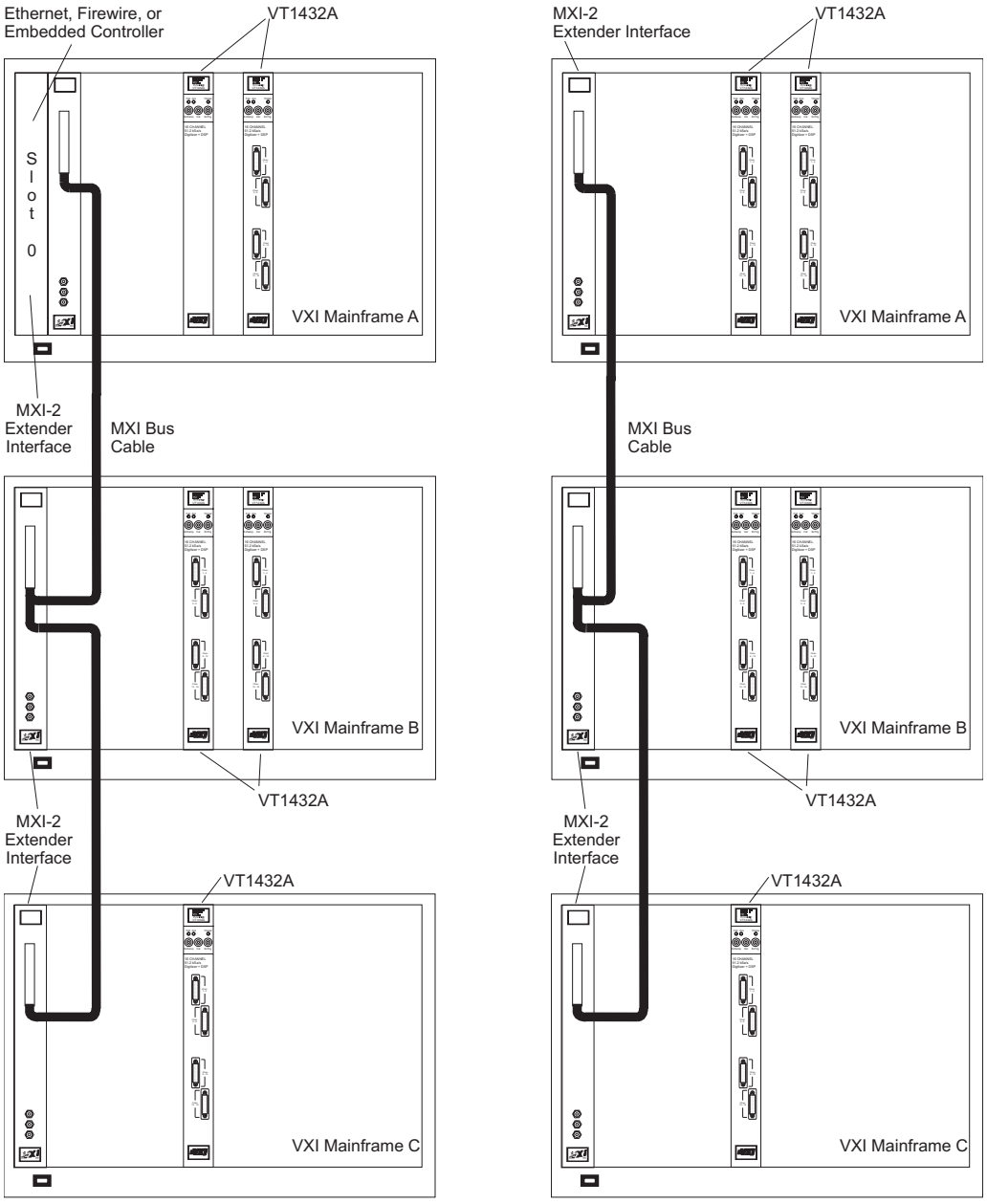
Phase Performance in Multiple Mainframe Measurements

Phase specifications are degraded by the delay that the inter-mainframe interface gives the sample clock. This delay is insignificant for many low-frequency applications because the phase error is proportional to frequency. A system with two VXI-MXI modules and a one-meter cable, typically has a 76 nanosecond (ns) sample clock delay in Mainframe B. This corresponds to an additional 0.007 degree phase error at 256 Hz and an additional 0.55 degree phase error at 20 kHz.

Using a four-meter cable (which adds approximately 18 ns of delay) causes a total of 94 ns clock delay in Mainframe B. This corresponds to an additional 0.0087 degree phase error at 256 Hz and an additional 0.68 degree phase error at 20 kHz.

The cable adds approximately 6 ns per meter of cable.

Each daisy-chained mainframe adds another increment of delay, but only for the additional cabling length.



Example 3: Embedded Slot 0 Controller

Example 4: MXI-2 Daisy Chain

Figure 3-10: Multiple mainframes - three mainframes

Synchronization in Multiple-mainframe Measurements

A TTL Trigger line between VT1432As making group measurements keeps all modules synchronized. This is an open-collector line where each module holds the one designated as the SYNC line low until the module is ready to advance to the next measurement state. Another TTL Trigger line is designated to carry the sample clock to all modules. This shared sample clock may come from any VT1432A module in Mainframe A or from an external signal routed through the Slot 0 Commander in Mainframe A.

One module is responsible for pulling the SYNC line low to start each group's state transition. Then, each module holds the line low until it is ready. When all modules are ready, the SYNC line drifts high. The unidirectional line prevents modules in Mainframe B from holding-off modules in Mainframe A.

The lowest logical address must be in Mainframe A because of VXI-MXI and Resource Manager (RM) constraints. Group constraints with the VXI*plug&play* Library force modules in Mainframe A to have their FIFOs emptied last. The VXI*plug&play* library reads data in channel order, so the highest channel is read last. To get this to work automatically, the call to hpe1432_init must list the logical addresses in descending order.

Channel triggering must be done only by modules in Mainframe A. A trigger in any other mainframe would not be communicated back on the SYNC line to Mainframe A. The VXI*plug&play* Library itself selects the VT1432A with the highest channel number for synchronization.

VXI-MXI Module Setup and System Configuration

The VXI-MXI Module setup in Mainframe A needs to be changed from those set by the factory. The VXI-MXI module is not the Slot 0 Controller for Mainframe A. See Table 2-1: Configuration Settings in the Agilent/HP E1482B VXI-MXI Bus Extender User's Manual. This requires changing several switch settings.

- ❑ Set the module as not being the Slot 0 Controller.
- ❑ Set the VME timeout to 200 μs.
- ❑ Set the VME BTO chain position to 1 extender, non-slot0.
- ❑ Do not source CLK10.
- ❑ Set the proper logical address.

Module Features

Data Flow Diagram and FIFO Architecture

The illustration on the next page shows data flow in the VT1432A. In this example there are four 4-channel input assemblies for a total of 16 input channels. The data for all channels is sent to the FIFO. The FIFO is divided into sections, one for each channel. The data moves through a circular buffer (first-in-first-out) until a trigger causes it to be sent on to the VME Bus. The data can also be sent to the Local Bus if option UGH is present.

The size of the sections in the FIFO is flexible. The amount of DRAM memory for each channel is the total DRAM memory divided by the number of channels. The standard DRAM size is 4 MB; an optional 32 MB DRAM is available.

The trigger can be programmed to trigger on the input or on information from the software. The following are examples of ways a trigger can be generated.

- input level or bound
- source
- external trigger
- RPM level (with tachometer option AYE)
- ttl_trigger (VXI backplane)
- freerun (automatic)

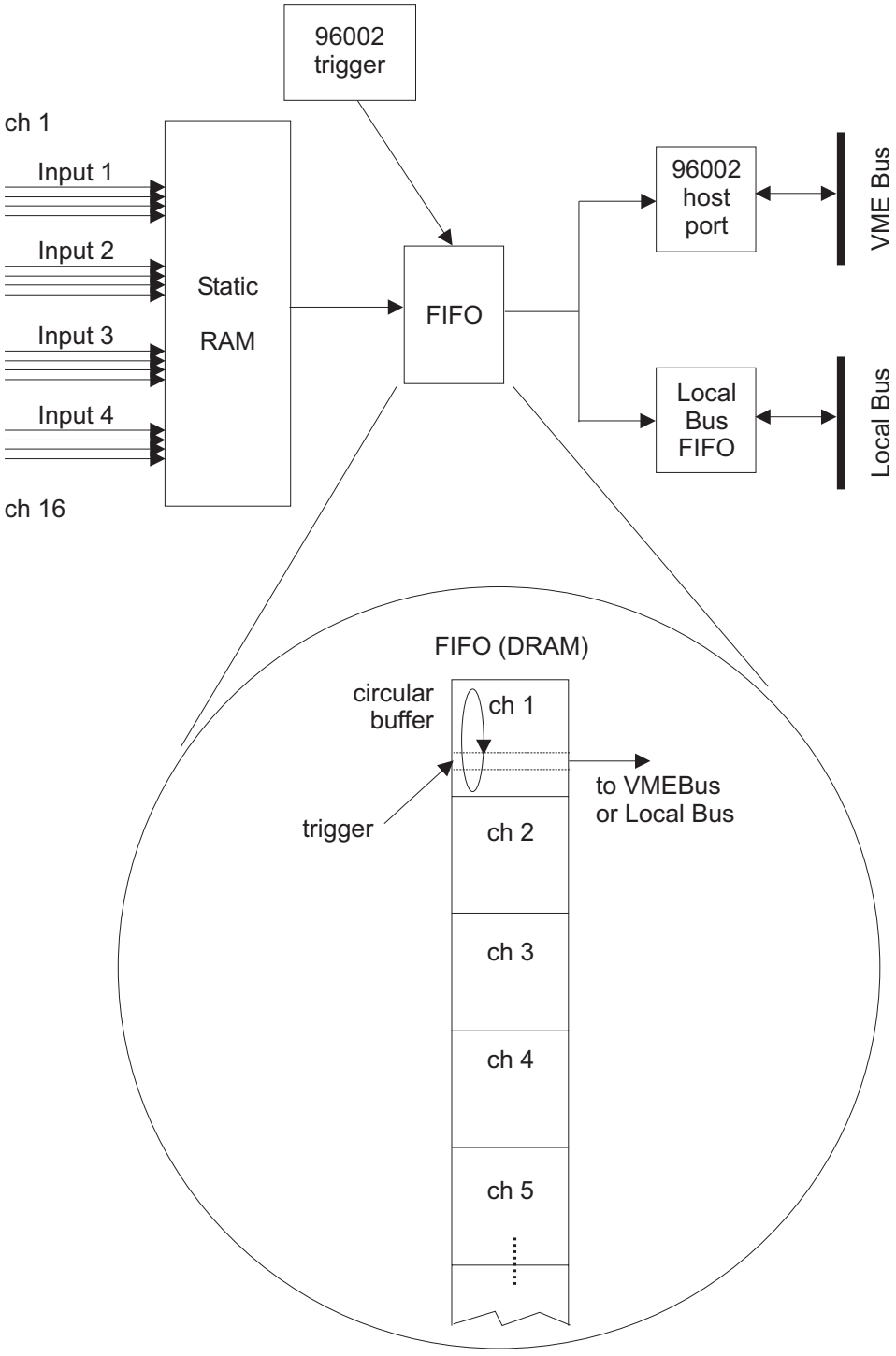


Figure 3-11: Data flow and FIFO architecture

Base Sample Rates

Baseband Measurement Spans

The table on the following page shows the measurement spans available for base sample rates, for baseband measurements.

"F_S" is the sample frequency or sample rate. The value for zero divide-by-two steps and no divide-by-5 step is the top measurement span corresponding to the sample rate. This is with no decimation and using 400 lines to avoid alias. The other values on the table are for this top span decimated by five and/or two.

For a VT1432A which has option VT1432A-1D4, the Arbitrary Source, the sample rate for the source is automatically set to be the same as the sample rate selected for the inputs. When the source is active the sample rate cannot be greater than 65.536 kHz.

Decimation Filter Diagram

The drawing below illustrates the way the spans in the table are generated. In the case of baseband spans (lower limit of span fixed at zero), the frequency can (optionally) be divided by five and then (optionally) divided by two up to eight times.

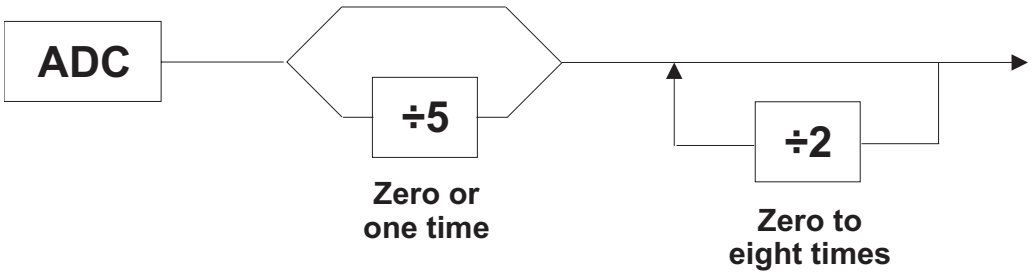


Figure 3-12: Decimation filter diagram - baseband

Table of Baseband Measurement Spans (Hz)

sample frequency (Fs) —>	32000		32768		40000		40960	
number of ÷2 steps	with ÷5	without ÷5	with ÷5	without ÷5	with ÷5	without ÷5	with ÷5	without ÷5
0	2500	12500	2560	12800	3125	15625	3200	16000
1	1250	6250	1280	6400	1562.5	7812.5	1600	8000
2	625	3125	640	3200	781.25	3096.25	800	4000
3	312.5	1562.5	320	1600	390.625	1953.125	400	2000
4	156.25	781.25	160	800	195.3125	976.5625	200	1000
5	78.125	390.625	80	400	97.65625	488.2813	100	500
6	39.0625	195.3125	40	200	48.82813	244.1406	50	250
7	19.53125	97.65625	20	100	24.41406	122.0703	25	125
8	9.765625	48.828125	10	50	12.20703	61.03516	12.5	62.5

sample frequency (Fs) —>	48000		50000		51200	
number of ÷2 steps	with ÷5	without ÷5	with ÷5	without ÷5	with ÷5	without ÷5
0	3750	18750	3906.25	19531.25	4000	20000
1	1875	9375	1953.125	9765.625	2000	10000
2	937.5	4687.5	976.5625	4882.813	1000	5000
3	468.75	2343.75	488.2813	2441.406	500	2500
4	234.375	1175.875	244.1406	1220.703	250	1250
5	177.1875	585.9375	122.0703	610.3516	125	625
6	58.59375	292.9688	61.03516	305.1758	62.5	31.25
7	29.29688	146.4844	30.51758	152.5879	31.25	156.25
8	14.64844	73.24219	15.25879	76.29395	15.625	78.125

* For the top span the bandwidth is 1.15 times span shown.

Additional Notes on Measurement Spans

Bottom reference is 10 Hz, max span is 20 kHz.
Top span 23000 Hz = 460 lines.

To select a sample frequency for time domain data, first divide the desired sample frequency by 2.56 to convert it to a measurement span. Then locate the closest measurement span on this table and choose the corresponding sample frequency at top of the table.

The VT1432A can use measurement spans that match those of the Agilent/HP E1431A, although not for all of the Agilent/HP E1431A's range. Maximum span for the Agilent/HP E1431A is 25.6 kHz. The maximum Agilent/HP E1431A-compatible span for the VT1432A 12.8 kHz.

For a VT1432A which has option VT1432A-1D4, the Arbitrary Source, the sample rate for the source is automatically set to be the same as the sample rate selected for the inputs. When the source is active the sample rate cannot be 40.000 kHz.

Measurement Process

Measurement Setup and Control

When the VT1432A makes a measurement, the measurement itself consists of two phases: the measurement initialization and the measurement loop. Each of these phases consists of several states, through which the measurement progresses.

The transition from one state to the next is tied to a transition in the Sync/Trigger line (one of the TTL trigger lines on the VXI back plane). A state (such as Idle) begins when the Sync/Trigger line goes low. The Sync/Trigger line then remains low as long as the state is in effect. When the Sync/Trigger line goes high it signals the transition to the next state. See the sections “Measurement Initialization” and “Measurement Loop” below for more details about these transitions. During all the transitions of the Sync/Trigger line, the clock line continues with a constant pulse.

The Sync/Trigger line is “wire-OR’d” such that all modules in a multiple-module system (within one mainframe) must release it for it to go high. Only one VT1432A is required to pull the Sync/Trigger line low. In a system with only one VT1432A, the Sync/Trigger line is local to the module and not is routed to a TTL TRIGGER line on the VXI back plane.

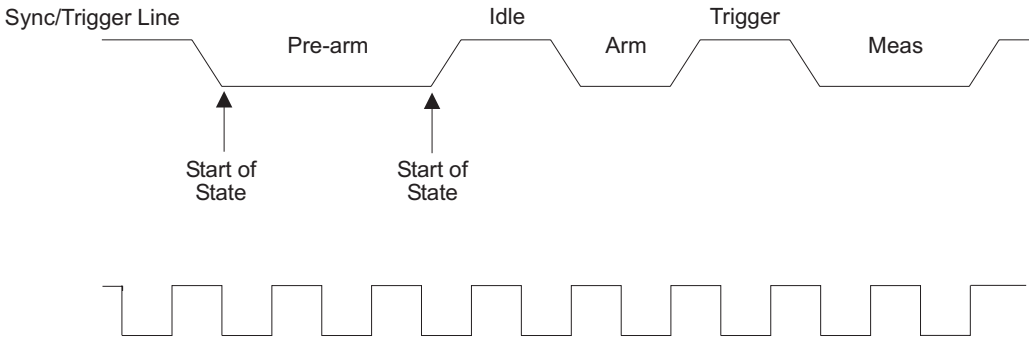


Figure 3-13: Transitions between states

Parameter Settings

Many parameters are channel-dependent, meaning that each channel can be set independently of the others in the module. Other parameters are module-dependent; changing a module-dependent parameter for a channel will change it for all channels on that module. For example, changing blocksize, a module-dependent parameter, for input Channel 3 will also change the block size for all other channels in the same VT1432A module as Channel 3.

When possible, parameters are written to the hardware as soon as they are received. Sometimes, the parameter can't be written to the hardware until the start of a measurement; in this case the value of the parameter is saved in RAM in the VT1432A module until the measurement is started with hpe1432_initMeasure. Some parameters can be changed while a measurement is running, but many do not take effect until the next start of a measurement.

Measurement Initiation

This section describes the measurement initiation process in the VT1432A.

The measurement initialization states and the corresponding Sync/Trigger line transitions (with 'H' for high, 'L' for Low) are:

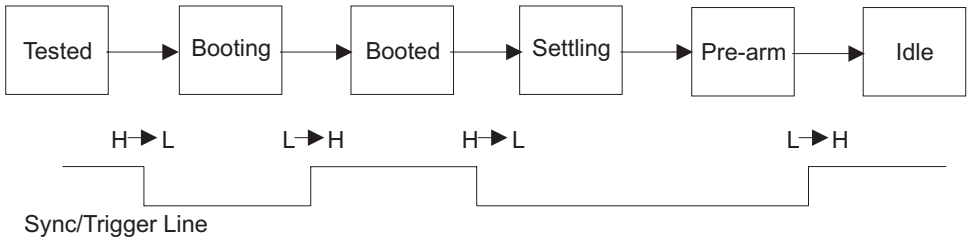


Figure 3-14: Measurement initialization

The module enters the TESTED state after a reset. In this state, all of the module parameters may be set. The VT1432A stays in the TESTED state until it sees a high-to-low transition of the Sync/Trigger line.

In the BOOTING state, the digital processors of the module load their parameters and their program. Once done, the module releases the Sync/Trigger line and moves to the BOOTED state. The VT1432A stays in the BOOTED state until it sees a high-to-low transition of the Sync/Trigger line (that is, all the VT1432As in the system have booted).

In the SETTLING state, the digital filters are synchronized and the digital filter output is 'settled' (it waits N samples before outputting any data). Once the module is settled, it advances to the PRE_ARM state.

In the PRE_ARM state, the module waits for a pre-arm condition to take place. The default is to auto-arm, so the module would not wait at all in this case. When the pre-arm condition is met, the module releases the Sync/Trigger line and advances to the IDLE state.

This complete measurement sequence initialization, from TESTED through BOOTING, BOOTED, SETTLING, PRE-ARM and IDLE, can be performed with a call to the function hpe1432_initMeasure.

Measurement Loop

This section describes the measurement loop in the VT1432A.

The progression of measurement states and the corresponding Sync/Trigger line transitions are:

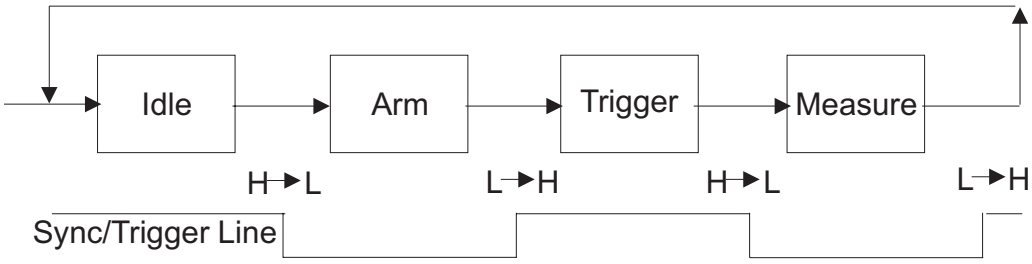


Figure 3-15: Measurement loop

In the IDLE state the VT1432A writes no data into the FIFO. The VT1432A remains in the IDLE state until it sees a high-to-low transition of the Sync/Trigger line or an RPM arm/trigger point is calculated. If any of the VT1432As in the system is programmed for auto arming (with hpe1432_setArmMode), the Sync/Trigger line is immediately pulled low by that VT1432A. The VT1432A may also be moved to the ARM state by an explicit call to the function hpe1432_armMeasure.

Upon entering the ARM state the VT1432A starts saving new data in its FIFO. It remains in the ARM state until the Sync/Trigger line goes high. If the VT1432A is programmed with a pre-trigger delay, it collects enough data samples to satisfy this pre-trigger delay and then releases the Sync/Trigger line. If no pre-trigger delay has been programmed, it releases the Sync/Trigger line immediately. When all modules in a system have released the Sync/Trigger line (allowing it to go high), a transition to the TRIGGER state occurs.

Upon entering the TRIGGER state the VT1432A continues to collect data into the FIFO, discarding any data prior to the pre-trigger delay. The VT1432A remains in the TRIGGER state until it sees a high-to-low transition of the Sync/Trigger line. The Sync/Trigger line is pulled low by any VT1432A which encounters a trigger condition and is programmed to pull the Sync/Trigger line. If any VT1432A is programmed for auto triggering (with hpe1432_setAutoTrigger), the Sync/Trigger line is pulled low immediately. The Sync/Trigger line may also be pulled low by an explicit call to the function hpe1432_triggerMeasure.

Upon entering the MEASURE state the VT1432A continues to collect data. The VT1432A also presents the first data from the FIFO to the selected output port, making it available to the controller to read. The VT1432A holds the Sync/Trigger line low as long as it is actively collecting data. In overlap block mode the VT1432A stops taking data as soon as a block of data has been collected, including any programmed pre- or post-trigger delays. (It starts again when another trigger occurs). In continuous mode, the VT1432A stops taking data only when the FIFO overflows. When data collection stops, the VT1432A releases the Sync/Trigger line. When all VT1432As are finished and the Sync/Trigger line goes high, the VT1432A goes into the IDLE state again.

The measurement initialization and loop may be interrupted at any time with a call to hpe1432_resetMeasure, which puts the module in the TESTED state.

Register-based VXI Devices

The VT1432A is a register-based VXI device. Unlike message-based devices which use higher-level programming using ASCII characters, register-based devices are programmed at a very low level using binary information. The greatest advantage of this is speed. Register-based devices communicate at the level of direct hardware manipulation and this can lead to much greater system throughput.

Users do not need to access the registers in order to use the VT1432A. The VT1432A’s functions can be more easily accessed using the VT1432A Host Interface Library software. However, more information about the registers are provided in Appendix A: Register Definitions.

Arm and Trigger

This section explains some terminology relating the the “Arm” and “Trigger” steps in the measurement loop. As an example a measurement might be set up to arm at a certain RPM level and then subsequently trigger at an external event corresponding to top dead center (TDC). The settings would be:

- Arm: RPM Step Arm
- Trigger: External Trigger

To begin a throughput session at this same RPM/TDC event, the first external trigger after a specified RPM should start a continuous mode measurement. Now (using overlap block mode) the settings would be:

- Pre-Arm: RPM Step Arm
- Arm: Auto
- Trigger: Auto

In the measurement loop, an arm must take place before a trigger. The number of triggers that occur before waiting for another arm condition can be defined. The default is one trigger for each arm. For each trigger, a block of data is sent to the host.

The first arm in a measurement is the pre-arm. By default, the pre-arm condition is the same as the regular arm conditions.

Valid Arm (and Pre-Arm) conditions are:

- Auto Arm
- Manual Arm
- RPM Step Arm

Valid trigger conditions are:

- Auto Trigger
- Input Trigger
- Source Trigger
- External Trigger
- Manual Trigger
- Tachometer Edge Trigger

VT1432A Triggering

The following is a short discussion of triggering for the VT1432A.

Triggering is defined as the transition from the armed state to the triggered state. This transition is caused by a low going edge on a TTL trigger line. The function hpe1432_getTtltrgLines selects which of the eight TTL trigger lines is to be used.

The low-going transition of the TTL trig line can be caused by any of the following items:

trigger type	enabling function
the AUTO TRIGGER circuitry	hpe1432_setAutoTrigger
the hpe1432_triggerMeasure function	hpe1432_triggerMeasure
a source trigger	hpe1432_setTriggerChannel
a tach trigger	hpe1432_setTriggerChannel
an external trigger	hpe1432_setTriggerExt
an input level or bound trigger event	hpe1432_setTriggerChannel and hpe1432_setTriggerMode

Each of these trigger sources can be enabled or disabled independently, so quite complex trigger setups are possible. In all cases, however, the first trigger event kicks off the measurement and the following trigger events become superfluous.

Note that for hpe1432_setAutoTrigger the setting HPE1432_MANUAL_TRIGGER really means “do not auto trigger” not “expect a manual trigger”.

For single-VT1432A systems, the TTL trigger signal is not connected to the VXI backplane. For multiple VT1432A systems, the hpe1432_initMeasure function connects the VT1432A trigger lines to the VXI backplane and at that point, the selection of which TTL trigger lines through hpe1432_getTtltrgLines is relevant. Multiple mainframe systems will need to account for the unidirectional nature of the inter-mainframe MXI extenders which will prevent all but the “upstream” mainframe from triggering the system.

Trigger Level

To set the trigger level, use hpe1432_setTriggerMode to select “level” or “bound” mode; and use hpe1432_setTriggerLevel twice to set both the upper and lower trigger levels. The difference between the upper and lower trigger levels must be at least 10% of full scale (and 10% is usually the best amount).

Also use hpe1432setTriggerSlope to specify a positive or negative trigger slope.

Level mode

If the mode is set to “level” and the trigger slope is positive, then the module triggers when the signal crosses both the upper and lower trigger levels in the positive direction. If the trigger slope is negative, the module triggers when the signal crosses both levels in the negative direction. Setting two trigger levels prevents the module from triggering repeatedly when a noisy signal crosses the trigger level.

Bound mode

If the mode is set to “bound” and the trigger slope is positive, then the module triggers when the signal exits the zone between the upper and lower trigger levels in either direction. If the trigger slope is negative, the module triggers when the signal enters the zone between the upper and lower trigger levels.

Data Transfer Modes

The VT1432A can be programmed to use either of two data transfer modes: overlap block mode and continuous mode. Block mode will be described first.

Block Mode (Agilent/HP E1431A)

The VT1432A's overlap block mode is similar the block mode which is used in some Agilent instruments such as the Agilent/HP E1431A. In block mode, the input hardware acquires one block after getting an arm and trigger. It does not allow the system to trigger until it is ready to process the trigger and it acquires pre-trigger data if necessary. The hardware does not accept a new arm and trigger until the acquired block is sent to the host. There is no provision for overlap or queuing up more than one block when in block mode. There is also no way for a FIFO overflow to occur.

The VT1432A's overlap block mode can be configured to act exactly like traditional block mode. It also has additional capabilities as described below.

Continuous Mode.

Both the VT1432A and the Agilent/HP E1431A use continuous mode. In this mode, the input hardware waits for an arm and trigger and then starts acquiring data continuously. If the host is slow, several blocks can be queued up in the input hardware. If the host gets far enough behind, a FIFO overflow occurs and the input stops acquiring data.

The VT1432A's overlap block mode can be configured to act similarly to continuous mode, but not identically. The VT1432A can also use the traditional continuous mode.

Overlap Block Mode

Overlap block mode combines features of both block mode and continuous mode. The main difference between overlap block mode and traditional block mode is that overlap block mode allows additional arms and triggers to occur before an already-acquired block is sent to the host. A trigger can occur before the end of the previous block, so overlapping blocks are possible (hence the name "overlap block mode"). As in continuous mode, there is an overlap parameter which controls how much overlap is allowed between consecutive blocks.

Limit on Queuing of Data

In overlap block mode, a number of trigger events may be queued up before the host reads the data for those triggers. The host may get further and further behind the data acquisition.

However, if the host gets far enough behind that the FIFO fills up, data acquisition must momentarily stop and wait for data to get transferred to the host. This places a limit on how far in time the host can be behind the data acquisition. By setting the size of the FIFO, it is possible control how far behind the host can get.

Making Overlap Block Mode Act Like Traditional Block Mode

If the FIFO size is set the same as the block size or if the number of pending triggers is limited to zero, then overlap block mode becomes identical to traditional block mode.

Making Overlap Block Act Like Continuous Mode

If the module is in auto-arm and auto-trigger mode, then overlap block mode becomes nearly the same as continuous mode.

One difference is that traditional continuous mode has a single arm and trigger, while overlap block mode may have multiple arms and triggers. Another is that continuous mode can be configured to start at any type of trigger event, while overlap block mode must be in auto-trigger mode to act like continuous mode. Finally, continuous mode always stops when a FIFO overflow occurs, but overlap block mode does not.

VT1432A Interrupt Behavior

Interrupt Setup

For an example of interrupt handling see the program event.c in the examples directory.

The VT1432A VXI module can be programmed to interrupt a host computer using the VME interrupt lines. VME provides seven such lines. Using hpe1432_setInterruptPriority, the VT1432A can be set up to use any one of them.

The VT1432A can interrupt the host computer in response to different events. Use hpe1432_setInterruptMask to specify a mask of events on which to interrupt. This mask is created by OR'ing together the various conditions for an interrupt. The following table shows the conditions that can cause an interrupt:

Interrupt Mask Bit Definitions

Define (in e1432.h)	Description
HPE1432_IRQ_BLOCK_READY	Scan of data ready in FIFO
HPE1432_IRQ_MEAS_ERROR	FIFO overflow
HPE1432_IRQ_MEAS_STATE_CHANGE	Measurement state machine changed state
HPE1432_IRQ_MEAS_WARNING	Measurement warning
HPE1432_IRQ_OVERLOAD_CHANGE	Overload status changed
HPE1432_IRQ_SRC_STATUS	Source channel interrupt
HPE1432_IRQ_TACHS_AVAIL	Raw tach times ready for transfer to other modules
HPE1432_IRQ_TRIGGER	Trigger ready for transfer to other modules

VT1432A Interrupt Handling

To make the VT1432A module do the interrupt, both a mask and a VME Interrupt line must be specified, by calling hpe1432_setInterruptMask and hpe1432_setInterruptPriority respectively. Once the mask and line have been set and an interrupt occurs, the cause of the interrupt can be obtained by reading the HPE1432_IRQ_STATUS_REG register (using hpe1432_getInterruptReason). The bit positions of the interrupt mask and status registers match so the defines can be used to set and check IRQ bits.

Once it has done this interrupt, the module will not do any more VME interrupts until re-enabled with hpe1432_reenableInterrupt. Normally, the last thing a host computer’s interrupt handler should do is call hpe1432_reenableInterrupt.

Events that would have caused an interrupt, but which are blocked because hpe1432_reenableInterrupt has not yet been called, will be saved. After hpe1432_reenableInterrupt is called, these saved events will cause an interrupt, so that there is no way for the host to "miss" an interrupt. However, the module will only do one VME interrupt for all of the saved events, so that the host computer will not get flooded with too many interrupts.

For things like "HPE1432_IRQ_BLOCK_READY", which are not events but are actually states, the module will do an interrupt after hpe1432_reenableInterrupt only if the state is still present. This allows the host computer's interrupt handler to potentially read multiple scans from a VT1432A module and not get flooded with block ready interrupts after the fact.

Host Interrupt Setup

This is a summary of how to set up a VT1432A interrupt:

- Look at the Resource Manager to find out which VME interrupt lines are available.
- Tell the VT1432A module to use the a VME interrupt line found in step one, using hpe1432_setInterruptPriority.
- Set up an interrupt handler routine, using hpe1432_callBackInstall. The interrupt handler routine will get called when the interrupt occurs.
- Set up interrupt mask in the VT1432A module, using hpe1432_setInterruptMask.

Host Interrupt Handling

When the VT1432A asserts the VME interrupt line, the program will cause the specified interrupt handler to get called. Typically the interrupt handler routine will read data from the module and then re-enable VT1432A interrupts with hpe1432_reenableInterrupt. The call to hpe1432_reenableInterrupt must be done unless the host is not interested in any more interrupts.

Inside the interrupt handler, almost any VT1432A Host Interface library function can be called. This works because the Host Interface library disables interrupts around critical sections of code, ensuring that communication with the VT1432A module stays consistent. Things that are not valid in the handler are:

- Calling hpe1432_createChannelGroup to delete a group that is simultaneously being used by non-interrupt-handler code.
- Calling one of the read data functions (hpe1432_readRawData or hpe1432_readFloat64Data), if the non-interrupt-handler code is also calling one of these functions.
- Calling hpe1432_init to reset the list of channels that are available to the VT1432A library.

As is always the case with interrupt handlers, it is easy to introduce bugs into the program and generally difficult to track them down. Be careful when writing this function.

Data Gating

Sometimes it is desirable to monitor data from some input channels and not others. The function hpe1432_setEnable enables or disables data from an input channel (or group of channels). If data is enabled, then the data can be read using hpe1432_blockAvailable and hpe1432_readRawData or hpe1432_readFloat64Data. If data is disabled, data from the specified channel is not made available to the host computer.

This parameter can be changed while a measurement is running, to allow the host computer to look at only some of the data being collected by the VT1432A module. While data from a channel is disabled the input module continues to collect data but it is not made available to the host computer. The host can then switch from looking at some channels to looking at others during the measurement. In contrast, the function hpe1432_setActive completely enables or disables a channel and can’t be changed while a measurement is running.

For order tracking measurements this function can be used to switch between receiving order tracking data ordinary time data or both.

VT1432A Parameters

Some parameters, such as range or coupling, apply to specific channels. When a channel ID is given to a function that sets a channel-specific parameter, only that channel is set to the new value.

Some parameters, such as clock frequency or data transfer mode, apply globally to a module. When a channel ID is used to change a parameter that applies to a whole module, the channel ID is used to determine which module. The parameter is then changed for that module.

Starting and stopping a measurement is somewhat like setting a global parameter. Starting a measurement starts each active channel in each module that has a channel in the group.

After firmware is installed and after a call to hpe1432_preset, all of the parameters (both channel-specific and global) in a VT1432A module are set to their default values. For channel-specific parameters, the default value may depend on the type of channel. Some channel-specific parameters apply only to a specific type of channel. For example, tach holdoff applies only to tach channels. Setting such a parameter for a channel that doesn't make sense will result in an error.

At the start of a measurement, the VT1432A firmware sets up all hardware parameters and ensures that the input hardware is settled before starting to take data. The firmware also ensures that any digital filters have time to settle. This ensures that all data read from the module will be valid.

However, after a measurement starts, VT1432A parameters can still be changed. The effect of this change varies, depending on the parameter. For some parameters, changing the value aborts the measurement immediately. For other parameters, the measurement is not aborted, but the changed parameter value is saved and not used until a new measurement is started. For still other parameters, the parameter change takes place immediately and the data coming from the module may contain glitches or other effects from changing the parameter. For more information, please see the "Programming Information" chapter of the VT1432A VXI*plug&play* Library online help.

The module cannot be told to wait for settling when changing a parameter in the middle of a measurement. The only way to wait for settling is to stop and re-start the measurement. Also, the settling that takes place at the start of a measurement cannot be disabled.

Refer to the (on-line) VT1432A Function Reference for the parameters needed for each function. (See "Where to get more information" in this chapter.)

New features of the VT1432A/VT1433B software

The following features have been added to the VT1432A/33B software since the previous edition of this manual. These and other features are documented in the online Function Reference. For more information look in the Function Reference entries for the functions that are used by the feature.

Auto range

Auto range calculates the best range for each channel so that the signal is full scale but not overloaded. Auto range works only while the measurement is running.

Averaging

Averaging can be done for resampling measurements on frequency or order data. It uses the function hpe1432_setAvgMode. There are several averaging modes: RMS averaging, linear averaging, exponential averaging or peak hold averaging.

Continuous re-sampled data

Continuous re-sampling forces the blocks of data to be contiguous, with no gaps between them. It uses the existing function hpe1432_setArmMode. (Without continuous re-sampling, each block of data follows the previous block after some interval, depending on the next trigger event.

Fast span or range change

The span or range can now be changed while the measurement is running, using the existing functions hpe1432_setSpan or hpe1432_setRange. Previously, if these commands were sent while a measurement was running, it would wait until the next measurement. Now it will change the span or range when the command is sent.

Time arming

This uses a new function hpe1432_setArmTimeInterval. It allows for an arming time interval to be specified. For example, it could be set to get a block of data every second.

Weighting filters (VT1433B only)

For the VT1433B, any of three weighting filters (A-weighting, B-weighting or C-weighting) can be set. This feature uses the function hpe1432_setWeighting.

Zoom (VT1432A only)

Up to now the VT1432A has made only baseband measurements (from zero to some frequency.) Zoom allows for a center frequency to be set around which a window of frequencies can be viewed. It uses two new functions: hpe1432_setZoom (turns zoom on/off) and hpe1432_setCenterFreq. (Zoom has not been implemented for the VT1433B).

Zoom (for the Arbitrary Source, option VT1432A-1D4)

This is similar to zoom for the VT1432A input. Zoom for the source allows for a center frequency and a span to be set for the output signal. It uses the existing function hpe1432_setSourceMode with a new zoom parameter.

Zoom applies to random burst source mode and continuous source mode, for both the VT1432A and the VT1433B. When used with the VT1432A, if the source center frequency is set to zero, the source center frequency will be the same as the center frequency set for the VT1432A input. This is also true for the span.

Where to get more information

There is more information available about the VT1432A. This section shows how to access it and print it, if desired.

The Function Reference for VXI*plug&play*

On a PC: the VT1432A Function Reference is in Microsoft Help text. Select the Help icon in the "VXIPNP" folder. Refer to Microsoft Windows documentation (including Help text) for information on using and printing Help.

On a UNIX system: look at the README file at /opt/hyperhelp. It includes instructions on how to install and use the VXI*plug&play* help.

The Function Reference for the Host Interface Library (C-language version)

The VT1432A distribution includes manual pages for the VT1432A Host Interface library. These manual pages can be examined on-line, using the"ptman" command that is shipped in "/opt/e1432/bin." For example, the manual page for the "e1432_init_measure" function can read by typing:

ptman e1432_init_measure

The distribution also includes a nicely formatted set of these manual pages, that can be printed on any postscript printer. This manual in postscript form is in file "/opt/e1432/man/man.ps." Typically, this manual can be printed by typing:

lp -opostscript /opt/e1432/man/man.ps

Alternatively, if there is no postscript printer available, a plain text version of the manual is in file "/opt/e1432/man/man.txt." This can be printed on any line printer.

Users of the C-language library will also find useful information about theVT1432A in the VT1432A help text (see above).

4

The Host Interface Library

Introduction

The Host Interface Library for the VT1432A is a set of functions that allows the register-based VT1432A to be programmed at a higher level than register reads and writes. The library allows groups of VT1432As to be set up and programmed as if they were one entity.

Two version of the Host Interface Library are available. One is the HP-UX C-Language Host Interface Library which uses SICL (the Standard Instrument Interface Library) to communicate to the VT1432A hardware. The other is the VXI*plug&play* Library which communicates with the hardware using the VXI*plug&play* standard. This chapter covers the SICL version. If using the VXI*plug&play* version, this chapter will not need. Instead, see the chapters titled "Getting Started With the VT1432A" and "Using the VT1432A."

The library includes routines to set up and query parameters, start and stop measurements, read and write data, and control interrupts. Routines to aid debugging and perform low-level I/O are also included.

For information on diagnostics see the chapter titled "Troubleshooting the VT1432A."

Almost all functions in this library return 0 if they complete successfully and a negative error number if there is a problem. The return value of the function should always be checked and appropriate action taken for non-zero values. See the on-line man pages for more information on error messages.

Header and Library Files

The /opt/e1432/lib directory contains several versions of the VT1432A Host Interface library:

lib1432.a	A normal HP-UX archive library, used by host programs wanting to talk to VT1432A hardware.
lib1432.sl	An HP-UX shared library, used by host programs wanting to talk to VT1432A hardware. This and the above archive library do exactly the same things. Usually, host programs would use the shared library, because it makes the host program smaller.
llib-l1432.ln	A lint library for the VT1432A C-Language Host Interface Library. If lint (a UNIX tool for checking source codes for problems) is not used, this file is superfluous.

An application using the VT1432A C-Language Host Interface Library must link in one of these libraries, typically lib1432.sl. The HP-UX versions of the VT1432A library use SICL to communicate with the VT1432A hardware, so an application using the VT1432A library must also link in the SICL library. Normally, this is found in /usr/lib/libsicl.sl.

Any application source code which uses any of the VT1432A C-Language Host Interface Library functions must include the e1432.h include file, found in /opt/e1432/include. Internally, this file includes machType.h, which is also found in /opt/e1432/include. If the application refers to specific VT1432A error numbers, it must also include err1432.h.

Parameter Information

Description of VT1432A Parameters

Some parameters, such as range or coupling, apply to specific channels. When a channel ID is given to a function that sets a channel-specific parameter, only that channel is set to the new value.

Some parameters, such as clock frequency or data transfer mode, apply globally to a module. When a channel ID is used to change a parameter that applies to a whole module, the channel ID is used to determine which module. The parameter is then changed for that module.

Starting and stopping a measurement is somewhat like setting a global parameter. Starting a measurement starts each active channel in each module that has a channel in the group.

After firmware is installed, and after a call to e1432_preset, all of the parameters (both channel-specific and global) in a VT1432A module are set to their default values. For channel-specific parameters, the default value may depend on the type of channel. Some channel-specific parameters apply only to a specific type of channel. For example, tach holdoff applies only to tach channels. Setting such a parameter for a channel that doesn't make sense will result in an error.

At the start of a measurement, the VT1432A firmware sets up all hardware parameters, and ensures that the input hardware is settled before starting to take data. The firmware also ensures that any digital filters have time to settle. This ensures that all data read from the module will be valid.

However, after a measurement starts, VT1432A parameters can still be changed. The effect of this change varies, depending on the parameter. For some parameters, changing the value aborts the measurement immediately. For other parameters, the measurement is not aborted, but the changed parameter value is saved and not used until a new measurement is started. For still other parameters, the parameter change takes place immediately, and the data coming from the module may contain glitches or other effects from changing the parameter.

The module cannot be told to wait for settling when changing a parameter in the middle of a measurement. The only way to wait for settling is to stop and re-start the measurement. Also, the settling that takes place at the start of a measurement cannot disabled.

Parameter Lists

This section shows which parameters are global parameters, which are channel-specific, and what types of channels the channel-specific parameters apply to. Default values are shown for all of these parameters. In addition, each parameter is categorized as “abort”, “wait”, “immediate”, or “glitch” depending on the behavior when this parameter is changed during a running measurement. Those with “abort” cause the measurement to abort. Those with “wait” don’t take effect until the start of the next measurement. Those with “immediate” take effect immediately. Those with “glitch” take effect immediately, and may cause glitches in the data that is read back, or on the source output if the parameter is applied to a source channel.

Global Parameters

Parameter	Default Value	Changes
append_status	Off	Immediate
arm_channel	None	Immediate
arm_mode	Auto Arm	Immediate
arm_time_interval	1 second	Immediate
auto_group_meas	On	Wait
avg_mode	None	Wait
avg_number	10	Wait
auto_trigger	Auto Trigger	Abort
avg_update	10	Wait
avg_weight	1	Immediate
blocksize	1024	Abort
cal_dac	0	Immediate
cal_voltage	0 volts	Immediate
calin	Grounded	Immediate
center_freq	2 kHz	Immediate

Parameter	Default Value	Changes
clock_freq	51.2 kHz	Abort
clock_master	Off	Abort
clock_source	Internal	Abort
data_mode	Block Mode	Abort
data_port	VME	Abort
data_size	16-Bit Integer	Abort
decimation_output	Single Pass	Wait
decimation_oversample	Off	Wait
decimation_undersamp	1	Wait
delta_order	0.1	Wait
fifo_size	0 (Use All DRAM)	Wait
filter_settling_time	64 samples	Wait
internal_debug	0x100	Immediate
interrupt_mask	0	Immediate
interrupt_priority	None	Immediate
lbus_mode	Pipe	Immediate
lbus_reset	Off (Not Reset)	Immediate
max_order	10	Wait
meas_time_lengh	0 (run forever)	Immediate
mmf_delay	0	Immediate
multi_sync	Off	Abort
overlap	0	Wait
pre_arm_mode	Auto Arm	Immediate
ramp	Off	Immediate

Parameter	Default Value	Changes
span	20000 Hz	Wait
sumbus	Off	Immediate
trigger_delay	0	Wait
trigger_ext	Off	Immediate
trigger_master	Off	Immediate
triggers_per_arm	1	Immediate
ttltrg_clock	TTLTRG1	Abort
ttltrg_gclock	TTLTRG1	Abort
ttltrg_satrg	TTLTRG0	Abort
ttltrg_trigger	TTLTRG0	Abort
window	Uniform	Glitch
xfer_size	0 (Use Blocksize)	Wait
zoom	Off	Wait

51.2 kHz 4-channel Input Parameters

Parameter	Default Value	Changes
active	Off	Abort
anti_alias_digital(*)	On	Abort
auto_range_mode	Up/Down	Immediate
calc_data	Time	Wait
coupling	DC	Glitch
enable	On	Immediate
filter_freq	200 kHz	Immediate
input_high	Normal	Glitch
input_low	Floating	Glitch
input_mode(*)	Volt	Glitch
range	10 volts	Glitch
range_charge	50,000 picocoulombs	Glitch
range_mike	10 volts	Glitch
trigger_channel	Off	Immediate
trigger_level_lower	-10%	Immediate
trigger_level_upper	0%	Immediate
trigger_mode	Level	Immediate
trigger_slope	Positive	Immediate

(*) Input mode is listed as channel-specific, but it actually applies to all channels within an SCA (such as a 4-channel input assembly).

Option 1D4 Single-channel Source Parameters

Parameter	Default Value	Changes
active	Off	Abort
amp_scale	1.0	Immediate
anti_alias_digital	On	Wait
duty_cycle	0.5	Immediate
filter_freq	25.6 kHz	Wait
ramp_rate	1 second	Wait
range	0.041567 volts	Immediate
sine_freq	1000 Hz	Immediate
sine_phase	0 degrees	Immediate
source_blocksize	0 (Use Input Blocksize)	Wait
source_centerfreq	0 Hz	Wait
source_cola	Off	Wait
source_mode	Sine	Abort
source_output	Normal	Abort
source_seed	3	Wait
source_span	0 (Use Input Span)	Wait
source_sum	Off	Wait
srcbuffer_init	Empty	Wait
srcbuffer_mode	Periodic_A	Wait
srcbuffer_size	1024	Wait
srcparm_mode	Immediate	Immediate
trigger_channel	Off	Wait

Option AYF Tachometer Parameters

Parameter	Default Value	Changes
active	Off	Abort
input_high	Normal	Immediate
pre_arm_rpm	600 RPM	Immediate
rpm_high	6000 RPM	Immediate
rpm_interval	25 RPM	Immediate
rpm_low	600 RPM	Immediate
rpm_smoothing	0	Immediate
tach_decimate	0	Immediate
tach_holdoff	10 microseconds	Immediate
tach_max_time	30 seconds	Immediate
tach_ppr	1	Immediate
trigger_channel	Off	Wait
trigger_level_lower	-0.05 volts	Immediate
trigger_level_upper	0 volts	Immediate
trigger_slope	Positive	Immediate

Channel and Group IDs

Most functions in the VT1432A C-Language Host Interface Library take an ID parameter which specifies what channel or group of channels the function should apply to. The ID can either be a channel ID or a group ID. If a group ID is used, then the function is applied to each channel contained in the group.

Channel IDs

Channel IDs are logical IDs which are created by a call to e1432_assign_channel_numbers. The e1432_assign_channel_numbers function must be called exactly once, following the call to e1432_init_io_driver, in order to declare to the library the logical addresses of the VT1432A modules that will be used.

This function checks the existence of a VT1432A module at each of the logical addresses given in a list of logical addresses, and allocates logical channel identifiers for each channel in all of the VT1432As. Input channels, source channels, and tach/trigger channels are kept logically separated. Channel numbers for each type of channel are numbered starting from one, so there will be input channels 1 through M, source channels 1 through N, and tach/trigger channels 1 through P, where M is the number of input channels, N is the number of source channels, and P is the number of tach/trigger channels.

As an example, suppose two logical addresses 100 and 101 are passed to the function, and the logical address 100 has two 4-channel input SCAs and a 2-channel tach/trigger board, while logical address 101 has three 4-channel input SCAs and a 1-channel source board. In this case, input channel IDs 1 through 8 are assigned to the eight input channels at logical address 100, while input channel IDs 9 through 20 are assigned to the twelve input channels at logical address 101. Tach/trigger channel IDs number 1 and 2 are assigned to the two tach/trigger channels at logical address 100, and Source channel ID number 1 is assigned to the source channel at logical address 101.

To use the ID of an input channel, the input channel number is given as an argument to the E1432_INPUT_CHAN() macro. (For backwards compatibility with the Agilent E1431A, the macro does nothing.) To use the ID of a source channel, the source channel number is given as an argument to the E1432_SOURCE_CHAN() macro. To use the ID of a tach/trigger channel, the tach/trigger channel number is given as an argument to the E1432_TACH_CHAN() macro. A channel ID is always positive.

For example, to set the range of the third input channel to 10 volts, the source code would look something like:

```
status = e1432_set_range(hwid, E1432_INPUT_CHAN(3), 10.0);
```

Group IDs

Group IDs are logical IDs which are created by a call to e1432_create_channel_group. This function can be called multiple times to create multiple groups, and each group can contain any combination of channels, including mixtures of different types of channels. The channel groups can overlap as well.

This function creates and initializes a channel group. A channel group allows commands to be issued to several VT1432A channels at once, simplifying system setup. The state of an individual VT1432A channel that is in more than one channel group, is determined by the most recent operation performed on any group to which this channel belongs.

If successful, this function returns the ID of the group that was created, which is then used to reference the channel group in most other functions in this library. A group ID is always negative.

As a side effect, this function makes all input channels in the channel group active, and all source and tach channels in the channel group inactive. Unlike the Agilent/HP E1431A library, this function does not inactivate other channels within the modules that the channels are in. Also unlike the Agilent/HP E1431A library, this function does not preset the channels in the new group.

As an example, to create a group consisting of the first three input channels and the eighth and ninth input channels, the code would like something like this:

```
SHORTSIZ16 chan_list[5];
SHORTSIZ16 input_group;
chan_list[0] = E1432_INPUT_CHAN(1);
chan_list[1] = E1432_INPUT_CHAN(2);
chan_list[2] = E1432_INPUT_CHAN(3);
chan_list[3] = E1432_INPUT_CHAN(8);
chan_list[4] = E1432_INPUT_CHAN(9);
input_group = e1432_create_channel_group(hw, 5, chan_list);
```

To create a group consisting of the first two source channels, the code would look something like this:

```
SHORTSIZ16 chan_list[2];
SHORTSIZ16 source_group;
chan_list[0] = E1432_SOURCE_CHAN(1);
chan_list[1] = E1432_SOURCE_CHAN(2);
source_group = e1432_create_channel_group(hw, 2, chan_list);
```

Multiple-module/Mainframe Measurements

Grouping of Channels/Modules

The interface library for the VT1432A is designed to allow programming of several channels from one or several distinct modules, as if they were one entity. Each VT1432A module has up to 16 channels. The library may control up to a maximum of 255 VT1432A modules (8160 channels).

When initializing the interface library, all module logical addresses are passed in the call to e1432_assign_channel_numbers. This function associates a logical channel ID with each channel. From then on, library functions use these logical IDs rather than the logical address.

The function e1432_create_channel_group can be used to declare any number of groups of channels, possibly overlapping. Each group can be uniquely identified by a group ID.

The 'target' of a library function is either a channel, a group, or (rarely) a module, depending on the nature of the call. When the same library function may be called with either a channel or a group identifier, it's 'target' is shown by a parameter named ID.

Multiple-module Measurements

A channel group that spans more than one module will need to be configured to use the TTL trigger lines on the VXI Bus for inter-module communications. This configuration automatically performed in the e1432_init_measure call unless defeated using e1432_set_auto_group_meas.

The following discussion outlines what e1432_init_measure does automatically. This must be done by the user if e1432_set_auto_group_meas has been used to defeat auto configuration.

There are eight VXI TTL trigger lines that can be used for multi-module synchronization. Often, these lines are used in pairs, one for sample clock and one for Sync/Trigger. The e1432_set_ttltrg_lines function selects which TTL trigger lines to use; this function always uses the TTL trigger lines in pairs. Calling e1432_set_clock_source with the group ID will set all modules to the same pair.

All modules need to be set to use the shared sync line rather than the default setting of internal sync. This can be done with the e1432_set_multi_sync function, using the group ID.

One module of the set of modules needs to be set to output the sync pulse. The module with the lowest VXI logical address is called the “system module” and assigned this duty. This can be set with the e1432_set_multi_sync function call, using the lowest channel ID in the group (NOT the group ID).

All modules except the “system module” need to be set to use the VXI TTL trigger lines as the clock source. Use e1432_set_clock_source for this.

Set the “system module” to output the clock. Use e1432_set_clock_master for this. After this is done, all system sync pulses come from the “system module” and drive the measurement state machines on all boards in the group.

Possible Trigger Line Conflict

The following describes a scenario where VT1432A modules might conflict and prevent a proper measurement. The conditions allowing the conflict are complex but must be understood by the user.

After a measurement has completed, the modules are left set up. If a module (call it module ‘A’) is driving the TTL trigger lines and a different group is started which also drives the TTL trigger lines (and that different group does not include module ‘A’), then module ‘A’ will conflict and prevent the other group from functioning. In this case make a call to e1432_finish_measure (using the old group ID which includes ‘A’) to turn off module ‘A’ and allow the new group to function.

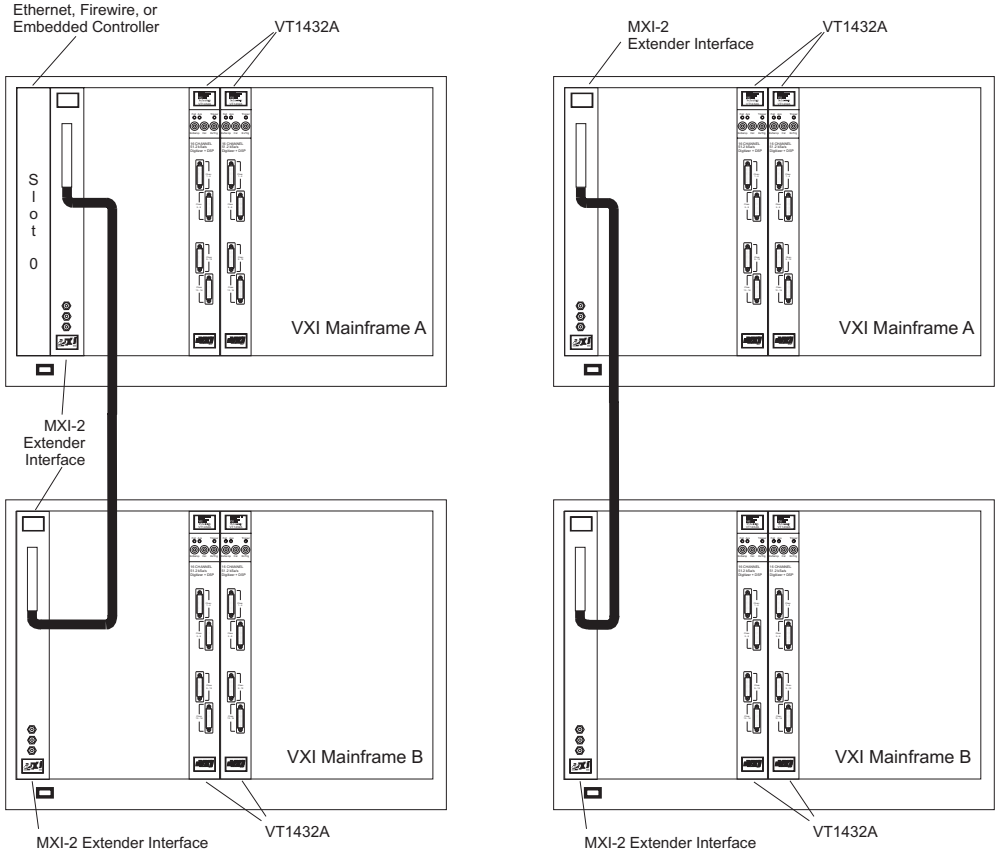
Note that if the new group includes all modules of the old group, the conflict will not occur since e1432_init_measure will reset all modules as needed. Also note that single module groups do not drive the TTL trigger lines, so single modules groups are immune from causing or receiving this conflict.

Managing Multiple-mainframe Measurements

In a single mainframe measurement, the VT1432A communicates with other VT1432As through the TTLTRG lines. However, when using the VXI-MXI bus extender modules, the TTLTRG lines, which carry the group synchronization pulse and sample clock, are extended only in one direction. This unidirectional signal connection restricts the types of measurements that can be made in a multiple mainframe environment.

The following types of multiple mainframe measurements cannot be performed:

- ❑ Unequal pre-trigger delay settings between mainframes
- ❑ Channel triggering by channels in Mainframe B
- ❑ Lower spans or longer blocksizes in Mainframe B
- ❑ Different digital filter settling times between VT1432A modules



Example 1: Slot 0 Controller **Example 2: MXI-2 Daisy Chain**

Figure 4-1: Multiple mainframes - two mainframes

In the example above, Mainframe A contains the Slot 0 Controller for a multiple mainframe system. Mainframe A is connected to Mainframe B with a VXI-MXI interface, Agilent/HP E1482B. To successfully manage this multiple mainframe environment, use the following guidelines.

- Locate modules with logical addresses less than 128 in Mainframe A.
- Locate modules with logical addresses greater than 127 in Mainframe B.
- Locate the highest-numbered channels in Mainframe A.
- Locate the last module in the module list specified in the call to e1432_assign_channels() in Mainframe A.
- Locate the module that generates the group synchronization pulse in Mainframe A.
- Locate the channels performing channel triggering in Mainframe A.
- Locate the module with the shared sample clock in Mainframe A.
- If a groupID is not used with the call e1432_read_data(), empty the VT1432As' FIFOs in Mainframe B before Mainframe A. In other words, do not empty the FIFOs in Mainframe A unless the FIFOs in Mainframe B have been emptied. For more information about groupID see "Grouping of Channels/Modules."
- If more than two mainframes are needed, daisy-chain them together. Treat each mainframe after the first as a Mainframe B. See the example on the next page.

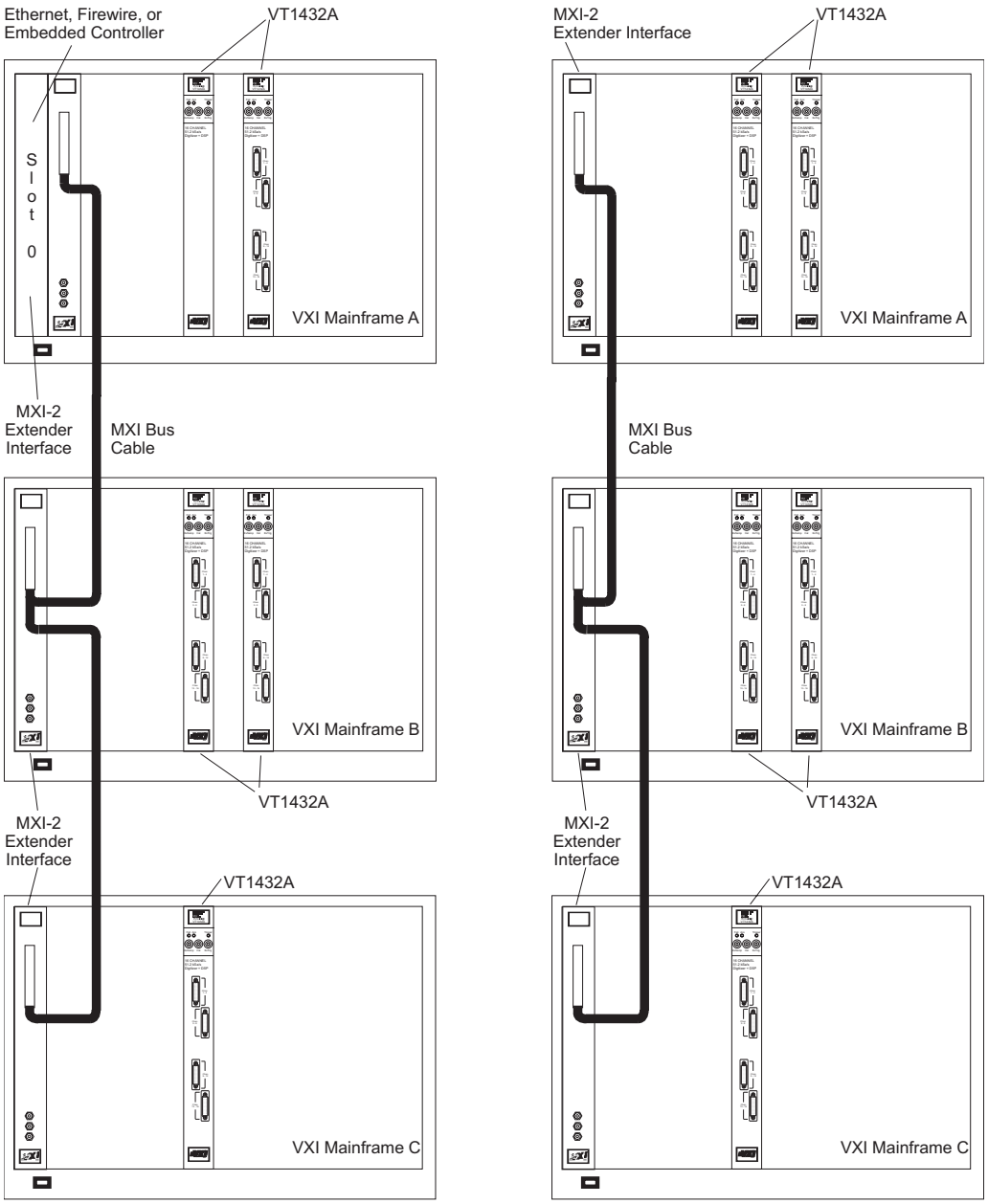
Phase Performance in Multiple Mainframe Measurements

Phase specifications are degraded by the delay that the inter-mainframe interface gives the sample clock. This delay is insignificant for many low-frequency applications because the phase error is proportional to frequency. A system with two VXI-MXI modules and a 1 meter cable, typically has a 76 nanosecond (ns) sample clock delay in Mainframe B. This corresponds to an additional 0.007 degree phase error at 256 Hz and an additional 0.55 degree phase error at 20 kHz.

A 4 meter cable adds approximately 18 ns of delay for a total of 94 ns clock delay in Mainframe B. This corresponds to an additional 0.0087 degree phase error at 256 Hz and an additional 0.68 degree phase error at 20 kHz.

The cable adds approximately 6 ns per meter of cable.

Each daisy-chained mainframe adds another increment of delay, but only for the additional cabling length.



Example 3: Embedded Slot 0 Controller

Example 4: MXI-2 Daisy Chain

Figure 4-2: Multiple mainframes - three mainframes

Synchronization in Multiple-mainframe Measurements

A TTL Trigger line between VT1432As making group measurements keeps all modules synchronized. This is an open-collector line where each module holds the one designated as the SYNC line low until the module is ready to advance to the next measurement state. Another TTL Trigger line is designated to carry the sample clock to all modules. This shared sample clock may come from any VT1432A module in Mainframe A or from an external signal routed through the Slot 0 Commander in Mainframe A.

One module is responsible for pulling the SYNC line low to start each group's state transition. Then, each module holds the line low until it is ready. When all modules are ready, the SYNC line drifts high. The unidirectional line prevents modules in Mainframe B from holding-off modules in Mainframe A.

The lowest logical address must be in Mainframe A because of VXI-MXI and Resource Manager (RM) constraints. Group constraints with the C-Library force modules in Mainframe A to have their FIFOs emptied last. The C-Library reads data in channel order, so the highest channel is read last. To get this to work automatically, the call to e1432_assign_channels() must list the logical addresses in descending order.

Channel triggering must be done only by modules in Mainframe A. A trigger in any other mainframe would not be communicated back on the SYNC line to Mainframe A. The C-Library itself selects the VT1432A with the highest channel number for synchronization.

VXI-MXI Module Setup and System Configuration

To set up a multiple mainframe system, follow the "Hardware Installation Rules" which appear in Chapter 2 of the Agilent/HP E1482B VXI-MXI Bus Extender User's Manual. This allows the Resource Manager to configure the system.

The VXI-MXI Module setup in Mainframe A needs to be changed from those set by the factory. The VXI-MXI module is not the Slot 0 Controller for Mainframe A. See Table 2-1. Configuration Settings in the Agilent/HP E1482B VXI-MXI Bus Extender User's Manual. This requires changing several switch settings.

- Set the module as not being the Slot 0 Controller.
- Set the VME timeout to 200 μs.
- Set the VME BTO chain position to 1 extender, non-slot0.
- Do not source CLK10.
- Set the proper logical address.

Measurement Process

Measurement Setup and Control

When the VT1432A makes a measurement, the measurement itself consists of two phases: the measurement initialization, and the measurement loop. Each of these phases consists of several states, through which the measurement progresses.

The transition from one state to the next is tied to a transition in the Sync/Trigger line (one of the TTL trigger lines on the VXI back plane). A state (such as Idle) begins when the Sync/Trigger line goes low. The Sync/Trigger line then remains low as long as the state is in effect. When the Sync/Trigger line goes high it signals the transition to the next state. See the sections “Measurement Initialization” and “Measurement Loop” below for more details about these transitions. During all the transitions of the Sync/Trigger line, the clock line continues with a constant pulse.

The Sync/Trigger line is “wire-OR’d” such that all modules in a multiple-module system (within one mainframe) must release it for it to go high. Only one VT1432A is required to pull the Sync/Trigger line low. In a system with only one VT1432A, the Sync/Trigger line is local to the module and not is routed to a TTL TRIGGER line on the VXI back plane.

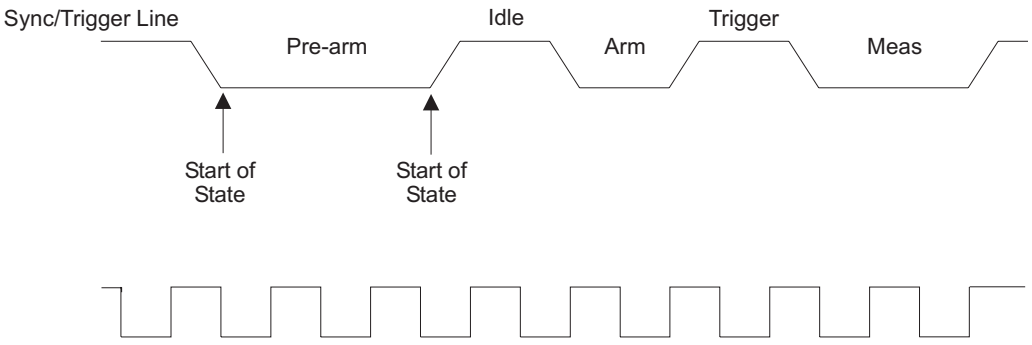


Figure 4-3: Transitions between states

Parameter Settings

Many parameters are channel-dependent, meaning that each channel can be set independently of the others in the module. Other parameters are module-dependent; changing a module-dependent parameter for a channel will change it for all channels on that module. For example, changing blocksize, a module-dependent parameter, for input channel 3 will also change the block size for all other channels in the same VT1432A module as channel 3.

When possible, parameters are written to the hardware as soon as they are received. Sometimes, the parameter can't be written to the hardware until the start of a measurement; in this case the value of the parameter is saved in RAM in the VT1432A module until the measurement is started with e1432_init_measure. Some parameters can be changed while a measurement is running, but many do not take effect until the next start of a measurement.

Measurement Initiation

This section describes the measurement initiation process in the VT1432A.

The measurement initialization states, and the corresponding Sync/Trigger line transitions (with 'H' for high, 'L' for Low) are:

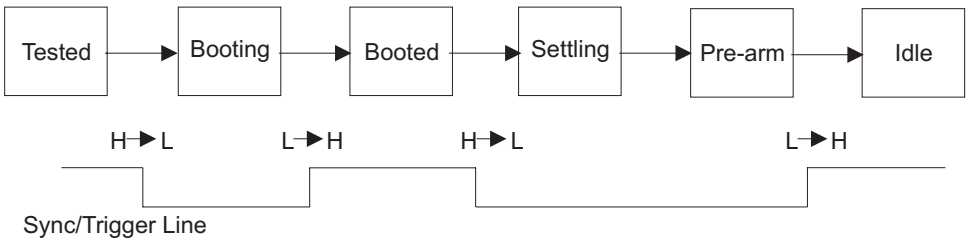


Figure 4-4: Measurement initialization

The module enters the TESTED state after a reset. In this state, all of the module parameters may be set. The VT1432A stays in the TESTED state until it sees a high-to-low transition of the Sync/Trigger line.

In the BOOTING state, the digital processors of the module load their parameters, and their program. Once done, the module releases the Sync/Trigger line and moves to the BOOTED state. The VT1432A stays in the BOOTED state until it sees a high-to-low transition of the Sync/Trigger line (that is, all the VT1432As in the system have booted).

In the SETTLING state, the digital filters are synchronized, and the digital filter output is 'settled' (it waits N samples before outputting any data). Once the module is settled, it advances to the PRE_ARM state.

In the PRE_ARM state, the module waits for a pre-arm condition to take place. The default is to auto-arm, so the module would not wait at all in this case. When the pre-arm condition is met, the module releases the Sync/Trigger line and advances to the IDLE state.

This complete measurement sequence initialization, from TESTED through BOOTING, BOOTED, SETTLING, PRE-ARM, and IDLE, can be performed with a call to the function e1432_init_measure.

Measurement Loop

This section describes the measurement loop in the VT1432A.

The progression of measurement states and the corresponding Sync/Trigger line transitions are:

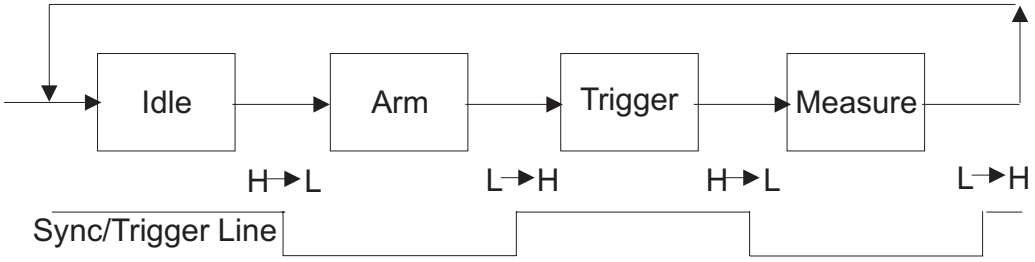


Figure 4-5: Measurement loop

In the IDLE state the VT1432A writes no data into the FIFO. The VT1432A remains in the IDLE state until it sees a high-to-low transition of the Sync/Trigger line or an RPM arm/trigger point is calculated. If any of the VT1432As in the system is programmed for auto arming (with e1432_set_auto_arm), the Sync/Trigger line is immediately pulled low by that VT1432A. The VT1432A may also be moved to the ARM state by an explicit call to the function e1432_arm_measure.

Upon entering the ARM state the VT1432A starts saving new data in its FIFO. It remains in the ARM state until the Sync/Trigger line goes high. If the VT1432A is programmed with a pre-trigger delay, it collects enough data samples to satisfy this pre-trigger delay, and then releases the Sync/Trigger line. If no pre-trigger delay has been programmed, it releases the Sync/Trigger line immediately. When all modules in a system have released the Sync/Trigger line (allowing it to go high), a transition to the TRIGGER state occurs.

Upon entering the TRIGGER state the VT1432A continues to collect data into the FIFO, discarding any data prior to the pre-trigger delay. The VT1432A remains in the TRIGGER state until it sees a high-to-low transition of the Sync/Trigger line. The Sync/Trigger line is pulled low by any VT1432A which encounters a trigger condition and is programmed to pull the Sync/Trigger line. If any VT1432A is programmed for auto triggering (with e1432_set_auto_trigger), the Sync/Trigger line is pulled low immediately. The Sync/Trigger line may also be pulled low by an explicit call to the function e1432_trigger_measure.

Upon entering the MEASURE state the VT1432A continues to collect data. The VT1432A also presents the first data from the FIFO to the selected output port, making it available to the controller to read. The VT1432A holds the Sync/Trigger line low as long as it is actively collecting data. In overlap block mode the VT1432A stops taking data as soon as a block of data has been collected, including any programmed pre- or post-trigger delays. (It starts again when another trigger occurs). In continuous mode, the VT1432A stops taking data only when the FIFO overflows. When data collection stops, the VT1432A releases the Sync/Trigger line. When all VT1432As are finished and the Sync/Trigger line goes high, the VT1432A goes into the IDLE state again.

The measurement initialization and loop may be interrupted at any time with a call to e1432_reset_measure, which puts the module in the TESTED state.

Register-based VXI Devices

The VT1432A is a register-based VXI device. Unlike message-based devices which use higher-level programming using ASCII characters, register-based devices are programmed at a very low level using binary information. The greatest advantage of this is speed. Register-based devices communicate at the level of direct hardware manipulation and this can lead to much greater system throughput.

It is not necessary to access the registers in order to use the VT1432A. The VT1432A's functions can be more easily accessed using the VT1432A Host Interface Library software. However, more information about the registers is available in Appendix A: Register Definitions.

Arm and Trigger

This section explains some terminology relating the the “Arm” and “Trigger” steps in the measurement loop. As an example a measurement might be set up to arm at a certain RPM level and then subsequently trigger at an external event corresponding to top dead center (TDC). The settings would be:

- Arm: RPM Step Arm
- Trigger: External Trigger

To begin a throughput session at this same RPM/TDC event, the first external trigger after a specified RPM should start a continuous mode measurement. Now (using overlap block mode) the settings would be:

- Pre-Arm: RPM Step Arm
- Arm: Auto
- Trigger: Auto

In the measurement loop, an arm must take place before a trigger. The number of triggers that occur before waiting for another arm condition can be specified. The default is one trigger for each arm. For each trigger, a block of data is sent to the host.

The first arm in a measurement is the pre-arm. By default, the pre-arm condition is the same as the regular arm conditions.

Valid Arm (and Pre-Arm) conditions are:

- Auto Arm
- Manual Arm
- RPM Step Arm

Valid trigger conditions are:

- Auto Trigger
- Input Trigger
- Source Trigger
- External Trigger
- Manual Trigger
- Tachometer Edge Trigger

VT1432A Triggering

The following is a short discussion of triggering for the VT1432A.

Triggering is defined as the transition from the armed state to the triggered state. This transition is caused by a low going edge on a TTL trigger line. Which one of the eight TTL trigger line is chosen by e1432_get_ttltrg_lines().

The low-going transition of the TTL trig line can be caused by any of the following items:

trigger type	enabling function
the AUTO TRIGGER circuitry	e1432_set_auto_trigger()
the e1432_trigger_measure() function	e1432_trigger_measure()
a source trigger	e1432_set_trigger_channel()
a tach trigger	e1432_set_trigger_channel()
an external trigger	e1432_set_trigger_ext()
an input level or bound trigger event	e1432_set_trigger_channel() and e1432_set_trigger_mode()

Each of these trigger sources can be enabled or disabled independently, so quite complex trigger setups are possible. In all cases, however, the first trigger event kicks off the measurement and the following trigger events become superfluous.

Note that for e1432_set_auto_trigger() the setting E1432_MANUAL_ARM really means “don’t auto trigger” not “expect a manual trigger”.

For single VT1432A systems, the TTL trigger signal is not connected to the VXI backplane. For multiple VT1432A systems, the e1432_init_measure() function connects the VT1432A trigger lines to the VXI backplane, and at that point, the selection of which TTL trigger lines through e1432_get_ttltrg_lines() is relevant. Multiple mainframe systems will need to account for the unidirectional nature of the inter-mainframe MXI extenders which will prevent all but the “upstream” mainframe from triggering the system.

Data Transfer Modes

The VT1432A can be programmed to use either of two data transfer modes: overlap block mode and continuous mode. Block mode will be described first.

Block Mode (Agilent/HP E1431A)

The VT1432A's overlap block mode is similar the block mode which is used in other Agilent instruments such as the Agilent/HP E1431A. In block mode, the input hardware acquires one block after getting an arm and trigger. It does not allow the system to trigger until it is ready to process the trigger, and it acquires pre-trigger data if necessary. The hardware does not accept a new arm and trigger until the acquired block is sent to the host. There is no provision for overlap or queuing up more than one block when in block mode. There is also no way for a FIFO overflow to occur.

The VT1432A's overlap block mode can be configured to act exactly like traditional block mode. It also has additional capabilities as described below.

Continuous Mode.

Both the VT1432A's and the Agilent/HP E1431A use continuous mode. In this mode , the input hardware waits for an arm and trigger, and then starts acquiring data continuously. If the host is slow, several blocks can be queued up in the input hardware. If the host gets far enough behind, a FIFO overflow occurs and the input stops acquiring data.

The VT1432A's overlap block mode can be configured to act similarly to continuous mode, but not identically. The VT1432A can also use the traditional continuous mode.

Overlap Block Mode

Overlap block mode combines features of both block mode and continuous mode. The main difference between overlap block mode and traditional block mode is that overlap block mode allows additional arms and triggers to occur before an already-acquired block is sent to the host. A trigger can occur before the end of the previous block, so overlapping blocks are possible (hence the name "overlap block mode"). As in continuous mode, there is an overlap parameter which controls how much overlap is allowed between consecutive blocks.

Limit on Queuing of Data

In overlap block mode, a number of trigger events may be queued up before the host reads the data for those triggers. The host may get further and further behind the data acquisition.

However, if the host gets far enough behind that the FIFO fills up, data acquisition must momentarily stop and wait for data to get transferred to the host. This places a limit on how far in time the host can be behind the data acquisition. By setting the size of the FIFO, one can control how far behind the host can get.

Making Overlap Block Mode Act Like Traditional Block Mode

If the FIFO size is set the same as the block size, or if the number of pending triggers is limited to zero, then overlap block mode becomes identical to traditional block mode.

Making Overlap Block Act Like Continuous Mode

If the module is in auto-arm and auto-trigger mode, then overlap block mode becomes nearly the same as continuous mode.

One difference is that traditional continuous mode has a single arm and trigger, while overlap block mode may have multiple arms and triggers. Another is that continuous mode can be configured to start at any type of trigger event, while overlap block mode must be in auto-trigger mode to act like continuous mode. Finally, continuous mode always stops when a FIFO overflow occurs, but overlap block mode does not.

VT1432A Interrupt Behavior

Interrupt Setup

The VT1432A VXI module can be programmed to interrupt a host computer using the VME interrupt lines. VME provides seven such lines, and the VT1432A module can be told to use any one of them (see e1432_set_interrupt_priority).

The VT1432A can interrupt the host computer in response to different events. A mask of events can be specified on which to interrupt. This mask is created by OR'ing together the various conditions that are desired. The following table, copied from the e1432_set_interrupt_mask manual page, shows the conditions that can cause an interrupt:

Interrupt Mask Bit Definitions

Define (in e1432.h)	Description
E1432_IRQ_MEAS_WARNING	Non-fatal measurement warning
E1432_IRQ_BLOCK_READY	Block of data ready in FIFO
E1432_IRQ_MEAS_STATE_CHANGE	Measurement state machine changed state
E1432_IRQ_TRIGGER	Trigger ready for transfer to other modules
E1432_IRQ_OVERLOAD_CHANGE	Overload status changed
E1432_IRQ_MEAS_ERROR	FIFO overflow
E1432_IRQ_TACHS_AVAIL	Raw tach-times available
E1432_IRQ_SRC_STATUS	Source status change

VT1432A Interrupt Handling

To make the VT1432A module do the interrupt, both a mask and a VME Interrupt line must be specified, by calling e1432_set_interrupt_mask and e1432_set_interrupt_priority respectively. Once the mask and line have been set, and an interrupt occurs, the cause of the interrupt can be obtained by reading the E1432_IRQ_STATUS_REG register (using e1432_read_register). The bit positions of the interrupt mask and status registers match so the defines can be used to set and check IRQ bits.

Once it has done this interrupt, the module will not do any more VME interrupts until re-enabled with e1432_reenable_interrupt. Normally, the last thing a host computer's interrupt handler should do is call e1432_reenable_interrupt.

Events that would have caused an interrupt, but which are blocked because e1432_reenable_interrupt has not yet been called, will be saved. After e1432_reenable_interrupt is called, these saved events will cause an interrupt, so that there is no way for the host to "miss" an interrupt. However, the module will only do one VME interrupt for all of the saved events, so that the host computer will not get flooded with too many interrupts.

For things like "E1432_IRQ_BLOCK_READY", which are not events but are actually states, the module will do an interrupt after e1432_reenable_interrupt only if the state is still present. This allows the host computer's interrupt handler to potentially read multiple scans from a VT1432A module, and not get flooded with block ready interrupts after the fact.

Host Interrupt Setup

The VT1432A Host Interface library normally uses the SICL I/O library to communicate with the VT1432A hardware. To receive VME interrupts, a variety of SICL setup calls must be made. The "examples" directory of the VT1432A distribution contains an example of setting up SICL to receive interrupts from a VT1432A module.

This is a summary of how to set up SICL to receive a VT1432A interrupt:

- Query SICL for which VME interrupt lines are available, using ivxibusstatus and ivxirminfo.
- Tell the VT1432A module to use the VME interrupt line found in step one, using e1432_set_interrupt_priority.
- Set up an interrupt handler routine, using ionintr and isetintr. The interrupt handler routine will get called when the interrupt occurs.
- Set up interrupt mask in the VT1432A module, using e1432_set_interrupt_mask.

Host Interrupt Handling

When the VT1432A asserts the VME interrupt line, SICL will cause the specified interrupt handler to get called. Typically the interrupt handler routine will read data from the module, and then re-enable VT1432A interrupts with e1432_reenable_interrupt. The call to e1432_reenable_interrupt must be done unless the host is not interested in any more interrupts.

Inside the interrupt handler, almost any VT1432A Host Interface library function can be called. This works because the Host Interface library disables interrupts around critical sections of code, ensuring that communication with the VT1432A module stays consistent. Things that are not valid in the handler are:

- Calling e1432_delete_channel_group to delete a group that is simultaneously being used by non-interrupt-handler code.
- Calling one of the read data functions (e1432_read_raw_data, e1432_read_float32_data, or e1432_read_float64_data), if the non-interrupt-handler code is also calling one of these functions.
- Calling e1432_assign_channel_numbers to reset the list of channels that are available to the VT1432A library.

As is always the case with interrupt handlers, it is easy to introduce bugs into the program, and generally difficult to track these bugs down. Be careful when writing this function.

Data Gating

Sometimes it is desirable to monitor data from some input channels and not others. The function e1432_set_enable enables or disables data from an input channel (or group of channels). If data is enabled, then the data can be read using e1432_block_available and e1432_read_xxx_data. If data is disabled, data from the specified channel is not made available to the host computer.

This parameter can be changed while a measurement is running, to allow the host computer to look at only some of the data being collected by the VT1432A module. While data from a channel is disabled the input module continues to collect data but it is not made available to the host computer. The host can then switch from looking at some channels to looking at others during the measurement. In contrast, the function e1432_set_active completely enables or disables a channel and can’t be changed while a measurement is running.

For order tracking measurements this function can be used to switch between receiving order tracking data, ordinary time data, or both.

VT1432A Parameters

Some parameters, such as range or coupling, apply to specific channels. When a channel ID is given to a function that sets a channel-specific parameter, only that channel is set to the new value.

Some parameters, such as clock frequency or data transfer mode, apply globally to a module. When a channel ID is used to change a parameter that applies to a whole module, the channel ID is used to determine which module. The parameter is then changed for that module.

Starting and stopping a measurement is somewhat like setting a global parameter. Starting a measurement starts each active channel in each module that has a channel in the group.

After firmware is installed, and after a call to e1432_preset, all of the parameters (both channel-specific and global) in a VT1432A module are set to their default values. For channel-specific parameters, the default value may depend on the type of channel. Some channel-specific parameters apply only to a specific type of channel. For example, tach holdoff applies only to tach channels. Setting such a parameter for a channel that doesn't make sense will result in an error.

At the start of a measurement, the VT1432A firmware sets up all hardware parameters, and ensures that the input hardware is settled before starting to take data. The firmware also ensures that any digital filters have time to settle. This ensures that all data read from the module will be valid.

However, after a measurement starts, VT1432A parameters can still be changed. The effect of this change varies, depending on the parameter. For some parameters, changing the value aborts the measurement immediately. For other parameters, the measurement is not aborted, but the changed parameter value is saved and not used until a new measurement is started. For still other parameters, the parameter change takes place immediately, and the data coming from the module may contain glitches or other effects from changing the parameter.

There is no way to tell the module to wait for settling when changing a parameter in the middle of a measurement. The only way to wait for settling is to stop and re-start the measurement. Also, there is no way to disable the settling that takes place at the start of a measurement.

For More Information

Refer to the (on-line) VT1432A Function Reference for a list of all functions and the parameters needed for each function. (See "Where to get more information" in the chapter titled "Using the VT1432A)."

5

Module Description

Module Features

The VT1432A 16 Channel 51.2 kSamples/s Digitizer plus DSP is a VXI C-sized, scalable input module. The VT1432A may contain up to four 4-channel input assemblies so that the module may have a total of up to 16 inputs.

The following is a list of some of the features of the VT1432A. See "Specifications" for more detailed information.

The standard VT1432A is described in this chapter. The Arbitrary Source and Tachometer options are described in other chapters.

General Features

- ❑ Fundamental sample rate selectable within the range of 32768 Hz to 51200 Hz.
- ❑ Digital sample rate decimation in a 1, 2, 5 sequence.
- ❑ Variable Block Size (binary)
- ❑ Optional Large Data Buffer (2 MSamples, expandable to 16 MSamples)
- ❑ Data from FIFO available with overlap
- ❑ VXI Shared Memory
- ❑ Flexible triggering, including pre- and post-triggering
- ❑ AC/DC coupling
- ❑ ICP® power supplies, with the optional ICP® 8-Channel Input (breakout box)
- ❑ Overload detection
- ❑ Synchronous sampling over multiple channels and VT1432A modules
- ❑ Large FIFO for long pre-trigger delays
- ❑ D32 VME Bus data transfer
- ❑ VXI Local Bus data transfer (with Local Bus option)

Arbitrary Source Features (option VT1432A-1D4)

- ❑ Sine output
- ❑ Random noise output
- ❑ Arbitrary output

Tachometer Features (option VT1432A-AYF)

- ❑ Current RPM value measurements
- ❑ Up/Down RPM triggered measurements

Other Options

- ❑ Local Bus, option VT1432A-UGV
- ❑ 32 MB total RAM, option VT1432A-ANC (standard is 4 MB)

Block Diagram

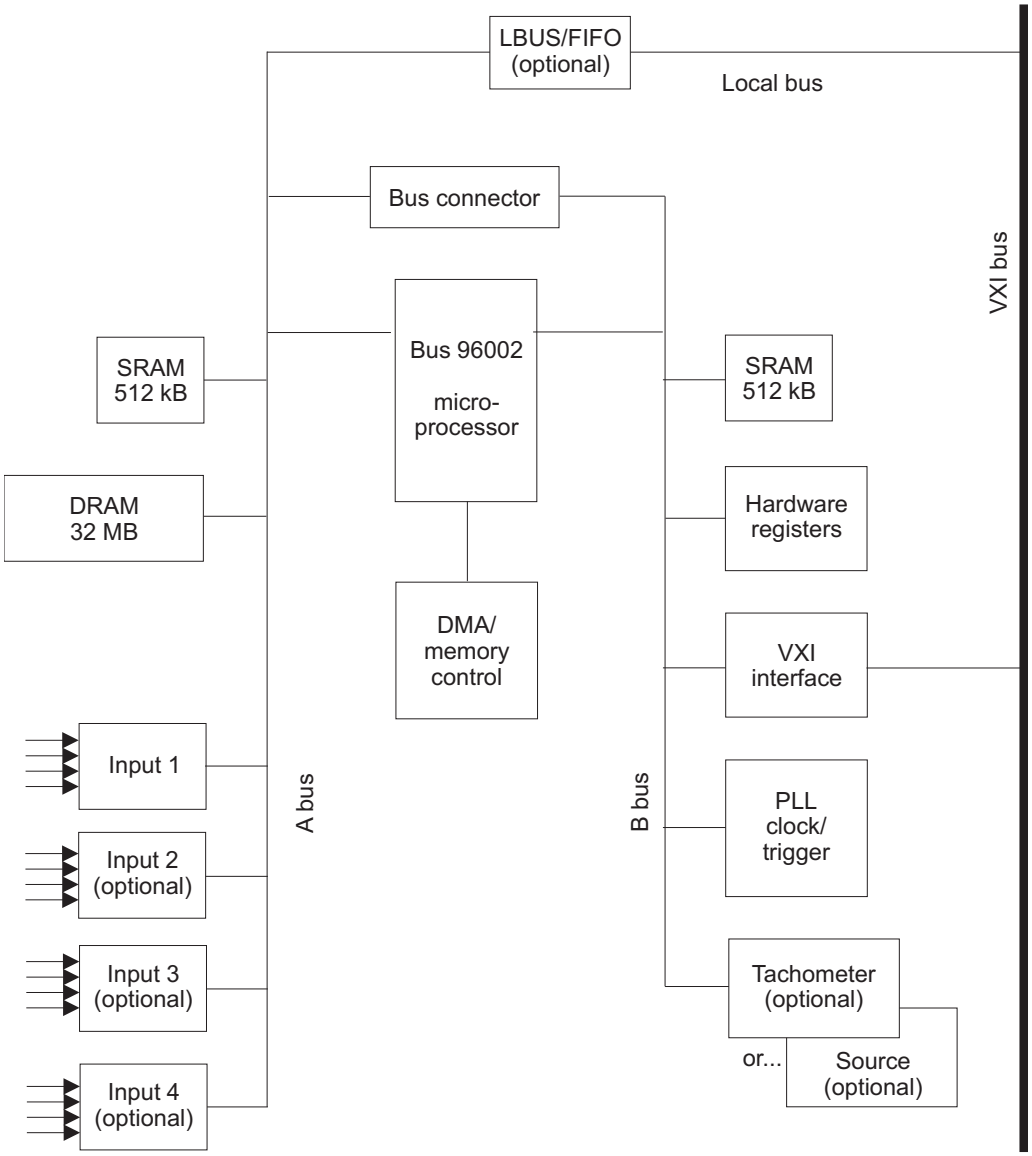


Figure 5-1: VT1432A block diagram

For block diagrams of the Arbitrary Source and the Tachometer, see the chapters on the Arbitrary Source option and the Tachometer option.

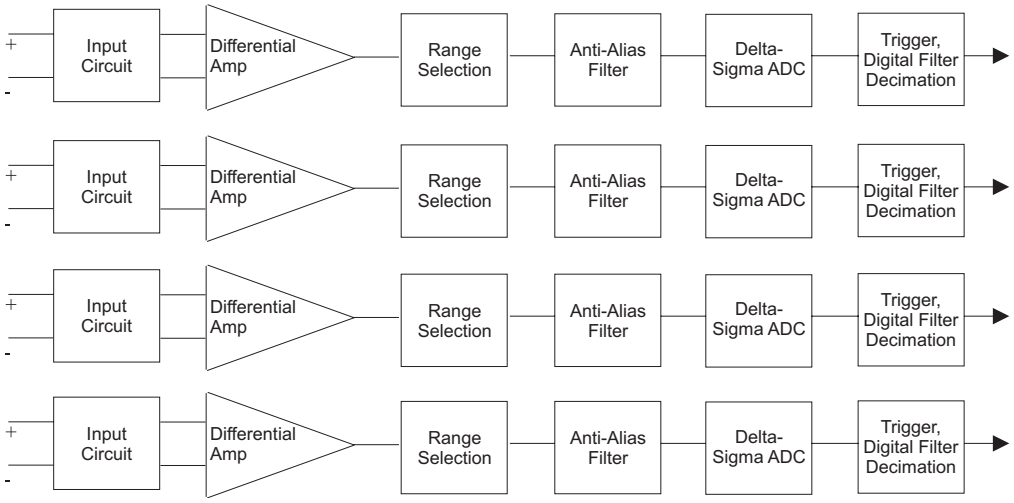


Figure 5-2: Input section diagram

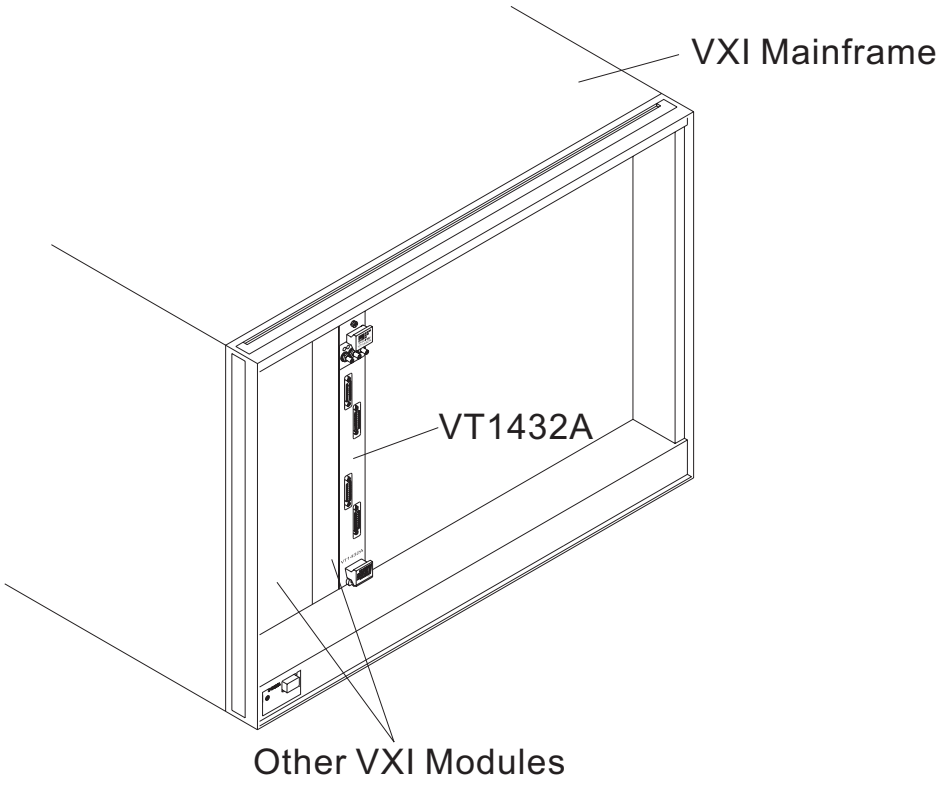


Figure 5-3: The VT1432A in a VXI mainframe

VT1432A Front Panel Description

Front Panels for 4, 8, and 16 Channels

The VT1432A may have any of several front panels depending on options and number of input channels. The following illustration shows front panels for 4, 8, and 16 channels.

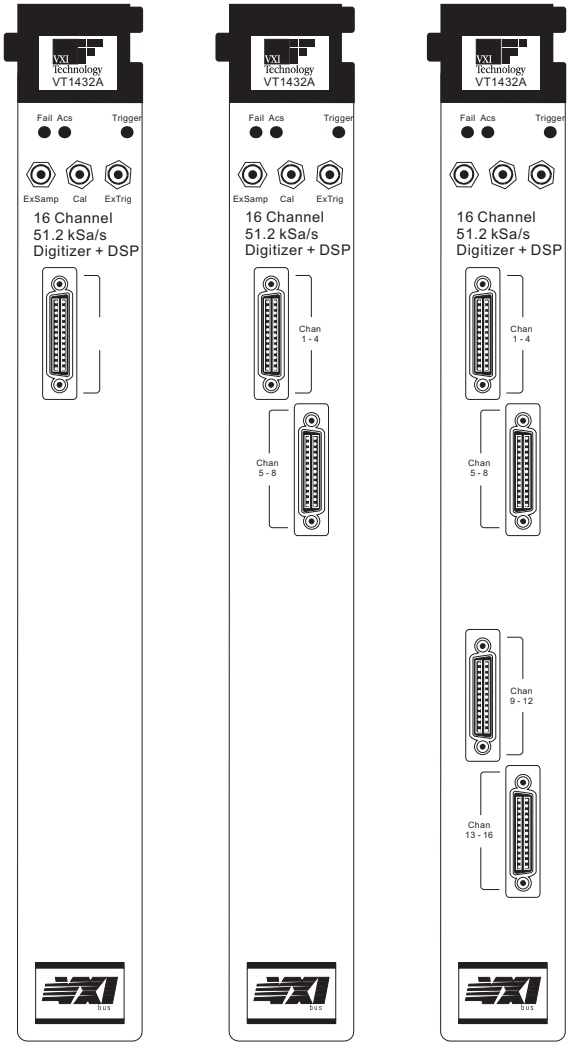


Figure 5-4: Front panels for 4, 8 and 16 channels

Standard Front Panel

This is the front panel for a standard VT1432A (this example has 16 inputs). The LEDs and connectors are described on the next page.

If the VT1432A has an Arbitrary Source (Option VT1432A-1D4) or a Tachometer (Option VT1432A-AYF) its front panel will be different. See the chapter on the Arbitrary Source or the chapter on the Tachometer for a description of its front panel.

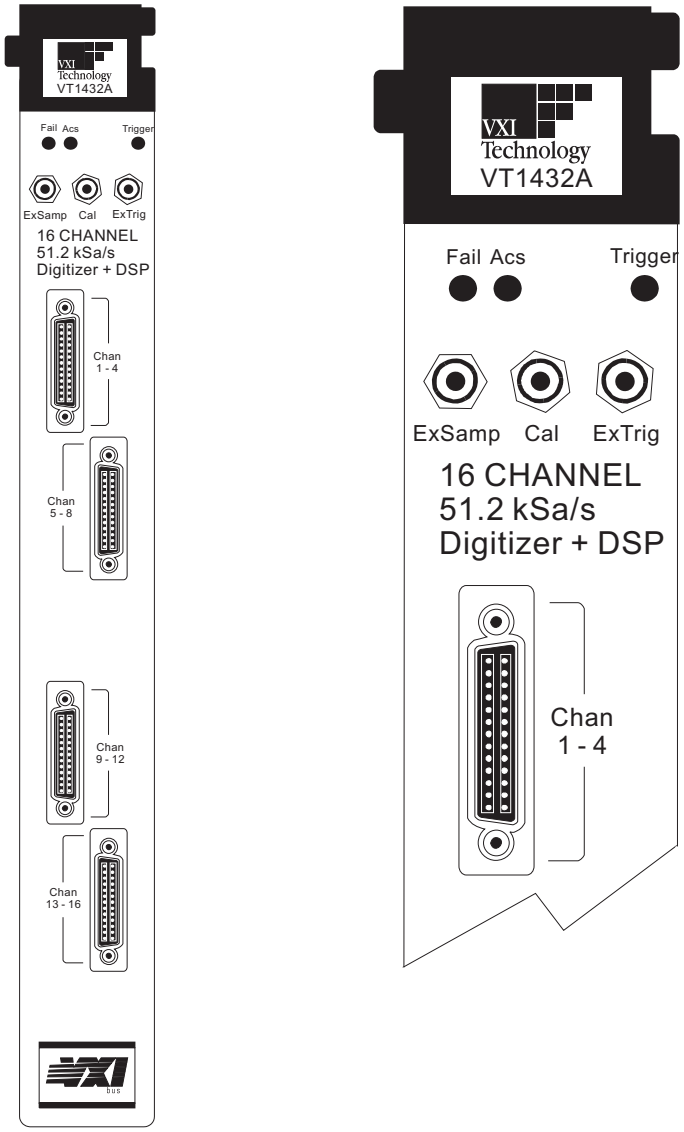


Figure 5-5: VT1432A standard front panel

Status LEDs

- Fail: This is the standard VXI "Failed" indicator. It lights briefly when powering up and normally goes out after a few seconds. If it stays on it indicates a hardware failure in the module.
- Acs: This is the standard VXI "Access" indicator. When it is on, it indicates that another device on the bus is contacting the module, for example to transfer data or read registers.
- Trigger: This LED flashes on each time the measurement triggers, so when it is blinking it indicates that the measurement is triggering.

  If the VT1432A has the Tachometer option, this LED is defined differently. See the chapter: The Tachometer Option (VT1432A-AYF).

SMB Connectors

- ExSamp: This is an input connector for an external sample clock. The sample clock must be TTL level and have a frequency between 40.96 kHz and 100 kHz. Internally this frequency can be decimated.

- Cal: This connector is used for calibration. It can be configured to output a calibration signal or to accept an input calibration signal. See the calibration section in this chapter.

- ExTrig: This allows for an external trigger input to the VT1432A. The input signal must be TTL, other characteristics can be defined in software. ExTrig can be enabled or disabled in software.

Input Connectors (1, 2 or 4)

These connectors are attached to the cables from an 8-Channel Input (breakout box) — two input connectors for each 8-Channel Input. They connect the input signal to the VT1432A. Each connector carries four channels. Depending on options, there can be 1, 2 or 4 input connectors (4 - 16 channels).

VXI Backplane Connections

Power Supplies and Ground

The VT1432A conforms to the VME and VXI specifications for pin assignment. The current drawn from each supply is given in the specifications chapter.

Data Transfer Bus

The VT1432A conforms to the VME and VXI specifications for pin assignment and protocol. A16, A24, D16 and D32 data transfers are supported.

DTB Arbitration Bus

The VT1432A 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 VT1432A 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 VT1432A 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), except that pulling the SYSRESET* line low also resets the Control Register itself, while a software reset does not .

The Local Bus (Option VT1432A-UGV)

The VXI specification includes a 12-wire Local Bus between adjacent module slots. Using the Local Bus, a standard byte-wide ECL protocol has been defined which can transfer data from left to right at up to 15.7 MB/s using VT1432A. If equipped with option VT1432A-UGV, the VT1432A 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.

Local Bus vs VME Transfers

With this option, data can be transferred from the VT1432A two different ways; via the VME Bus or via the Local Bus.

The VME Bus is the universal data bus for VXI architecture. It provides flexibility and versatility in transferring data. Transfers over the VME Bus can be 16 or 32 bits wide.

The Local Bus supports faster transfer rates than the VME Bus. For example, if data is transferred from the VT1432A to the VT2216A VXI/SCSI Interface Module, the Local Bus provides a direct pipeline to the VT2216A.

Using the Local Bus, data can be transferred in the background while processing data in a signal-processing module.

All Local Bus data-transfers originate in an input module and move towards a signal processing or disk throughput module to the right of the input module. If other modules generate data to the left of the input module, the input module will pass the data to its right and append its own data to the data blocks from previous modules.

The VT1432A VXI Device

Address Space

The VXI system architecture defines two types of address space. A16 space consists of 64 kilobytes (kB) and A24 consists of 16 megabytes (MB).

The VT1432A has a 32-bit port through which it has access to the A16 and A24 space. It can also use D32 to send and receive data though the port. Or it can use the port for 16-bit data transfers by using only 16 of the 32 bits available. The VT1432A performs a different type of VME cycle depending on the number of bits transferred per cycle (two cycles for 16-bit transfers and one cycle for 32-bit).

Shared Memory

Shared memory provides a way for the VT1432A to transfer data to a controller. The shared memory in the VT1432A is mapped to the A24 VXI address space. The controller can then access that same address space to receive or write data. A function can be called to get the data. See “The Host Interface Library” chapter.

Memory Map

The following discussion of memory mapping is included as supplemental information. It is not needed to operate the VT1432A as this functionality is hidden when using the VT1432A Host Interface Library software.

Refer to the VT1432A block diagram (Figure 5-1). The VXI interface maps some of the VT1432A’s B-bus internal memory space so that it is visible to the VXI Bus. The port connecting the A and B busses also allows the VXI Bus access to the SRAM, DRAM and inputs which are on the A bus. (SRAM stands for Static RAM; DRAM is Dynamic RAM.)

The VXI interface has two “windows” on the B bus memory space. Each is 512 kB, which is 128 32-bit words. One of the windows is fixed and the other is movable. The movable window allows the VXI Bus access to many different parts of the memory space. The fixed window contains:

- The A16 registers
- The B-bus SRAM
- The hardware registers
- The FIFO (which is in DRAM)

The mapping of the fixed and movable windows is illustrated as follows:

Address		
$FFFFF_{16}$ $8000\ 0_{16}$	Movable DSP Bus Window	Movable
$7FFFF_{16}$ $3000\ 0_{16}$	Fixed DSP Bus Window	Fixed
$2FFFF_{16}$ $2000\ 0_{16}$	Send/Receive Data Registers	
$1FFFF_{16}$ $0004\ F_{16}$	Fixed DSP Bus Window	
$0003\ F_{16}$ $0000\ 0_{16}$	VXI Bus A16 Registers	

For more information, see “The A24 Registers” in the chapter titled Register Definitions.

List of A16 Registers

The following lists the A16 registers. For more information see "The A16 Registers" in the chapter titled Register Definitions.

Address	Read	Write
$3E_{16}$	Parameter 7 Register	
$3C_{16}$		
$3A_{16}$	Parameter 6 Register	
38_{16}		
36_{16}	Parameter 5 Register	
34_{16}		
32_{16}	Parameter 4 Register	
30_{16}		
$2E_{16}$	Parameter 3 Register	
$2C_{16}$		
$2A_{16}$	Parameter 2 Register	
28_{16}		
26_{16}	Parameter 1 Register	
24_{16}		
22_{16}	Query Response Register	Command Register
20_{16}		
$1E_{16}$	FIFO Count	
$1C_{16}$		
$1A_{16}$	Send Data	Receive Data
18_{16}		
16_{16}	RAM 1	
14_{16}		
12_{16}	RAM 0	
10_{16}		
$0E_{16}$	IRQ Status Register	IRQ Reset Register
$0C_{16}$	IRQ Config Register	
$0A_{16}$	Page Map Register	
08_{16}	Port Control Register	
06_{16}	Offset Register	
04_{16}	Status Register	Control Register
02_{16}	Device Type	
00_{16}	ID Register	Logical Address Register

Trigger Lines (TTLTRG)

TTLTRG consist of eight TTL lines on the VXI backplane on connector P2. They are available to provide synchronization between devices. VXI devices can use the TTLTRG lines for simple communication with other devices. For example, a device can wait for a line to go high before taking an action or it can assert a line as a signal to another device.

The VT1432A uses two trigger lines. These can be placed on any two of the eight TTLTRG lines available on the VXI backplane. The lines are:

- Sync/Trigger line
- Free-running clock line

When programmed in a multiple-module configuration, only one of the VT1432A modules can provide the clock signal but any of them can trigger.

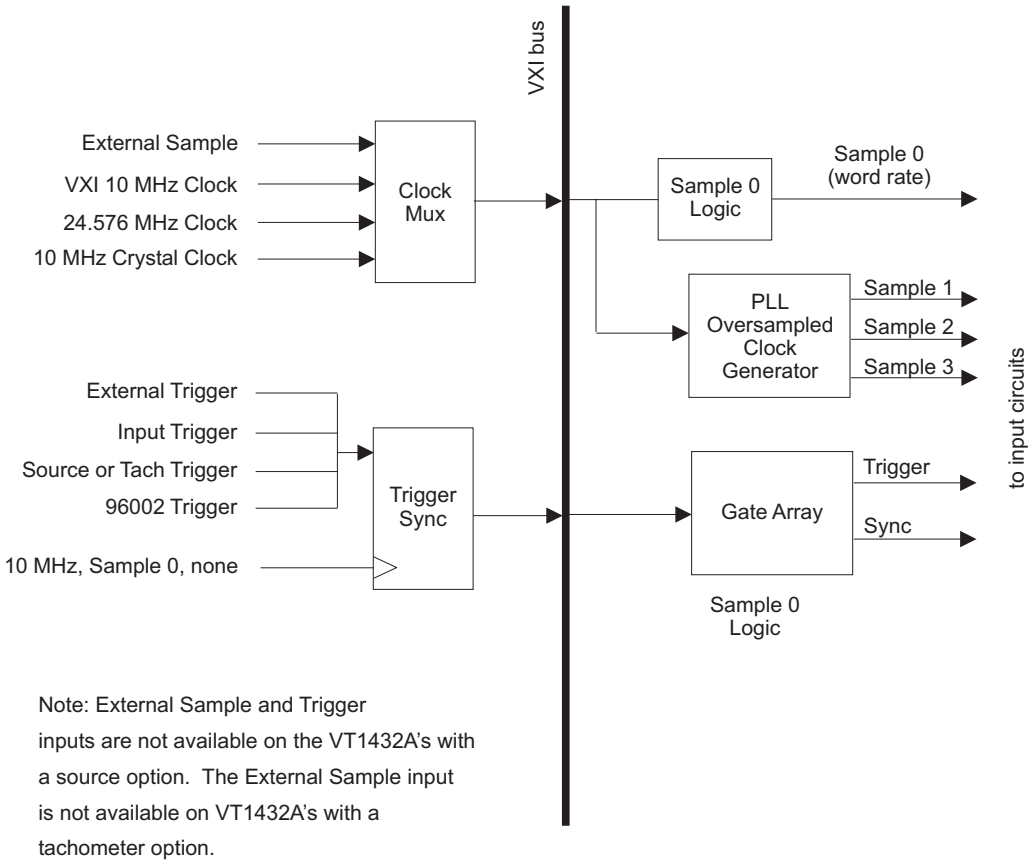


Figure 5-6: Clock/sync diagram

Providing an External Clock

The VT1432A can be programmed to accept an external word rate clock from the Sample 0 line on the VXI Bus. The digital filters are still functional, providing a range of effective word rates. All sampling is done simultaneously and is not multiplexed.

To connect an External Sample Clock, use the External Sample SMB connector on front panel of the VT1432A. External Sample at word rate and External Trigger are available on the front panel of VT1432A's which do not have an arbitrary source or tachometer option.

The external clock must be a fixed frequency. Its maximum frequency must not be higher than 100 kHz. Its minimum frequency must be at least 40.96 kHz.

Calibration Description

The Cal connector on the front panel of the standard VT1432A can be configured (in software) as either an input or an output. It can be set to any of four settings:

- DC - The VT1432A outputs a DC calibration signal from the millivolt range up to 15 volts.
- AC - The VT1432A outputs a signal from an Arbitrary Source option (in the same module or a different VT1432A module in the system.)
- Ground - The connector is shunted to ground for a zero-volt reference.
- Open Circuit - In this mode the connector becomes an input which can receive a calibration signal up to ±15 volts.

The VT1432A is calibrated at the factory and the calibration placed in EPROM memory for use at each power-up. In addition an auto-zero function is provided.

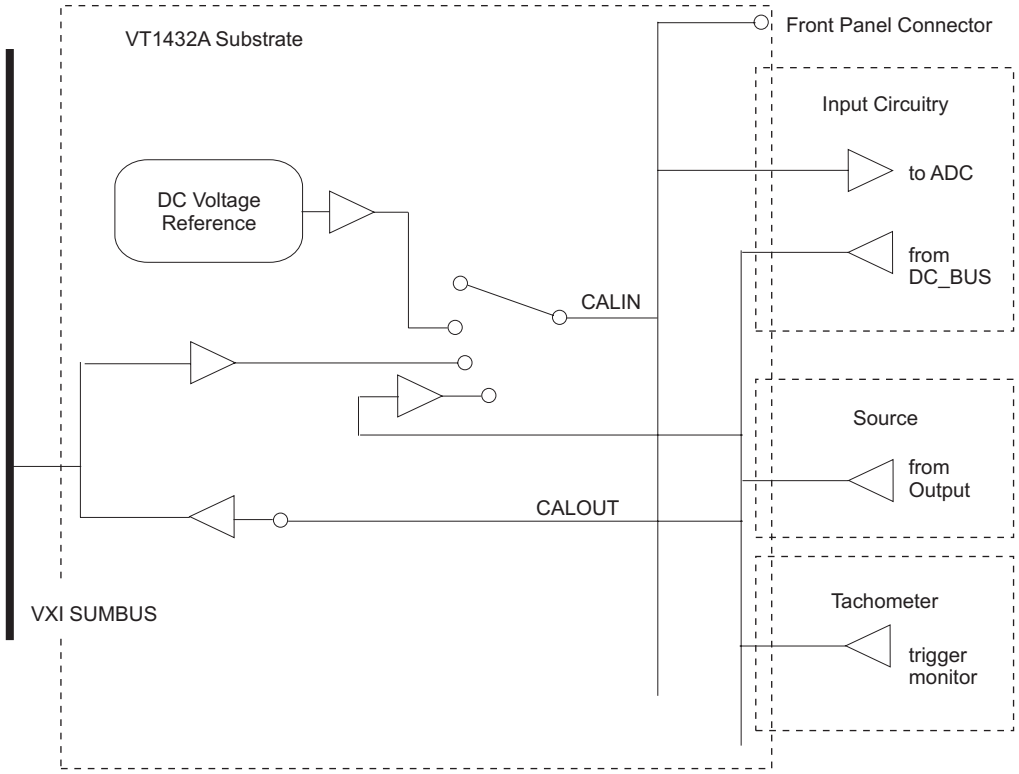


Figure 5-7: Calibration block diagram

6

The Arbitrary Source Option (VT1432A-1D4)

Arbitrary Source Description

An arbitrary source can be included with the VT1432A 16 Channel 51.2 kSamples/s Digitizer plus DSP as Option VT1432A-1D4. (It cannot be installed with a Tachometer, Option VT1432A-AYF.) The Arbitrary Source Option can supply arbitrary or sine signals under control of measurement software.

Trigger

The Arbitrary Source can be used to trigger the measurement and to trigger other modules in the measurement system.

Arbitrary Output

The Arbitrary Source can be programmed to output any signal that is described by data downloaded by the software.

Source Output Modes

The Arbitrary Source has several output modes including the following:

- arbitrary
- sine
- noise
- random
- burst

COLA (and Summer)

The COLA (Constant Output Level Amplifier) output supplies a signal similar to the Source “Out” output except that it is at a constant output level of about one volt peak.

The same connector (labeled “COLA”) can also be programmed as a summer input. A signal connected to this input is summed with the internal source output to create the final output.

External Shutdown

Shorting the center pin of the shutdown connector to its shield causes the source to ramp down and shut off.

Block Diagram

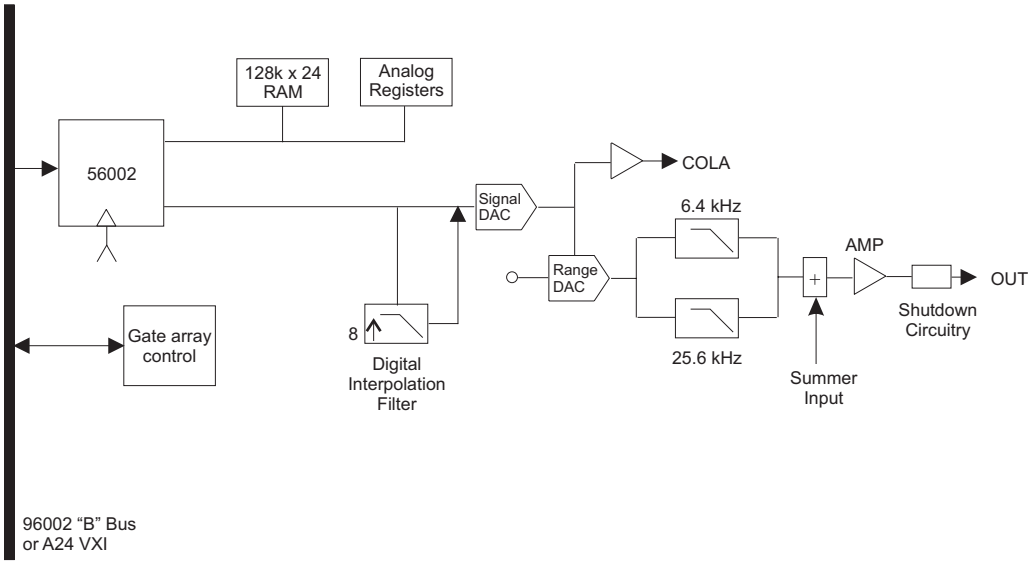


Figure 6-8: Arbitrary source option block diagram

The Arbitrary Source Option Front Panel

The VT1432A with the Arbitrary Source Option may have 4, 8, and 16 input channels. The following illustration shows a front panel for 16 channels. The LEDs and connectors are described on the next page.

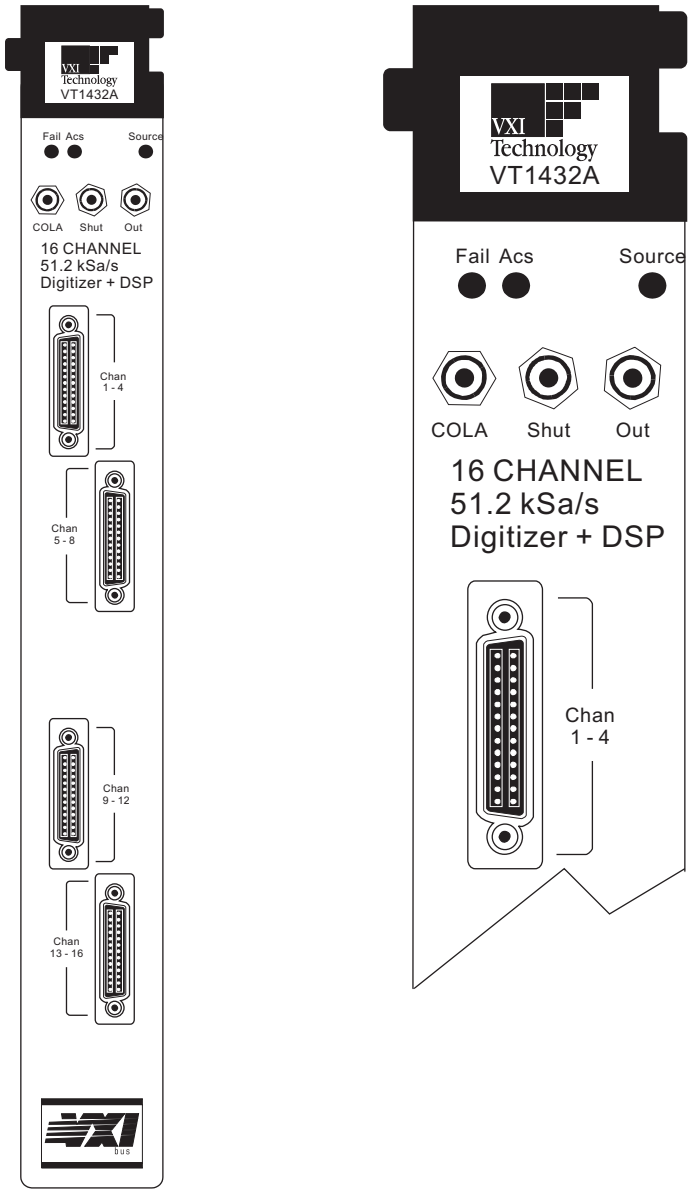


Figure 6-9: VT1432A with source option - front panel

LEDs and Connectors for the Arbitrary Source Option

Status LEDs

- Fail: This is the standard VXI “Failed” indicator. It lights briefly when powering up and normally goes out after a few seconds. If it stays on it indicates a hardware failure in the module.
- Acs: This is the standard VXI “Access” indicator. When it is on, it indicates that another device on the bus is contacting the module, for example to transfer data or read registers.
- Source: If this LED is lighted it indicates that the source is on and producing output.

SMB Connectors

- COLA: This is the output connector for the COLA (Constant Output Level Amplifier) output.

  This connector can also be configured as a Summer input. A signal connected to this input is summed with the internal source output to create the final output.

- Shut (Shutdown): Shorting the center pin of this connector to its shield causes the source to ramp down and shut off.

- Out: This is the main output of the Arbitrary Source.

  The Out connector can also be configured to output a calibration signal. This is not quite the same as the calibration signal described in Chapter 5 because it comes directly from the internal source without going through the other circuitry of the calibration section.

Input Connectors (1, 2 or 4)

These connectors are attached to the cables from an 8-Channel Input (breakout box.) There are two input connectors for each 8-Channel Input. They connect the input signal to the VT1432A. Each connector carries four channels. Depending on options, there can be 1, 2 or 4 input connectors (4 - 16 channels).

Updating the arbitrary source firmware

When updated firmware for the arbitrary source is available, the ROM in the VT1432A can be updated by using the procedure documented in /usr/e1432/arbsrc/README.

7

The Tachometer Option (VT1432A-AYF)

Tachometer Description

A tachometer input can be included with the VT1432A 16 Channel 51.2 kSamples/s Digitizer plus DSP as Option VT1432A-AYF. (It cannot be installed with a Source, Option VT1432A-1D4.) The Tachometer Option is a two channel tachometer input used to capture the contents of a freerun counter whenever an external input crosses a programmable threshold.

Tachometer Inputs

The tachometer has two inputs which connect to analog conditioning, holdoff and FIFO circuitry. See the block diagram in this chapter. The inputs can be configured so that one input connector (Tach 2) becomes an external trigger input and the other (Tach 1) remains a tachometer input. (The Tach 1 connector cannot be a trigger input.) The switch that determines this configuration is controlled by software.

External Trigger Input

A VT1432A without a tachometer option can accept a TTL external trigger signal (see “Trigger Lines” in the chapter titled “Module Description”). With the tachometer option the VT1432A still has that capability and is also able to accept an analog external trigger signal at the Tach 2 input.

Trigger Level

The trigger level of the tachometer can be set by software.

Tachometer Monitoring

The tachometer is capable of sending its analog input signal onto the VT1432A module’s internal calibration line. The calibration line can be connected to the 51.2 kHz 4-channel input assembly, so that the signal on the tachometer’s connector can be monitored via an input channel. This can be useful when deciding where to set the trigger level of the tachometer. An example program is supplied with the VT1432A Host Interface library, which shows how to perform this tachometer monitoring.

Exact RPM Triggering

The tachometer can be used to create exact RPM triggering, controlled by software. The RPM of the tach channel is calculated from tach transition times. Then the sample numbers in the data FIFO are determined for exact RPM triggering.

Input Count Division

The tachometer can be programmed to divide the input signal. For example if a signal is coming in at 100 counts per second, the tachometer can be set to look at only every 10th count for a result of 10 counts per second.

Holdoff Time

The tachometer can be programmed to wait for a specified period of time between counts that it will detect. After a count is detected, subsequent counts will be ignored until the holdoff time has passed.

Block Diagram

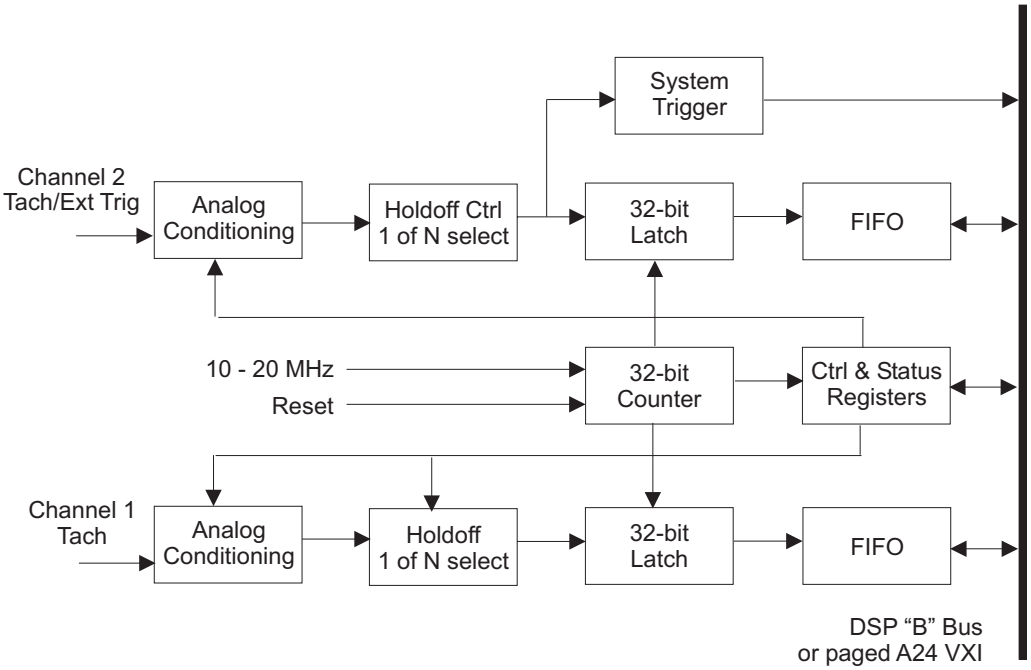


Figure 7-10: Tachometer option block diagram

The Tachometer Option Front Panel

The VT1432A with the Tachometer Option may have 4, 8 and 16 input channels. The following illustration shows a front panel for 16 channels. The LEDs and connectors are described on the next page.

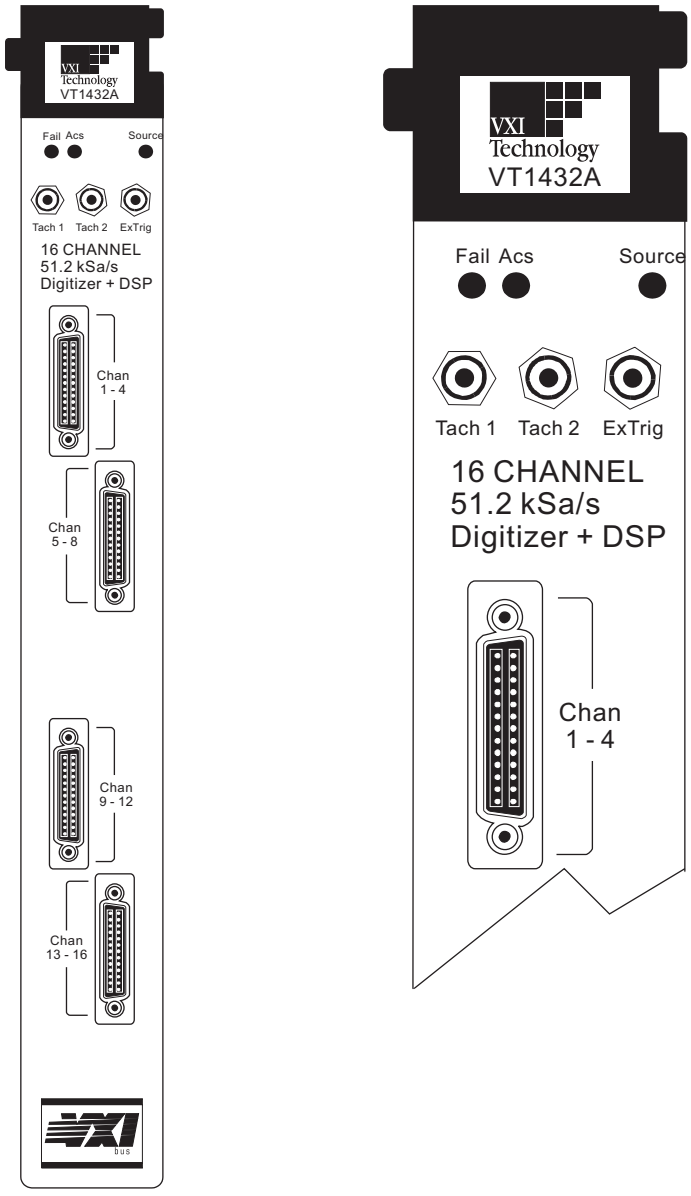


Figure 7-11: VT1432A with tachometer - front panel

LEDs and Connectors for the Tachometer Option.

Status LEDs

- Fail: This is the standard VXI “Failed” indicator. It lights briefly when powering up and normally goes out after a few seconds. If it stays on it indicates a hardware failure in the module.
- Acs: This is the standard VXI “Access” indicator. When it is on, it indicates that another device on the bus is contacting the module, for example to transfer data or read registers.
- Trigger: This LED flashes on each time an edge is detected on the tachometer signal, so when it is blinking it indicates that the tachometer signal is on. (For a VT1432A that does not have the Tachometer option, this LED is defined differently.)

SMB Connectors

- Tach1: This is one of the two tachometer inputs. Tach1 cannot be configured as an external trigger.
- Tach2: This is the second of the two tachometer inputs. Tach2 can also be configured (via software) to be an external trigger input
- ExTrig: This allows for an external trigger input to the VT1432A. The input signal must be TTL, other characteristics can be defined in software. ExTrig can be enabled or disabled in software.

Input Connectors (1, 2 or 4)

These connectors are attached to the cables from an 8-Channel Input (breakout box) — two input connectors for each 8-Channel Input). They connect the input signal to the VT1432A. Each connector carries four channels. Depending on options, there can be 1, 2 or 4 input connectors (4 - 16 channels).

8

Break Out Boxes

Introduction

A Break Out Box connects the VT1432A or VT1433B to a set of connectors to receive input signals.

Several types of Break Out Boxes are available. This chapter covers:

- VT3240A Voltage Break Out Box
- VT3241A ICP® Break Out Box

Other Break Out Boxes include the VT3242A Charge Break Out Box and the VT3243A Microphone Break Out Box. See the documentation supplied with those products for more information.

Service

For service on the Break Out Boxes contact the nearest VXI Technology Customer Support center.

The VT3240A and VT3241A Break Out Boxes

Each of the Break Out Boxes described in this section has eight BNC connectors for input. They each have two cables which connect to the sub-miniature “D” connectors on the front panel of the VT1432A/33B. Each of the two cables carries four channels. For a 4-channel VT1432A or VT1433B, one Break Out Box is used but only one of its cables is used; and only connectors 1-4 are used (or connectors 5-8, depending on which cable is used). For a 16-channel VT1432A, two Break Out Boxes are used.

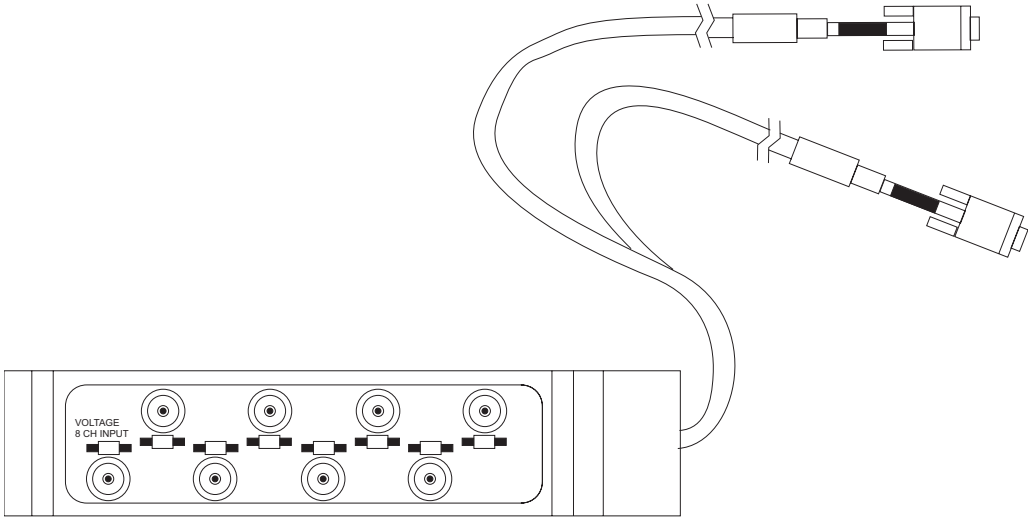


Figure 8-1:VT3240A Voltage Break Out Box

VT3240A Voltage-type Break Out Box

In this type of Break Out Box the signal is sent straight through to the sub-miniature “D” connectors on the VT1432A/33B.

VT3241A ICP®-type Break Out Box

Each of the eight connectors in this type of Break Out Box is connected to an independent, floating current source. These are intended to power integrated-circuit piezo-electric (ICP®) transducers. They supply 4.5 mA (nominal) at up to 28 volts. The current sources are controllable by software in groups of four. That is, the current sources for connectors 1-4 can be turned on or off as a group, as can the current sources for connectors 5-8.

Break Out Box Grounding

Each connector on the VT3240A and VT3241A Break Out Box has a small manual switch next to it. When this switch is in the “GND” position the outer shell of the connector is grounded to the chassis ground of the VXI mainframe. When it is in the “DIFF” position it is not grounded to the mainframe and will float if not grounded elsewhere in the system (such as at the sensor). The connector shell should not be allowed to float: if the switch is in the “DIFF” position the shell should be grounded elsewhere in the system.

Break Out Box Cables

Making a Custom Break Out Box Cable

A cable to connect the Break Out Box with the VT1432A/33B is supplied with the each of the Break Out Boxes described in this chapter. However, this section is included for those users who may want to make their own connecting cable. The drawing on this page shows the AMP part numbers for the parts needed to make the plug end of the cable. This illustration shows a VT3240A Voltage Break Out Box, a VT3242A Break Out Box requires a single cable with connectors at both ends.

The next page shows the pinout for the connector.

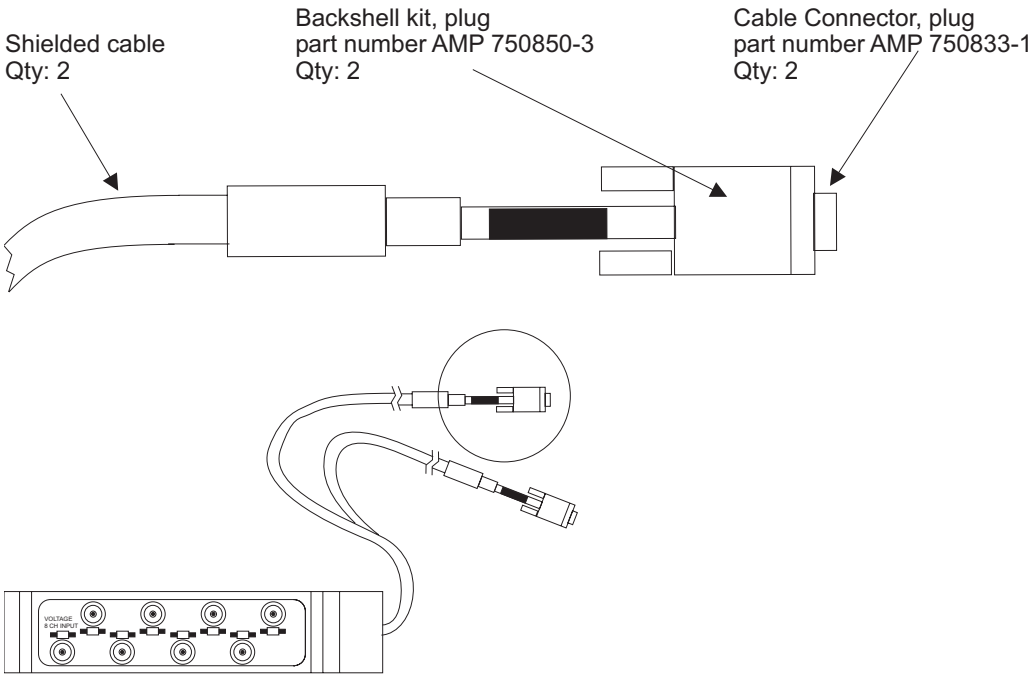
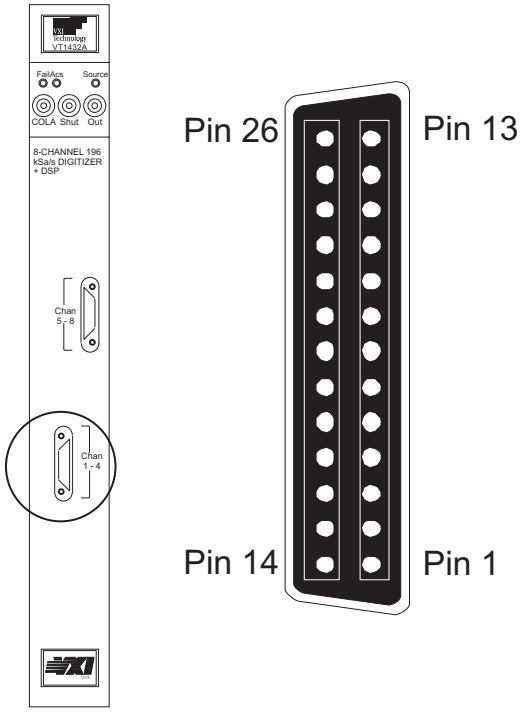


Figure 8-2: Break Out Box cable and part numbers



Pin definitions for input connector

definition	pin #	pin #	definition
RFI GND/Cable Shield	26	13	- Diff 1
+24 V Power	25	12	+Diff 1
GND Return for ±24 V	24	11	RFI GND/Drain Shield 1
-24 Power	23	10	RFI GND/Drain Shield 2
RFI GND	22	9	- Diff 2
I2C SCL	21	8	+ Diff 2
CAL HIGH	20	7	CAL LOW
BoB_EN	19	6	- Diff 3
RFI GND	18	5	+ Diff 3
I2C SDA	17	4	RFI GND/Drain Shield 3
RFI GND	16	3	RFI GND/Drain Shield 4
I2C_EN	15	2	- Diff 4
RFI GND/Cable Shield	14	1	+ Diff 4

Recommendations on wiring for the VT1432A/33B 4 Channel Input Connector

Allowed Connections

Differential Input Channels
Connect at VT1432A/33B end of cabling and at DUT
Recommended: shielded twisted pair

1	+ Diff 4
2	- Diff 4
5	+ Diff 3
6	- Diff 3
8	+ Diff 2
9	- Diff 2
12	+ Diff 1
13	- Diff 1

Input Channel Shielding
Connect at VT1432A/33B end of cabling ONLY

3	RFI GND/Drain Shield 4
4	RFI GND/Drain Shield 3
10	RFI GND/Drain shield 2
11	RFI GND/Drain Shield 1

Additional shielding of entire cable
GND for grounded measurements if required

14	RFI GND/Cable Shield
26	RFI GND/Cable Shield

Dis-allowed Connections

Do NOT connect these pins on VT1432A/33B end of cabling. These signals and supplies are provided for VXI Technology specified break out boxes and are unspecified for other usage.

Do not use:	
15	I2C_EN
17	I2C_SDA
21	I2C_SCL
16	RFI GND/I2C Shield
18	RFI GND/I2C Shield
22	RFI GND/I2C Shield
19	BOB_EN
7	CAL_LOW
20	CAL_HIGH
23	±24 V Power
24	±24 V GND Return
25	+24 V Power

In general:

- ± DIFF n lines are the differential inputs for each channel. Shielded twisted-pair is recommended.
- RFI GND/Drain Shield n are the grounds for the shield on the twisted-pair for each input channel. Connect at the VT1432A/33B end of the cable only.
- RFI GND/Cable Shield are the grounds for a shield around the entire cable and the ground points for making individual channels single-ended.
- I2C_xxx supply control signals to the active break out boxes. Support for other usage is not provided. These are not used with the VT3240/1A Voltage and Voltage/ICP® break out boxes.
- RFI GND/I2C Shield protects the analog input lines.
- BOB_EN is another break out box control signal. Support for the usage of these break out boxes is restricted to those which are VXI Technology-specified.
- CAL_HIGH/LOW are signal lines to send calibration signals to the VXI Technology-specified break out boxes. The signals available on these lines are not specified and their usage is discouraged.
- ±24 V Power and GND supply power to the signal conditioning circuitry in the active break out boxes and ICP® in the active ICP® break out box. The power available on these lines is not specified and their usage is discouraged.

9

Troubleshooting the VT1432A

Diagnostics

The following describes a limited diagnostic program for the VT1432A, VT1433B, and VT1434A. It is to be run from an HP-UX host. The program is called “hostdiag.exe.” It can be found with the VT1432A Host Interface Software Library at location C:\VXIPNP\winNT\HPE1432\bin.

location: /usr/e1432/bin

Usage: hostdiag [-hPsuvV] [-f file] [-L laddr] [-S slot] [-O list]

-h
Does a quick, partial test by bypassing the tests which involve downloading code to the module.

-f file
Uses “file” as the source of code to download to the module instead of the default sema.bin.

-L logical_addr
Specifies the logical address of the module to be tested. The default value is 8.

-O option_list
Tests the module against a list model/options. For example -O “E1432,1DE,VT1432A-AYF” tests the module as an 8 channel VT1432A with the tachometer option. Without this option, hostdiag only tests what it finds present. Hardware which has failed in such a way that it appears to be absent will not be detected without this option.

-P
Prints only a pass/fail message - no diagnostic printouts.

-s
Additionally runs the “standard input/output” tests. Sources finish testing with 1 V_{PK}, 1 kHz sine on each output for manual verification of output functionality. Input testing (both VT1432A and VT1433B inputs and the Tachometer input) assumes 1 V_{PK}, 1 kHz sine input on each channel. This allows testing of additional portions of the signal path which inaccessible from the internal tests.

-S vxi_slot
Test the module in the vxi slot, vxi_slot. Default is to test the module at logical address 8.

-u
Display usage message.

-v
Specifies the verbose printing. Normally, hostdiag does not print anything unless an error is found. With this option, hostdiag prints status messages as it operates. This option also enables additional diagnostic information which is not generally useful.

-V
Print version info.

Hostdiag returns 0 upon success or returns non-zero if an error is detected.

Coverage:

- Main board
- DRAM SIMMs
- Input SCAs (Signal Conditioning Assemblies)
- Source SCAs (VT1434A)
- Optional source
- Optional tachometer (VT1432A and VT1433B)

Notes:

- Tests are somewhat limited but will catch many hardware errors
- No errors printed means that all tests passed

10

Replacing Assemblies

Replaceable Parts

For information on upgrading the VT1432A or replacing parts, contact VXI Technology Customer Support Services. See the Support Resources section of the preface of this manual for contact information.

Replacement parts are listed in the following tables:

- Assemblies: without option VT1432A-AYF or VT1432A-1D4
- Assemblies: with option VT1432A-AYF
- Assemblies: with option VT1432A-1D4
- Cables: without option VT1432A-AYF or VT1432A-1D4
- Cables: with option VT1432A-AYF
- Cables: with option VT1432A-1D4
- Front Panel

Ordering Information

To order a part listed in one of the tables, quote the part number (VTI Part Number), indicate the quantity required and address the order to the nearest VXI Technology sales and service office (see the inside back cover of this guide). The first time a part is listed in the table, the quantity column (Qty) lists the total quantity of the part used in the module. For the corresponding name and address of the manufacturer's codes shown in the tables, see "CAGE Code Numbers."

Caution The module is static sensitive. Use the appropriate precautions when removing, handling and installing to avoid unnecessary damage.

Direct Mail Order System

Within the U.S.A., VXI Technology can supply parts through a direct mail order system. Advantages of the Direct Mail Order System are:

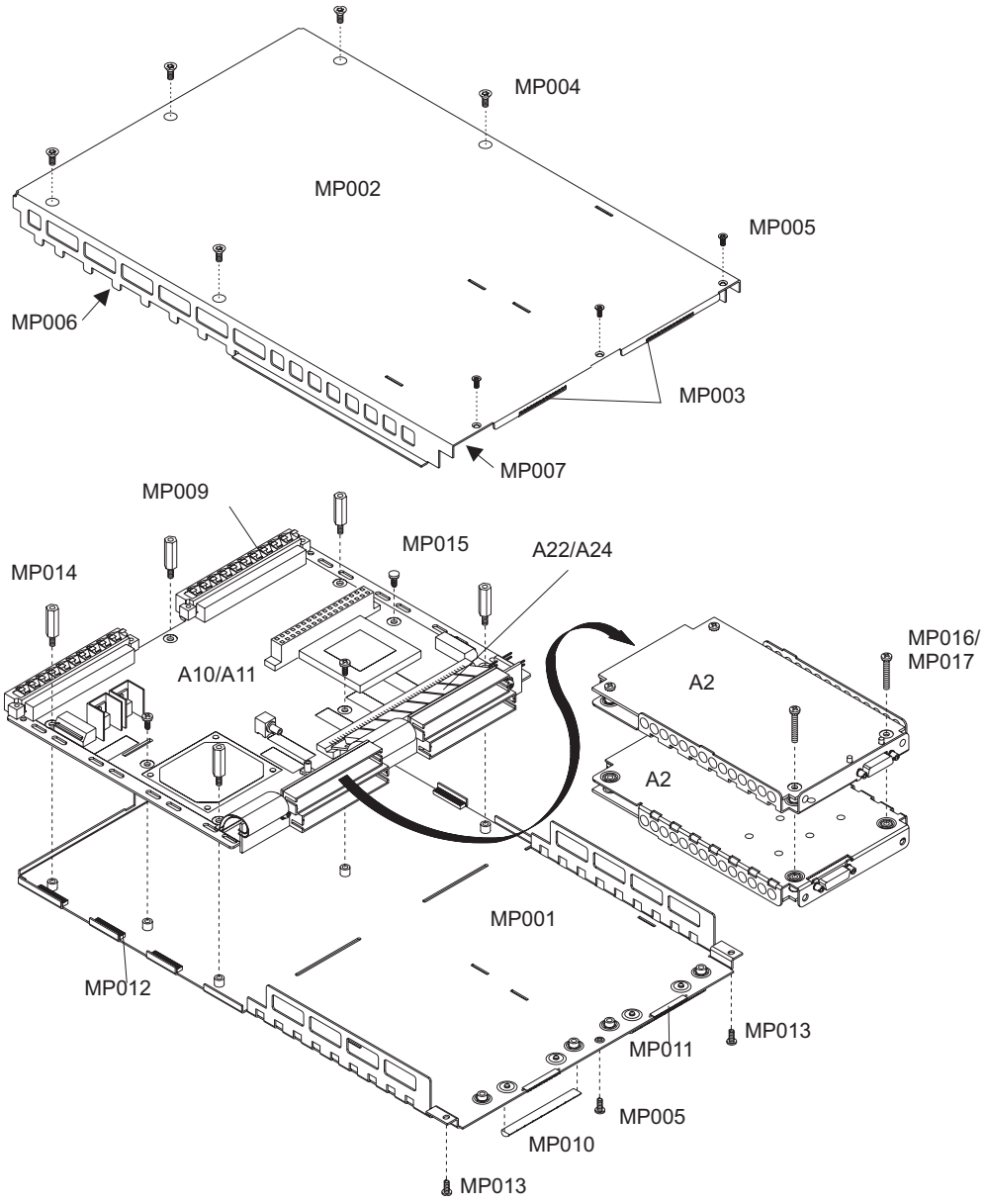
- Direct ordering and shipment from VXI Technology.
- No maximum or minimum on any mail order.
- Transportation charges are prepaid. A small handling charge is added to each order.
- No invoicing. A check or money order must accompany each order.
- Mail order forms and specific ordering information are available through VXI Technology, Inc. See "Need Assistance" chapter of this guide for a list of VXI Technology sales and service office locations and phone numbers.

CAGE Code Numbers

The following table provides the name and address for the manufacturers' CAGE code numbers (Mfr Code) listed in the replaceable parts tables.

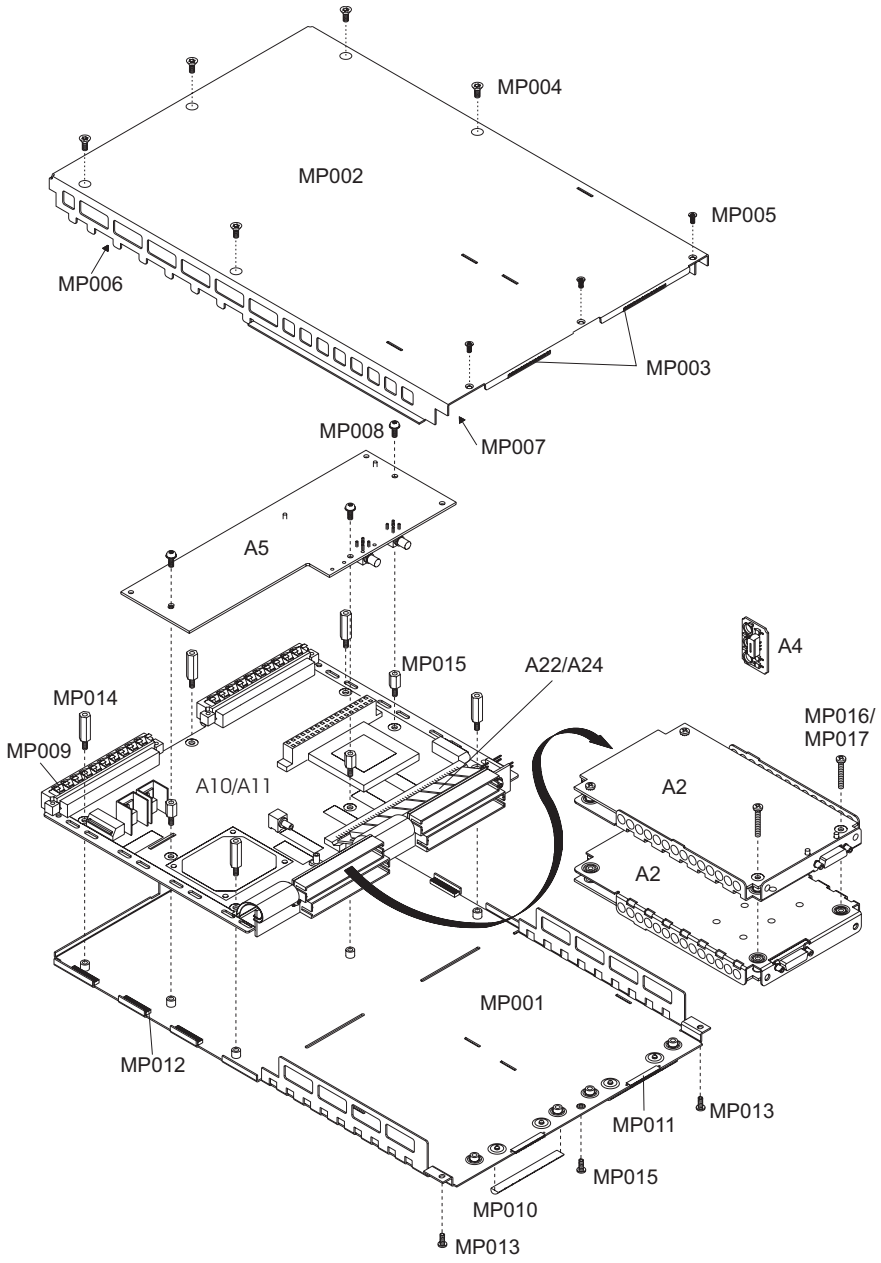
Mfr Code	Mfr Name	Address
03LB1	VXI Technology, Inc.	Irvine, CA 92614 U.S.A.
30817	Laird Technologies	Delaware Water Gap, PA 18327 U.S.A.

Assemblies: without option VT1432A-AYF or VT1432A-1D4



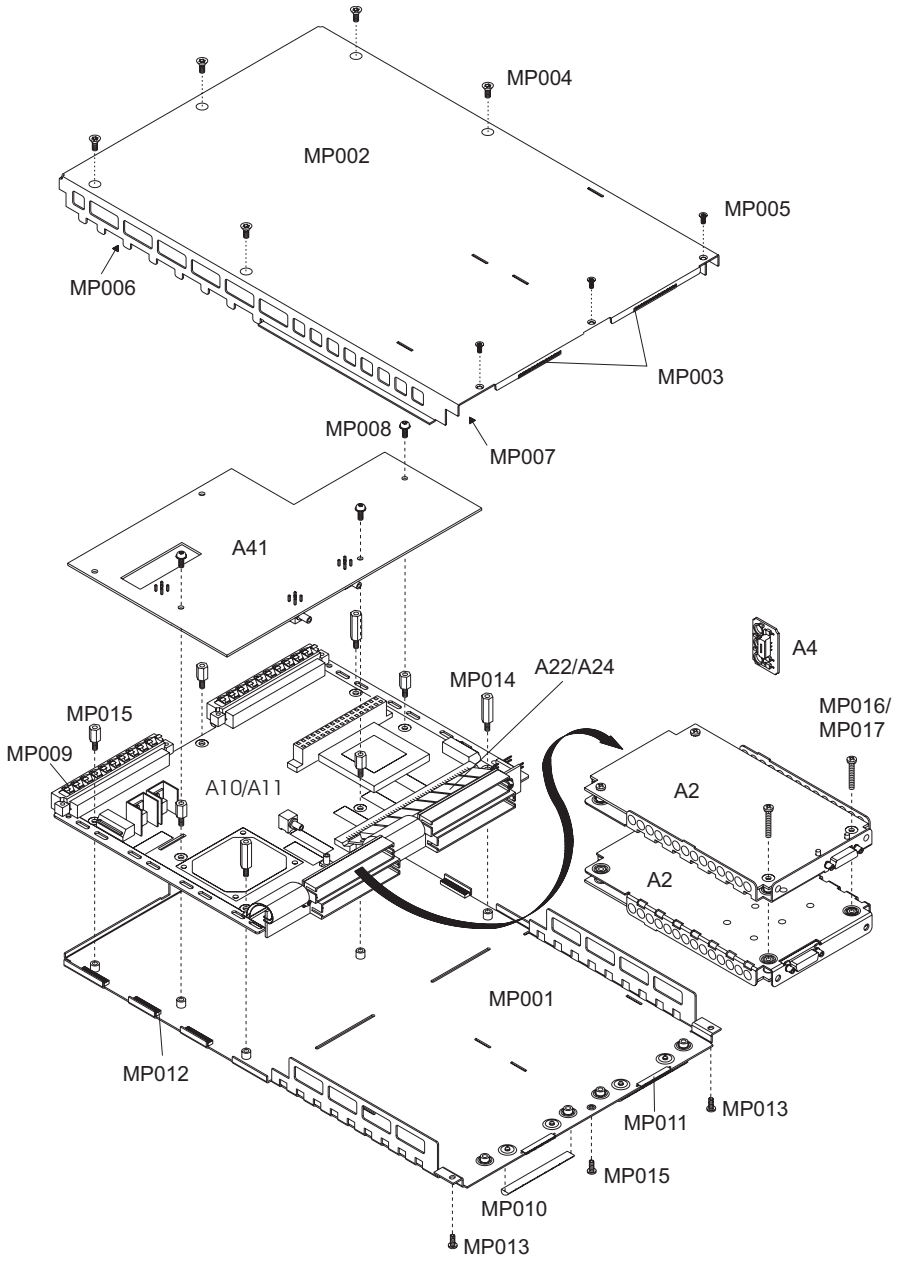
Ref Des	VTI Part Number	Qty	Description	Mfr Code	Mfr Part Number
A2	E1432-66502	4	PC ASSY-INPUT	03LB1	E1432-66502
A4	E1432-66504	1	PC ASSY-LED	03LB1	E1432-66504
A10	E1433-66510	1	PC ASSY-MAIN OPT UGV	03LB1	E1433-66510
A11	E1433-66511	1	PC ASSY-MAIN	03LB1	E1433-66511
A22	1818-5622	1	ICM DRAM, SIMM, 8x32	03LB1	1818-5622
A24	1818-5624	1	ICM DRAM, SIMM, 1x32	03LB1	1818-5624
MP001	41-0403-000	1	SHTF CVR-BTTM ALSK	03LB1	41-0403-000
MP002	41-0402-000	1	SHTF CVR-TOP	03LB1	41-0402-000
MP003	8160-0862	0	GSKT RFI STRIP FNGRS	30817	0097-553-17-020
MP004	0515-2033	5	SCR-MCH M3.0 10MMLG	03LB1	0515-2033
MP005	0515-2028	4	SCR-MCH M2.5 6MMLG	03LB1	0515-2028
MP006	E1432-44101	1	GSKT THERMAL CONDUCTOR	03LB1	E14320-44101
MP007	E1485-40601	1	GSKT-RFT, TOP CVR ADH SHT	03LB1	E1485-40601
MP008	0515-0372	3	SCR-MCH M3.0 8MMLG	03LB1	0515-0372
MP009	E1450-01202	4	STMP SHLD-RFI GRND	03LB1	E1450-01202
MP010	8160-0686	1	STMP FNGRS-RFI STRP BECU	30817	786-185
MP011	8160-0683	0	STMP STRP-SPNG FLTR GRD	30817	0097-551-17-X
MP012	8160-0869	6	GSKT RFI, 2MM X 4MM	03LB1	8160-0869
MP013	0515-0368	2	SCR-MCH M2.5 X 0.45	03LB1	0515-0368
MP014	0380-4042	5	STDF-HXMF M3.0 16.7MMLG	03LB1	0515-4042
MP016	0515-0664	2	SCR-MCH M3.0 12MMLG	03LB1	0515-0664
MP017	0515-0667	4	SCR-MCH M3.0 25MMLG	03LB1	0515-0667

Assemblies: with option VT1432A-AYF



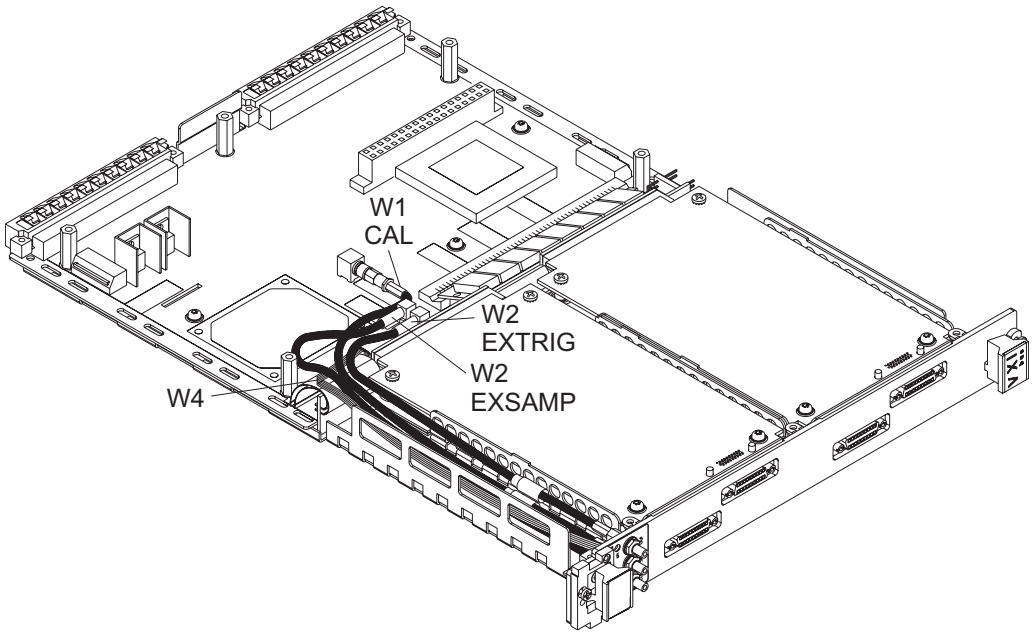
Ref Des	VTI Part Number	Qty	Description	Mfr Code	Mfr Part Number
A2	E1432-66502	4	PC ASSY-INPUT	03LB1	E1432-66502
A4	E1432-66504	1	PC ASSY-LED	03LB1	E1432-66504
A5	E1432-66505	1	PC ASSY-OPT AYF	03LB1	E1432-66505
A10	E1433-66510	1	PC ASSY-MAIN OPT UGV	03LB1	E1433-66510
A11	E1433-66511	1	PC ASSY-MAIN	03LB1	E1433-66511
A22	1818-5622	1	ICM DRAM, SIMM, 8x32	03LB1	1818-5622
A24	1818-5624	1	ICM DRAM, SIMM, 1x32	03LB1	1818-5624
MP001	41-0403-000	1	SHTF CVR-BTTM ALSK	03LB1	41-0403-000
MP002	41-0402-000	1	SHTF CVR-TOP	03LB1	41-0402-000
MP003	8160-0862	0	GSKT RFI STRIP FNGRS	30817	0097-553-17-020
MP004	0515-2033	5	SCR-MCH M3.0 10MMLG	03LB1	0515-2033
MP005	0515-2028	4	SCR-MCH M2.5 6MMLG	03LB1	0515-2028
MP006	E1432-44101	1	GSKT THERMAL CONDUCTOR	03LB1	E14320-44101
MP007	E1485-40601	1	GSKT-RFT, TOP CVR ADH SHT	03LB1	E1485-40601
MP008	0515-0372	3	SCR-MCH M3.0 8MMLG	03LB1	0515-0372
MP009	E1450-01202	4	STMP SHLD-RFI GRND	03LB1	E1450-01202
MP010	8160-0686	1	STMP FNGRS-RFI STRP BECU	30817	786-185
MP011	8160-0683	0	STMP STRP-SPNG FLTR GRD	30817	0097-551-17-X
MP012	8160-0869	6	GSKT RFI, 2MM X 4MM	03LB1	8160-0869
MP013	0515-0368	2	SCR-MCH M2.5 X 0.45	03LB1	0515-0368
MP014	0380-4042	5	STDF-HXMF M3.0 16.7MMLG	03LB1	0515-4042
MP015	0380-4041	3	STDF-HXMF M3.0	03LB1	0515-4041
MP016	0515-0664	2	SCR-MCH M3.0 12MMLG	03LB1	0515-0664
MP017	0515-0667	4	SCR-MCH M3.0 25MMLG	03LB1	0515-0667

Assemblies: with option VT1432A-1D4



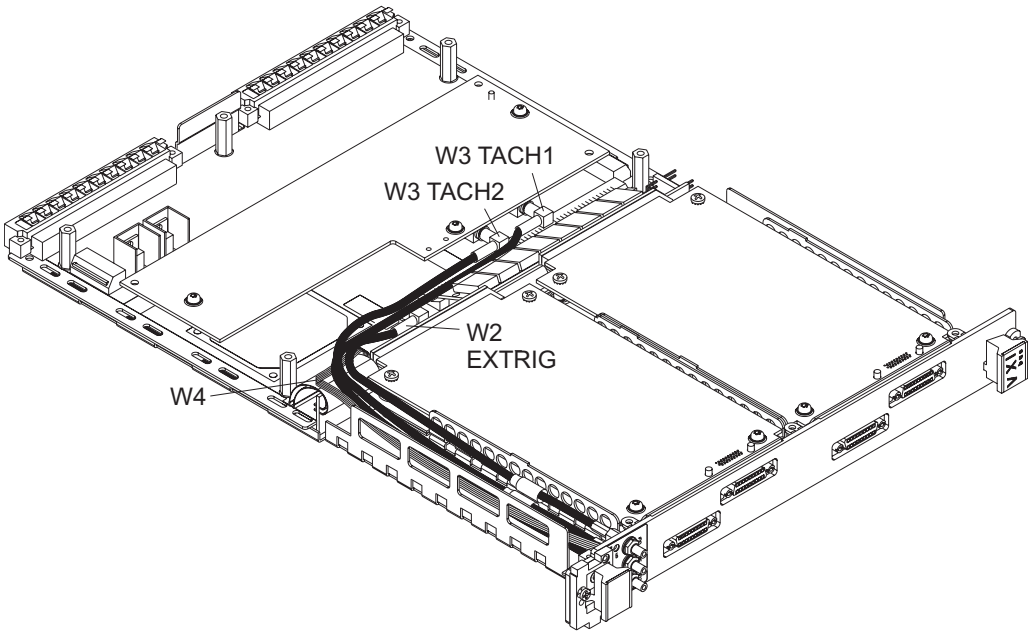
Ref Des	VTI Part Number	Qty	Description	Mfr Code	Mfr Part Number
A2	E1432-66502	4	PC ASSY-INPUT	03LB1	E1432-66502
A4	E1432-66504	1	PC ASSY-LED	03LB1	E1432-66504
A10	E1433-66510	1	PC ASSY-MAIN OPT UGV	03LB1	E1433-66510
A11	E1433-66511	1	PC ASSY-MAIN	03LB1	E1433-66511
A22	1818-5622	1	ICM DRAM, SIMM, 8x32	03LB1	1818-5622
A24	1818-5624	1	ICM DRAM, SIMM, 1x32	03LB1	1818-5624
A41	E1432-66541	1	PC ASSY-OPT 1D4	03LB1	E1432-66541
MP001	41-0403-000	1	SHTF CVR-BTTM ALSK	03LB1	41-0403-000
MP002	41-0402-000	1	SHTF CVR-TOP	03LB1	41-0402-000
MP003	8160-0862	0	GSKT RFI STRIP FNGRS	30817	0097-553-17-020
MP004	0515-2033	5	SCR-MCH M3.0 10MMLG	03LB1	0515-2033
MP005	0515-2028	4	SCR-MCH M2.5 6MMLG	03LB1	0515-2028
MP006	E1432-44101	1	GSKT THERMAL CONDUCTOR	03LB1	E14320-44101
MP007	E1485-40601	1	GSKT-RFT, TOP CVR ADH SHT	03LB1	E1485-40601
MP008	0515-0372	3	SCR-MCH M3.0 8MMLG	03LB1	0515-0372
MP009	E1450-01202	4	STMP SHLD-RFI GRND	03LB1	E1450-01202
MP010	8160-0686	1	STMP FNGRS-RFI STRP BECU	30817	786-185
MP011	8160-0683	0	STMP STRP-SPNG FLTR GRD	30817	0097-551-17-X
MP012	8160-0869	6	GSKT RFI, 2MM X 4MM	03LB1	8160-0869
MP013	0515-0368	2	SCR-MCH M2.5 X 0.45	03LB1	0515-0368
MP014	0380-4042	3	STDF-HXMF M3.0 16.7MMLG	03LB1	0515-4042
MP015	0380-4041	5	STDF-HXME M3.0	03LB1	0515-4041
MP016	0515-0664	2	SCR-MCH M3.0 12MMLG	03LB1	0515-0664
MP017	0515-0667	4	SCR-MCH M3.0 25MMLG	03LB1	0515-0667

Cables: without option VT1432A-AYF or VT1432A-1D4



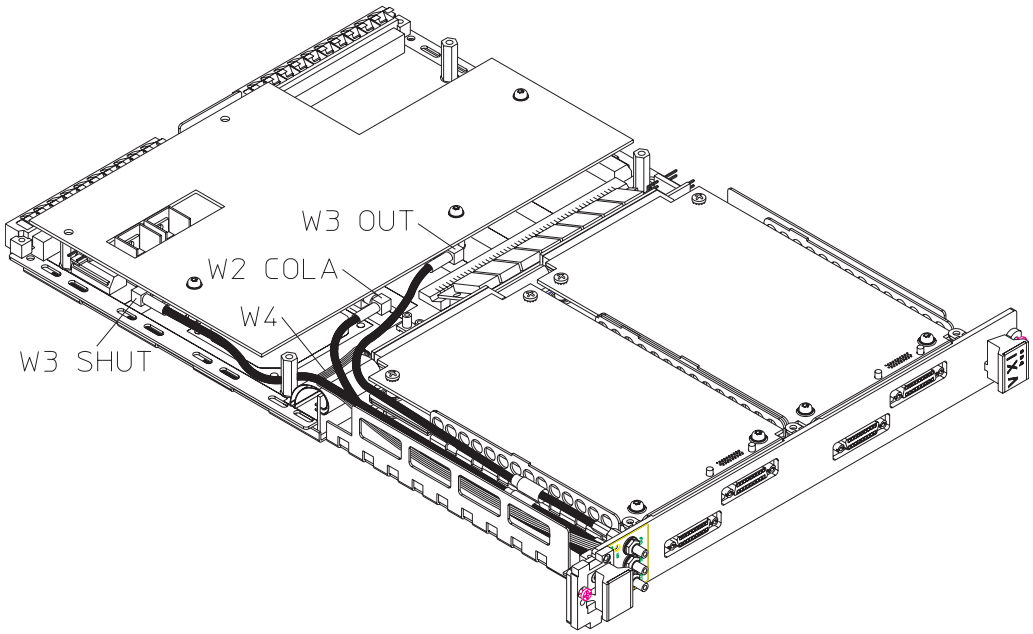
Ref Des	VTI Part Number	Qty	Description	Mfr Code	Mfr Part Number
W1	8120-6767	1	CBL-ASM CXL, 290MM	03LB1	8120-6767
W2	8120-6765	2	CBL-ASM CXL, 255MM	03LB1	8120-6765
W4	8120-6762	1	CBL-FLEX, 5-COND, 225MML	03LB1	8120-6762

Cables: with option VT1432A-AYF



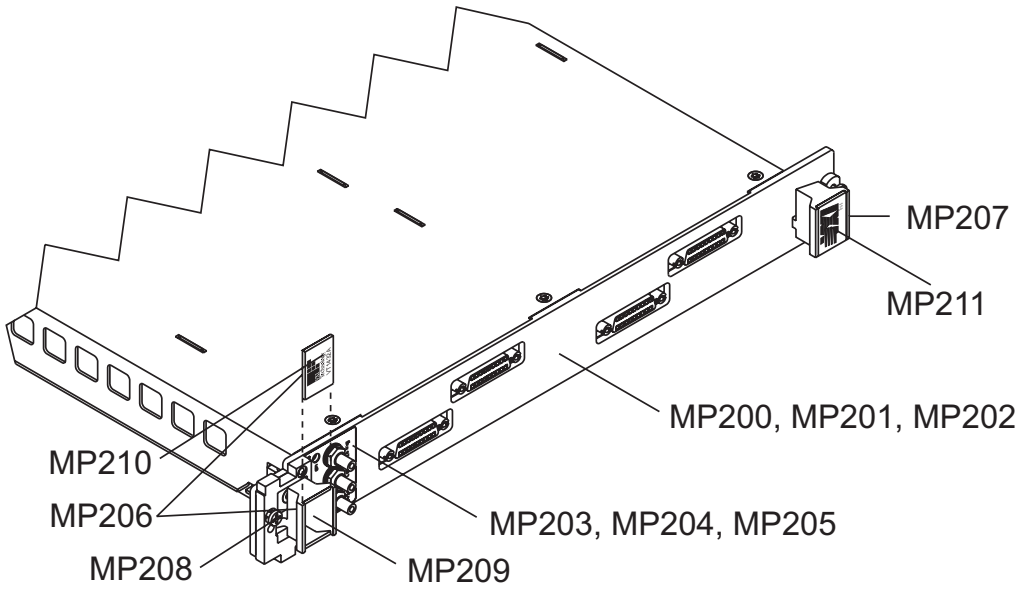
Ref Des	VTI Part Number	Qty	Description	Mfr Code	Mfr Part Number
W2	8120-6765	1	CBL-ASM CXL, 255MM	03LB1	8120-6765
W3	8120-6766	2	CBL-ASM CXL, 03LB1	03LB1	8120-6766
W4	8120-6762	1	CBL-FLEX, 5-COND, 225MML	03LB1	8120-6762

Cables: with option VT1432A-1D4



Ref Des	VTI Part Number	Qty	Description	Mfr Code	Mfr Part Number
W2	8120-6765	1	CBL-ASM CXL, 255MM	03LB1	8120-6765
W3	8120-6766	2	CBL-ASM CXL, 03LB1	03LB1	8120-6766
W4	8120-6762	1	CBL-FLEX, 5-COND, 225MML	03LB1	8120-6762

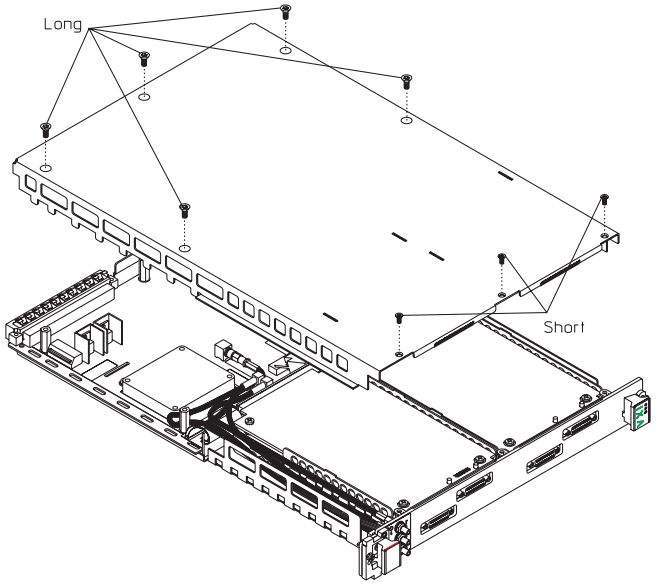
Front Panel



Ref Des	VTI Part Number	Qty	Description	Mfr Code	Mfr Part Number
MP200	E1432-00204	1	PNL-FRT, STANDARD	03LB1	E1432-00204
MP201	E1432-00202	1	PNL-FRT, OPT 1DE	03LB1	E1432-00202
MP202	E1432-00201	1	PNL-FRT, OPT 1DD	03LB1	E1432-00201
MP203	E1432-44301	1	LBL-FRT PNL SMB'S, STD	03LB1	E1432-44301
MP204	E1432-44302	1	LBL-FRT PNL SMB'S, OPT 1D4	03LB1	E1432-44302
MP205	E1432-44303	1	LBL-FRT PNL SMB'S, OPT AYF	03LB1	E1432-44303
MP206	E1400-84106	1	MOLD KIT-TOP EXTR HNDL	03LB1	E1400-84106
MP207	E1400-84105	1	MOLD KIT-BTTM EXTR HNDL	03LB1	E1400-84105
MP208	0515-1968	2	SCR-MCH M2.5 6MMLG	03LB1	0515-1968
MP209	0515-1375	2	SCR-MCH M2.5 6MMLG	03LB1	0515-1375
MP210	43-0016-003	1	LABEL, VXI EXT, VXI TECH, NEW LOGO	03LB1	43-0016-003
MP211	43-0016-002	1	LABEL, VXI EXT, VXIBUS	03LB1	43-0016-002

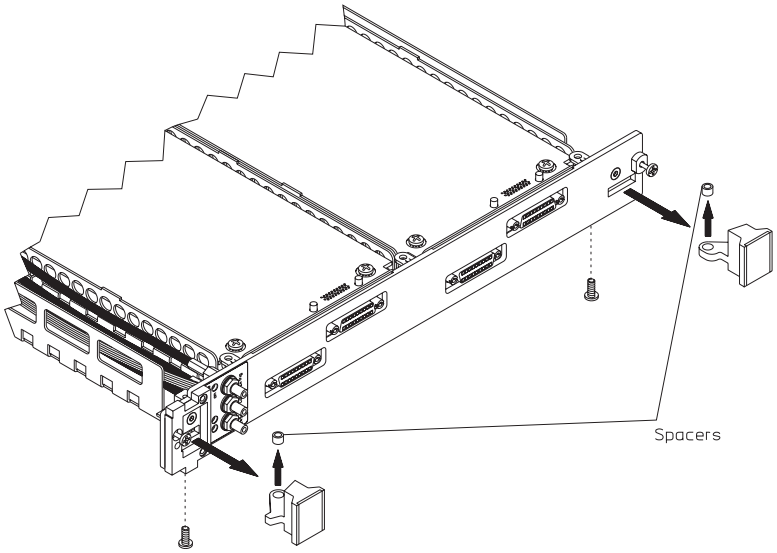
To remove the top cover

1 Remove the five long screws using a T-10 Torx driver and remove the three short screws using a T-8 Torx driver. Lift cover off.

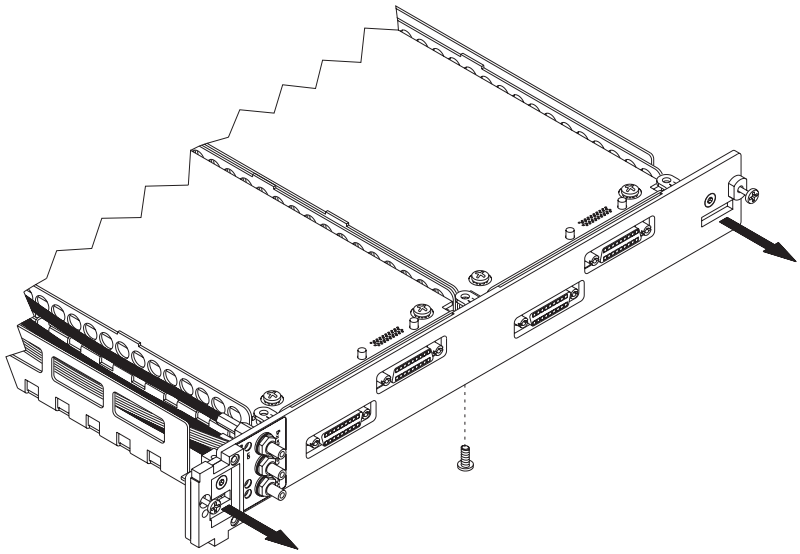


To remove the front panel

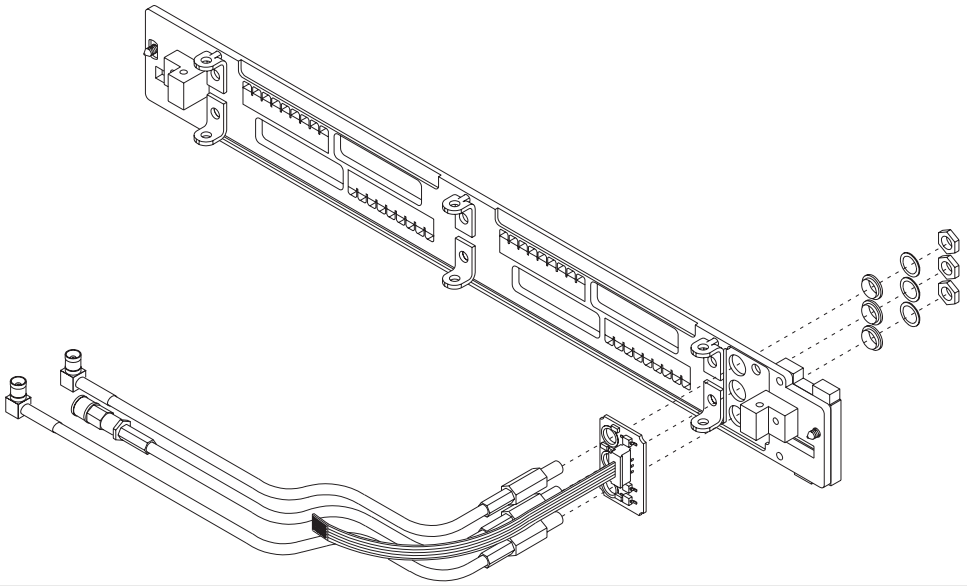
1 Remove top cover, see “To remove the top cover.” Gently disconnect cables from the printed circuit assemblies. Using a T-8 Torx driver, remove the two screws that attach the handles to the assembly. Pull out the handles making sure not to lose the two spacers.



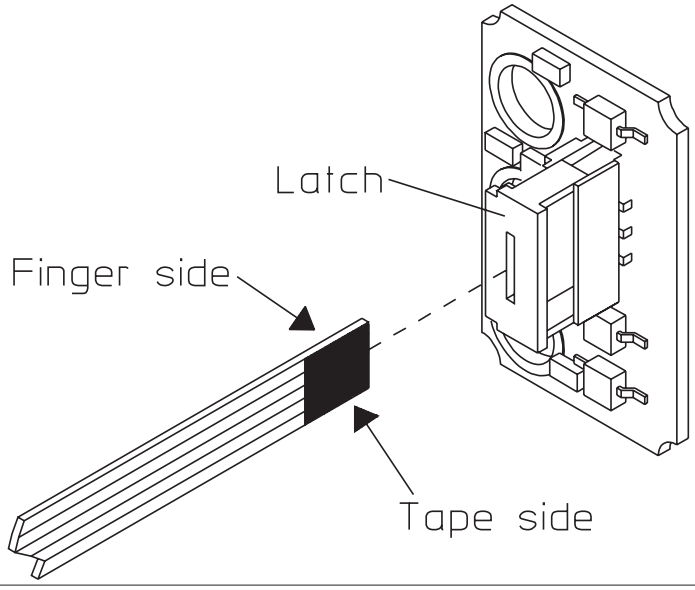
2 Using a T-8 Torx driver, remove the screw that attaches the front panel to the bottom cover. Gently pull the front panel off.



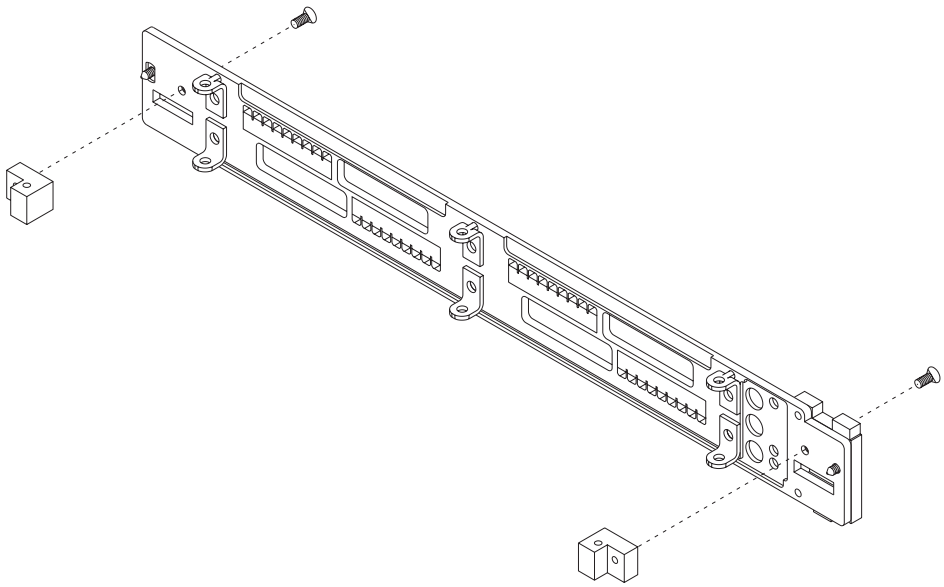
3 Remove the nuts that fasten the cables and assembly to the front panel. Using a 1/4-inch nut driver.



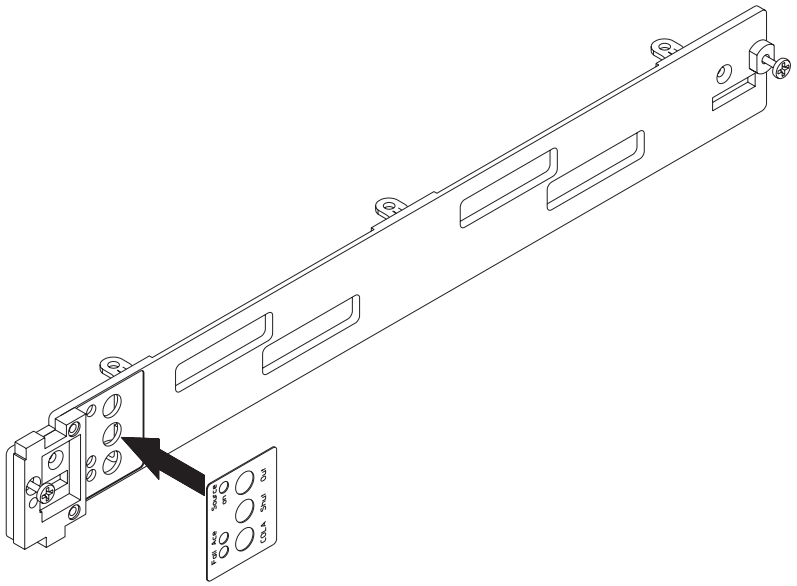
4 Remove ribbon cable from the A4 assembly, by pulling back the latch on the connector and removing cable. Be sure to note the orientation of the cable.

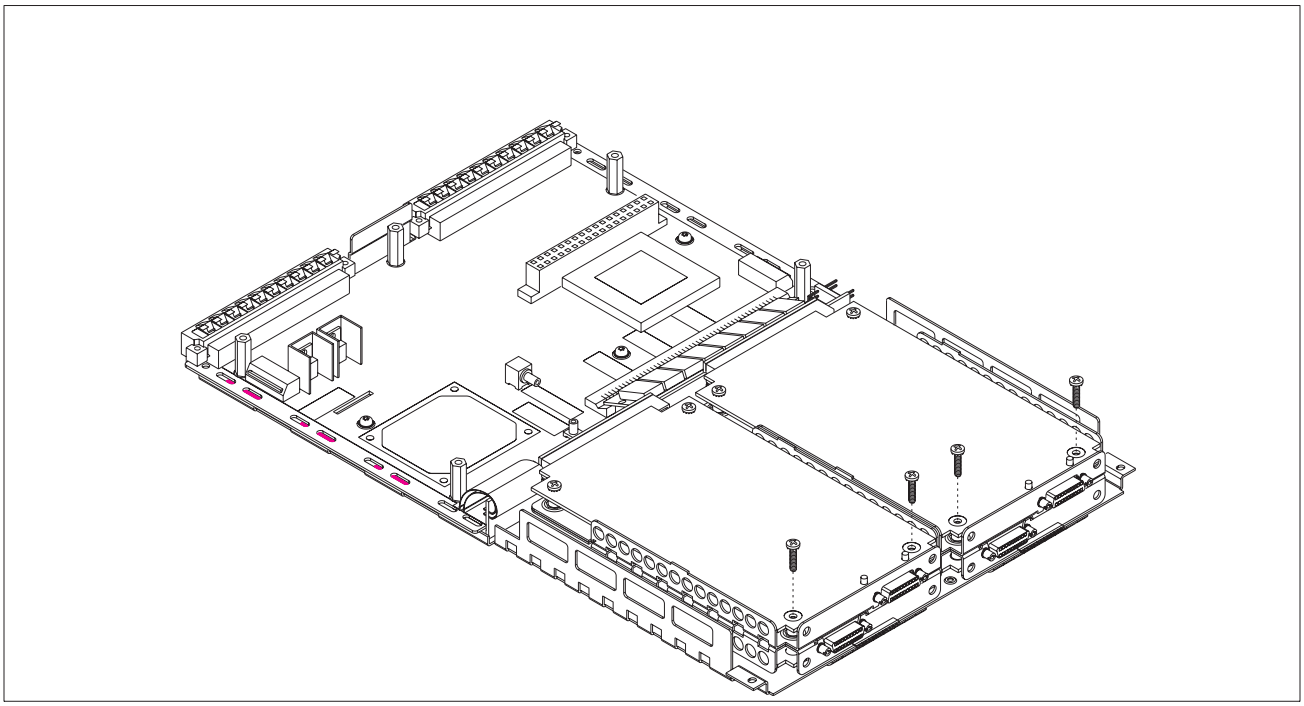


5 To replace the front panel with another that does not have its own side brackets, remove the brackets from the old front panel using a T-8 Torx driver. Be sure to note the positioning of the brackets, alignment is critical.

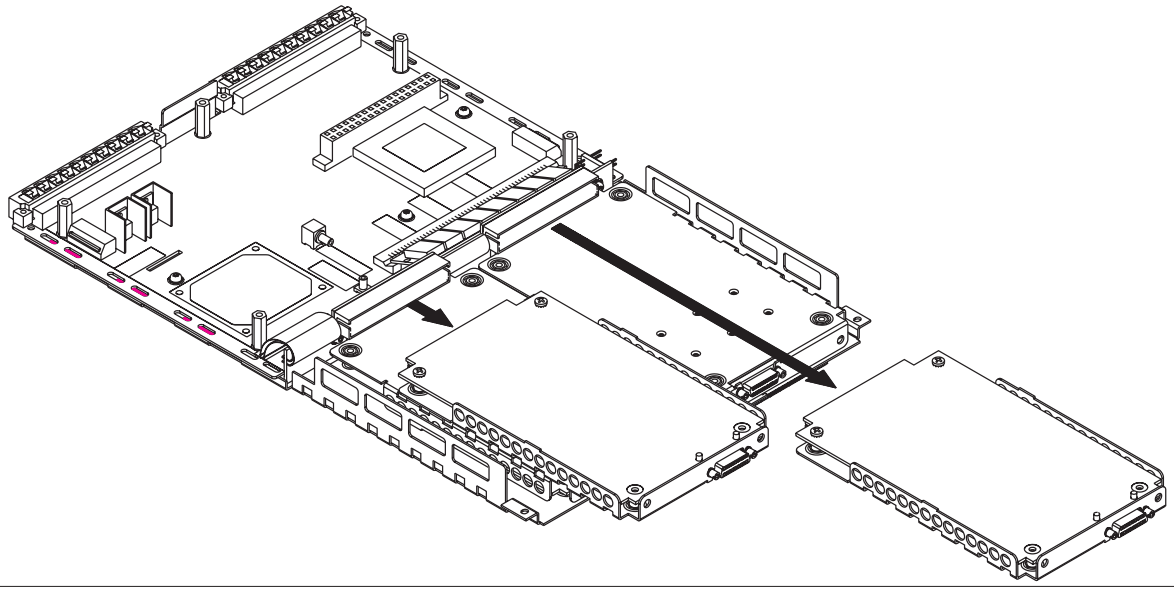


6 To replace the front panel with another that does not have the label already attached, remove the tape backing and place it on the front panel as shown.

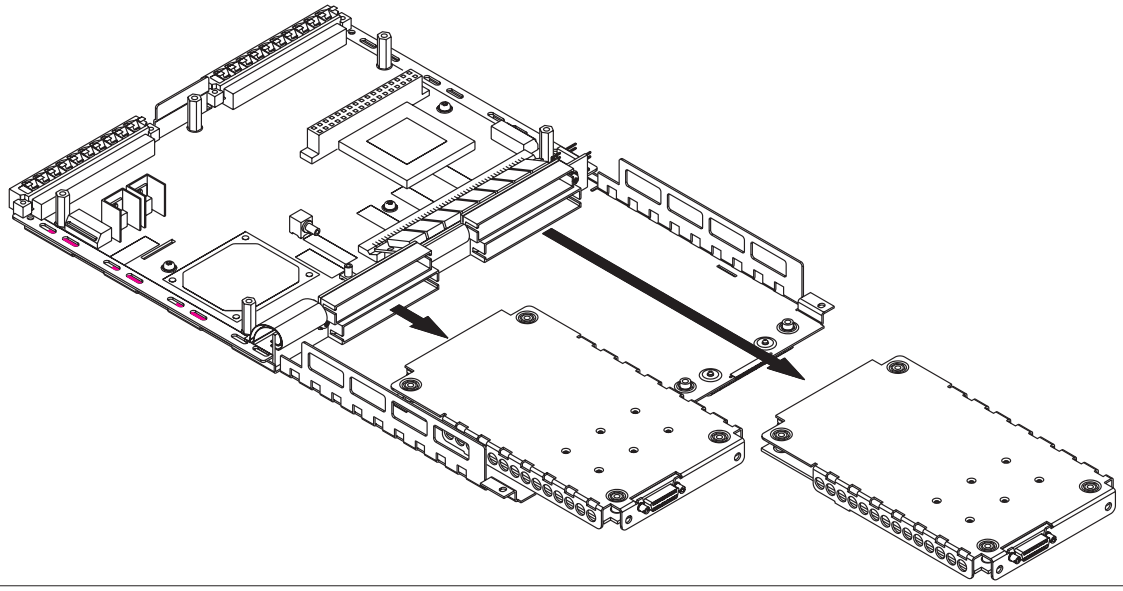




3 Remove the top two assemblies by gently pulling them forward, releasing them from the connectors.



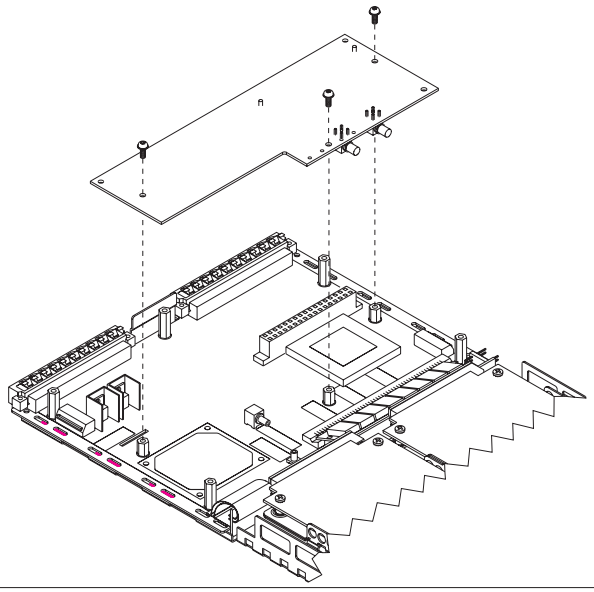
4 Remove the remaining input assemblies.



To remove the option VT1432A-AYF assembly

1 Remove the top cover, see “To remove the top cover.” Disconnect the two cables leading to the A5 assembly and move cables aside.

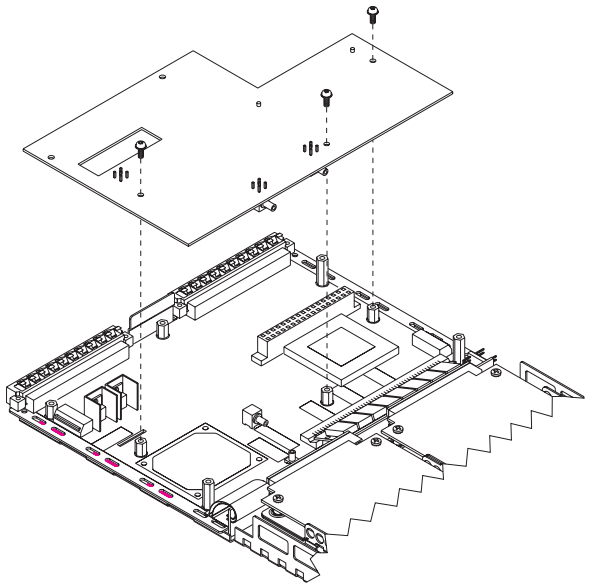
2 Using a T-10 Torx driver, remove the three screws that attach the assembly to the VT1432A and lift the assembly off.



To remove the option VT1432A-1D4 assembly

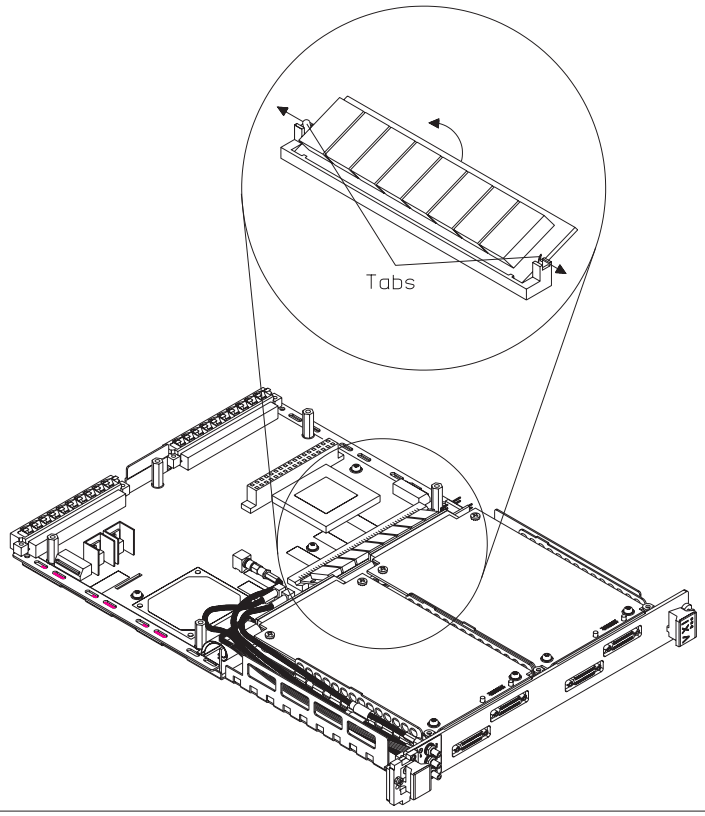
1 Remove the top cover, see “To remove the top cover.” Disconnect the three cables leading to the A41 assembly and move cables aside.

2 Using a T-10 Torx driver, remove the three screws that attach the assembly to the VT1432A and lift the assembly off.



To remove the A22/A24 assembly

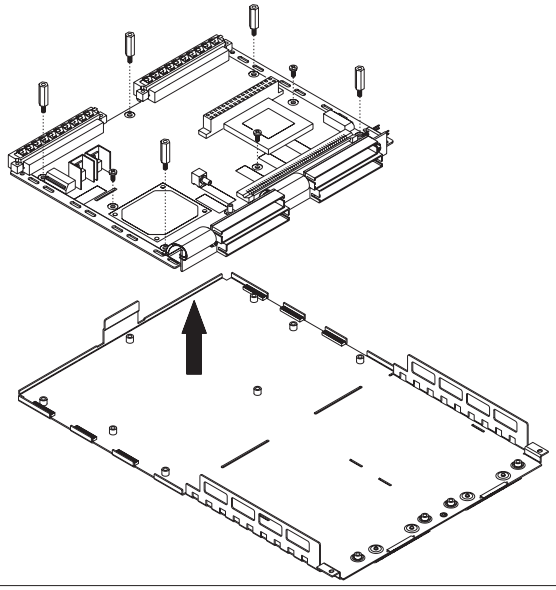
1 Remove the top cover, see “To remove the top cover.” Gently push the silver tabs outward and tilt the A22/A24 assembly forward releasing it from the connector.



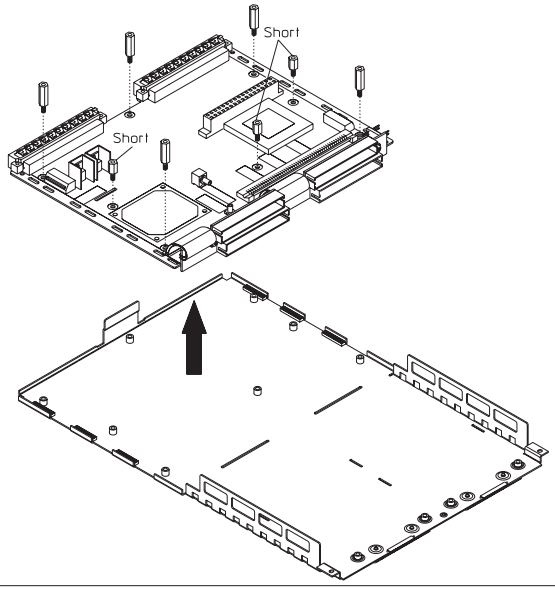
To remove the A1/A11 assembly

1 Remove top cover and input assemblies. See “To remove the top cover,” and “To remove the input assemblies.”

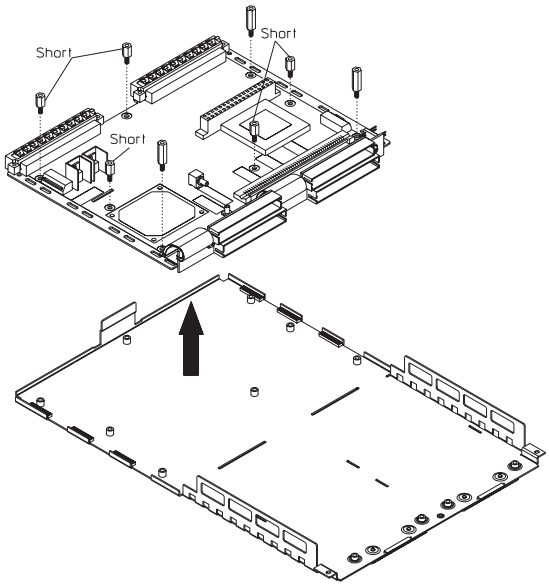
2A *If the VT1432A does NOT have option VT1432A-AYF or option VT1432A-1D4, do the following:* Remove the five standoffs using a 1/4-inch nut driver and remove the three screws using a T-8 torx driver.



2B *If the VT1432A has option VT1432A-AYF, do the following:* Remove the VT1432A-AYF option assembly, see "To remove the option VT1432A-AYF assembly." Remove the five long and the three short standoffs using a 1/4-inch nut driver.



2C *If the VT1432A has option VT1432A-1D4, do the following:* Remove the VT1432A-1D4 option assembly, see "To remove the option VT1432A-1D4 assembly." Remove the three long and the five short standoffs using a 1/4-inch nut driver.



11

Backdating

Backdating

This chapter documents modules that differ from those currently being produced. With the information provided in this chapter, this guide can be modified so that it applies to any earlier version or configuration of the module.

Main PC assembly change

VT1432A's with option VT1432A-UGV (optional Local Bus) up through serial number US36470676 were built with main PC assembly A1 (part number E1432-66501). This was replaced by main PC assembly A10 (part number E1433-66510) starting with serial number US3647677. This new main PC assembly is backward compatible with older VT1432A's.

Appendix A

Register Definitions

The VT1432A VXI Registers

The VT1432A 16 Channel 51.2 kSamples/s Digitizer plus DSP is a register-based VXI device. Unlike message-based devices which use higher-level programming using ASCII characters, register-based devices are programmed at a very low level using binary information. The greatest advantage of this is speed. Register-based devices communicate at the level of direct hardware manipulation and this can lead to much greater system throughput.

Users do not need to access the registers in order to use the VT1432A. The VT1432A's functions can be more easily accessed using the VT1432A Host Interface Library software. However, this chapter describing the registers is provided as supplemental information.

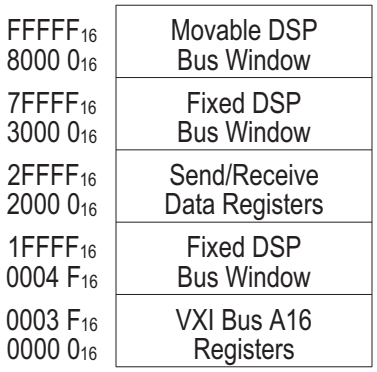
The A16 Registers

The following A16 registers are accessible at the base address defined by the device's logical address. The register at offsets 00_{16} to E_{16} are not accessible using longword (D32) accesses. The registers at offsets 10_{16} to $3E_{16}$ may be accessed by any of the D08(EO), D16 or D32 modes. All of these registers are also accessible at the device A24 base address.

Address	Read	Write
$3E_{16}$	Parameter 7 Register	
$3C_{16}$		
$3A_{16}$	Parameter 6 Register	
38_{16}		
36_{16}	Parameter 5 Register	
34_{16}		
32_{16}	Parameter 4 Register	
30_{16}		
$2E_{16}$	Parameter 3 Register	
$2C_{16}$		
$2A_{16}$	Parameter 2 Register	
28_{16}		
26_{16}	Parameter 1 Register	
24_{16}		
22_{16}	Query Response Register	Command Register
20_{16}		
$1E_{16}$	FIFO Count	
$1C_{16}$		
$1A_{16}$	Send Data	Receive Data
18_{16}		
16_{16}	RAM 1	
14_{16}		
12_{16}	RAM 0	
10_{16}		
$0E_{16}$	IRQ Status Register	IRQ Reset Register
$0C_{16}$	IRQ Config Register	
$0A_{16}$	Page Map Register	
08_{16}	Port Control Register	
06_{16}	Offset Register	
04_{16}	Status Register	Control Register
02_{16}	Device Type	
00_{16}	ID Register	Logical Address Register

The A24 Registers

The following A24 registers are accessible at the base address defined by the device's offset Register. The registers at offsets 0 to E_{16} are not accessible using longword (D32) accesses. The registers at offsets 10_{16} to $FFFFF_{16}$ may be accessed by any of the of the D08(EO), D16 or D32 modes.



The A24 registers are defined as follows:

- VXI Bus A16 Registers: These are the same registers accessed at the device's A16 base address.
- Fixed DSP Bus Window: Accesses to this region are mapped to the corresponding locations at the base of the internal DSP's memory map, also accessible through Page 0 of the moveable DSP bus window.
- Send/Receive Data Registers: Accesses to any address in this region will read/write the Send and Receive Data registers defined in the A16 register set. VME Bus D32 Block Transfers are supported for these addresses only.
- Movable DSP Bus Window: Accesses to this region are mapped (by the Page Map register) to different 512 kB regions of the internal DSP bus.

The VXI Bus Registers are defined as follows:

- ID Register: A read of this 16-bit register provides information about the device's configuration. Its value is always CFFF16 as defined in the following table.

Bit	15-14	13-12	11-0
Contents	11 (Register Based Device)	00 (A16/A24)	111111111111 (HP's ID)

- Logical Address Register: A write to this register changes the device's logical address according to the VXI Bus Dynamic Configuration protocol. Its format is defined in the following table.

Bit	15-8	7-0
Contents	No effect	Logical Address

- Device Type Register: A read of this register provides information about the device's configuration. Its format is defined in the following table.

Bit	15-12	11-0
Contents	0011 (1 MB of A24)	Model Code (201_{16} for VT1432A)

- Status Register: A read of this register provides information about the device's status as defined in the following table.

Bit	15	14	13-12	11	10	9	8
Contents	A24 Active	MODID*	Unused	Block Ready	Data Ready	ST Done	Loaded

Bit	7	6	5	4	3	2	1	0
Contents	Done	Err*	Unused	HW OK	Ready	Passed	Q Resp Ready	Cmd Ready

A24 Active: A one (1) in this field indicates that the A24 registers can be accessed. It reflects the state of the Control register's A24 Enable bit.

MODID*: A one (1) in this field indicates that the device is not selected via the P2 MODID line. A zero (0) indicates that the device is selected by a high state on the P2 MODID line.

Unused: A read of these bits will always return zero (0).

Block Ready: A one (1) indicates that there is a block of data available to be read from the Send Data registers. A zero (0) indicates that less than a full block is available.

Data Ready: A one (1) indicates that there is at least one word (32 bits) of data available in the Send Data register. A zero (0) indicates that there is not valid data in the Send Data register.

ST Done: A one (1) indicates that the internal DSP has competed and passed its self test.

Loaded: A one (1) indicates that the internal DSP has successfully booted and has loaded a valid model code.

Done: A zero (0) indicates that the on-card microprocessor has not finished processing the last command and the Err* bit is not valid. This bit is set and cleared by the DSP.

Err*: A zero (0) indicates that an error has occurred in communicating with the DSP (for example: invalid parameters). This bit is set and cleared by the DSP.

Ready: The meaning of this depends on the state of the Passed bit. While Passed is false, a one(1) indicates that the device is in the Config Reg Init state and the Model Code bits of the Device Type register are not valid, while a zero (0) indicates that the device is in either the self test or failed state. When Passed is true, a one (1) indicates that the DSP has finished its initialization and is ready for normal operation, while a zero (0) indicates that the device is in the passed state.

Passed: A zero (0) indicates that the device is in either the Hard Reset, Soft Reset, Config Reg Init, Failed or Init Failed state. A one (1) indicates that the device is in the passed state.

HW OK: A one (1) indicates that all the on-card FPGAs have successfully be initialized.

Q Resp Ready (Query Response Ready): A one (1) indicates that the Query Response Register is loaded and ready to be read. It is set by the DSP and cleared in hardware by a write to the Command Register.

Cmd Ready: A one (1) indicates that the command register and parameter register are available for writing. It is set by the DSP microprocessor and cleared in hardware by a write to the Command Register. This bit, when zero (0) additionally indicates that the Done bit is not valid.

- Control Register: A write to this register causes specific actions to be executed by the device. The actions are described in the following table.

Bit	15	14-2	1	0
Contents	A24/A32 Enable	Unused	Sysfail Inhibit	Reset

A24/A32 Enable: A one (1) in this field enables access to the device's A24 VME Bus registers. A zero (0) disables such access.

Sysfail Inhibit: A one (1) disables the device from driving the SYSFAIL* line.

Reset: A one (1) forces the device into a reset state.

- Offset Register: This read/write register defines the base address of the device's A24 registers. The four most significant bits of the Offset register are the values of the four most significant bits of the device's A24 register addresses. The 12 least significant bits of the Offset register are always zero (0). Thus, the Offset register bits 15-12 map the VME Bus address lines A23-A20 for A24 register accesses. A read of the Offset register always returns the address offset most recently written to the Offset register.
- Port Control Register: This register is used to override the Local Bus control of the device. (This applies to VT1432A modules that are equipped to use Local Bus). It has the following format:

Bit	15-2	1	0
Contents	Unused	LBus Pipe	LBus Enable

LBus Pipe: Writing a one (1) puts the Local Bus into pipeline mode, if the LBus Enable bit is also set. Writing a zero (0) allows the Local Bus to operate in some other mode.

LBus Enable: Writing a one (1) enables the Local Bus interface. Writing a zero (0) disables the local bus interface. RESET VALUE: 0

- Page Map Register: This read/write register defines the internal location of the movable window into the device's DSP bus. (This 512 kB window begins at 512 kB into the device's A24 registers.) The eight least significant bits of the Page Map register are the page number. These bits are mapped to the internal DSP bus address lines as follows:

  Bit 0: DSP A(17)
  Bit 1: DSP A(18)
  Bit 2: DSP A(19)
  Bit 3: DSP A(20)
  Bit 4: DSP A(21)
  Bit 5: DSP A(22)
  Bit 6: DSP A(30) and A(24)
  Bit 7: DSP A(31)

The eight most significant bits of the Page Map Register are always zero (0).

- IRQ Config Register: This register configures the first VME Bus interrupt source. It provides for selection of the VME Bus IRQ level used and a bit mask. It has the following format:

Bit	15-8	7-4	3	2-0
Contents	Mask	Unused	IRQ Enabled	IRQ Line

Mask: This is a bit mask used to enable up to eight interrupt causes. A bit value of zero (0) disables the corresponding interrupt source. RESET VALUE: 0

IRQ Enable: A one (1) in this bit enables the generation of IRQ's. A zero (0) resets each of the eight interrupt causes and status bits. RESET VALUE: 0

IRQ Line: This field select which VME Bus IRQ line is driven by this device. A value of zero (0) disconnect the interrupt source. RESET VALUE: 0

- IRQ Status Register: This read-only register indicates the reason for asserting the VME Bus interrupt. The format of the data is identical to that of the Status/ID word returned by an interrupt acknowledge (IACK) cycle. It differs from the IACK cycle in that the IACK cycle will clear the status bits and cause the de-assertion of the IRQ line. The register has the following format:

Bit	15-8	7-0
Contents	Status	Logical Address

Status: Each of these bits indicates the status of a cause of interrupt. A one (1) in a bit position indicates that the corresponding source is actively requesting and interrupt.

Logical Address: This is the device's current logical address.

- IRQ Reset Register: This register is used to resent the interrupt function. It has the following format:

Bit	15-8	7-0
Contents	Reset Bits	Unused

Reset Bits: Writing a one (1) to any of these bits will clear the corresponding bit in the IRQ status register . This will not disable subsequent interrupt generation. Clearing all of the IRQ status bits will cause the de-assertion of the IRQ line. Writing a zero (0) has no effect.

- Ram 0-1: These are 32-bit general purpose RAM locations which are also accessible to the on-board DSP. See the following section regarding D16/D08 access of 32-bit registers.
- Send Data Register: Reading this register gets the next available word from the measurement data FIFO. The measurement data FIFO is a 32-bit device. See the following section regarding D16/D08 access of 32-bit registers.
- Receive Data Register: Writing to this register puts a word into the source data FIFO. The source data FIFO is a 32-bit device. See the following section regarding D16/D08 access of 32-bit registers.
- Count Register: The Count register contains an unsigned 16-bit integer which is the number of 16-bit words of data which are currently available from the Send Data register or which the Receive Data register is currently ready to accept. While a device is generating or accepting data, the Count register may indicate fewer than the actual number of words available.
- Query Response/Command Register: This register is used to send commands to and receive responses from the device. It is implemented as a 32-bit RAM location. Writing the least significant byte (highest address) clears the Command/Parameter Ready and Query Response Ready bits in the status register and interrupts the on-board DSP. See the following section regarding D16/D08 access of 32-bit registers and the communication protocol.

- Parameter 1-7 Registers: These are 32-bit RAM locations used to pass parameters along with commands to the device or query responses from the device. See the following section regarding D16/D08 access of 32-bit registers and the communication protocol.

32-bit Registers

Several of the A16 registers (and all other 24-bit registers) are implemented as 32-bit-only resources. These are accessible using VME Bus D16 and D08(EO) accesses. However certain restrictions apply. The affected A16 registers are:

- RAM 0-1
- Send Data
- Receive Data
- Query Response Command
- Parameter 1-7

Reading 32-bit Registers

When reading a 32-bit register using 8- or 16-bit modes, a simple caching mechanism is used. On any read including the most significant byte (lowest address), the 32-bit register is read and all 32-bits are latched into the read cache. A read not including the most significant byte fetches data from the read cache, without re-reading the register. This insures that the data will be unchanged by any intervening write by the DSP (which would result in garbled data).

This mechanism also introduces a hazard. Reads of less significant bytes get data from the 32-bit register last read by a most-significant-byte read. In other words, the least significant byte can’t be read first or by itself. Thus, there are two important rules:

1. Always read all 32 bits of a 32-bit register.
2. Always read the most significant part first.

Writing 32-bit Registers

When writing to a 32-bit register using 8- or 16-bit modes, a simple caching scheme is also employed. On any write not including the least significant byte (highest address), the data is latched into the write cache. A write to the least significant byte causes the cached data to be written to the 32-bit register (in parallel with the current data for the least significant bytes(s).

This mechanism has its own hazards. Writes to the least significant byte will always include the most recently cashed data, whether intended for that register or not. Lone writes to the most significant part of a 32-bit register will be lost if not followed by a write to the least significant part of the same register. Thus there are two important rules:

1. Always write all 32 bits of a 32-bit register.
2. Always write the least significant part last.

Command/Response Protocol

The Command/Response protocol uses the following resources:

- Command/Query Response register implemented as a general purpose RAM
- Three parameter registers implemented as a general purpose RAM
- Additional A24 accessible RAM contiguous with the parameter registers
- The Command Ready, Query Response Ready, Err* and Done bits of the Status register.

The RAM registers are the communications media, while the Status register bits provide synchronization. In general, a controller sends a command to the DSP by first writing any parameters to the parameter registers and the following RAM location. It then writes the command to the command register, which clears the Command/Parameter Ready bit and interrupts the DSP. At this point, the DSP has exclusive access to the RAM registers. The controller may not access that RAM again until the Command/Parameter Ready bit is true.

When interrupted, the DSP reads the command and its parameters, writes any response data back to the Query Response Register and any other data to the parameter registers and the following RAM and set the Command/Parameter Ready bit true.

The Query Response Ready bit is used to indicate that the DSP has written query data to the RAM registers. It is set by the software and cleared by a write of the Command Register.

The Done bit is set by DSP software when it finishes execution of a command or a command sequence. This may by long after it has set the Command/Parameter Ready bit. The DSP software clears the Done bit immediately on receipt of a new command, before it sets the Command/Parameter Ready bit.

The Err* bit is asserted (to 0) by the DSP software to indicate an error in the decoding or execution of a command. It is asserted (to 1) if the command was executed with no error. This bit must be valid before Done is set at the end of a command.

In order to avoid contention and/or invalid data reads, there are certain rules that must be observed:

1. A controller must not write to any of the RAM registers when Command/Parameter Ready is false.
2. The DSP must not write to any of the RAM registers when either Command/Parameter Ready or Query Response Ready is true.
3. A controller must not read any of the RAM registers when Query Response Ready is false.
4. The DSP must not read any of the RAM registers when Command/Parameter Ready is true.
5. When writing a command together with parameter, a controller must always write to the Command Register last.
6. When executing a command that requires it to return response data, the DSP must set the Query Response Ready bit no later than the Command/Parameter Ready bit.
7. The DSP must not clear the Done bit while Command/Parameter Ready is true.
8. The DSP must not change the Err* bit while Done is true.
9. A controller must not regard the done bits a valid while Command/Parameter Ready is false.
10. A controller must not regard the Err* bit as valid while Done is false.

Controller Protocol Examples

There are three basic procedures used by a controller, Write Command, Read Response and Wait for Done. These can be combined for more complex sequences.

Write Command

This is the procedure to send a command to the DSP.

1. Wait for Command/Parameter Ready true.
2. Write any parameters to the Parameter registers and RAM.
3. Write the command to the Command register.

Read Response

This is the procedure for reading a response to query command.

1. Wait for Query Response Ready true.
2. Read the data from the Query Response register and any additional data from the Parameter registers and RAM.

Wait for Done

This is the procedure to wait for command completion and check for error.

1. Wait for Command/Parameter Ready true.
2. Wait for Done true.
3. If Err* = 0 , handle error.

Complex Sequences

A robust procedure for sending a query and reading the response would look like this:

1. Send Command.
2. Wait for Done.
3. If no error then Read Response.

Multiple commands may be sent with a test for errors at the end of the sequence. This example sends three commands before checking for errors.

1. Send Command.
2. Send Command.
3. Send Command.
4. Wait for Done.

DSP Protocol

When a controller writes to the Command register, a DSP interrupt is generated. When responding to this interrupt, the DSP will follow this procedure.

1. Clear the Done bit.
2. Read and decode the command from the Command register.
3. Read any parameters from the Parameter registers and RAM.
4. If a response data is required:

   a. Write the data to the Query Response register, Parameter registers and RAM.
   b. Set Query Response Ready true.

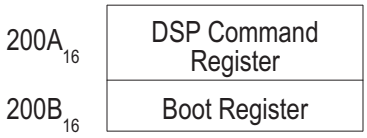
7. Set Command/Parameter Ready true.
8. Finish command execution.
9. If any errors are pending set Err* = 0, else set Err* = 1.
10. Set Done true.

There are two additional requirement for the DSP:

1. Once it begins processing a command interrupt, the DSP must defer processing subsequent commands until it has finished.
2. The DSP software maintains an error(s) pending flag (and possibly and error queue) that is set by any command decoding or execution error and cleared by some other method such as an error query.

DSP Bus Registers

There are two 32-bit registers in the DSP bus address space. The VXI FPGA does not assert TA* when these registers are accessed.



Note that these registers appear multiple times in the memory map, since only the address lines A31-30, A17-13, A9-8 and A3-0 are used for decoding.

The A24 registers are defined as follows:

- Boot Register: This read/write register is used to configure the device after a device reset. It has the following format:

Bit	31-16	15	14	13	12	11-0
Contents	Unused	Spare	ST Done	Loaded	Ready	Model Code

Spare: This read/write bit has no pre-defined function.

ST Done: This bit should be written to a one (1) when the DSP successfully competes its self-test, within five seconds after SYSRESET* is de-asserted. Its initial value is zero (0).

Loaded: This bit should be written to a one (1) when (or immediately after) the DSP loads the model code, before competing its self-test. Its initial value is zero (0).

Ready: This bit is written to a one (1) to indicate that the device is ready for normal operation. Its initial value is zero (0).

Model Code: As soon as possible and within 25 ms after coming out of reset, when the DSP has valid code loaded, it should write the VXI model code to these bits. Their initial value is 0x0200.

- DSP Command Register: This register is used to assert VXI interrupts and toggle various status register bits. Many of the bits in this register are grouped into related Clock and Value pairs. This allow the bits to be modified independently with single register writes. In order to change an output value, the Clock bit must be written as a one (1), while the Value is written as the desired output value. Writing the Clock bit as a zero (0) will not change the output state. The current state is read from the Value bit.

The DSP Command register has the following format:

Bit	31-24	23	22	21	20	19	18	17	16
Contents	Unused	FIFO Enable Clock	FIFO Enable Value	FIFO In Clock	FIFO In Value	DONE Clock	DONE Value	ERRn Clock	ERRn Value

Bit	15	14	13	12	11	10	9-8	7-0
Contents	Q Resp Ready Clock	Q Resp Ready Value	Cmd Ready Clock	Cmd Ready Value	IRQ Enable Clock	IRQ Enable Value	Unused	IRQ7-0

Glossary

A16 registers

Address space using 16 address lines. The VXI definition gives each VXI module 64 bytes of A16 registers.

A24 registers

Address space using 24 address lines. VXI modules can configure how much A24 address space they use.

Agilent VEE

An Agilent program for graphical programming. See *VEE*.

arbitrary source

A signal source capable of producing an arbitrary waveform according to the way it is programmed.

arbitration bus

See DTB arbitration bus.

ASCII

American Standard Code for Information Interchange, a standard format for data or commands.

backplane

A set of lines that connects all the modules in a VXI system.

baseband

A band in the frequency spectrum that begins at zero. In contrast a zoomed band is centered on a specified center frequency.

block mode

A mode of data-collecting used in instruments such as the Agilent/HP E1431A. The instrument stops taking data as soon as a block of data has been collected. Overlap block mode in the VT1432A and VT1433B can be configured to act exactly like block mode.

block size

The number of sample points in a block of data.

breakout box

Another name for the 8-channel input connector.

C-Library (interface library)

A library of functions, written in C language, which can be used to operate the VT1432A and VT1433B.

C-size

One of several possible sizes for VXI modules. The VT1432A and VT1433B are C-size modules.

channel-dependent commands

Commands that are channel-dependent change a parameter for each channel independently.

COLA

Constant Output Level Amplifier.

continuous mode

A mode of data-collecting used in the VT1432A, the VT1433B and in other instruments such as the Agilent/HP E1431A. The instrument collects data continuously and stops only if the FIFO overflows.

D32, D16 and D08 (EO)

The VXI Bus provides 32 data lines. Modules can use all 32 lines or 16 lines or 8 lines. For example, “D16 access” refers to data read across 16 lines.

daisy-chain

A set of instruments or modules connected together in a line. Data and instructions enter each one before being buffered and passed out to the next module in line.

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 *Spectrum & Network Measurements* available through VXI Technology.

delta sigma

A method for converting an analog input to digital data. It involves using a difference of two voltages (delta) and a summation of signals (sigma) to improve accuracy.

digitizer

An instrument which converts analog signals into digital data suitable for digital signal processing.

DRAM

Dynamic Random Access Memory.

DSP
Digital Signal Processing.

DTB arbitration bus
The VT1432A does not use the arbitration bus. The arbitration bus is part of the VXI specification and is used by some modules to request bus control.

ECL
Emitter-Collector Logic, a standard for electrical signals.

Engineering Unit (EU)
A scale factor used to convert the output of a transducer (in volts) into another unit (for example: g's).

FFT
Fast Fourier Transform.

FIFO
First-In First-Out. A buffer and controller used to transmit data. The FIFO in the VT1432A/VT1433B input is implemented using DRAM.

freerun counter
A counter in which the bits always increment. When the freerun counter reaches all ones it resets to all zeros and continues counting.

Fs
Sample Frequency or sample rate.

group ID
Any number of channels may be declared and uniquely identified by a groupID. A channel can be a member of more than one group.

holdoff time
A circuit that detects a trigger signal will not respond to another trigger until the holdoff time has passed. This prevents a ringing signal from be detected as multiple triggers.

IACK
Interrupt ACKnowledge.

ICP®
Integrated-Circuit Piezo-electric transducer.

IRQ
Interrupt ReQuest.

kSamples/s
Kilosamples per second.

LED
Light Emitting Diode.

Local Bus
A high-speed port that is defined as a standard byte-wide ECL protocol which can transfer measurement data at up to 2.62 MSamples per second from left to right on the VXI backplane.

logical address
The VXI logical address identifies where each module is located in the memory map of the VXI system.

message-based VXI device
Message-based devices communicate with the VXI Bus using high-level ASCII commands. Programming is easier and more sophisticated, but communication is slower than with register-based devices. Message-based devices can also be programmed at the register level. The VT1432A and VT1433B are register-based VXI devices.

module-dependent commands
Commands that are module-dependent change a parameter for all channels of the module; even when only one channel has been specified in the channel list.

MXI bus
A bus standard which can be used to connected multiple VXI mainframes.

overlap block mode
A mode of data-collecting in used in the VT1432A and VT1433B. It is similar to block mode except that it allows additional arms and triggers to occur before an already-acquired block is sent to the host.

pipeline mode
A Local Bus mode in which data is sent through a module and on to the next one.

Plug&Play
See VXI*plug&play*

RAM
Random Access Memory.

register-based VXI device

Register-based devices communicate with the VXI Bus by way of registers. They must be programmed with low-level binary commands but they can communicate faster than message-based devices. The VT1432A and VT1433B are register-based VXI devices.

registers

Memory locations in the hardware of a VXI module which can be used to program the module at a low level.

RPM

Revolutions Per Minute.

ROM

Read-Only Memory.

SCA

Signal Conditioning Assembly. An example is the 4-channel input assemblies used in the VT1432A (also called Vibrato).

sample rate

The rate at which the measurement data is sampled. For the VT1432A, the sample rate is 2.56 times the frequency span. Sample rate is abbreviated “Fs” (for “sample Frequency”).

settling

When settling, the digital filter waits a designated number samples before outputting any data.

SFP

SDee Soft Front Panel.

shared memory

Memory locations in both a VXI module and in a host or controller which are shared and can be used to transmit data between the host and module.

slot 0 commander

The module which occupies the left-most slot in a VXI mainframe. It supplies important signals for the rest of the system.

SMB

Sub-Miniature “B”; a type of connector.

Soft Front Panel (SFP)
A VXI*plug&play* program which provides and easy-to-use interface for the VT1432A. It can be used in Microsoft Windows 95 or later or Windows NT.

SRAM
Static Random Access Memory.

summer
A circuit that outputs the sum of two input signals.

sync/trigger line
A TTL line on the VXI back plane, used for synchronization or triggering signals.

SYSRESET*
SYStem RESET line, part of the VXI Bus.

system module
The module with the lowest VXI logical address. It needs to be set to output the synchronization pulse for a multiple module group. All system sync pulses come from the system module.

tachometer
The tachometer produces a signal which is proportional to the rotation of a device. It can be programmed to produce one or more signals per revolution.

target
The ‘target’ of a library function is either a channel, a group or (rarely) a module, depending on the nature of the call. When the same library function may be called with either a channel or a group identifier, its ‘target’ is shown by a parameter named ID.

TTL
Transistor-Transistor Logic, a standard for electrical signals.

TTLTRG
TTL TRiGger lines, part of the VXI Bus.

VEE
Virtual Engineering Environment, a program which facilitates the setup and programming of instruments by employing a graphic user interface.

VME Bus

An industry-standard bus on the VXI backplane for module control, setup and measurement data transfers. For measurement data transfers, the Local Bus offers higher transfer rates.

VXI

VME Extensions for Instrumentation, a standard specification for instrument systems.

VXI*plug&play*

A set of standards which provides VXI users with a level of standardization across different vendors beyond what the VXI standard specifications spell out.

zoom

In instruments that support zoom, one can select a frequency span around a specified center frequency in order to focus on a specific frequency band.

Index

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W