User Manual

Tektronix

TVS600 & TVS600A Series Waveform Analyzers

070-9916-00

VXI bus

This document supports firmware version 1.5
and above.

CE

Printed in the U.S.A.

Tektronix, Inc., P.O. Box 1000, Wilsonville, OR 97070–1000

WARRANTY

Tektronix warrants that the products that it manufactures and sells will be free from defects in materials and workmanship for a period of three (3) years from the date of shipment. If a product proves defective during this warranty period, Tektronix, at its option, either will repair the defective product without charge for parts and labor, or will provide a replacement in exchange for the defective product.

In order to obtain service under this warranty, Customer must notify Tektronix of the defect before the expiration of the warranty period and make suitable arrangements for the performance of service. Customer shall be responsible for packaging and shipping the defective product to the service center designated by Tektronix, with shipping charges prepaid. Tektronix shall pay for the return of the product to Customer if the shipment is to a location within the country in which the Tektronix service center is located. Customer shall be responsible for paying all shipping charges, duties, taxes, and any other charges for products returned to any other locations.

This warranty shall not apply to any defect, failure or damage caused by improper use or improper or inadequate maintenance and care. Tektronix shall not be obligated to furnish service under this warranty a) to repair damage resulting from attempts by personnel other than Tektronix representatives to install, repair or service the product; b) to repair damage resulting from improper use or connection to incompatible equipment; c) to repair any damage or malfunction caused by the use of non-Tektronix supplies; or d) to service a product that has been modified or integrated with other products when the effect of such modification or integration increases the time or difficulty of servicing the product.

THIS WARRANTY IS GIVEN BY TEKTRONIX IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEKTRONIX' RESPONSIBILITY TO REPAIR OR REPLACE DEFECTIVE PRODUCTS IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Service Assurance

If you have not already purchased Service Assurance for this product, you may do so at any time during the product's warranty period. Service Assurance provides Repair Protection and Calibration Services to meet your needs.

Repair Protection extends priority repair services beyond the product's warranty period; you may purchase up to three years of Repair Protection.

Calibration Services provide annual calibration of your product, standards compliance and required audit documentation, recall assurance, and reminder notification of scheduled calibration. Coverage begins upon registration; you may purchase up to five years of Calibration Services.

Service Assurance Advantages

- Priced well below the cost of a single repair or calibration
- Avoid delays for service by eliminating the need for separate purchase authorizations from your company
- Eliminates unexpected service expenses

For Information and Ordering

For more information or to order Service Assurance, contact your Tektronix representative and provide the information below. Service Assurance may not be available in locations outside the United States of America.

Name
Company
Address
City, State, Postal code
Country
Phone

VISA or Master Card number and expiration date or purchase order number
Repair Protection (1,2, or 3 years)
Calibration Services (1,2,3,4, or 5 years)
Instrument model and serial number
Instrument purchase date

Table of Contents

Getting Started

Operating Basics

Reference

Status and Events

Appendices

Glossary

Index

List of Figures

List of Tables

General Safety Summary

Review the following safety precautions to avoid injury and prevent damage to this product or any products connected to it. To avoid potential hazards, use this product only as specified.

Only qualified personnel should perform service procedures.

While using this product, you may need to access other parts of the system. Read the *General Safety Summary* in other system manuals for warnings and cautions related to operating the system.

To Avoid Fire or Personal Injury

Connect and Disconnect Properly. Do not connect or disconnect probes or test leads while they are connected to a voltage source.

Ground the Product. This product is indirectly grounded through the grounding conductor of the mainframe power cord. To avoid electric shock, the grounding conductor must be connected to earth ground. Before making connections to the input or output terminals of the product, ensure that the product is properly grounded.

Observe All Terminal Ratings. To avoid fire or shock hazard, observe all ratings and markings on the product. Consult the product manual for further ratings information before making connections to the product.

Do not apply a potential to any terminal, including the common terminal, that exceeds the maximum rating of that terminal.

Do Not Operate Without Covers. Do not operate this product with covers or panels removed.

Use Proper Fuse. Use only the fuse type and rating specified for this product.

Avoid Exposed Circuitry. Do not touch exposed connections and components when power is present.

Do Not Operate With Suspected Failures. If you suspect there is damage to this product, have it inspected by qualified service personnel.

Do Not Operate in Wet/Damp Conditions.

Do Not Operate in an Explosive Atmosphere.

Keep Product Surfaces Clean and Dry.

Provide Proper Ventilation. Refer to the manual’s installation instructions for details on installing the product so it has proper ventilation.

Symbols and Terms

Terms in this Manual. These terms may appear in this manual:

WARNING. *Warning statements identify conditions or practices that could result in injury or loss of life.*

CAUTION. *Caution statements identify conditions or practices that could result in damage to this product or other property.*

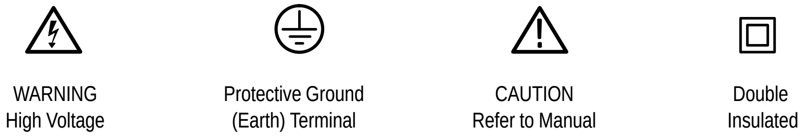
Terms on the Product. These terms may appear on the product:

DANGER indicates an injury hazard immediately accessible as you read the marking.

WARNING indicates an injury hazard not immediately accessible as you read the marking.

CAUTION indicates a hazard to property including the product.

Symbols on the Product. The following symbols may appear on the product:



Preface

This is the user manual for the TVS600 and TVS600A Waveform Analyzers. It covers the following information:

- Describes the capabilities of the waveform analyzer, how to install it, and how to use it in a programming environment
- Explains how to operate the waveform analyzer: how to control acquisition of, processing of, and input/output of information
- Lists and describes the syntax of SCPI (Standard Commands for Programmable Instruments) commands and lists IEEE 488.2 Common Commands, both of which set up and control the waveform analyzer
- Introduces the TVS600A VXI*plug&play* Driver and the TVS600A VXI*plug&play* Soft Front Panel (SFP), the two other features for set up and control of the waveform analyzer

***NOTE**. The driver and SFP are the VXI*plug&play* instrument driver and the VXI*plug&play* soft front panel required of VXI*plug&play*-compliant instruments. They are included in the TVS600A VXI*plug&play* software.*

About This Manual

This manual is composed of the following chapters:

- *Getting Started* shows you how to configure and install your waveform analyzer and provides an incoming inspection procedure.
- *Operating Basics* uses maps to describe the various interfaces for controlling the waveform analyzer, including the front panel, the SCPI and IEEE 488.2 Common Commands languages, the driver, and the soft front panel. These maps provide overviews of the product and its functions from several viewpoints. The chapter concludes with tutorial examples on programming the waveform analyzer using the SCPI commands.
- *Reference* comprises an encyclopedia of topics (see *Overview* on page 3–1) that describe the waveform analyzer interface and features and how to use them. (Detailed descriptions of all programming commands are found in the *TVS600 & TVS600A Command Reference* manual.)
- *Status and Events* describes how the status and events reporting system operates and lists all possible system errors.

- *Appendices* provides additional information including the specifications, calculation algorithms, and SCPI conformance information.

Related Manuals and Online Documents

This manual is part of a document set of standard-accessory manuals and online documentation; this manual mainly focuses on installation and usage. Please read the following list of documents supporting TVS600A operation and service. (See *Accessories* on pages 1–3 for manual part numbers.)

Manual Name	Description
TVS600 & TVS600A Series Waveform Analyzers Reference	Provides an alphabetical listing of the programming commands. It is the quick command reference and is a standard accessory.
TVS600 & TVS600A Series Waveform Analyzers Command Reference	Provides an alphabetical listing of the programming commands and details. It is the comprehensive command reference and is a standard accessory.
TVS600A Online SFP Help for the VXI*plug&play* Soft Front Panel	Documents the Soft Front Panel, an application that ships with this product. The TVS600A VXI*plug&play* software is a standard accessory.
TVS600A Online Driver Help for the VXI*plug&play* Driver	Documents the robust library of functions that ships with this product. The TVS600A VXI*plug&play* software is a standard accessory.
TVS600 & TVS600A Series Waveform Analyzers Service Manual	Describes how to service the instrument to the module level. This optional manual must be ordered separately.

For more information on how the product documentation relates to the waveform-analyzer operating elements, see *Operating Interfaces Map* on page 2–3.

Default Model

This manual documents the TVS621, TVS621A, TVS625, TVS625A, TVS641, TVS641A, TVS645, and TVS645A waveform analyzers. Take note of the following conventions used when referencing these products:

- Generally, the name “TVS600A” (or just “waveform analyzer”) is used when providing information common to the TVS600 and TVS600A series of waveform analyzers.
- The labels “TVS600 only” and “TVS600A only” are used when providing information that pertains only to those models.
- The more specific names, listed above, are used when providing information that pertains only to a specific model, such as the TVS625A.

Contacting Tektronix

Product Support	For application-oriented questions about a Tektronix measurement product, call toll free in North America: 1-800-TEK-WIDE (1-800-835-9433 ext. 2400) 6:00 a.m. – 5:00 p.m. Pacific time
	Or contact us by e-mail: tm_app_supp@tek.com
	For product support outside of North America, contact your local Tektronix distributor or sales office.
Service Support	Contact your local Tektronix distributor or sales office. Or visit our web site for a listing of worldwide service locations.
	http://www.tek.com
For other information	In North America: 1-800-TEK-WIDE (1-800-835-9433) An operator will direct your call.
To write us	Tektronix, Inc. P.O. Box 1000 Wilsonville, OR 97070-1000

Product Description

This chapter describes the TVS600A Waveform Analyzer and its options. Following this description are two sections:

- *Installation* shows you how to configure and install the waveform analyzer and the VXI*plug&play* software included with the product.
- *Incoming Inspection* provides a procedure for verifying basic operation and functionality.

Key Features

The TVS600A Waveform Analyzers are a family of C-size, VXI modules that provide high-speed signal acquisition, real-time measurements, and Fast-Data-Channel data transfer. They are VXI*plug&play*-compliant instruments for use in the WIN, WIN95, and WINNT frameworks, and they include VXI*plug&play*-compliant software. Key features include:

- Fully programmable with an extensive SCPI command set and message-based interface
- A VXI*plug&play* compliant instrument driver and soft front panel (see *VXIplug&play Software* on page 1–2)
- A maximum real time digitizing rate of up to 5 GSample/second with an analog bandwidth up to 1 GHz. See Table 1–1 for models and rates available.
- Acquisition modes such as normal, envelope, peak-detect, and average. See Table 1–1 for models and modes available.
- Fast throughput features: Fast Data Channel (FDC) and auto-advance acquisition cycle, which allows fast rearm
- A full compliment of internal triggering types, including main, delayed, edge, setup and hold, transition (runt and slew rate), and pulse triggering
- Trigger sources from input channels, front-panel external trigger, and VXI-backplane triggers
- Standard acquisition memory that allows 15,000 samples to be taken in realtime acquisition mode and 30,000 samples in extended-realtime acquisition mode
- An extensive calculation system for measurements, template testing, waveform math, and waveform transformations

Differences by Model

Table 1–1 lists some key feature-differences between the TVS models that this manual covers.

Table 1–1: Key-feature differences among models

Feature	621	621A	625	625A	641	641A	645	645A
No. of channels	2				4			
Digitizing rate, max.	1 GS/s		5 GS/s		1 GS/s		5 GS/s	
Analog Bandwidth	250 MHz		1 GHz		250 MHz		1 GHz	
Template and Mask Testing	No	Yes	No	Yes	No	Yes	No	Yes
Timeout, logic, & transition triggering types	No	Yes	No	Yes	No	Yes	No	Yes
Peak-detect acquisition mode	No	Yes	No	Yes	No	Yes	No	Yes
Reference Waveforms	No	Yes	No	Yes	No	Yes	No	Yes
Measurement Zones	No	Yes	No	Yes	No	Yes	No	Yes

Firmware Upgrade

Owners of TVS600 model waveform analyzers may purchase a firmware upgrade kit that adds most of the features of the TVS600A models. Contact your Tektronix service representative for more information (see *Contacting Tektronix* on page xv).

VXI*plug&play* Software

***NOTE**. This manual ships with the TVS600A version of the VXIplug&play software; TVS600 Waveform Analyzers with version 2.0 or earlier firmware cannot use all the features supported by the VXIplug&play driver and Soft Front Panel. You may prefer to use your previous versions of the driver and Soft Front Panel or, you may prefer to upgrade your firmware (see* Firmware Upgrade, *above).*

The waveform analyzer is a VXI*plug&play* instrument and comes with VXI*plug&play* software that runs on the Win, Win95, and WinNT platforms.

The waveform analyzer software installs the VXI*plug&play* instrument driver and the VXI*plug&play* soft front panel required by the VXI*plug&play* standard:

- TVS600A VXI*plug&play* Driver. This driver contains functions that can greatly simplify programming the waveform analyzer. The software includes an online reference (help file) and the source code for this driver, which you can adapt to your specific applications. The source code also illustrates the use of many SCPI commands.
- TVS600A VXI*plug&play* Soft Front Panel (SFP). This SFP application provides a graphical user interface for controlling many of the waveform-analyzer features and a display for the waveforms that the waveform analyzer acquires. Online help is available from the SFP help menu.

Occasionally new versions of the software may become available at our web site. See *Contacting Tektronix* on page xv in *Preface.*

Accessories

This section lists the standard and optional accessories available for the TVS600A Waveform Analyzers.

Standard Accessories

The following accessories are shipped with the waveform analyzer:

- *TVS600 & TVS600A Series User Manual* (Tektronix part number 070-9916-XX)
- *TVS600 & TVS600A Series Reference Manual* (Tektronix part number 070-9918-XX)
- *TVS600 & TVS600A Series Command Reference Manual* (Tektronix part number 070-9917-XX)
- TVS600A VXI*plug&play* Software (Tektronix part number 063-2811-XX, 16-bit, and 063-2812-XX, 32-bit)

Optional Accessories

The following accessories are available for use with the waveform analyzer:

- Probes: the waveform analyzer can use a wide range of passive, active, and differential voltage probes; current probes; O/E converters; and electrical communications adapters. Please refer to your Tektronix catalog for more information on probes and other accessories.
- *TVS600A Waveform Analyzers Service Manual* (Tektronix part number 070-9915-XX)
- Opt. C3: Three years of Calibration Services.

- Opt. C5: Five years of Calibration Services.
- Opt. D3: Test Data for Opt. C3.
- Opt. D5: Test Data for Opt. C5.
- Opt. R5: Repair warranty extended to cover five years.

Installation

This chapter covers installation of the waveform analyzer. It first illustrates the common system configurations and then addresses the following topics:

- *Hardware Configuration* on page 1–6
- *Hardware Installation* on page 1–8
- *Software Installation* on page 1–12

See your system manuals for system configuration instructions; a VXIbus system is shown in Figure 1–1 and a MXI/VXI or GPIB system is shown in Figure 1–2.

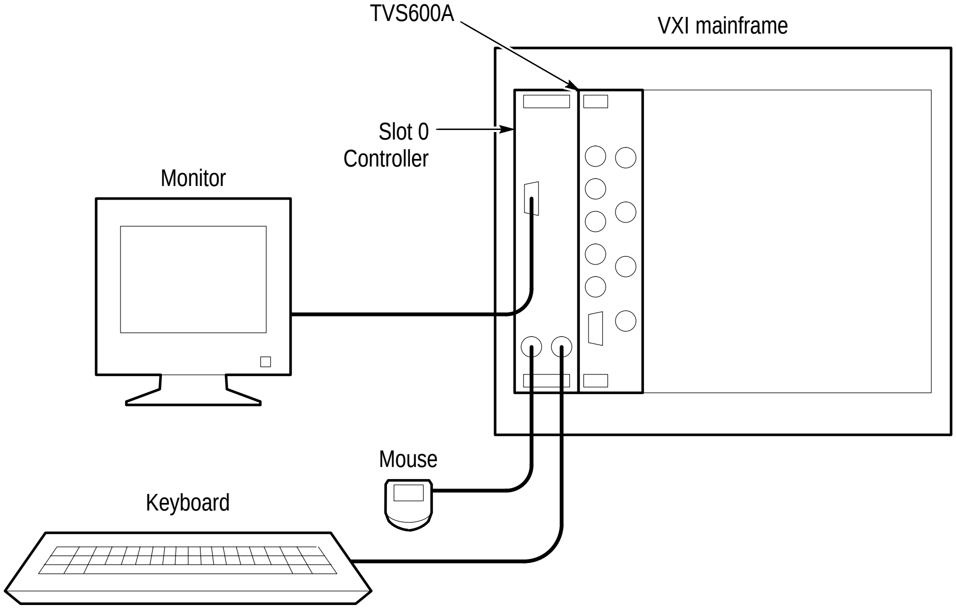


Figure 1–1: VXIbus system configuration with embedded controller

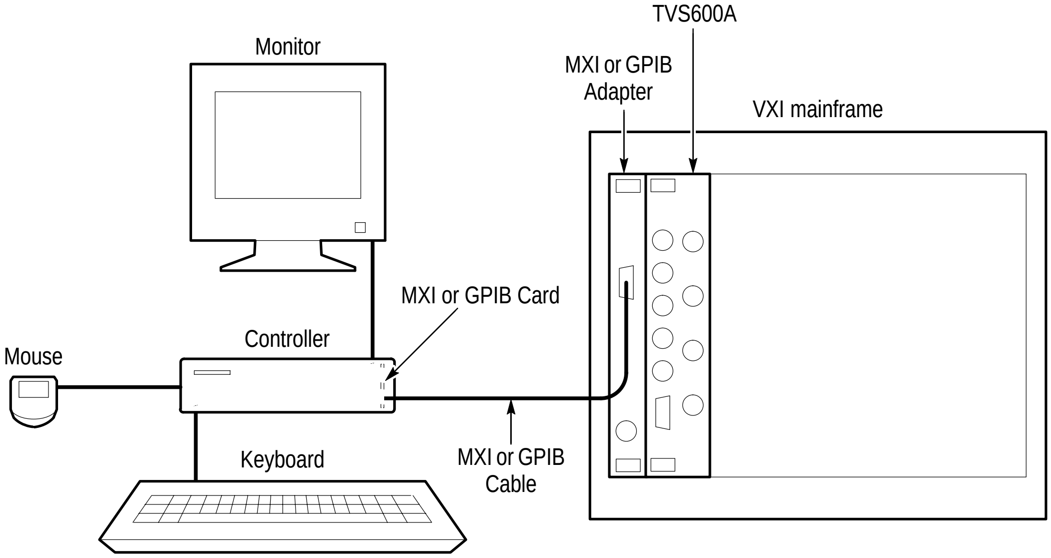


Figure 1–2: MXI/VXI or GPIB system configuration

Hardware Configuration

You must configure the VXIbus module and the VXIbus mainframe before installing the module. To configure the waveform analyzer, set its logical address on the VXIbus. To make ready your VXIbus mainframe, you ensure power and capacity requirements are met. You may also have to set the Bus Grant and Interrupt Acknowledge jumpers, if your mainframe uses them. This section describes how to perform the necessary configuration.

Setting the Logical Address

Every module within a VXIbus system must have a unique logical address; no two modules can have the same address. On the waveform analyzer, you rotate two switches on the rear panel to select the logical address. Refer to Figure 1–3 for the switch locations. You can select a static address or Dynamic Auto Configuration. The default address is FF hexadecimal, which selects Dynamic Auto Configuration.

Static Logical Address. Static logical address selections set the address to a fixed value. The range for static addresses is from 01 to FE hexadecimal (1 to 254 decimal). A static logical address ensures that the waveform-analyzer address remains fixed for compatibility with systems that require a specific address value. Remember that each device within your system must have a unique address to avoid communication problems. The factory default address is FF hexadecimal, which selects Dynamic Auto Configuration.

Dynamic Auto Configuration. With Dynamic Auto Configuration selected (hexadecimal FF or decimal 255), the system resource manager automatically sets the address to an available value in your system. For example, if you already have devices set to addresses 01 and 02, the resource manager might automatically assign address 03 to the waveform analyzer at power on.

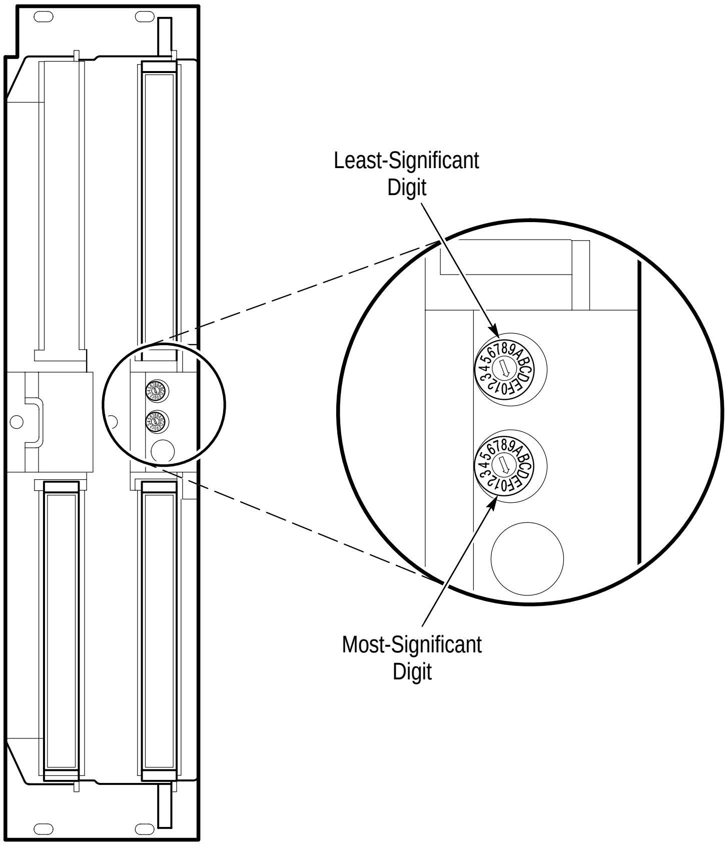


Figure 1–3: Logical address switches (FF setting shown)

Configuring the VXIbus Mainframe

This section describes how to install the waveform analyzer into a Tektronix VXIbus mainframe. If you are installing the waveform analyzer into a different mainframe, refer to the instruction manual for that mainframe for any pertinent installation or capacity information.

Voltage, Current, and Cooling Requirements. You will find voltage, current, and cooling requirements for the waveform analyzer in *Appendix A*: *Specifications* at the following locations:

- Voltage and current requirements. See Table A–6 on page A–12.
- Cooling requirements. See *Airflow Resistance* in Table A–7 on page A–13.

These requirements also appear on the left cover of the waveform analyzer. Be sure your mainframe can supply adequate current and cooling to the waveform analyzer as well as the other modules you plan to install into the same mainframe.

⚠ ***WARNING.*** *Shock hazards exist due to high currents within the mainframe compartment. Do not change configuration of the Bus Grant and Interrupt Acknowledge jumpers unless you are qualified to do so. Consult your VXI mainframe manual for safety warnings and configuration information.*

Jumper Settings. Most newer VXIbus mainframes, such as the Tektronix VX1410 Intelliframe, have an auto-configurable backplane with no mechanical jumpers. You do not need to set jumpers on these VXIbus mainframes.

Some earlier VXIbus mainframes contain daisy-chain jumper straps that you must configure before installing the waveform analyzer. The jumper straps, located beside the P1 connectors, set up the Bus Grant (BG0–BG3) and Interrupt Acknowledge (IACK) signals. If you are using a Tektronix mainframe that uses such straps, the names of the jumper straps (BG0–BG3 and IACK) are often printed on the circuit board facing the front of the mainframe. Access these jumpers from the front of the mainframe.

If your VXIbus mainframe has jumper straps, set the IACK and BG0–BG3 jumpers for the waveform analyzer as described below:

- Remove the jumper straps for the left-most slot in which you will install the waveform analyzer (retain the strap for future configurations).
- Leave the jumpers for the right-most slot installed in the mainframe.

For example, if you want to install the waveform analyzer into the third and fourth mainframe slots, remove all jumper straps for the third slot and make sure the jumpers are installed in the fourth slot.

Hardware Installation

This section describes how to install the waveform analyzer into a VXIbus mainframe and how to install the product software. For hardware configuration information, refer to *Hardware Configuration* on page 1–6.

You may install the waveform analyzer into any empty slot in the mainframe except Slot 0. Be sure to set the logical address before installation (see *Setting the Logical Address* on page 1–6).

⚠ **CAUTION.** *If you install the waveform analyzer into a D-size mainframe, be sure to connect the P1 and P2 connectors of the module to the P1 and P2 connectors on the mainframe. Electrical damage will result if you connect the P1 and P2 connectors on the module to the P2 and P3 connectors on the mainframe.*

To avoid damage, check for bent pins on P1 and P2 before installation.

Use the following installation procedure and Figure 1–4 on page 1–10 to install the waveform analyzer into the mainframe:

1. On the mainframe, set the power ON/STANDBY switch to STANDBY.

2. Insert the waveform analyzer into the mainframe top and bottom module guides and push it partially into the mainframe (Figure 1–4). Then slide the waveform analyzer into the mainframe as far as it will go without forcing it.

3. Make sure the front panel is flush with the front of the mainframe chassis, and then use a flat-blade screwdriver to install the top and bottom retainer screws. Alternately tighten the screws, applying only a few turns at a time to fully seat the module. If it is not flat, remove the module and check for mechanical problems with the VXIbus connector or the module enclosure.

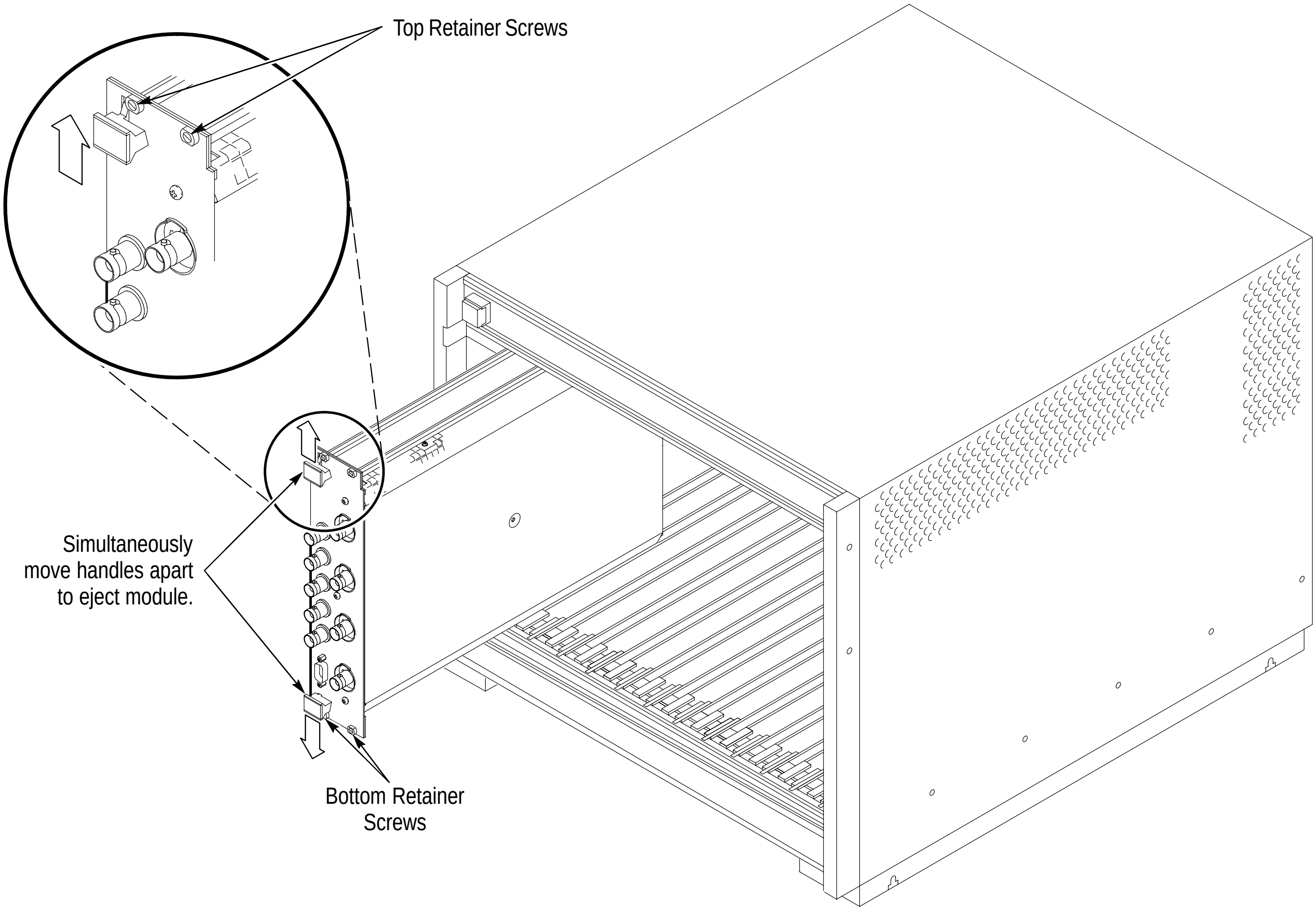


Figure 1–4: Module retainer screws and ejector mechanism

Removal from VXIbus Mainframe

Use the following procedure to remove the waveform analyzer from a Tektronix VXIbus mainframe. If you are using a different mainframe, you may need to modify this procedure. Refer to your mainframe manual for instructions.

1. On the mainframe, set the power ON/STANDBY switch to STANDBY.
2. Using a flat-blade screwdriver, loosen the top and bottom retainer screws (Figure 1–4).
3. Grasp both handles of the waveform analyzer. At the same time, move the top handle upward and the bottom handle downward to eject the waveform analyzer.
4. Pull the waveform analyzer out of the mainframe.

Power-On Procedure

This section describes how to check that your waveform analyzer powers up properly. Be certain that the waveform analyzer is properly configured before applying power. Refer to *Hardware Configuration* on page 1–6.

1. Before applying power to your waveform analyzer and VXIbus system, check the following items:

   - Ensure that all VXIbus modules are properly installed.
   - Check that all connected signal sources are set to an appropriate output level to avoid damaging inputs on your waveform analyzer.
   - Power up your controller, if it is external to the VXIbus mainframe.

2. Apply power to your VXIbus mainframe.

   During power on, the waveform analyzer performs a self test to verify functionality. The self test requires approximately five seconds to complete. The front-panel ARM’D and TRIG’D indicators blink during the self test. After testing successfully completes, the green READY indicator on the front panel should be on.

***NOTE**. The READY indicator does not light if the power-on self test fails.*

   If your waveform analyzer does not pass the power-on self test (READY indicator does not light), power off the VXIbus mainframe and check that all modules are fully seated in the VXIbus mainframe. If the problem persists, remove the waveform analyzer and check that its address setting does not conflict with another module. If failures continue, the module might require service.

3. Once the power-on self tests are complete, the waveform analyzer recalls the settings that were active when the waveform analyzer was powered off. There is one exception: input protection is always set ON, its reset value. Power-on settings are stored in nonvolatile memory.

Most parameters have a default value that you can restore by sending the `*RST` command.

Software Installation

This section describes how to install the TVS600A VXI*plug&play* software that accompanies this manual. The product software includes the TVS600A VXI*plug&play* Soft Front Panel and TVS600A VXI*plug&play* Driver. It is installable on WIN, WIN95, and WINNT platforms.

***NOTE**. This manual ships with the TVS600A version of the VXI*plug&play *software; TVS600 Waveform Analyzers with versions earlier than version 2.0 firmware cannot use all the features supported by the VXI*plug&play *Driver and Soft Front Panel. You may prefer to use your previous versions of the driver and Soft Front Panel or you may prefer to upgrade your firmware (see* Firmware Upgrade*, on page 1–2).*

The instructions here are sufficient for installing TVS600 or TVS600A software; for detailed descriptions of TVS600 software, refer to the online help that came with that software.

Description

The TVS600A VXI*plug&play* software provides a virtual front panel and an instrument driver that supports several programming environments.

TVS600A Soft Front Panel (SFP). The SFP (TKSFS600.EXE) is a robust application that runs in Windows 3.1, Windows NT, and Windows 95; a few of its uses follow:

- Display and update waveforms acquired by the waveform analyzer.
- Set up and acquire those waveforms using normal acquisition mode and edge triggering.
- Transfer waveforms from any waveform analyzer channel or reference to another reference or to a .DIF (Data Interchange Format standard) file. Load any reference with a waveform from a file.
- Copy waveform-analyzer control setups to a file or from an existing file of setups to another file. Load the waveform analyzer with a setup from a file.
- Load waveform test templates stored in the SCPI DIF format and configure simple template testing.
- Run any command using the talker/listener utility of the SFP.

For detailed descriptions of these and other SFP features, refer to the Windows online help documents present with this application.

TVS600A Driver. The driver is a library of functions that provide a high-level means of controlling most waveform-analyzer operations. A single function can perform an instrument setup that would require many more SCPI commands to effect the same setup.

The driver supports the following programming environments:

- National Instruments LabWindows/CVI for Windows, Windows95, and Windows NT
- National Instruments LabVIEW for Windows
- Microsoft Visual BASIC
- Microsoft Visual C
- Borland Turbo C
- Hewlett Packard HP-VEE for Windows95 and WindowsNT

Other programming languages may also be used if they are compatible with Windows 16-bit and/or 32-bit DLLs.

TVS600A Driver Source Code. The driver library includes source code for the driver to help you better understand driver functions and to allow you to compile them in your programming environment. The source code conforms to ANSI C which enhances portability to other computer environments. (The source code also is useful in that it illustrates usage of many of the low-level commands underlying each function.)

Online Help. The TVS600A software includes three Windows online documents and release notes:

SFP600A.HLP	*TVS600A VXI*plug&play *Soft Front Panel Help*
TKTVS600.HLP	*TVS600A VXI*plug&play *Driver Reference*
DSTVS600.HLP	*TVS600A Data Sheet*
TKTVS600.TXT	*TVS600A Release Notes*

Requirements

The software requires one of the following VXI*plug&play* frameworks:

- WIN Framework Version 4.0 or better
- WIN95 Framework Version 4.0 or better
- WINNT Framework Version 4.0 or better

The framework comes with software provided by the manufacturer of VXI*plug&play* WIN, WIN95, and WINNT Framework Ver. 4 compliant controllers and is required to run the Soft Front Panel software and the driver library.

The following system requirements should be met before installing the TVS600A VXI*plug&play* software.

- 5 Mbyte of disk space
- Windows 3.1, Windows 95, or Windows NT
- 16 MBytes minimum required; 32 MBytes recommended

Set Up

The TVS600A VXI*plug&play* software is located on the DOS-format, high-density, 3-1/2 inch floppy disks included with this manual.

To install the VXI*plug&play* software from Windows, perform the following steps:

1. Insert disk 1 in the 3-1/2 inch disk drive.

   If your operating system is Windows 3.1, start the Program Manager and select Run in the File menu; if your operation system is Windows 95, display the START menu and select RUN. If your operating system is NT, use the Program Manager or the START menu, depending on which operation you NT version supports.

2. In the field below Command Line:, enter [drive]\setup. Enter the letter for [drive] that corresponds to the drive in which you inserted the VXI*plug&play* disk (usually A:\).

3. Click on OK to start the setup program. Setup loads all files on disk 1 to your C drive, unless you specify another, and then prompts you to insert additional disks. (The target directory is \VXIPNP.) If you previously installed a Tektronix VXI*plug&play* disk on your system, then you might not be prompted for all disks and the setup program might not need to install all files.

4. When installation completes, you are prompted to either run the soft front panel application or exit to Windows. If you have not previously installed the WIN Framework software, then you should exit to Windows. Install WIN Framework software before using the VXI*plug&play* software.

NOTE*. Before running the soft front panel, you must run your system resource manager.*

5. You should also review the software release notes. See *Software Release Notes* following this procedure.

Setup creates a VXIPNP program group and installs the soft-front-panel icon named TKTVS600 FRONT PANEL. It also installs an uninstall icon. For a list of the installed files, refer to the software release notes.

Software Release Notes

The installation disk installs software release notes in an ASCII file named TKTVS600A.TXT. This file contains additional installation and operation information that supercedes other product documentation. Installation places the TKTVS600A.TXT file at

C:\VXIPNP\WIN\TKTVS600\TKTVS600.TXT.

Note that “\WIN\” in the above directory name applies to the WIN framework and is replaced with \WIN95\ or \WINNT\ for the Windows 95 and Windows NT frameworks, respectively.

To view the TKTVS600.TXT file, open the Notepad Windows accessory and select the appropriate path to open the TKTVS600.TXT document.

Using the Software

To learn to use the TVS600A Soft Front Panel, start it and use its online help, accessed from its online help menu. The soft front panel help is included as a Windows Help document, which is available once you start the application.

***NOTE**. Before running the soft front panel, configure and run the VXI resource manager. See your controller and VXI framework documentation for instructions.*

To start the TVS600A Soft Front Panel, open the VXIPNP program group and double click on the TKTVS600 FRONT PANEL icon (Windows 3.1 and earlier versions of Windows NT) or traverse the Start menu to Programs, then VXIPNP, and then select the TKTVS600 FRONT PANEL menu entry (Windows 95 and later versions of Windows NT).

At start up, the software finds the installed TVS600A modules and asks you to pick one or all for connection. If you select All, then all TVS600A waveform analyzers are connected, though only one waveform analyzer is active and in communication with the soft front panel. For instructions on using the soft front panel, select Help –> Display SFP Help.

To learn about the driver functions, open (double click) the stand-alone, online reference, VXIPNP\WIN[95][NT]\TKTV600\TKTVS600.HLP. To use the library functions, include calls to them in the programs you write from your programing environment.

Incoming Inspection Procedure

This section contains instructions for performing the *Incoming Inspection Procedure*. This procedure verifies that the waveform analyzer is operating correctly after shipment.

If the waveform analyzer fails any test within this section, the module may need service. To contact Tektronix for service, see *Contacting Tektronix* on page xv of *Preface.*

Preparation

Read the following information before performing the incoming inspection procedure.

Description

The *Incoming Inspection Procedure* is divided into four parts:

- *Connect the VXIbus Test System* on page 1–19 provides instructions for setting up an example test system for this procedure.
- *Self Tests* on page 1–19 provides instructions for performing the internal self tests.
- *Functional Tests* on page 1–21 measures the time- and amplitude-reference signals at the REFERENCE OUTPUT connector.
- *Self Cal* on page 1–24 provides instructions for performing internal self calibration.

Test Equipment

The *Incoming Inspection Procedure* requires the following test equipment:

- VXIbus mainframe, such as the Tektronix VX1410 IntelliFrame
- Slot 0 controller, such as the National Instruments VXIpc-486 Series Model 566
- Computer peripherals for suggested Slot 0 controller: keyboard, monitor and, mouse
- Software capable of sending and receiving SCPI message-based commands, such as the TVS600A Soft Front Panel included with this product’s software
- One coaxial cable with BNC connectors
- One dual banana to BNC adapter

- Frequency counter (measures 10 MHz at <0.0025% accuracy), such as the Hewlett Packard 5314A
- Digital multimeter (measures +8 V at 0.25% accuracy), such as the Fluke 8842A

You can perform these tests using any system components that allow you to send commands to the waveform analyzer.

System Setup

The *Incoming Inspection Procedure* is designed for a VXIbus system that contains an embedded Slot 0 controller. During the procedure you will communicate with the waveform analyzer using message send/receive software such as TVS600A Soft Front Panel software, a standard accessory. Figure 1–5 on page 1–19 shows a typical setup.

The procedure can be performed using other Slot 0 controllers and talk/listen software. If you use an alternate setup, you might need to reformat the commands to work with your message send/receive software.

You may choose to perform the *Incoming Inspection Procedure* using the SERIAL INTERFACE connector on the front panel of the waveform analyzer. To do so, connect a terminal or computer COM port directly to the SERIAL INTERFACE connector. For terminal connection, use a 9-pin to 25-pin serial interface cable (Tektronix part number 012-1380-XX). A similar cable might work to connect your computer. On the computer, run message send/receive software that provides PC terminal emulation.

The waveform analyzer recalls the last RS-232 settings from memory at power on. Table 1–2 lists the factory default settings. Use the SYSTem:COMMunicate:SERial commands to modify the RS-232 parameters. The commands are listed in *System Commands*, on page 3–76, and in your *TVS600 & TVS600A Command Reference.*

Table 1–2: Factory default RS-232 settings

Parameter	Default Setting
Baud rate	9600
Stop bits	1
Parity	None
DCD	Off
Echo	On
LBUF	On
Pace	XON
RTS	On
ERES	On

Connect the VXIbus Test System

Perform the following steps to connect a VXIbus test system similar to the example shown in Figure 1–5. If you use a different system controller and slot 0 controller, you may need to reformat the commands used in this procedure.

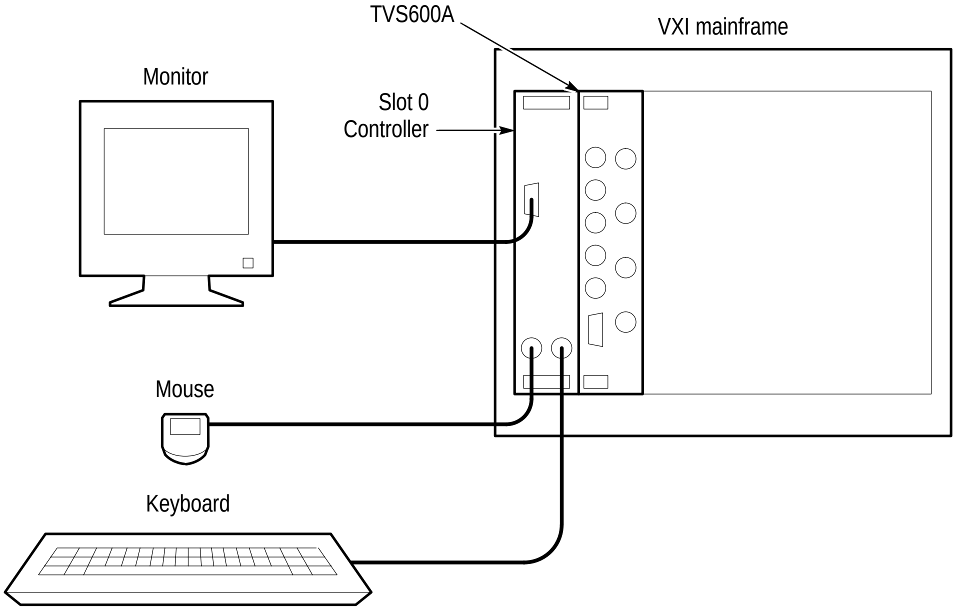


Figure 1–5: Example VXIbus test system for the *Incoming Inspection Procedure*

1. Perform the *Power-On Procedure* located on page 1–11.
2. Allow a 20-minute warmup. Then perform the *Self Tests* procedure that follows.

Self Tests

The self tests use internal routines to verify that the waveform analyzer is functional. No test equipment is required.

1. Send the following command to execute the internal self test routines:

   ```
   TEST
   ```

2. Wait for the self tests to complete.

   - When running self tests, the ARM’D, and TRIG’D indicators blink.
   - These tests take approximately 2 minutes to complete.

3. Send the following query to check the self test results:

```
TEST:RES?
```

4. Read the self test results.

- A 0 result indicates all tests passed successfully.
- A –1 result indicates the self tests are still in progress; wait two minutes and send the `TEST:RES?` query again to read the test results.
- A 1000 to 2999 result indicates self test failures and a need for service on the failed module. (See the page *Contacting Tektronix* on page xv.)

5. Proceed to the *Functional Tests* to continue the *Incoming Inspection Procedure*.

Functional Tests

The following procedures test the internal time and voltage references of the waveform analyzer. You will need a frequency counter, digital multimeter, coaxial cable with BNC connectors, and a dual-banana to BNC adapter to perform the *Functional Tests*. See the equipment requirements on page 1–17.

Measure Time Reference

This procedure tests the accuracy of the internal time reference (10 MHz ± 1 kHz).

1. Connect a coaxial cable from the frequency counter input to the REFERENCE OUTPUT connector (see Figure 1–6).
2. Send the following command to initialize the waveform analyzer:

   ```
   *RST
   ```

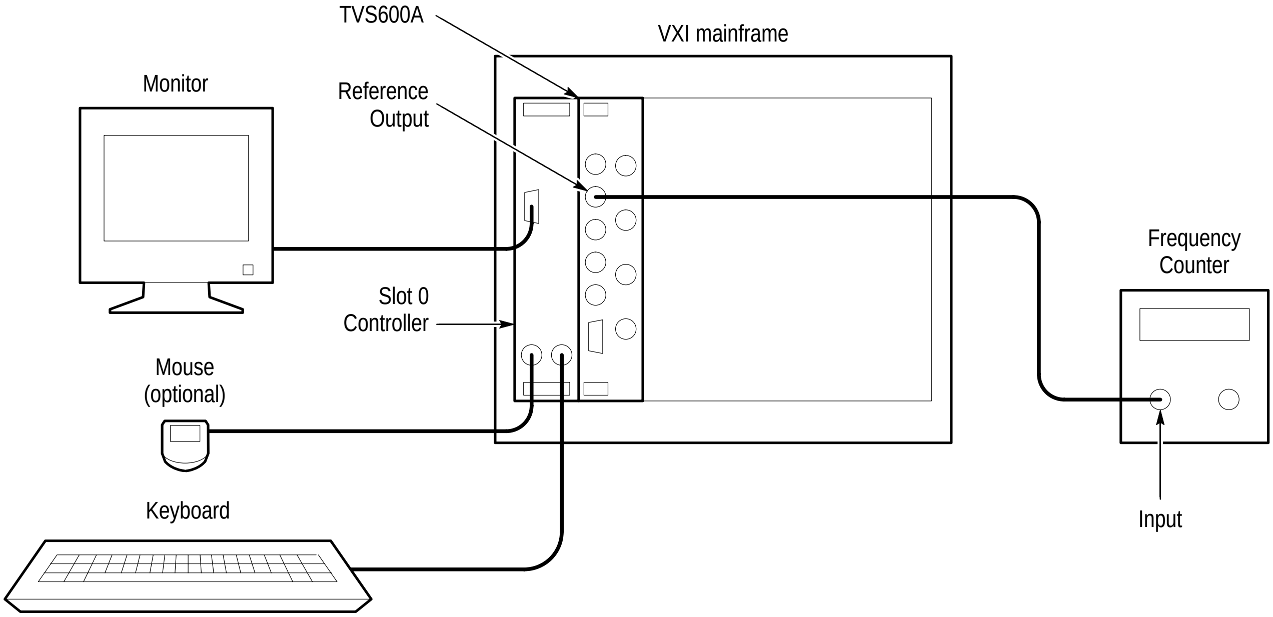


Figure 1–6: Time reference test setup

3. Select the following settings for the frequency counter:

Mode	Frequency
Trigger	Internal
Attenuation	X1

4. Send the following command to turn on the time reference:

```
OUTP:REF:FUNC CLOC
OUTP:REF ON
```

5. Check the frequency counter display. The frequency must be between 9,999,000 Hz and 10,001,000 Hz.

6. Disconnect the frequency counter from the REFERENCE OUTPUT connector.

7. Proceed to *Measure Voltage Reference* to continue the *Incoming Inspection Procedure*.

Measure Voltage Reference

This procedure tests the accuracy of the internal voltage reference (+8 V ± 1%).

1. Use a coaxial cable and dual-banana to BNC adapter to connect the digital multimeter input to the REFERENCE OUTPUT connector (see Figure 1–7).

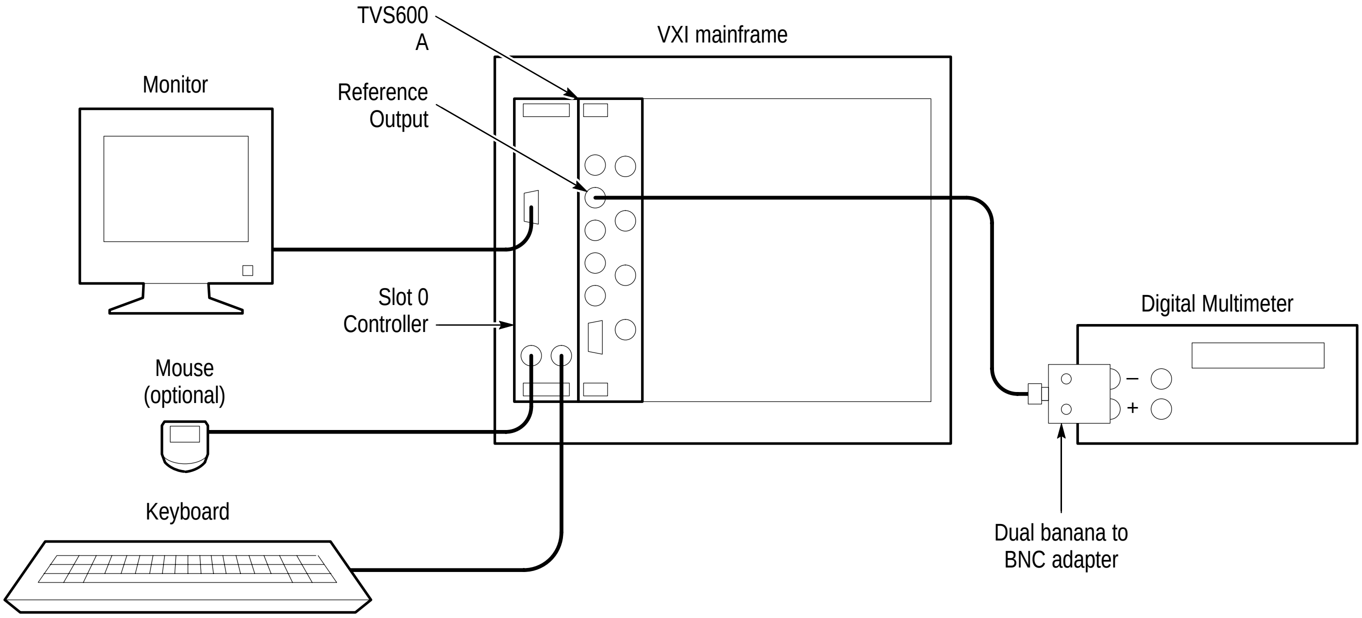


Figure 1–7: Voltage reference test setup

2. Send the following command to initialize the waveform analyzer:

```
*RST
```

3. Select the following digital multimeter control settings:

Mode	`DC Volts`
Scale	`20`

4. Send the following command to turn on the voltage reference:

```
OUTP:REF:FUNC VOLT
OUTP:REF ON
```

5. Check the digital multimeter display. The voltage must be between +7.92 V and +8.08 V.

6. Disconnect the digital multimeter from the REFERENCE OUTPUT connector.

7. Proceed to *Self Cal* to continue the *Incoming Inspection Procedure*.

Self Cal

The Self Cal, using internal routines and an internal time and voltage reference, generates data, such as gain and offset values, that optimizes the waveform analyzer performance at the current ambient temperature. The data is stored in memory and used until you perform another self cal. No test equipment is required.

1. Send the following command to execute the internal self cal routines:

   ```
   CAL
   ```

2. Wait for the self cal to complete.

   - When running self cal, the READY, ACCESSED, ARM'D, and TRIG'D indicators blink.

   - The self cal takes 5 to 7 minutes to complete.

3. Send the following query to check the self cal results:

   ```
   CAL:RES?
   ```

4. Read the self cal results:

   - A 0 result indicates all tests passed successfully.

   - A –1 result indicates the self cal is still in progress; wait five minutes and send the `CAL:RES?` query again to read the test results.

   - A 2000 to 2999 result indicates self cal failures and a need for service on the failed module.

This completes the *Incoming Inspection Procedure*. If all tests passed, the waveform analyzer is ready for use. If any test failed, refer the module for service (see *Contacting Tektronix* on page xv of *Preface).*

Operational Maps

This chapter acquaints you with how the TVS600A Waveform Analyzer functions and operates. It consists of two parts: *Operational Maps* and *Tutorial*.

This section, *Operation Maps*, includes the following information:

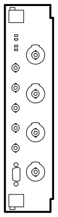
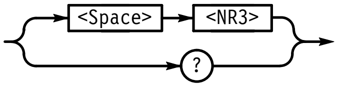
- *Operating Interfaces Map,* on page 2–3, describes the different tools provided for controlling the waveform analyzer and lists the documentation that supports each tool.
- *Hardware Interface Map,* on page 2–5, describes the elements of the waveform-analyzer front panel and cross references information relevant to each element.
- *Operations Overview and SCPI Model Map,* on page 2–9, describes the high-level operating process of waveform analyzer and relates it to the SCPI-command model.
- *Data Flow Model Map,* on page 2–11, describes how data flows between the input, the storage and calculation, and the output blocks.

Tutorial, on page 2–15, presents step-by-step procedures that shows you how to use the command set to configure the waveform analyzer to acquire waveforms and to perform many of its functions.

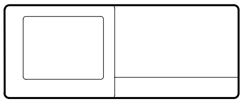
For information on configuring and installing your waveform analyzer, refer to Chapter 1, *Getting Started*.

Operating Interfaces Map

Although the SCPI commands form its main control “interface,” the waveform analyzer comes with other interfaces that offer some degree of control. Each interface is listed below and cross referenced to the document that supports it.

These Interfaces...	have these uses...	Check these documents...	to...
Hardware Interface 	Connect input signals Monitor acquisition progress Provide serial and VXIbus control interfaces	***User Manual***	Read about these inputs and indicators and their specifications. This manual also describes all features.
SCPI Commands 	Provide total setup and control of the TVS600A features Require most product knowledge to use Can be included in programs or sent using the SFP or other talker/listener program	***Quick Reference***	Quickly remind yourself of the syntax of a command
		User Manual, Command Reference	Explore the SCPI commands in depth: their syntax, options, and so on

TVS600A VXI*plug&play* Driver
`#include <tktvs600.h>`
`main(),`
`{`
`...`
`}`
	Provides specialized, application oriented, control of TVS600A features
	Require less product knowledge to use
	Can be called from programs
		Online Function Reference	Explore, on line, the usage and command syntax of the driver functions

TVS600A Soft Front Panel 	Provides a graphic user interface that can control many, but not all, features of the TVS600A Requires least product knowledge to use Can function as a talker/listener to send SCPI commands	***Online Help***	Explore, on line, the uses and operation of the soft front panel

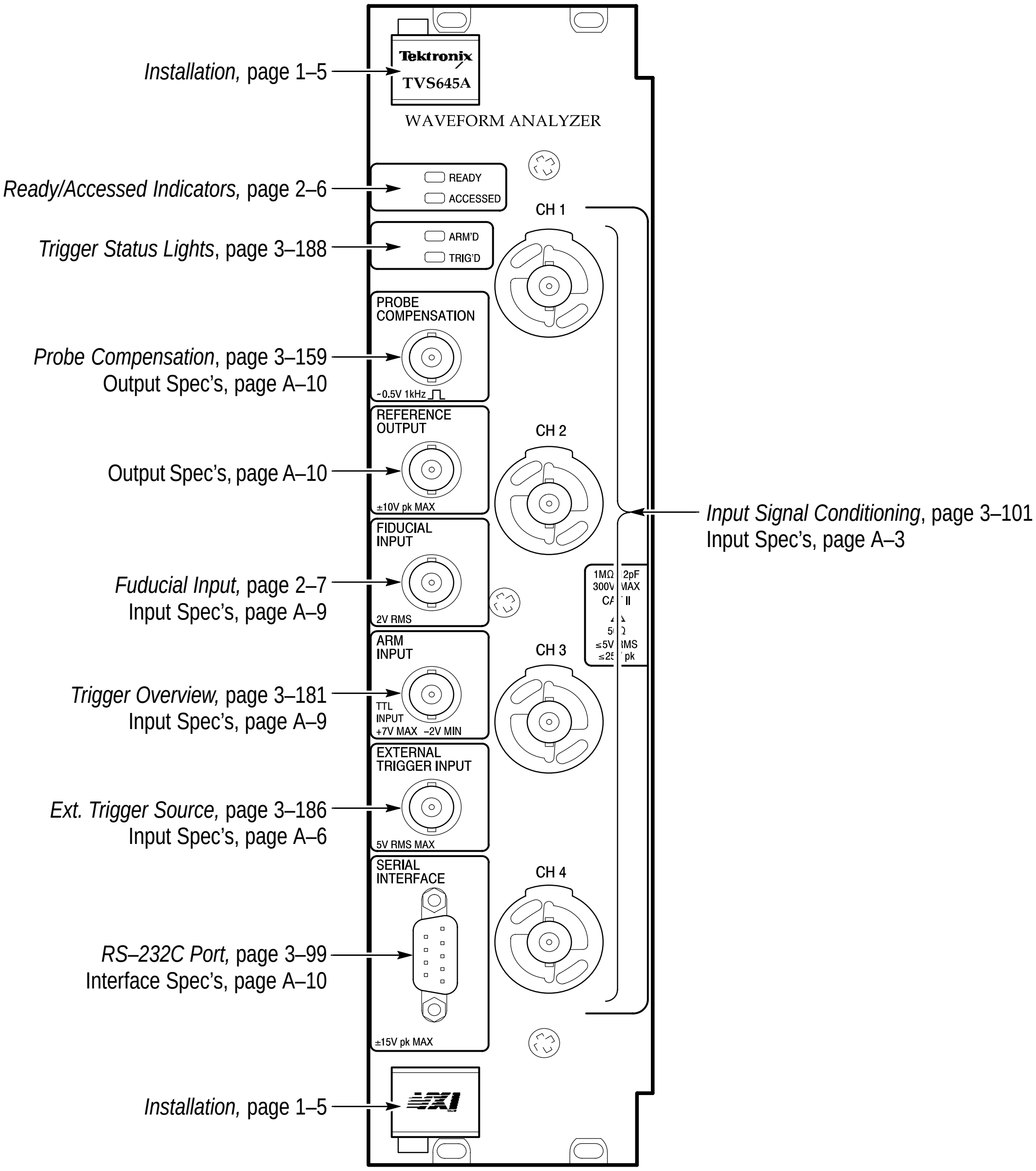
Refer to the documents just described for more information on each of the control interfaces listed.

Also, you can check the following references within this manual:

- For more information on the Hardware Interface, see the *Hardware Interface Map* on page 2–5.
- For more information on the Serial Interface and the VXIbus Interface, see the *Hardware Interfaces* on page 3–93.
- For more information on SCPI commands, see *Command Groups* on page 3–65 and *Command Syntax* on page 3–85. Also see your *TVS600 & TVS600A Series Command Reference* manual, which is your primary reference for TVS SCPI commands.

Hardware Interface Map

The hardware interface features a variety of input and output connectors for acquisition of and triggering on input signal,s and indicators for monitoring the acquisitions of those signals. The interface map below shows each connector and indicator with cross references to its description or its specification. Brief descriptions of each element follow the map.



CH1, CH2, CH3, and CH4 Channel Inputs. These BNC input connectors drive the vertical channel amplifiers and their dedicated digitizers. The TVS641A and TVS645A have four input channels as shown on page 2–5; the TVS621A and TVS625A provide only Ch1 and Ch 2. Each channel supports TEKPROBE Level 1 and Level 2 probes, which offer many features, including signal offset.

Each channel can be set to various input and signal coupling selections. See the references indicated on the map for more information; see also *Input Commands* on page 3–71 in this manual and in the *TVS600 & TVS600A Series Waveform Analyzers Command Reference*.

READY Indicator. The green LED lights continuously after the waveform analyzer powers up and completes power-on diagnostics successfully. During normal operation, READY blinks when an error occurs that generates a status message.

ACCESSED Indicator. The yellow LED blinks on and then off under the following conditions:

- When communication with the waveform analyzer occurs
- When the Slot 0 resource manager asserts the Module Identification (MODID) line

ARM’D Indicator. The green LED lights when the waveform analyzer is armed and ready to accept a trigger signal.

TRIG’D Indicator. The green LED lights briefly when a trigger occurs for the current acquisition. The TRIG’D Indicator lights continuously whenever trigger events occur more often than three events per second. Automatic triggers (automatic trigger mode) do not light the LED.

PROBE COMPENSATION. The BNC output provides a 1 kHz square wave for adjusting probe compensation. The signal amplitude is 0.5 V peak-to-peak into a 1 MΩ load. To enable the compensation signal, send the command `OUTP:PCOM ON`.

REFERENCE OUTPUT. The BNC output provides access to two internal references, the DC-calibrator reference voltage or the time-base clock. The precision-calibrator reference voltage (`VOLT`) is +8.0 V. The time-base clock (`CLOC`) is a 10 MHz square wave. Amplitude is ≥ 1 V into a 50 Ω load. To select a reference signal, send the command `OUTP:REF:FUNC CLOC` or `VOLT`. To enable the selected signal to the REFERENCE OUTPUT connector, send the command `OUTP:REF ON`.

FIDUCIAL INPUT. The BNC input provides a way to add a signal component to the Channel 1 input signal. Adding a common timing signal to Channel 1 on several instruments provides a way to improve cross timing between multiple instruments. The input range is ±1 V. The input resistance presents 0.01 μF in series with 50 Ω.

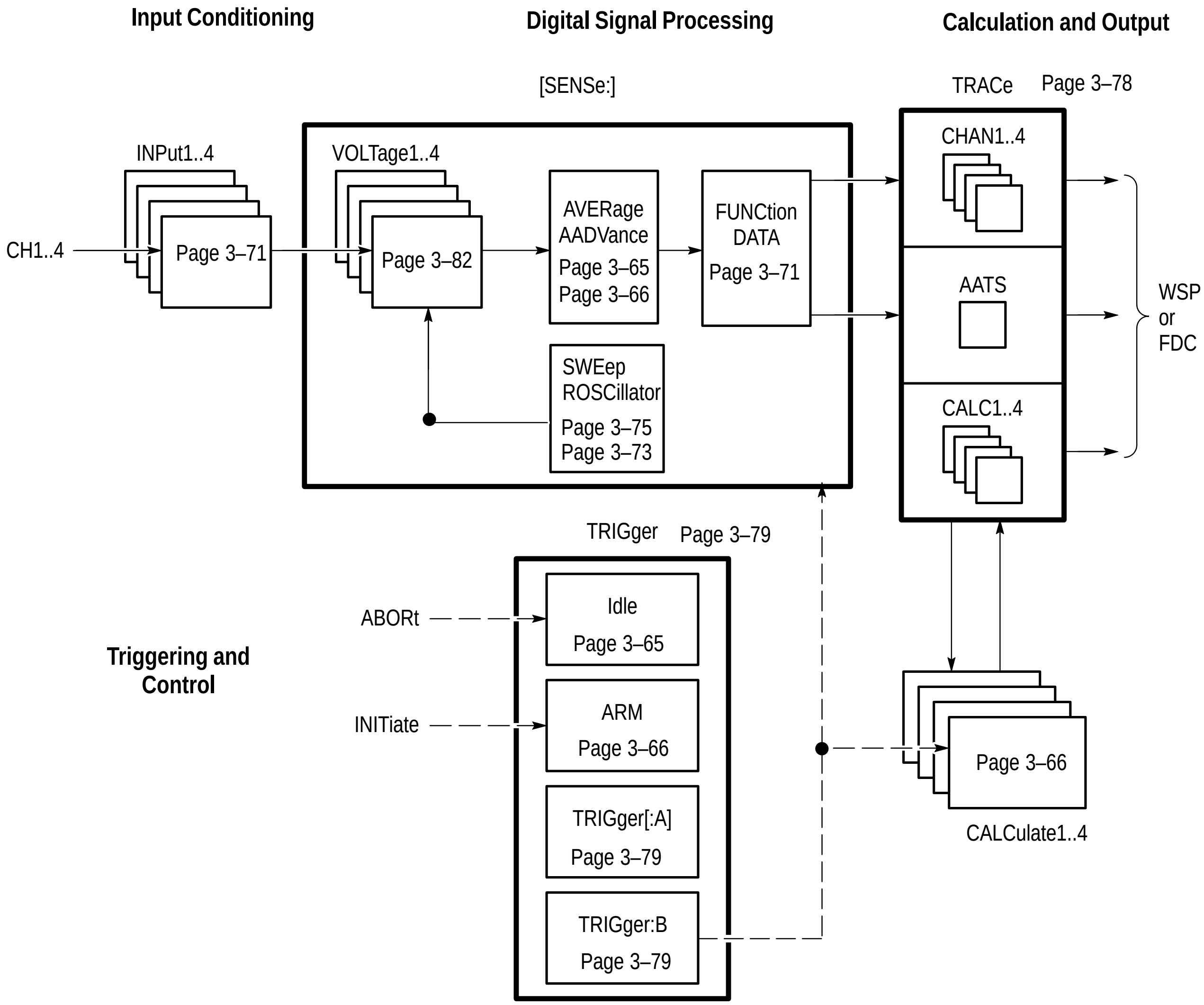
ARM INPUT. The BNC input allows you to arm the acquisition system by grounding the center lead. The ARM input is level sensitive and is not latched. An internal pull up resistor connected to +5 V maintains a high level until you ground the input. You must maintain a ground state on the center lead until the Trigger event occurs.

EXTERNAL TRIGGER INPUT. The BNC input provides a connection for an external trigger source. The input has 50 Ω impedance and is DC coupled only. Trigger signals as large as ±5 V (DC + Peak AC) may be applied.

SERIAL INTERFACE. The subminiature D connector provides a serial interface for controlling the waveform analyzer and reading acquired data. See *RS-232C Port* on page 3–99 for more information (including the RS-232 pin assignments in Figure 3–33). You can configure the serial interface with the commands in the `SYST:COMM:SER` subsystem.

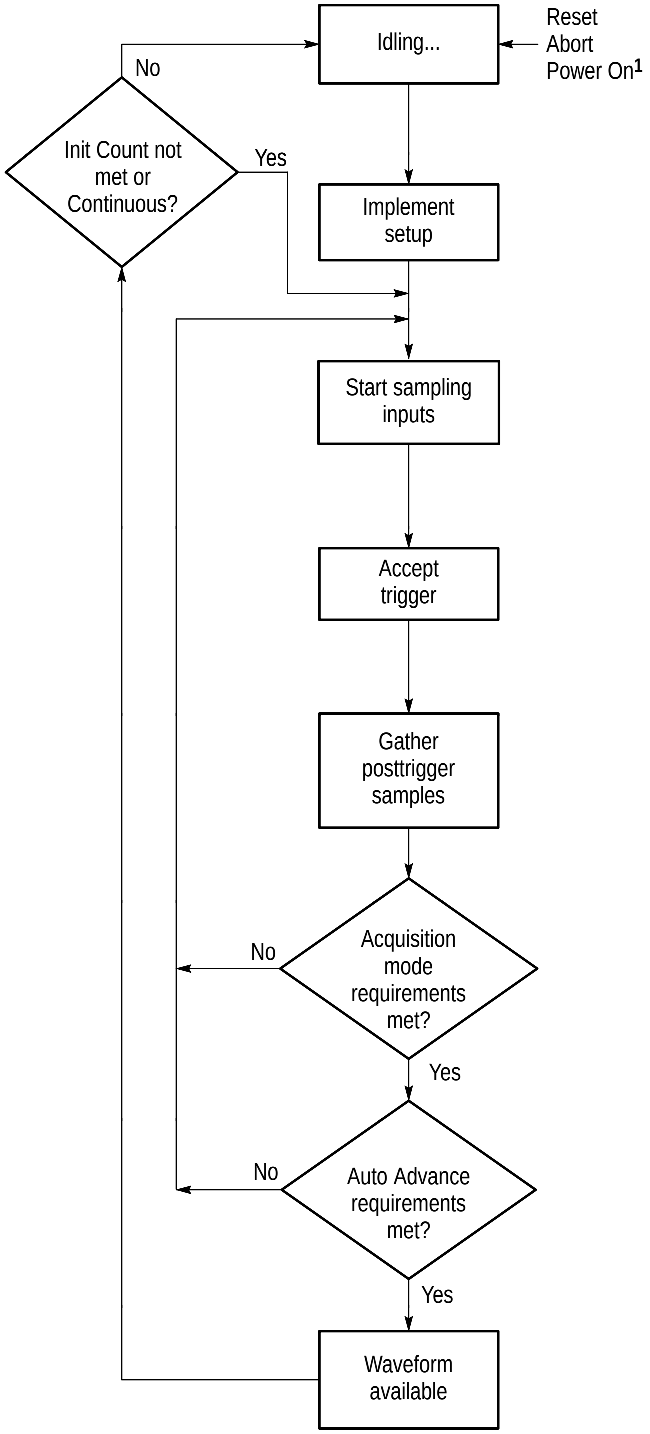
Operational Overview and SCPI Model Map

The TVS600A Waveform Analyzer is a VXI*plug&play* compliant product. It is fully programmable using commands that follow the SCPI 1995.0 standard for a sensing instrument.



The process overview that follows describes each step in the top-level cycle of waveform analyzer operation. When the description refers to a specific group of SCPI commands, these commands follow the organization in the *SCPI Model Map* above; refer to the map as you read the following process overview.

Process Overview


Idling...
Reset
Abort
Power On[1]
No
Init Count not met or Continuous?
Yes
Implement setup
Start sampling inputs
Accept trigger
Gather posttrigger samples
No
Acquisition mode requirements met?
Yes
No
Auto Advance requirements met?
Yes
Waveform available

Process Block Description

1. The waveform analyzer starts in the idle state; it enters this state upon power up, upon receiving reset (*RST) or ABORt commands, or upon finishing acquisition tasks.

2. Upon receiving the initiate command, the TVS600A implements its setup based on defaults and any SCPI commands it has received, such as those shown in the SCPI model on page 2–9.

3. The waveform analyzer then begins sampling signals coupled to its input channels. Sampling continues until enough pretrigger samples are acquired to satisfy requirements established by the SWEep commands; at that point, the trigger system is armed.

4. Sampling continues until the trigger requirements set by the TRIGger commands are met; at that point the trigger system gates the acquisition system. Trigger is based on inputs picked from the input channels, the external trigger input, or the VXI trigger bus.

5. Sampling continues until enough posttrigger samples are acquired to fill record length requirements set by the SWEep commands. At that point, one waveform record exists. If normal acquisition mode is on and auto advance is off, processing skips to step 8.

6. If averaging, enveloping, or peak-detect acquisition mode is on, the record becomes part of the multi-acquisition record that these modes produce. The process loops back to step 3 above to acquire additional records until the number of acquisitions specified for the acquisition mode are processed and then processing continues as for step 8 below.

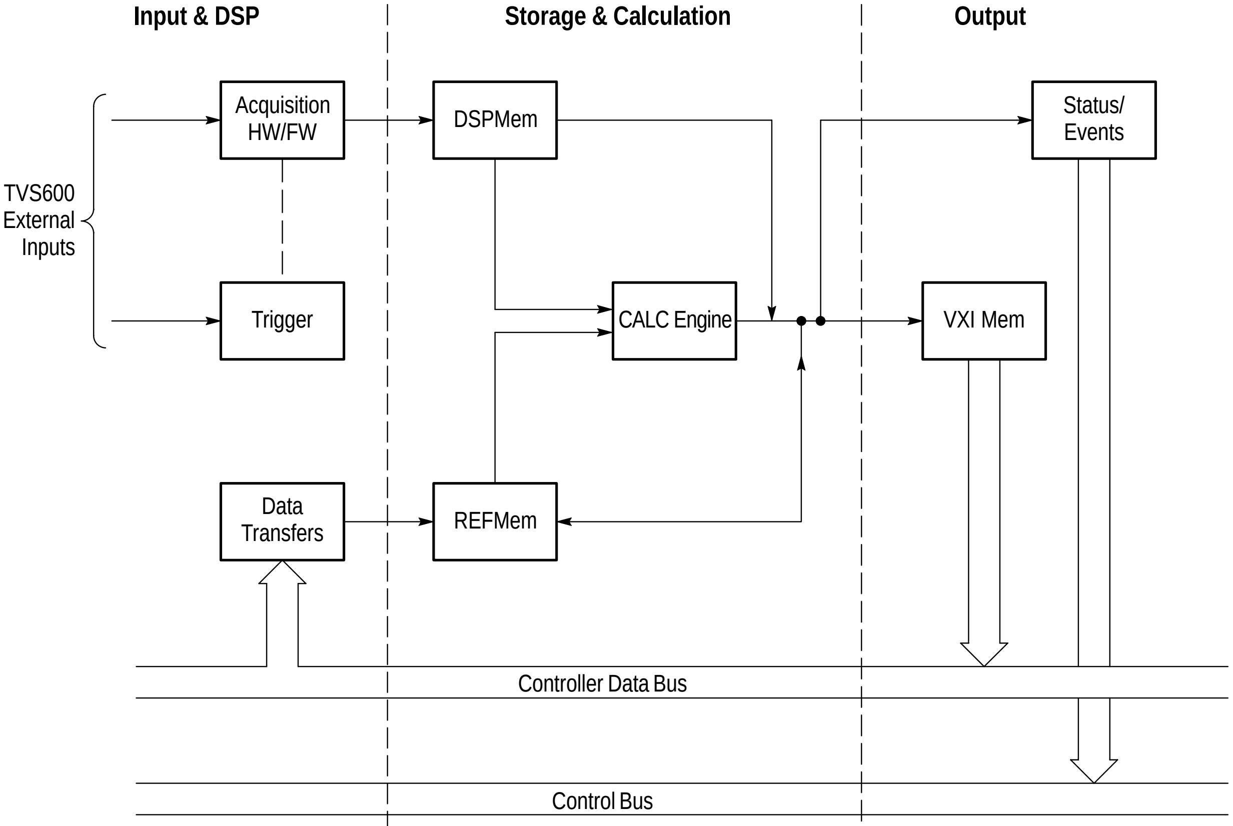
7. If auto-advance is on, the record is held in DSP memory while the process loops back to step 3 to acquire additional records specified for the auto-advance count. Processing then continues as for step 8 below.

8. At this point the acquisition record is in DSP memory and is available using the TRACE commands. The waveform analyzer then either returns to idle state or, if Init continuous is on or the init count is less than count, it returns to step 2 above.

[1] Note: if acquiring when powered down, the waveform analyzer skips the idle state and resumes acquisition starting with step 3.

Data Flow Model Map

You can study data flow in the waveform analyzer to learn how it functions and operates. The data flow through the functional blocks of the TVS600A is shown below. The various data-handling or processing tasks performed are also shown, each cross referenced to a description in this manual.



▪ Setup acquisition	page 3–3, 3–11	▪ Perform calculations	page 3–21, 3–43	▪ Output data to system	page 3–117
▪ Import waveform data	page 3–117	▪ Download waveforms	page 3–117	▪ Output control signals based on CALC results	page 3–62
▪ Setup input channels	page 3–101	▪ Measure waveforms	page 3–141		

Input & DSP

The input signals flow from left to right through their individual channels, where they are coupled, amplitude-scaled, and sometimes bandwidth-filtered by their respective input conditioning and voltage blocks (shown in *Operational Overview and SCPI Model Map* on page 2–9) before passing to the acquisition block. Samples of the input signals are routed to the trigger block for triggering acquisitions.

The Acquisition block samples and digitizes the signals from the channels to produce waveform records. (The timing of this process is described in the process overview on page 2–10 and in more detail under *Acquisition Cycle* on page 3–6). The Acquisition block can further process the acquired waveform records according to acquisition mode (average, envelope, or peak-detect) or into auto-advance records. (Acquisition modes and the auto-advance cycle are explained in *Acquisition Modes and Auto-Advance Cycle* starting on page 3–11.)

The Trigger system is shown below the Acquisition block. It receives samples of the input signals from the Input Conditioning block (as well as from other sources) and triggers the Acquisition block. (See *Triggering Overview* on page 3–181.)

Storage & Calculation

Within the Storage and Calculation block, newly acquired waveforms are stored in Acquisition memory. Waveforms and other data may also be downloaded over the system data bus and stored locally in individual references (REF1 – REF10). Downloadable data includes:

- Standard Y-vs.-T waveforms with preambles
- Envelope waveforms for use in template testing

To download the data just described, the waveform analyzer supports a limited subset of the Data Interchange Format (DIF) standard. Transfer of data is supported using the `TRACe` commands; see *I/O of Waveforms* on page 3–117.

The Storage and Calculation block can use all the data described, whether acquired or downloaded, for further processing according to user-specified criteria. The user can define calculations that use acquisition-memory waveforms and REF-memory waveforms or other data as data sources. Calculation results can be stored in REF memory or output to the data bus through the Output block. (The calculation system is described in *Calculation System Overview* on page 3–21.)

One other type of data may be stored in this block: data describing a waveform-analyzer setup can also be downloaded over the data bus. Such instrument setups are routed to and stored in DATA memories (10 available) under control of the `MEMory` commands; see *Saving and Recalling Settings* on page 2–23 and your *TVS600A Series Command Reference* for more information.

Output

Once acquired and processed, the Output block can transfer waveforms and calculation results to the system data bus. *Data Formats* on page 3–138 of *I/O of Waveforms* explains the format requirements of data transferred. Also, see the `TRACe`, `DATA?`, `CALCulate:DATA?` queries in *Command Groups* on page 3–65 of this manual and in your *TVS600 & TVS600A Series Command Reference*.

In TVS600A models, the Output block can also output signals based on calculation results using the Control/Notification functions. See *Control/Notification Functions* on page 3–62.

Tutorial

The examples presented here show you how to use the command set to acquire, average and envelope on, and perform calculations on an input signal. Each example builds on what you learned in the previous examples. Each example describes briefly what you will accomplish and then provides the step-by-step instructions to complete the task.

Required Equipment

You need the following equipment to run these examples:

- A signal generator capable of supplying a 10 MHz square wave.
- Two probes or BNC cables to connect the TVS600A input channels to the signal generator output.
- A signal splitter for BNC or probes to supply the square wave signal to two input channels. If using probes, you may need an adapter to properly connect the probes to the generator output. Refer to the manual for your probes to choose the correct adapter, if any are needed.

Host System Requirements

Before starting this Tutorial, make certain that your VXIbus system is installed and you have a talker/listener program running. Install your waveform analyzer and verify communication between it and your VXIbus controller before starting this tutorial. For instructions on how to install your waveform analyzer, refer to page 1–5.

You will need a way to send commands to the waveform analyzer. You can use the talker/listener program that accompanies the communications card in your controller. Refer to documentation that accompanies the card for instructions on starting the talker/listener application.

Alternately, you can use the talker/listener feature of the TVS600A Soft Front Panel software shipped with your system. It is installed as part of *Software Installation* process described on page 1–12. If you wish, you can enter commands from a terminal that you connect to the RS-232 connector on the waveform-analyzer front panel. For RS232 set up information, see *RS-232C Port* on page 3–99.

***NOTE**. All examples use the Word Serial protocol for communications.*

Entering Commands

In these procedures, SCPI commands are sometimes given in long form and sometimes in short form. You can type either form, just do not type an intermediate form. For abbreviation rules, see *Abbreviating Commands, Queries, and Parameters* on page 3–87.

Example 1 — Instrument Setup

This example describes the waveform analyzer settings you will typically make before starting to acquire signals. Correct initial set up is very important to ensure that you acquire good data. In this example, you will use the commands `INPut`, `VOLTage`, `SWEep`, and `TRIGger`. You will actually acquire a signal in Example 2 based on the settings you perform in this example.

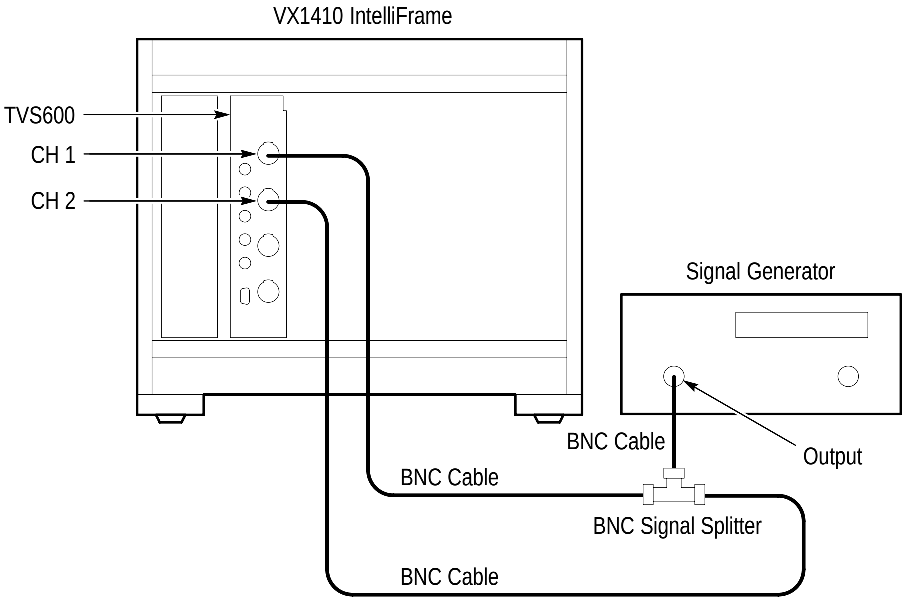


Figure 2–1: Initial equipment setup for the tutorial

1. Connect Channel 1 waveform analyzer input to the signal generator output using a probe or BNC cable and the signal splitter.

2. Configure the signal generator as follows:

   - Output level 4 Vp-p
   - Waveform type Sine Wave
   - Frequency 10 MHz
   - Offset 0 V
   - Enable output

3. Send the command `*RST` to reset the waveform analyzer.

   The `*RST` command resets instrument settings to their default values. Resetting the instrument is often the quickest way to set the instrument to a known state before starting to set up a new measurement. Communication port settings and data security commands are not affected by `*RST`.

4. Set signal coupling for CH 1 to DC by sending the command `INPut1:COUPling DC`.

   The `INPut1:COUPling` command sets the coupling to DC, which passes all signal components. Other commands in the INPut subsystem control input impedance (50 Ω or 1 MΩ) and a low-pass filter (20 MHz or 250 MHz).

   You can specify the identical command using the minimal spelling INP:COUP DC. To ensure portability of your test programs, use the short form of the commands. The short form is used through the remainder of this tutorial.

5. Set the input voltage range for the CH 1 input by sending the command `VOLT1:RANG:PTP 5`.

   The `VOLT1:RANG:PTP` command sets the full-scale range for the selected channel to 5 V. The voltage range setting should be just larger than the signal you wish to acquire to ensure the best accuracy and resolution of acquired data. Places where the input signal exceeds the range setting are recorded as overrange or underrange values in the waveform record.

6. Now set the time base by sending the command `SWE:TINT 2E–9`.

   With this `SWE:TINT` command you set the time between samples, the sample interval, to 2 ns. With the default record length of 1024 points, you will acquire just over 20 cycles of the 10 MHz input signal. To derive this value, multiply the record length of 1024 intervals/record by 2 ns/interval to get 2048 ns record duration.

7. Set the trigger level to 1 V by sending the command `TRIG:LEV 1`.

   The `TRIGger:LEVel` setting applies to the source `INTernal1`, which is the CH 1 input. `TRIGger:SOURce` was set to `INTernal1` by the `*RST` command sent in earlier in this example. You could set the `TRIGger:SOURce` to the CH 2 input by specifying `INT2`. Note that when you do not specify `:A` or `:B` in TRIGger commands, `TRIGger:A` is set.

8. Enable the input CH 1 by sending the command `FUNC CHAN1`.

   The command `FUNC CHAN1` enables, or connects, the CH 1 signal from the INPut subsystem to the acquisition hardware, which is controlled by the SENSe subsystem.

The commands you have used in this example are necessary for most acquisitions. In other cases, you might adjust for a DC offset with the command `VOLT<n>:RANG:OFFS` or place the trigger point in the middle of the acquisition record with the command `SWE:OREF:LOC 0.5`. The `<n>` in the `VOLT<n>` command is replaced by a channel number, such as `VOLT2` or `VOLT1`.

You have completed Example 1 and should proceed to Example 2 where you will acquire a waveform.

Example 2 — Acquiring a Signal

This example shows you how to start acquiring a signal based on the setup described in Example 1. You will initiate acquisition and retrieve the acquisition record. The example ends with a discussion of the steps necessary to acquire signals on multiple channels.

The hardware setup is the same as that used in Example 1.

1. Perform the initial set up described in Example 1.
2. Start acquisition by sending the command `INITiate`.

   The `INITiate` command starts the acquisition system. After initiation, the system looks for any defined ARM condition then looks for a Trigger A event.

   You defined the Trigger A event to be a positive transition through the 1 V level. If you had specified an ARM source, such as the front panel ARM INPUT, then the acquisition system would wait for the ARM signal before looking for a trigger event.
3. Get the acquisition record just acquired from CH 1 by sending the command `DATA? CHAN1`.

   Due to the `*RST` sent earlier, the acquisition record is returned as a series of 1024 ASCII numbers separated by commas. You can change the data format to binary with the `FORMat` command.
4. To acquire a second channel, simply set the CH 2 vertical range with `VOLT2:RANG:PTP 5` and enable the channel with the command `FUNC CHAN2`. Then send the INIT command to start acquisition.

   The settings for the time base (SWEep) and trigger (TRIGger A) are shared by all input channels.

The sequence of commands you used in this example is necessary to insure that signal acquisition starts correctly. Note that the `DATA?` command waits for completion of acquisition or calculations before it starts to transfer the resultant data record or measurement value.

You have completed Example 2 and should proceed to Example 3 where you will acquire an averaged signal and an enveloped signal.

Example 3 — Averaging and Enveloping a Signal

This example shows you how to acquire an averaged signal and an enveloped signal. Averaging removes noise from your signal which improves the dynamic range of the acquired waveform and the accuracy of any measurements you perform on it. Enveloping creates a waveform that contains the maximum and minimum values at each sample point over a number of acquisitions.

1. Reset the waveform analyzer with the `*RST` command and then perform the initial setup in Example 1.
2. Enable signal averaging by sending the command `AVER ON`.

   Averaging acquires a specified number of waveforms and averages them together to produce an averaged waveform.
3. Set the number of acquisitions to average by sending the command `AVER:COUN 16`.
4. To acquire an averaged waveform send the initiate command `INIT`.
5. To transfer the averaged waveform out of the instrument, use the same `DATA` command you used in Example 2, `DATA? CHAN1`.
6. To configure the instrument to acquire an envelope waveform send the command `AVER:TYPE ENV`.
7. Start envelope acquisition with the command `INIT`.
8. Now get the envelope waveform with the command `DATA? CHAN1`.

   Envelope is a special type of averaging acquisition. The default type of averaging is `SCALar`. Enveloping saves the maximum and minimum values acquired for each sample point during the 16 acquisitions (`AVER:COUN 16`) you specified in step 3. The result is an acquisition record containing 512 pairs of interleaved maximum and minimum points. The waveform record still contains 1024 points and covers the same time period as a normal acquisition record.

You have completed Example 3 and should proceed to Example 4. In Example 4 you will use the CALCulate system to perform a rise time measurement and integrate a signal in real time.

Example 4 — Performing Basic Calculations

This example shows you how to perform two types of SCPI calculations. First, you will configure the CALCulate system to measure the rise time on the CH 1 signal and retrieve the result. Then you will integrate a signal and retrieve the waveform record of the integrated signal.

1. Perform the steps in Example 1 in preparation to acquire CH 1.

2. Specify the CH 1 signal as the input to the CALCulate1 block by sending the command `CALC1:FEED CHAN1`.

   The `FEED` command specifies the source that the CALCulate1 block will operate on. The source here is the CH 1 signal vertically scaled by the VOLTage1 block. Hence, the "source" or `FEED` is the VOLTage 1 block. The waveform analyzer instruments have four CALC blocks that operate on any of the input channels. For example, we could perform the rise time measurement by specifying `CALC2:FEED CHAN1`. You may find it easier to keep track of operations when you match CALC blocks to the input channel numbers.

3. Set up the CALC1 block to perform the rise time measurement by sending the command `CALC1:WML RTIM`.

   `RTIM` specifies the rise time measurement. You can specify several measurements at once by adding commands after `RTIM` separated by commas. For instance, the command `CALC1:WML RTIM,PTP,MEAN` specifies rise time, peak-to-peak and mean measurements. The parameters proximal, mesial, and distal used by the measurement system, are controlled by `CALCulate :WMParameter` commands. The defaults are used here.

4. Enable the measurement list for CALC1 block by sending the command `CALC1:WML:STAT ON`.

   This command activates the CALC1 block so it will perform its list of measurements, `RTIM` in this example. You must set the STATe of a CALC block to ON before the CALC block will perform any measurements.

5. Initiate the acquisition process as you did in the previous Examples.

   Active measurements, `RTIM` in this instance, are performed when the acquisition process ends. If you are averaging a waveform, the measurements are performed when all acquisitions have been averaged.

6. Retrieve the rise time result by sending the command `CALC1:DATA?`.

   `CALC1:DATA?` returns all results from the CALC1 block. In this case, the rise time value is returned as a floating point number in ASCII format. When you send `CALC1:DATA?` all results from calculations and measurements defined for the CALC1 block are returned as a comma separated list.

7. Set the CALC2 block to operate on the CH 1 signal by sending the command `CALC2:FEED CHAN1`.

   Note that the CALC1 block is still configured to perform the rise time measurement (`RTIM`).

8. Set the CALC2 block to integrate the CH 1 signal by sending the command `CALC:INT:STAT ON`.

9. Initiate the acquisition process as you previously did.

10. Retrieve the integrated waveform result by sending the command `CALC2:DATA?`.

    `CALC2:DATA?` returns all results from the CALC2 block. In this case the returned result is the integral waveform calculated on the CH 1 input signal. If you send the command `CALC1:DATA?` you will get the rise time measurement performed by the CALC1 block on this last acquisition. Note that all data returned from a CALC block are composed of ASCII characters separated by commas.

You have completed Example 4 and should proceed to Example 5. In Example 5 you will define a calculation using an algebraic expression.

Example 5 — Performing Advanced Calculations

This example introduces the expression model for defining calculations. The expression model provides a powerful tool for measurement and signal processing. You will find that setting up a measurement or calculation is easier with the expression model than with the standard SCPI model. Note that the expression model may not be available on other digitizer products with a SCPI command set.

In this example, you will calculate a waveform that is the CH 1 signal minus its mean value.

1. Reset your waveform analyzer and perform the initial set up in Example 1.

2. Set the CALC1 block to calculate the signal minus its mean by sending the command `CALC1:PATH:EXPR (CHAN1–MEAN(CHAN1))`.

   This command sets the CALC1 block to first compute the arithmetic mean (or DC level) of the CH 1 acquisition record. Then it subtracts the mean value from each sample value in the record. Incidentally, this calculation cannot be defined using the standard calculation model defined by SCPI.

   Note that, when using the expression model, you do not need to set the `WMList`, `STATe`, or `FEED` for the chosen CALC block.

3. Initiate the acquisition process as you previously did.

4. Retrieve the calculated waveform by sending the command `CALC1:DATA?`.

You have completed Example 5 and should proceed to Example 6. In Example 6 you will save the current settings of the waveform analyzer into an on-board settings location. Then you will recall the setting.

Example 6 — Saving and Recalling Settings

This example shows you how to save the current waveform analyzer settings in an on-board location and then recall the settings. You will configure CH 1 and CH 2 to acquire signals and perform measurements on those signals.

1. Perform the initial set up described in Example 1 to enable CH 1.

2. Connect the signal generator to CH 2. CH 1 should already be connected. Set the CH 2 vertical range and enabling the channel with the command chain `VOLT2:RANG:PTP 5;:FUNC CHAN2`.

   A command chain is a series of commands separated by semicolons (;). You can send commands and queries as command chains.

3. Configure the CALC1 block to measure the rise time on CH 1 by sending the CALC1 source and measurement commands `CALC1:FEED CHAN1;:CALC1:WML RTIM`.

4. Configure the CALC2 block to measure the peak-to-peak value on CH 2 by sending the CALC2 source and measurement commands `CALC2:FEED CHAN2;:CALC2:WML PTP`.

5. Enable the measurement list for both CALC blocks by sending the command `CALC1:WML:STAT ON;:CALC2:WML:STAT ON`.

6. Initiate the acquisition process as you did earlier.

7. Retrieve the two measurement results by sending the command `CALC1:DATA?;:CALC2:DATA?`.

   The data returned by these chained queries are separated by a semicolon. You do not need to test your settings before saving them, though it is always good to verify your setup before making a measurement.

8. Save the current waveform analyzer instrument settings in location 1 by sending the command `*SAV 1`.

   All acquisition and calculation settings are saved in the settings location number 1. You can reuse a complex test setup by saving your final instrument configuration in a settings location. Instrument settings that control communications interfaces and input protection are not saved. You may store ten instrument settings in on-board memory that is retained when power is off.

9. In order to test the restore capability, reset the instrument by sending the command `*RST`.

10. Check that CH 1 signal feed is now disabled by sending the command `FUNC?`.

With the instrument reset, you should get the reply `""`, which is an ASCII null string that indicates no channels are enabled. If channels were enabled you would get a string such as `"XTIM:VOLT 1","XTIM:VOLT 2"`, which indicates CH 1 and CH 2 are enabled.

11. Now restore your settings from location 1 by sending the command `*RCL 1`.

12. Check that CH 1 and CH 2 signal feed is now enabled by sending the command `FUNC?`.

13. You can test the retrieved settings by initiating acquisition and retrieving the the new measurement results as you did in step 7.

You have completed Example 6 and should proceed to Example 7. In Example 7 you will work with the waveform analyzer status and event reporting system.

Example 7 — Using Status and Events

This example shows you how to use the status and event commands to determine the status of your waveform analyzer. Command, execution, and other system errors plus other system events are stored in a memory buffer on-board. You can use the event information to better control the waveform analyzer and enhance overall system performance.

***NOTE**. If you are entering commands over the RS-232 port, send the command* `SYST:COMM:SER:ERES OFF`*. This command disables automatic return of events from the error/event queue to the RS-232 port. Automatic echoing of events occurs only on the RS-232 port.*

1. Reset your waveform analyzer and perform the initial set up in Example 1 to enable CH 1.
2. Enable all events in the Standard Event Status Register (SESR) by sending the command `*ESE 255`. Figure 2–2 shows the SESR. You can enable any or all events in the SESR so they register in the Status Byte Register.
3. Create a command error by sending the incorrect command `CALC1:XXX?`.

   This syntax error for the root node CALC causes a command error. Figure 2–2 shows that when a command error occurs, its bit is set to one in the SESR.

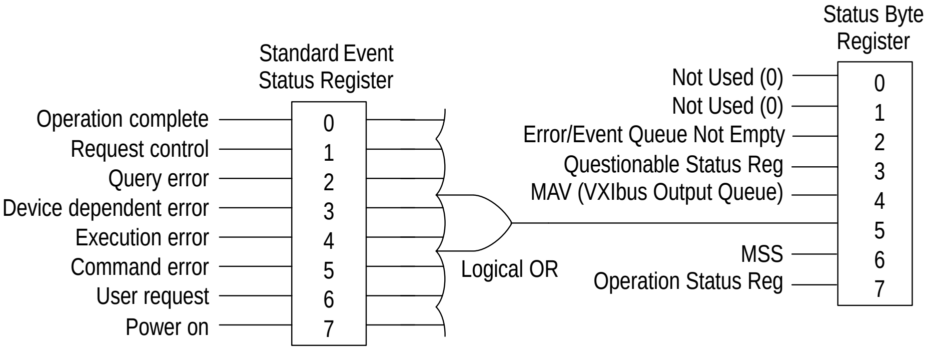


Figure 2–2: Standard Event and Status Byte Registers

4. Check the Status Byte Register (SBR) for the command error by sending the query `*STB?`.

   This query returns the status byte from the SBR. The status byte contains a decimal number that indicates errors or other significant events. Figure 2–2 shows the bit assignments. Your response should have bit five set for a

Command error. Unlike the other waveform analyzer status registers, the SBR output simply mirrors its inputs so it is not cleared when you read it.

5. To get specific information on the type of error, read the SESR by sending the command `*ESR?`.

   The returned decimal number has bit five set and possibly others. Note that reading the SESR, or any other event register, clears it.

6. To get the actual error message from the error/event queue send the command `SYST:ERR?`.

   The response should be the error `Execution Error -113, "Undefined header, unrecognized command - CALCI:XXX"`. Reading an event message from the event/error queue removes it from the queue. By repeating the `SYST:ERR?` query, or by sending the command `SYST:ERR:ALL?`, you can read all buffered events.

   Note that you can read the next event from the event/error queue without first checking the Standard Event Status Register or the Status Byte Register.

7. Generate a command error by sending the command `CALC1:WML FTM`

   This command has an incorrect spelling for the parameter FTIM (fall time).

8. Try clearing the events by sending the reset command `*RST`.

9. To check if the error message is still in the error/event queue send the command SYST:ERR?.

   You should get the error `Execution Error -141, "Invalid character data, CALC1:WMC FTM"`. To ensure that only current events and errors are in the event queue you should always clear the errors before you start a new measurement or test.

10. To clear the event registers and all messages from the error/event queue, send the command `*CLS`.

    Besides reading the registers and queue, only the *CLS command, or shutting off power, can clear the status registers and error/event queue.

11. Check for the command error that you created in step 7 by sending the command `*ESR?`.

    You will get a response of `"0"` unless you have generated other errors since the last reset.

You have completed Example 7, which is the last of the Tutorial examples. For more information on specific commands refer to the command descriptions in your *TVS600 & TVS600A Series Waveform Analyzer Command Reference*. For more detailed information on the waveform-analyzer features, refer to Chapter 3 *Reference*.

Overview

This chapter describes in depth how the many features of the TVS600A Waveform Analyzer operate. Sections for the features describe them, list procedures that show how to set up and operate the feature using SCPI commands, and list driver functions that provide similar capability as the SCPI commands. (Driver functions are part of the TVS600A VXI*plug&play* software included with the waveform analyzer.)

The table that follows on page 3–2 lists operating tasks and the sections in this chapter that document those tasks.

Tasks or Topics	Subtasks or Subtopics	Section Title	Contents	Page No
Data Input	Acquiring waveforms	*Acquisition Overview*	Background on acq. modes and cycles	3–3
		Acquisition Modes and Auto-Advance Cycle	Set up of the acquisition system	3–11
		Autoset and Reset	Automatic setup of the acquisition, triggering systems and input channels	3–17
		Triggering Overview	Background on basic trigger operation	3–181
		Trigger Types	Set up of triggering system for edge, pulse, logic, and transition triggering	3–193
	Downloading waveforms, templates, or other data	*I/O of Waveforms*	Set up for downloading of any data to the waveform analyzer	3–117
		Template Testing (TVS600A Only)	Set up for downloading of templates to use in testing of waveforms	3–165
Data Processing (Calculation)	Background needed to do the subtasks below	*Calculation System Overview*	Background on the Calculation system needed to understand math, DSP, measurement, and limit-test operations	3–21
	Math (+.–,/,*) advanced DSP operations	*Calculation Functions*	Functions for processing waveforms, extracting segments of waveforms, etc.	3–43
	Taking Measurements	*Measurements*	Set up for measuring parameters	3–141
	Limit tests	*Template Testing (TVS600A Only*	Set up for testing waveforms against templates	3–165
		Limit Testing Measurements	Set up for testing measurement results against limits	3–152
Data Fetching	All uploading of waveforms, calculation results, and other data	*I/O of Waveforms*	Primary reference for uploading of any data from the waveform analyzer	3–117
Probe Cal	Compensation or calibration of probes	*Probe Calibration*	Procedures for compensating passive probes and calibrating active, voltage probes	3–159
Reference	Commands	*Command Syntax*	Primer on constructing SCPI commands and queries	3–85
		Command Groups	Brief Reference for waveform-analyzer commands by functional group	3–65
	Communications Interfaces	*Hardware Interface*	Descriptions of the RS-232C port and VXIbus interface	3–93
	Calc expressions	*Expression Syntax*	Definition of the BNF forms allowed for CALC expressions	3–34
		Expression Operators	Descriptions of all math, Boolean, assignment, etc. operators that CALC expressions support	3–36
		Calculation Functions	Descriptions of all math, advanced DSP, waveform, and control/notification functions of the Calculation system	3–43

Acquisition Overview

To process data, the waveform analyzer must either import it over the system data bus or acquire it, by sampling and digitizing signals connected to its input channels. This section provides background on the acquisition process:

- Describes the acquisition hardware
- Defines the sampling process, sampling modes, and the waveform record
- Details the acquisition cycle in normal and auto-advance versions
- Briefly describes the acquisition modes

See the topic *Acquisition Modes and Auto-Advance Cycle* on page 3–11 for information on use of the acquisition modes (normal, envelope, peak-detect, and average).

Acquisition Hardware

Before a signal can be acquired, it must pass through the input channel where it is filtered, scaled, and digitized. Each channel has a dedicated input amplifier and digitizer as shown in Figure 3–1; each channel can produce a stream of digital data from which waveform records can be extracted. (See *Input Signal Conditioning* on page 3–101 for further description of input channels.)

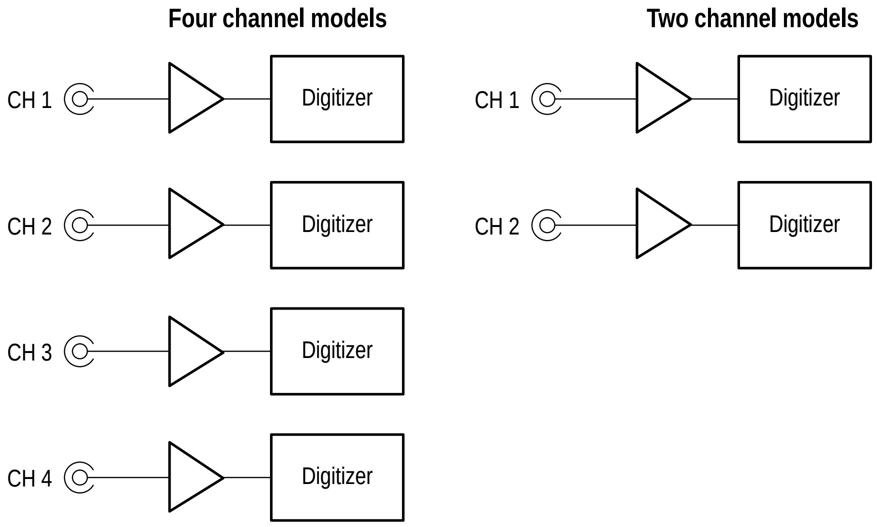


Figure 3–1: Digitizer configuration

Sampling Process

Acquisition is the process of sampling an analog input signal of an input channel, converting it into digital data, and assembling it into a waveform record, which is stored in acquisition memory. Sampling, then, is the process that provides a continuous stream of digitized signal data from which the waveform analyzer assembles the waveform record (see Figure 3–4 on page 3–6).

The waveform analyzer samples the voltage level of the input signal at regular time intervals. The signal parts within the vertical range of the amplifier are digitized. See Figure 3–2.

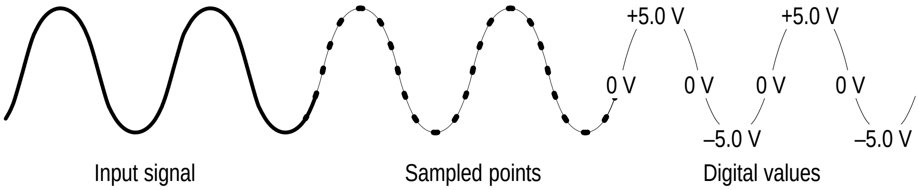


Figure 3–2: Digital acquisition — sampling and digitizing

The resulting flow of sampled points becomes a “data stream” from which the waveform analyzer extracts a waveform record and stores it in acquisition memory. The waveform record includes the signal source, trigger point location, and horizontal and vertical scaling.

The waveform analyzer acquisition system can process the data as its acquired, averaging or enveloping the waveform data to produce enhanced waveform records. (Refer to *Acquisition Modes and Auto-Advance Cycle* on page 3–11.) Once the waveform record exists (enhanced or not), you can use the CALCulate subsystem to further process that record: perform measurements, waveform math, data transforms, and so on.

Sampling Modes

The waveform analyzer provides two modes of input sampling: real time and extended real time. Either sampling mode may be used with both the normal and auto-advance acquisition cycles. (Figure 3–5 on page 3–7 shows both normal and auto-advance acquisition cycles.) Real-time sampling provides acquisitions with up to 15,000 point record lengths; extended real-time sampling provides acquisitions with up to 30,000 point record lengths.

Real Time Sampling. With real-time sampling, the waveform record is filled serially by the digitizers based on a single trigger event. Figure 3–3 shows how real-time sampling occurs in a linear fashion. Because a real-time acquisition requires only one trigger event, real-time sampling is effective for capturing rare or non-repeating events. Real-time sampling is the normal sample mode after the reset command, *RST.

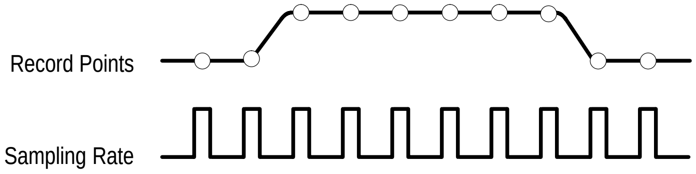


Figure 3–3: Real-time acquisition

Extended Real-Time Sampling. With extended real-time sampling, you can acquire a waveform record up to 30,000 points. The waveform record fills serially with only one trigger event. Extended real-time sampling is available whenever the sample interval is 100 ns or longer (≤10 M Samples/sec.); no explicit command is necessary. However, you do need to explicitly set the record length to 30000 after switching to extended real-time acquisition if you want the longest record.

Waveform Record

The waveform record for any given channel is taken from the continuous stream of samples (data stream) digitized by the waveform analyzer (see *Sampling Process* on page 3–4). The section or segment of this data stream that the waveform analyzer cuts out to form each channel's waveform record is determined by a set of common parameters ("common" means they affect the waveforms in all channels). These common parameters include the sample interval, record length, trigger point, and the number of samples acquired before (pretrigger) and after (posttrigger) the trigger point.

Figure 3–4 shows how the common parameters define the waveform record; as shown in the figure, they define where in the data stream and how much data is taken. Locate the following parameters in the figure:

- Sample Interval. The precise time between sample points taken during acquisition.
- Record Length. The number of samples required to fill a waveform record.
- Trigger Point. The trigger point marks the time zero in a waveform record. All waveform samples are located in time with respect to the trigger point.

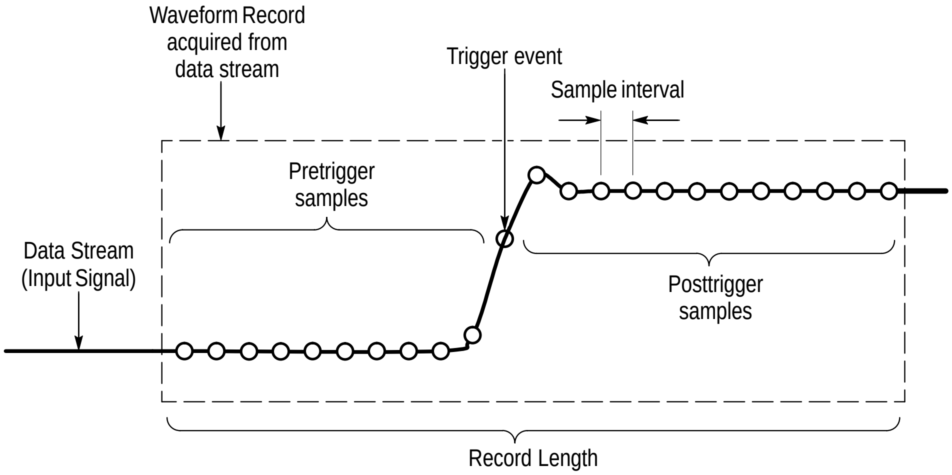


Figure 3–4: The waveform record and its defining parameters

These parameters are set with commands from the Sweep and Trigger groups; see the following titles for details:

- *Horizontal Scaling of Waveforms* on page 3–109
- *Triggering Types* on page 3–193

Also see the SWEEp and TRIGger:A commands in your *TVS600 & TVS600A Series Waveform Analyzers Command Reference.*

Acquisition Cycle

The waveform analyzer can be set to either of two acquisitions cycles: the normal cycle (default after *RST) or the auto-advance cycle. Both cycles have common steps, but loop through them differently (see Figure 3–5). A brief description of each step, or acquisition block, follows:

- The INITiate command is always required to start acquisition. INIT puts into effect any changes to the waveform analyzer setup, and then activates the acquisition system to await an arm or trigger signal.
- During the pretrigger time, the acquisition system initializes itself and the digitizer acquires any pretrigger record points specified with the SWEep commands.
- Before the trigger enabled event occurs, any specified ARM event must occur and all required pretrigger samples must be acquired. Once enabled, the next valid TRIGger:A event is accepted. (See Figure 3–62 on page 3–183.)

- During the posttrigger waveform acquisition period, the active channel digitizers acquire all posttrigger waveform samples needed to fill the defined waveform records.
- In the data out of acquisition memory block, the digitizers transfer acquired data out of acquisition memory and into DSP memory.

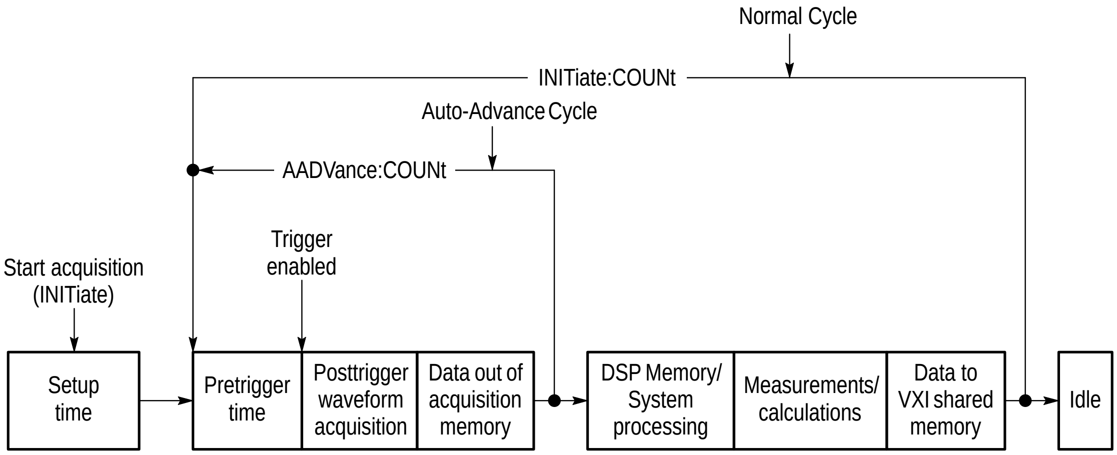

Normal Cycle
INITiate:COUNt
Auto-Advance Cycle
AADVance:COUNt
Trigger enabled
Start acquisition (INITiate)
Setup time
Pretrigger time
Posttrigger waveform acquisition
Data out of acquisition memory
DSP Memory/ System processing
Measurements/ calculations
Data to VXI shared memory
Idle

Figure 3–5: The acquisition cycle

Normal Acquisition Cycle. This acquisition cycle (default after *RST) performs the steps just described and then immediately performs post-acquisition processing (system processing, any measurements or calculations specified, and so on) of each waveform record after its acquired. (Refer to Figure 3–5). As processing completes, the resultant waveform records are moved to VXIbus shared memory for transfer to your system controller.

Auto-Advance Acquisition Cycle. This acquisition cycle performs the steps just described but then bypasses the post-acquisition processing (see Figure 3–5) until the number of waveforms you specify are acquired. This acquisition cycle provides the shortest acquisition rearm time. For more information, see *Auto-Advance Acquisition* on page 3–13.

Acquisition Modes

The waveform analyzer can use the following acquisition modes:

- Normal. Samples once per sample interval for each record point, with no further processing during acquisition.
- Average. Accumulates an average value for each record point over many acquisitions to provide higher vertical resolution.

- Peak Detect (TVS600A models only). Finds the highest and lowest sample values over a single acquisition, obtaining a maximum/minimum sample pair for every consecutive pair of sample intervals.
- Envelope. Finds the highest and lowest sample values over many acquisitions to produce a waveform record of maximum and minimum values (an envelope).

Sampling in normal mode (default after *RST) and in peak-detect mode is shown in Figure 3–6; averaging and envelope mode, shown in Figure 3–7, derive from processing normally sampled data during the acquisition cycle.

Averaging is valuable for improving the signal-to-noise ratio (SNR) on repetitive signals. If the noise is random, SNR improves approximately 3 db for each power of two (doubling) of the number of averages. The waveform analyzer uses floating-point data, so the added resolution is preserved.

Peak-detect mode is especially useful for detecting fast events at slow sample rates; lower sample rates allow observation of longer-duration records while the peak-detection hardware captures narrow events (glitches).

The commands in the AVERage subsystem control all three acquisition modes. All active channels are affected by the acquisition mode selected. Averaging, peak detecting, and enveloping occur in the acquisition system before waveform records are passed to the CALC blocks. Waveforms acquired in one of these modes are often more suitable for a particular application; for example, you might use averaging mode to average out random noise in your waveform.

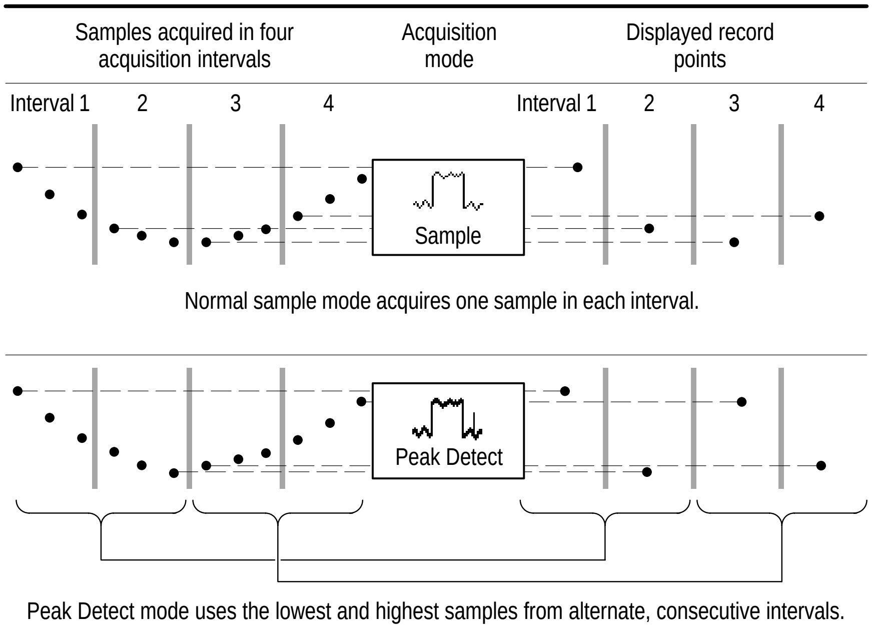

Samples acquired in four acquisition intervals
Acquisition mode
Displayed record points
Interval 1
2
3
4
Interval 1
2
3
4
Sample
Normal sample mode acquires one sample in each interval.
Peak Detect
Peak Detect mode uses the lowest and highest samples from alternate, consecutive intervals.

Figure 3–6: Normal and Peak Detect

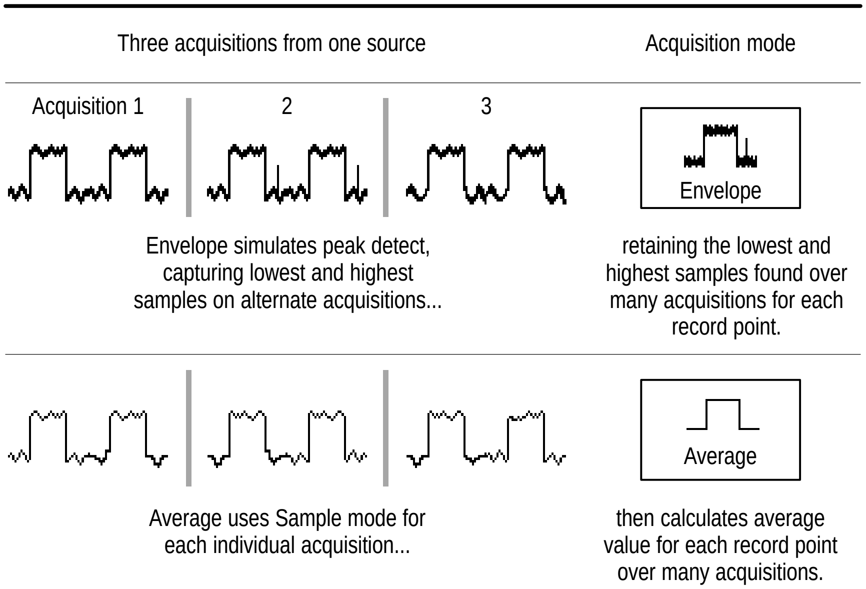

Three acquisitions from one source
Acquisition mode
Acquisition 1
2
3
Envelope
Envelope simulates peak detect, capturing lowest and highest samples on alternate acquisitions...
retaining the lowest and highest samples found over many acquisitions for each record point.
Average
Average uses Sample mode for each individual acquisition...
then calculates average value for each record point over many acquisitions.

Figure 3–7: Envelope and Average

Acquisition Modes and Auto-Advance Cycle

To acquire data (signals), the waveform-analyzer acquisition hardware must be set up properly. The modes and special acquisition cycle discussed here control how waveform data is acquired and processed. In other words, they control the nature of the waveform record that is output to the control bus or delivered to the calculate system for post-acquisition processing. This section includes:

- *Acquisition Modes* (Normal, Average, Envelope, and Peak-Detect), page 3–11
- *Auto-Advance Acquisition* (with Time Stamping), page 3–13

NOTE. *The acquisition parameters that set acquisition rates, record length, and offset are described in* Input Signal Conditioning *on page 3–101.*

Acquisition Modes

Average and envelope acquisition modes combine a number of acquisitions in one waveform record as follows:

- Normal (default) makes one acquisition per waveform record.
- Average accumulates an average value for each record point over many acquisitions to provide higher vertical resolution and noise suppression.
- Peak detect finds the highest and lowest sample values over a single acquisition to produce a waveform record of alternating maximum and minimum values.
- Envelope finds the highest and lowest sample values over one or more acquisitions to produce a waveform record of alternating maximum and minimum values.

All active channels are affected by the acquisition mode set. Averaging, peak-detecting, and enveloping occur in the acquisition system before waveform records are passed to the CALC blocks.

Why Use?

Use envelope mode when you want a waveform record that contains the extremes of waveform variations over many acquisitions; use peak-detect mode when you want a waveform record that contains the extremes over one or more acquisitions. Use average mode when you want a waveform record that averages the results of several acquisitions, such as when you want to reduce random noise in the record to produce a more accurate measurement.

***NOTE**. Peak-detect mode can capture subinterval spikes normally missed in other acquisition mode making peak-detect the optimal mode for glitch detection.*

To Use

The SCPI commands treat all the acquisition modes except normal as different types of averaging. You can set the acquisition modes as follows:

1. To select normal acquisition mode, send `AVERage OFF`; to enable one of the other acquisition modes, send `AVERage ON`. The mode enabled depends on the selection made in step 2.

2. To select the mode enabled when `AVERage` is `ON`, send one of the following commands:

   - `AVERage:TYPE SCALar` to select average mode
   - `AVERage:TYPE ENVelope` to select envelope mode
   - `AVERage:TYPE PEAK` to select peak-detect mode

3. To set the number of acquisitions to process into the waveform record, using the acquisition mode set in step 2, send `AVERage:COUNt:<arg>`, where `<arg>` is `MINimum`, `MAXimum`, or an integer lying within minimum (1) and maximum (4096).

Commands

The commands and functions to select the acquisition mode and count follow:

Used to:	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
select between normal mode or mode set by the `AVER:TYPE` command	`AVERage OFF or ON`	`tktvs600_setStdAcq`	Yes
select average mode	`AVERage:TYPE SCALar`		Yes
select envelope mode	`AVERage:TYPE ENVelope`		Yes
select peak detect mode	`AVERage:TYPE PEAK`		Yes
select number of acquisitions to average, envelope or peak detect	`AVERage:COUNt <n>`		Yes

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Functions Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application**

Usage Notes

Some usage notes follow:

- The commands in the AVERage subsystem control averaging, enveloping, and peak detecting; the AVERage subsystem treats these modes as *types* of averaging.
- Any commands, functions, or SFP-control changes that restart acquisition will restart the averaging, peak-detect, or enveloping process, discarding previously averages or envelopes and beginning anew.
- Averaging, peak-detecting, and enveloping are acquisition processes that affect all four channels simultaneously.
- You cannot use average, peak-detect, or envelope modes when using the auto-advance mode.

Auto-Advance Acquisition

Auto-advance acquisition acquires a sequence of waveform records with minimal delay between acquisitions. It behaves as follows:

- Auto-advance acquisition provides the fastest acquisition cycle time because the system processing, measurement, and data transfer tasks are delayed until the specified number of acquisitions complete (see Figure 3–5 on page 3–7).
- Auto-advance acquisition includes a timestamp which records the time between consecutive waveform records in the auto-advance acquisition sequence. Timestamp resolution is 125 ns.

Many of the basic settings for auto-advance acquisition are the same as for normal acquisition. You select the input channels and define the record length and sample interval.

Why Use?

Use auto-advance acquisition when you want to decrease the probability of missing short-term events in a data stream that might occur during the longer dead time required by the normal acquisition cycle. Consider auto-advance mode when you need to increase the waveform capture rate and can use normal acquisition mode.

To Use

To configure auto-advance acquisition, you must turn it on and specify the number of waveform records to acquire. First set the acquisition parameters and enable the input channels in the same manner as when making any acquisition. (See *Input Signal Conditioning* on page 3–101.) Then do the following steps:

1. To turn on auto-advance acquisition, send `AADVance ON`.

***NOTE**. Before setting the number of records to acquire (step 2), you must turn on the channels to acquire and set the waveform record length. See step 3 of the procedure on page 3–102 to turn on channels and the entire procedure for setting up the horizontal window on page 3–113 to set record length.*

2. Select the number of records to acquire before doing post-acquisition calculation: send `AADVance:COUNt <arg>`, where <arg> is :

- `1` to `MAXimum`
- `MINimum` (1 acquisition)
- `0` (an acquisition count sufficient to fill acquisition memory)
- `MAXimum` (a number that depends on overall instrument set up)

Both 0 (zero) and Maximum set up the acquisition count to fill acquisition memory, but 0 will adjust the acquisition count to keep filling memory when the number of active channels changes; MAXimum will not.

3. Select the number of records to return from those acquired: send `AADVance:RECord:COUNt <arg>`, where <arg> is :

- `1` to `MAXimum`
- `MINimum` (1 acquisition)
- `0` (transfer all records)
- `MAXimum` (a number that depends on overall instrument set up)

4. Select an index into the sequence of auto-advance records from which to count (count specified in step 3): send `AADVance:RECord:STARt <arg>`, where <arg> is :

- `–(MAXimum + 1)` to `(MAXimum – 1)`
- `MINimum` (1, the first record)
- `0` (last record)
- `MAXimum` (a number that depends on overall instrument set up)

Negative settings select waveform records referenced from the last one, the zero record. For example, the –1 record is the second from last and the –2 record is third from last.

5. Send `INITiate` start acquiring.

6. Send `DATA?` return the specified number of records.

The example *FDC Example* on page 3–128 also shows how auto advance works.

Commands

The commands and functions that set up auto-advance acquisition follow:

Used to:	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
switch from normal to auto-advance cycle	`AADVance ON (or OFF)`	`tktvs600_setAAdvAcq`	No
set number of auto-advance records to acquire	`AADVance:COUNt`		No
set number of auto-advance records to return	`AADVance:RECord:COUNt`		No
set start index for :RECord:COUNt	`AAVance:RECord:STARt`		No

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Functions Reference.***

[3] **If so indicated, feature can be set using the Soft Front Panel application**

Usage Notes

Some usage notes follow:

- You must turn on the channels as well as the record length you will use with auto-advance before setting number of records to return. See *Note* on page 3–14.
- You can set :`COUNt` to fill acquisition memory with waveform records by giving `:COUNt` the value `MAX` or `0` (zero).
- You cannot use average, peak detect, or envelope modes when using the auto-advance cycle.
- During auto-advance acquisition, the time between consecutive waveform records is recorded in a timestamp record. The timestamp record has the trace name `AATS` and is available with the transfer commands, such as `TRACe?` and `TRACe:COPY`.
- The timestamp record contains a sequence of times in seconds, from t0 to tn, separated by commas. Set the format of the AATS record to ASCII or 32-bit REAL numbers with the command `FORM:TRAC:AATS`.

Autoset and Reset (TVS600A only)

The TVS600A Waveform Analyzer can be automatically set up to trigger on and acquire a waveform. It can also be reset to its factory default setup. This section describes how to execute Autoset and Reset.

Autoset

Autoset automatically sets up the waveform analyzer control parameters based on the characteristics of the input signal. When operating waveform analyzer manually, autoset is typically much faster and easier to use than setting each control parameter individually. Autoset adjusts control settings in these categories: Vertical, Horizontal, and Trigger; definitions of each autoset follow.

- Voltage (vertical) autoset. Sets `VOLTage:RANGe:UPPer` and `VOLTage:RANGe:LOWer` so that the incoming signal fills the center 90% of the peak-to-peak range of the vertical window. Vertical offset is set to zero.
- Trigger autoset. Sets the `TRIGger[:A]:SOURce` to the channel you specify and the `TRIGger:[A:]LEVel` to 50% of the peak-to-peak value of the trigger signal.
- Sweep (horizontal) autoset. Sets `SWEep:TINTerval` to the nearest value such that an acquired record contains 2.5 cycles of the incoming signal on the specified channel.

Any autoset times out after approximately 20 ms in absence of an adequate input signal, leaving the target settings of the autoset unchanged. Incoming signals must be repetitive at 50 Hz or faster for autosets to reliably occur.

Why Use?

Provides a quick way to change basic vertical (`VOLTage`), horizontal (`SWEep`), and trigger (`TRIGger:A`) setups based on the input signal you want to acquire. Autoset is convenient to use when the characteristics of the signal being acquired are not known, such as when troubleshooting.

You can also use Autoset to invoke a quick, general setup by first sending a reset command (`*RST` common command), then using the `FUNCtion` command to turn on the channel you wish to autoset, and finally sending the three autoset commands. Then you need only to send the commands to modify the setup to best suit your purposes, rather than doing the complete setup from scratch.

To Use

Perform the following example procedure to learn how to use autoset:

1. To produce the quick, general autoset described above, first send `*RST`.

2. Turn on the channel you wish to autoset on; for example, send: `FUNC:ON CHAN1`.

3. Now send the command `SYSTem:AUToset:VOLTage CHAN<n>;SWEep CHAN<n>;TRIGger CHAN<n>`, where <n> is the number corresponding to the channel upon which to base the autoset. When part of a global autoset, `SYSTem:AUToset:VOLTage` should be sent first.

4. To autoset in only one of the categories, send the appropriate command:

   - `SYSTem:AUToset:VOLTage CHAN<n>`
   - `SYSTem:AUToset:SWEep CHAN<n>`
   - `SYSTem:AUToset:TRIGger CHAN<n>`

Commands

The commands and functions to autoset the waveform analyzer follow:

Use to:	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
autoset input characteristics for channel	`SYSTem:AUToset:VOLTage`	`tvs600_configAutoSetup`	Yes
autoset horizontal for channel	`SYSTem:AUToset:SWEep`		Yes
autoset trigger for channel	`SYSTem:AUToset:TRIGger`		Yes

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Functions Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be executed using the Soft Front Panel application.**

Usage Notes

Some usage notes follow:

- The channel you specify determines the input signal that autoset bases its settings on. Autoset ignores channels other than the one specified; you must turn on the channel you specify with the `FUNCtion:ON` command.
- You can specify different channels for voltage, sweep, and trigger autosets.
- To autoset vertical and horizontal scales and triggering, you must send all three commands. When sending more than one autoset command, send `SYSTem:AUToset:VOLTage` first.
- It is the user’s or programmer’s responsibility to preset all related settings such as enabling the channel, bandwidth filters, couplings, vertical settings, and so on.
- All three types of autoset can be executed from the TVS600A Soft Front Panel.

- After autoset completes, you can query the appropriate setup parameters to determine their setting changes as follows:

Type	Query

`:AUToset:VOLTage`	`VOLTage:RANGe:UPPer?`
	`VOLTage:RANGe:LOWer?`
	`VOLTage:RANGe:PTPeak?`
	`VOLTage:RANGe:OFFSet?`

`:AUToset:SWEep`	`SWEep:TINTerval?`
`:AUToset:TRIGger`	`TRIGger:LEVel?`

- `AUToset:TRIGger` sets trigger level as appropriate to trigger on the specified trigger source. If no triggers are found, then the trigger level will be unchanged. `AUToset:TRIGger` leaves the waveform analyzer set to that source, but does not change other trigger parameters that might prevent triggering. You can send reset before autosetting the triggers.

Reset

Reset automatically sets many of the waveform-analyzer control parameters to a default set up.

Why Use?

Sets the instrument to a known, “general purpose” setup, which you can tailor to meet your test setup needs. (Reset defaults may be determined by sending *LRN after a reset.)

Also, reset sent and followed immediately by sending autoset provides a quick way to set up the vertical (VOLTage), horizontal (SWEep), and trigger (TRIGger:A) control parameters based on the input signal you want to acquire.

To Use

To reset the waveform analyzer, send the system command *RST.

Reset affects most settings, but many settings in the SYStem and STATus subsystems are not changed.

Commands

The commands and functions to reset the waveform analyzer follow:

Used to:	System Commands[1]	Driver Functions[2]	SFP?[3]
reset to a default setup	`*RST`	`tktvs600_reset`	Yes
return ASCII listing of default settings after a reset	`*LRN?`	`tktvs600_getAsciiSettings`	Yes

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Functions Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be executed using the Soft Front Panel application.**

Listing Autoset or Reset Defaults

After sending a autoset or reset, you can determine the resulting settings by sending the IEEE 488.2 common command *LRN. The *LRN command returns the current state of the waveform analyzer as a sequence of ASCII settings.

Calculation System Overview

This section introduces the waveform-analyzer calculation system. After data is acquired (or imported into) the waveform analyzer, the CALCulate commands can be used to perform the following operations on the data:

- Calculate waveform functions: add, subtract, multiply, and divide waveforms, differentiate and integrate waveforms, perform Fast Fourier Transforms on waveforms, and filter frequency components of waveforms
- Calculate parametric measurements, such as amplitude, on waveforms
- Extract and process segments of waveforms
- Compare waveforms against limit templates

This section introduces the CALC engine in its two forms, the SCPI model and the Expression model. General procedures for using both models are included. This section also lists the expression syntax used with the later model and concludes with a table of operators that can be used in CALC expressions.

The section that follows, *Calculation System Functions,* describes the various waveform functions that the CALC engine can call; the sections *Measurements*, on page 3–141, and *Template Testing*, on page 3–165, discuss the calculation of measurements and the comparison of waveforms against limits.

The CALC Process

In general, a calculation is a post-acquisition process that occurs after a waveform record is acquired and stored in acquisition memory. TVS600A models can also perform calculations on waveforms stored in references (internal memory locations). The generic process for calculation follows:

1. You set up the waveform analyzer to acquire the waveform(s) you want to perform calculations on.

2. You send CALC commands to define one or more calculations using one or more CALC blocks. These CALC blocks define the inputs (calculation sources) and operations on these inputs. The CALC block *is* the calculation.

3. You initiate an acquisition.

4. You send the `CALC:DATA?` query (or `TRACE:DATA?`) to fetch your measurement results.

The procedure just described is the high-level process you go through to do calculations; the only variation is when you recalculate or perform new calculations on data already acquired or stored in references, in which case, you skip step 1 and replace step 3 with a `CALCulate:IMMediate` command (see step 6 on page 3–27).

The two methods or models for defining calculations follow:

- With the standard SCPI model, you define a signal source and a series of functions to execute on the source.
- With the expression model, you describe a calculation with an algebraic expression using signal sources as variables.

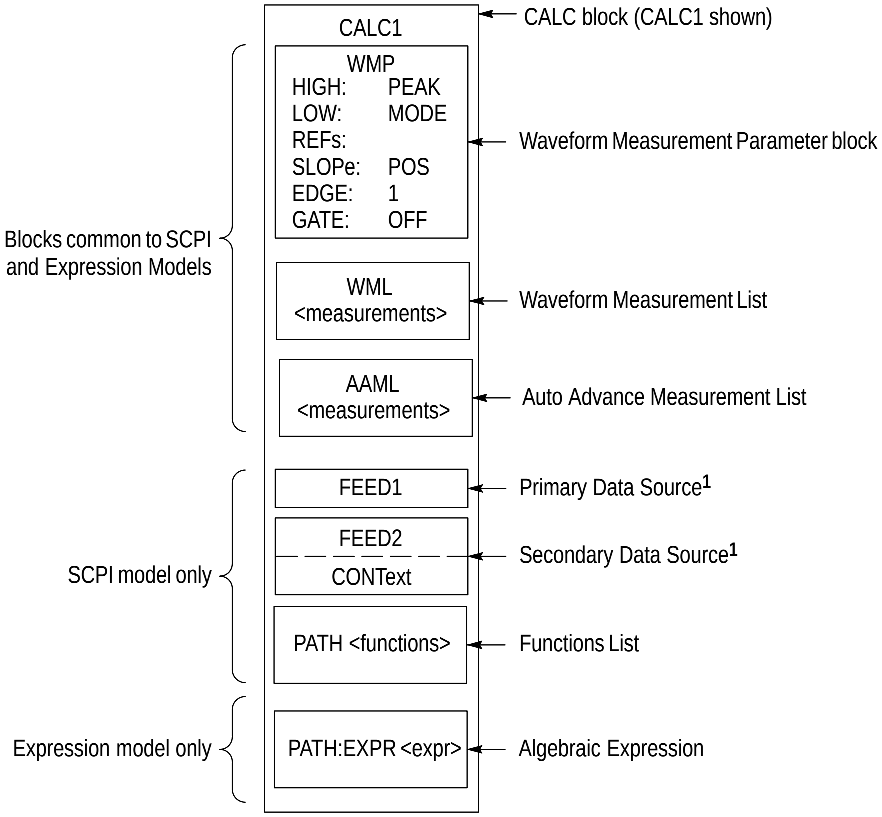
The two CALC models follow these rules:

- Only one of the two CALC models may be used at a time with an individual CALC block, but if you define multiple CALC blocks, each can use either model.
- Both models allow you to use all CALC functions, such as FILTer and TRANsform.
- In general, the SCPI model allows only calculations that operate on one source at a time. (Measurement calculations using two waveforms, such as gain, phase and delay, are exceptions to the one-source limitation.)
- The expression model allows calculations that operate on one or more than one source at a time, such as when adding the waveforms in channel 1 and channel 2.

A detailed discussion of each model follows, preceded by a description of the CALC block, which is used by both models.

The CALC Blocks

The waveform analyzer supports four separate CALC blocks, CALC1 through CALC4, which you use to define up to four separate calculations. Each CALC block is a calculation. Both SCPI and Expression models use the CALC blocks, but not to the same extent. Figure 3–8 illustrates the composition of a CALC block and the model-dependencies of its components; descriptions of the components follow the figure.



[1] These sources also have a limited use with the Expression model; see *Feed1* and *Feed2* descriptions in text. Feed2:CONText has no use in expressions.

Figure 3–8: Anatomy of a CALC block

CALC Block (CALC1 shown). One of four user-defined measurements.

Waveform Measurement Parameter Block (CALC1:WMP shown). Specifies parameters used to characterize the data source for both CALC models. Each of the four separate CALC blocks has its own, separately defined Waveform Measurement Parameters block (WMP). You must set the measurement parameters in each CALC block or accept the default settings. The `CALC:WMParameter` commands set up each CALC block.

The WMP block contains settings for measurement variables, levels, and methods that control how a waveform is characterized, for example, how its high-, middle-, and low-amplitude levels are defined. These measurement parameters are described in Appendix B, starting on page B–1; see also *Measurement Parameters* on page 3–141.

Waveform Measurement List (CALC1:WML shown). Specifies the measurements available to the CALC block. For both CALC models, the measurements must first be included in the WML of the CALC block if the WML function is to be used in that CALC block.

Once included in the WML, the measurements are then taken by including the WML in the functions list when using the SCPI model; when using the Expression model, you can apply the entire WML in an expression: `CALC2:PATH:EXPR "WML (CHAN1)"`.

***NOTE**. You do not need to include measurements in the WML to use them individually in expressions. For example, `CALC2:PATH:EXPR "RTIME (CHAN1)"` returns the rise time of channel one regardless of whether `RTIME` is included in the CALC2 WML.*

Auto Advance Measurement List. Specifies the measurements available to the CALC block when using the auto-advance acquisition cycle. See *Auto-Advance Acquisition* on page 3–13 for information on using auto advance.

FEED1. Specifies the data source when CALC1 is defined using the SCPI model.

Also, when the local value %1 is used with the Expression model, %1 is assigned the value of FEED1 as a default; see pages 3–33 and 3–37 for more information on local variables.

FEED2. Specifies a second data source, used only when CALC1 is making a two-source measurement, such as gain, delay, or phase, using the SCPI model. See *Measurements* section, page 3–141.

Also, when the local value %2 is used with the Expression model, %2 is assigned the value of FEED2 as a default; see pages 3–33 and 3–37 for more information on local variables.

FUNCTIONS List. Specifies one or more functions, such as input filtering, waveform differentiating, or measurements in the WML, to perform on the waveform specified by the CALC:FEED1 data source (SCPI model only).

Expression. A user-defined, algebraic expression that directly specifies data sources, operators, and functions without using FEED1 or FEED2 or the FUNCTION block (Expression model only).

Note from Figure 3–8 that both CALC models use the WMP parameter block and the WML list, but only the SCPI model depends on the Feeds and the

Functions blocks. The Expression model specifies data sources, and the functions that apply to them, directly in the expression.

SCPI Calculation Model

The waveform analyzer supports the standard SCPI calculation model. This model provides for processing a single data source through a sequence of one or more functions in a linear fashion. For example, you might first bandwidth-limit filter the channel 1 input waveform to reduce random noise and then measure the peak-to-peak amplitude of the filtered waveform.

Functions that may be used with the SCPI model are listed in the section *Calculation Functions* on page 3–43, and include the waveform measurement list and all the measurements it contains (measurements are covered in *Measurements* on page 3–141).

To Use

There are four calculations, called CALC blocks 1 through 4, that you can define. The following process describes in a general sense what's required to define one such CALC block using the SCPI model:

1. Set up to acquire the channel you will use as a data-source FEED. (See the procedures in *Input Signal Conditioning,* starting on page 3–101, and *Trigger Types,* starting on page 3–193.)

TVS600A-models only. If you will use a reference as the data-source FEED, acquire or import a waveform and store it in that reference (one of REF1–10, see *I/O of Waveforms* on page 3–117*)*.

2. Specify the functions and the sequence in which they are applied for the CALC block: send `CALC<n>:PATH <arg1>, <arg2>, ... <argn>`, where `CALC<n>` is one of CALC blocks `CALC1` through `CALC4` and `<arg1>` through `<argn>` are calculation functions.

Figure 3–9 shows how the `CALC:PATH` command sets the data flow through a set of functions.

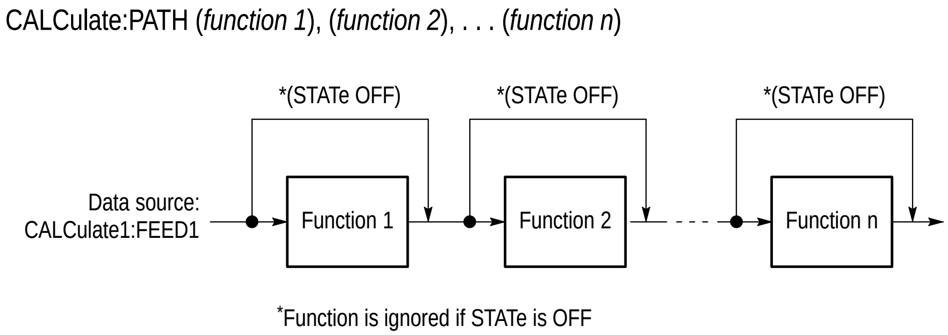

CALCulate:PATH (*function 1*), (*function 2*), . . . (*function n*)
*(STATe OFF)
*(STATe OFF)
*(STATe OFF)
Data source:
CALCulate1:FEED1
Function 1
Function 2
Function n
*Function is ignored if STATe is OFF

Figure 3–9: PATH definition for SCPI calculation model

3. Turn on those functions specified in step 2: send
CALC<n>:<function1>:STATe <arg1>;<function2>:STATe <arg2>;..
<functionn>:STATe <argn>,
where CALC<n> is one of CALC1 through CALC4<n> and <arg1> through <argn> are ON (1) or OFF (0) for the function listed.

   Functions are performed only when their individual STATe is set to ON; functions whose STATe is set to OFF are not executed and any data passed to OFF functions is routed unchanged to the next function in the PATH. See Figure 3–10.

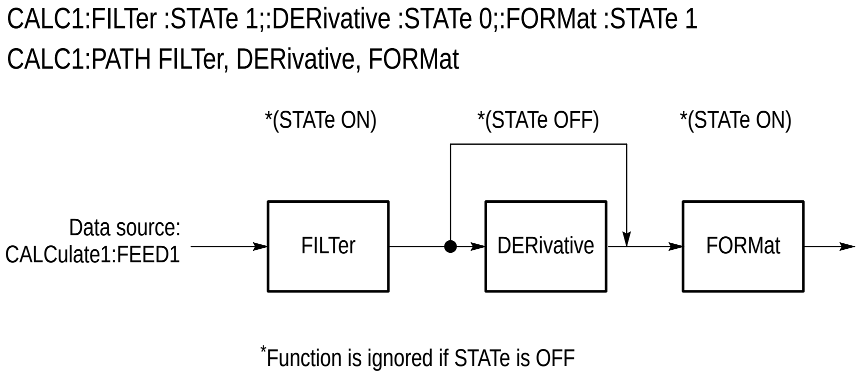

CALC1:FILTer :STATe 1;:DERivative :STATe 0;:FORMat :STATe 1
CALC1:PATH FILTer, DERivative, FORMat
*(STATe ON)
*(STATe OFF)
*(STATe ON)
Data source:
CALCulate1:FEED1
FILTer
DERivative
FORMat
*Function is ignored if STATe is OFF

Figure 3–10: Effect of STATe on CALC:PATH operation

4. Completely specify (or accept default values) for each function: send CALC<n>:<function 1>:<parameters/args>, where <n> is the CALC-block number, 1–4, and <parameters/args> varies with the functions used.

   Example: CALC1:FILTer:FREQuency BPASs;:CALC1:FILTer:FREQuency:STARt 100E+3;:CALC1:FILTer:FREQuency:STOP 10E+6 sets a bandpass filter that passes only frequencies from 100 KHz to 10 MHz for any waveform fed to the CALC1 block. (Note that commands are chained; see *Chaining Commands and Queries on* page 3–87.)

5. Specify the source (feed) for the CALC block to which the specified calculation will be applied. Send `CALC<n>:FEED1 CHAN<n>` or `REF<n>`, where `<n>` is the number of the CALC block, `CHAN<n>` is one of Ch 1 – Ch 4, and `REF<n>` is one of Ref 1 – 10.

6. Use the appropriate commands to return your calculation:

   - If you have specified a channel not yet acquired as the FEED, send the command `INITiate` and then the query `CALC:DATA?` to first acquire your data source and then to return your calculation.

   - If you have specified a channel already acquired as the FEED (as when you are recalculating the same data) or if the feed is a reference waveform, send the command `CALC:IMMediate` and then query `CALC:DATA?` to calculate without acquiring and to return your calculation. (Only TVS600A models come equipped with references.)

Commands

The commands and functions to set up calculations using the SCPI model follow:

Used to:	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
return calculation results for CALC<n>	`CALCulate<n>:DATA?`	`tktvs600_fetchCalculation`	No
return characterization data for CALC<n	`CALCulate<n>:DATA:PREamble?`	`tktvs600_getPreamble`	No
set the primary data source for CALC<n>	`CALCulate<n>:FEED[1]`	No[4]	No
set the secondary data source for CALC<n>	`CALCulate<n>:FEED2`	No[4]	No
set the WMP block used for FEED2 to WMP<n>	`CALCulate<n>:FEED2:CONText`	No[4]	No
force new CALCulation<n> without reacquiring	`CALCulate<n>:IMMediate`	`tktvs600_initiateCalculation`	No
specify functions and their order for CALC<n>	`CALCulate<n>:PATH`	No[4]	No
set waveform parameters used to characterize FEED 1 data source	`CALCulate<n>:WMParameter`	`tktvs600_setCalcWMParameter`	No

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Functions Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application**

[4] **Driver functions use the Expression CALC model only.**

SCPI Data Sources

Once you define a CALC block, it is available for calculating any source you want to feed it. You must define a single data source for each CALC block using the CALC:FEED1 command. See Figure 3–11. As shown in the procedure above, there are two sources for feeding the CALC block:

- Input channels. You can feed (connect) an input channel to one or more CALC blocks. Use the CALC<n>:FEED1 command; for example, the command CALC1:FEED1 CHAN1 connects the channel 1 waveform record to the CALC1 block. You may also specify the FEED1 source as "XTIM:VOLT <n>."

  Input channel data will be calculated only if the data acquires. For example, if you set up a calculation for channel 1, then send INITiate, and channel 1 does not trigger, you will get an error message.

- Reference waveforms (TVS600A-models only). You can feed (connect) a waveform (or other data) stored in any of REF 1 through REF10 to one or more CALC blocks. Use the CALC<n>:FEED1 command; for example, the command CALC1:FEED1 REF1 connects the waveform stored in REF 1 to the CALC1 block.

  You must make sure that references you feed to CALC blocks contain data. (Use TRACe:COPy commands to move waveform and other data to references.)

For more information, refer to the CALC:FEED1 command in your *TVS600A Command Reference.*

You may also define a second data source for each CALC block using the CALC:FEED2 command. The allowed sources are the same as for CALC:FEED1; for example, the command CALC1:FEED2 CHAN2 connects the CH 2 waveform record to the CALC1 block. SCPI-model calculations that make dual-waveform measurements (such as when measuring Gain or Delay) can make use of the second data source. See the procedure on page 3–146 of *Measurements*.

The functions you set using CALC:PATH apply only to the data source selected with FEED1. The source selected by FEED2 is ignored by the functions you select.

NOTE*. SCPI model calculations are disabled when you define an expression model calculation with CALC:PATH:EXPR.*

SCPI Calculations Variations

Defining CALC blocks (calculations) using the SCPI model is handy when you want to do the following calculation tasks:

- Switch between data sources quickly: simply change the FEED1 and send `CALC:IMMEDiate` to recalculate the acquired waveform.

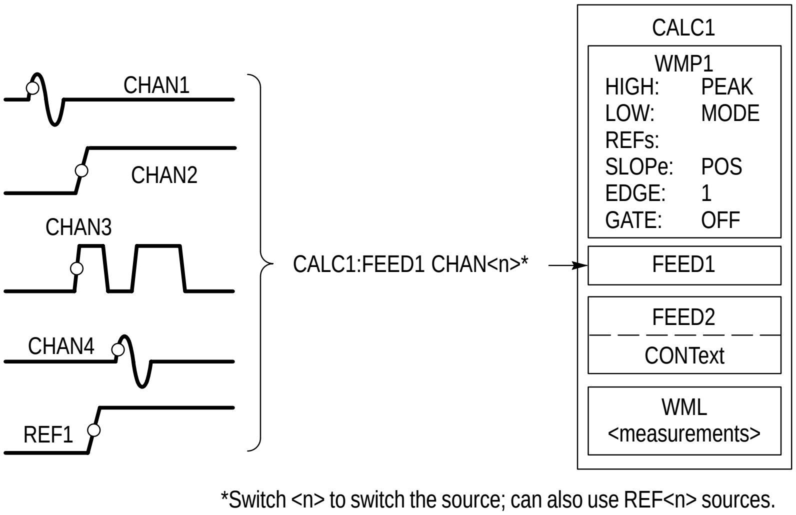


Figure 3–11: Different waveforms, same calculation

- Calculate a waveform in more than one way: define one CALC block for each way and feed both CALC blocks the same waveform. For example, perform the same measurement in two CALC blocks, but FILTer to remove high-frequency components from the waveform in one case but not the other.

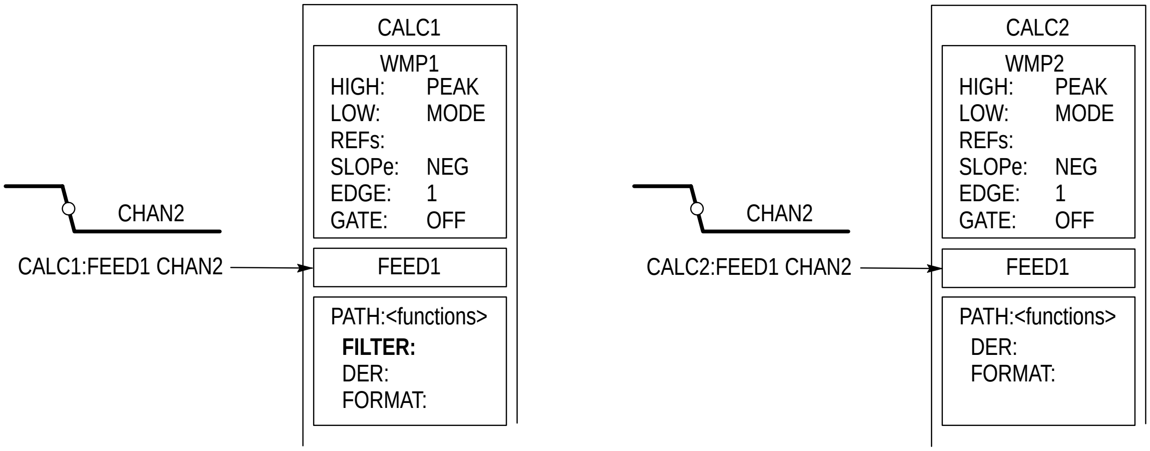


Figure 3–12: Different calculations, same waveform

- Calculate a waveform the same way, but characterize it differently: set two (or more) CALC blocks with the same functions list, but different waveform measurement parameter blocks. For example, measure the second positive-going edge in one CALC block, but the negative-going edge in the other.

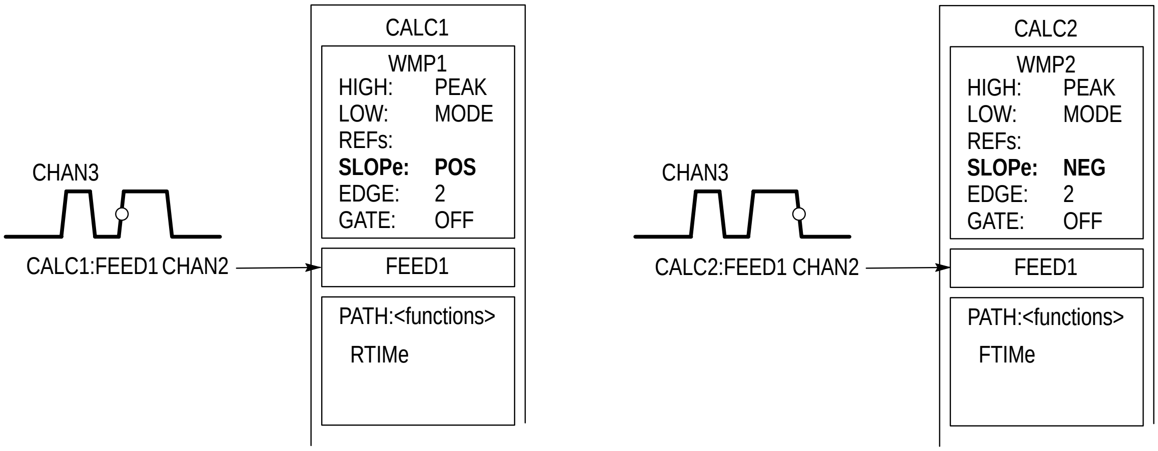


Figure 3–13: Same calculation, same waveforms, different characterization

- Calculate dual-waveforms measurements, using any WMP block to characterize the second waveform. For example, switch between measuring the delay between channel 1 edge and the first, second, and third edge of channel 4 by switching context between CALC2, CALC3, and CALC4.

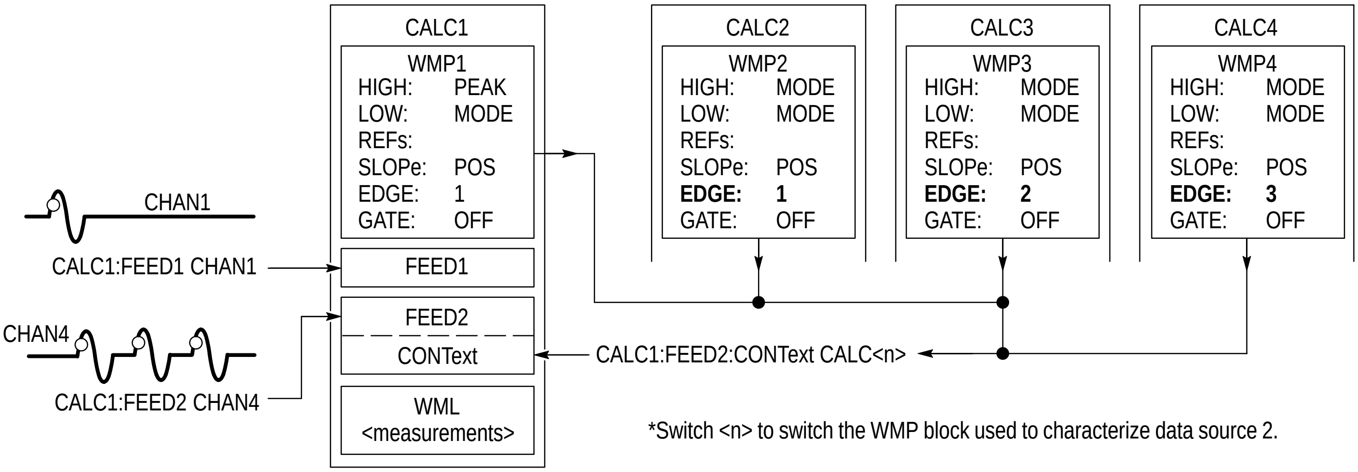


Figure 3–14: Dual waveform measurements, switching waveform characterization

Other variations are possible. For example, to perform the same measurement on two (or up to four) channels during the same acquisition, define the same measurement in two CALC blocks:

```
CALC1:WML FTIM,RTIM;:CALC2:WML FTIM,RTIM
```

In this case, you feed one waveform to one CALC block and the second to the another CALC block before acquiring the waveforms.

Expression Model

The waveform analyzer supports a second, expression-based model for defining calculations. The Expression model provides an alternative to the linear, single-source processing that the SCPI model provides. For example, using a calculation based on the expression, you might take the average of the sum of channel 1 and channel 2 waveforms using the following expression:

```
CALC2:PATH:EXPR "(CHAN1+CHAN2)/2"
```

The `CALC:PATH:EXPR` command forces the waveform analyzer to ignore FEED1 and FEED2 settings and the Functions block settings and to derive the data sources and the functions directly from the expression you define.

***NOTE**. The expressions may be enclosed in quotation marks, as shown the expression above, or in parentheses:* `CALC2:PATH:EXPR ((CHAN1+CHAN2)/2)`.

To Use

Like the SCPI model, the expression model uses the four calculations, called CALC blocks 1 through 4, that you can define. The following process describes the general procedure to define one such CALC block using the Expression model:

1. Set up to acquire the channels you will use as a data sources. (See procedures in *Input Signal Conditioning* starting on page 3–101 and in *Trigger Types* starting on page 3–193.)

   TVS600A-models only. If you will use references as data sources, acquire or import waveforms and store them in those references (any of REF1 through REF 10). (See *I/O of Waveforms* on page 3–117.)

2. Specify the expression, including data sources, functions, and operations. For example, send: `CALCulate<n>:PATH:EXPR <expr>,` where <n> is the CALC-block number, `1–4` and `<expr>` is any algebraic expression, as defined by the BNF description for expression syntax on page 3–34.

   - Expressions may use the input channels, CHAN1 through CHAN4 as data sources; with TVS600A-models only, expressions can also use references, REF1 through REF10, and the scratch-pad variables, %1 through 9% (see page 3–37), as data sources.

- Expressions may use any operators listed in *Expression Operators* tables starting on page 3–36.
- Expressions may use those functions listed for the Expression model in the section *Calculation Functions* on page 3–43.

3. Completely specify (or accept default values) for each function: send `CALC<n>:<function 1> :<parameters/args>`, where <n> is the CALC-block number, `1–4` and `<parameters/args>` varies with the function in use.

Example: `CALC1:FILTer:FREQuency BPASs;FREQuency:STARt 100E+3 ;STOP 10E+6` sets a bandpass filter that passes only frequencies from 100 KHz to 10 MHz for any waveform to which the filter is applied (by the CALC1 expression). (Note that commands are chained and root- and lower-level nodes are omitted; see *Chaining Commands and Queries* on page 3–87.)

4. Use the appropriate commands to return your calculation:

- If you have specified any channels not yet acquired in your expression, send the command `INITiate` and then the query `CALC:DATA?` to first acquire your data source and then to return your calculation.
- If you have specified only channels already acquired in your expression, (as when you are recalculating the same data) or if the data sources are all references, send the command `CALC:IMMediate` and then the query `CALC:DATA?` to calculate without acquiring and to return your calculation. (Only TVS600A models come equipped with references.)

Commands

The commands and functions to set up calculations using the Expression model follow:

Used to:	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
return calculation results for CALC<n>	`CALCulate<n>:DATA?`	`tktvs600_fetchCalculation`	Yes[4]
return characterization data for CALC<n	`CALCulate<n>:DATA:PREamble?`	`tktvs600_getPreamble`	No
set the %1 variable to Ch <n>	`CALCulate<n>:FEED[1]`	No	No
set the %2 variable to Ch <n>	`CALCulate<n>:FEED2`	No	No
force CALCulation<n> without reacquiring	`CALCulate<n>:IMMediate`	`tktvs600_initiateCalculation`	Yes
specify functions and their order for CALC<n>	`CALCulate<n>:PATH:EXPR`	`tktvs600_setCalcExpression`	Yes
set waveform parameters used to characterize data sources specified in the CALC expression	`CALCulate<n>:WMParameter`	`tktvs600_setCalcWMParameter`	No

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Functions Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application**

[4] **Returns only results that evaluate to a single number (scalars, not vectors such as waveforms).**

Expression Data Sources

There are three data sources that you may use in CALC expressions:

- Input channels. Input channel data will be calculated only if the data is successfully acquired. For example, `CALC2:PATH:EXPR "MEAN(CHAN1)"` will fail (return an error) if you send `INITiate` and channel 1 cannot acquire. (Or if you send CALC2:IMMediate and channel 1 has not been acquired.)
- References (TVS600A-models only). REF1 through REF10 can be used as data sources. You must make sure that references you include in your CALC expressions contain data. (Use `TRACe:COPY` commands to move waveform and other data to references.)
- Local variables (TVS600A-models only). %1 through %9 can be used locally within an expression as scratch-pad variables. You assign these variables using the assignment operators; see *Assignment Operators* on page 3–37.

Local variables, %1 and %2, default to the values of FEED 1 and FEED 2, respectively (%3 through %9 have no defaults.) If you assign FEED1 and FEED2 to Ch1 and Ch2, respectively, you can shorten your expressions by substituting %1 for CHAN1 and %2 for CHAN2 as long as you do not assign these variables other values. For example:

```
CALC1:PATH:EXPR "REF2:=REF1*AMPL(CHAN1)+LOW(CHAN1)"
```

reduces to:

```
CALC1:PATH:EXPR "REF2:=REF1*(AMPL(%1)+LOW(%1)"
```

There is a 200 character limit on expressions; local variables can help you stay within that limit when creating long expressions. Local variables also reduce calculation time when they replace multiple instances of a channel data source.

For more information, refer to the `CALC:PATH:EXPRession` command in your *TVS600A Command Reference*.

Expression Syntax

The syntax for CALCulate expressions is defined in the following BNF description (for meanings of BNF forms, see Table 3–31 *BNF symbols and meanings* on page 3–90). Note that TVS600 models cannot use elements that refer to TVS600A features only.

```
<statement>   ::=   expr“;”
                    <statement><statement>
                     /* epsilon */
<expr>        ::=   <statement><expr>
                    <factor>
                    <expr><binop><expr>
                    <expr><logop><expr>
                    <expr><relop><expr>
                    <lval><asgnop><expr>
                    <lval>“#=”<accumop>“(”<arg_list>“)”
                    <lval>“:=”<expr>
                    <expr>“..”<expr>
                    <expr>{“<?”“|”“>?”}<expr>
                    !<expr>
                    “{”<statement>“}”
                    /* epsilon */
factor        ::=   [–<number>|{+}<number>]{<unit>}
                    AATS
                    [CHAN1|CHAN2|CHAN3|CHAN4]{“[”<nrx>“]”}
                    <meas>
                    <parameter>
                    <ref>
                    “(”<expr>“)”
<lval>        ::=   <ref>|<parameter>
<meas>        ::=   <func>“(”<arg_list>“)”
                    <meas_func>“(”<m_arg_list>“)”
```

```
<arg_list>       ::=    <expr>{[, <expr>]...}
<m_arg_list>     ::=    <wmp_expr> {[, wmp_expr>]...}
<wmp_expr>       ::=    {WMP1:|WMP2:|WMP3:|WMP4:} <expr>
<parameter>      ::=    %1|%2|%3|%4|%5|%6|%7|%8|%9
<ref>            ::=    REF1|REF2|REF3|REF4|REF5|REF6|REF7|REF8|REF9|REF10|REF
                        “[”<nrx>“]”
<binop>          ::=    +|–|*|/
<asgnop>         ::=    +=|–=|*=|/=||=|&=|^=
<logop>          ::=    &|||^|AND|OR
<relop>          ::=    ==|!=|>|>=|<|<=|><|<>|EQ|NE|GT|GE|LT|LE|INSide|OUTside
<accumop>        ::=    AVERage|ENVelope|VECTor|STATistics
<meas_func>      ::=    AAMList|AC|ACRMS|AMPLitude|AREA|DC|DELay
                        |CARea|CMEan|COPulse|CPARea|CRMS|CROss
                        ||FTIMe|FREQuency|GAIN|HIGH|LOW|MAXimum
                        |MEAN|MID|MINimum|NCRoss|NDUTycycle|NWIDth
                        |OVERshoot|PARea|PCRoss|PDUTycycle|PERiod|PHASe
                        |PREShoot|PTPeak|PWIDth|RMS|RTIMe|SDEViation|TTRig|WMList
<func>           ::=    ABSolute|AVERage|BAT|DERivative|ENVelope|EVT|FFT|
                        FILTer|FORMat|HLT|INTegral|SEGMent|SMOothing|SRQ|
                        STATistics|TRANsform|TRG|VECTor|XDURation|XOFFset|XSCale|
                        XSIZe
<nrx>            ::=    <number>{<unit>}
<unit>           ::=    PCT|XUnit|Second|PS|NS|US|MS
<number>         ::=    [0–9]...{.[0–9]...}{[E|e]{+|–}[0–9]...}
```

An example of the expression syntax follows:

```
CALC1:PATH:EXPR "FORM(TRAN(FILT(CHAN1)))"
```

This command performs frequency filtering on CHAN1, then performs an FFT transform, and finally formats the result, perhaps in a logarithmic magnitude format.

The settings for `CALC:FILTer`, `:TRANsform`, and `:FORMat` functions must be completed in separate command statements prior to starting acquisition. The settings might include enabling the low-pass filter with `CALC:FILT:FREQ:LPAS`, setting the `TRANsform WINDow` type, and setting the result format for the resulting FFT waveform record with `CALC:FORM MLOG`.

The following example produces a waveform with an average value of zero and an amplitude normalized to one:

```
CALC2:PATH:EXPR "(CHAN1–MEAN(CHAN1))/AMPL(CHAN1)"
```

In this example, you could use local variables for faster execution. Assuming that %1 is FEED1 (the default) and FEED1 is assigned to channel 1, you can use the following calculation:

```
CALC2:PATH:EXPR "(%1–MEAN(%1))/AMPL(%1)"
```

Expressions for Auto-Advance Records

Expressions for auto-advance acquisition are similar to those for normal acquisition. You can specify a set of measurements to perform on all auto-advance waveform records with the command `CALC1:AAMList`. An example follows:

```
CALC1:PATH:EXPR "AAMList(CHAN1)"
```

If AAMList is set to measure RTIMe (rise time), the expression returns a vector of RTIMe measurements for all auto-advance acquisitions (as delimited by `AADV:REC:START,COUNT`).

You can also specify individual measurements or functions to perform on a single record. In the following example, MEAN is measured only on auto-advance record number five.

```
CALC1:PATH:EXPR "MEAN(CHAN1[5])"
```

Expression Operators

The CALC expression model supports a rich set of operators of various types. These operators are described in the tables that follow.

The following definitions are used in the tables:

<arg> an operand.

<bool> a single Boolean value or vector of Boolean values.

<expr> any expression allowed by the BNF syntax (see page 3–34).

<lval> A reference (REF1 through REF10) or a local (scratch-pad) variable (%1 through %9).

<scalar> a single numeric value.

<tuple> an ordered group of numbers; for example, an envelope waveform is a vector of 2-tuples, where each 2-tuple contains a max:min pair.

<vector> an array of numbers or of tuples of numbers. Vectors are typically waveforms or segments of waveforms.

Table 3–1: Arithmetic operators

Name	Syntax (<arg> opr <arg>)	Outputs	Description
+	<expr> + <expr>	<vector>	The arithmetic sum of the two operators.
–	<expr> – <expr>	<vector>	The arithmetic difference between two operands.
*	<expr> * <expr>	<vector>	The arithmetic product of two operands.
/	<expr> / <expr>	<vector>	The arithmetic quotient of the two operands.

Arithmetic operators follow standard math rules.

- If the operands are of equal lengths, the result is a vector of that length.
- If one operand is a scalar, it is replicated for the operation, and the result is a vector of length equal to that of the other operand.
- If the vectors are of differing length (not of a length equal to one), the mathematical operation is executed from left-to-right until the data from the shorter vector is exhausted. The result is not padded.

Table 3–2: Assignment operators (TVS600A models only)

Name	Syntax (<arg> opr <arg>)	Outputs	Description
Assignment (set equal)	<lval> := <expr>	<vector>	Assigns the value of <expr> to <lval> and returns the result.
Plus equals	<lval> += <expr>	<vector>	Assigns the value of (<lval> + <expr>) to <lval> and returns the result.
Minus equals	<lval> –= <expr>	<vector>	Assigns the value of (<lval> – <expr>) to <lval> and returns the result.
Times equals	<lval> *= <expr>	<vector>	Assigns the value of (<lval> * <expr>) to <lval> and returns the result.
Div equals	<lval> /= <expr>	<vector>	Assigns the value of (<lval> / <expr>) to <lval> and returns the result.
OR equals	<lval> |= <bool>	<vector>	Assigns the value of (<lval> OR <expr>) to <lval> and returns the result.
AND equals	<lval> &= <bool>	<vector>	Assigns the value of (<lval> AND <expr>) to <lval> and returns the result.
XOR equals	<lval> ^= <bool>	<vector>	Assigns the value of (<lval> XOR <expr>) to <lval> and returns the result.
Accumulate	<lval> #= <expr>	<vector>	Used for an open-ended accumulation of data over multiple acquisitions. Behavior is undefined for all cases in which <expr> is not of the form AVERage(<expr_list>), ENVelope(<expr_list>), STATistics(<expr_list>), or VECTor(<expr_list>),

The assignment operators assign a <vector> or <scalar> result to an <lval> (local value), which must be a reference (REF1–REF10) or a scratch-pad variable. Lval's have the following characteristics:

- An <lval> defaults (when previously undefined) to a zero length <scalar> of value 0.0.
- If lvals are REFs, they are retained after the calculation in progress completes until they are changed: either by subsequent assignments in subsequent calculations, by waveform I/O operations, or powering off the waveform analyzer.
- If lvals are local, scratch-pad variables, they are only retained until the calculation in which they are defined completes. Local variables disappear after the CALC block in which they occur is evaluated; REF values persist until changed.

Assignment operators lend themselves to a variety of uses. The following commands build a vector of 100 frequency measurements taken from a series of 100 auto-advance waveform records and stores it in REF1 using the := (set equal) operator in the expression:

```
CALC1:AAML FREQ
CALC1:PATH:EXPR "REF1:=AAML(CHAN1)"
AADV:COUNT 100
INIT
```

Table 3–3: Boolean operators (TVS600A models only)

Name	Syntax (<arg> opr <arg>)	Outputs	Description
OR	<bool> | <bool>	<bool>	The logical OR of the two Boolean operands.
AND	<bool> & <bool>	<bool>	The logical AND of the two Boolean operands.
XOR	<bool> ^ <bool>	<bool>	The logical exclusive OR of the two Boolean operands.
Unary NOT	!<bool>	<bool>	The logical NOT (inverse) of a Boolean operand.

Boolean operations follow standard C-language rules: a value is TRUE if it is not equal to zero. They behave as follows:

- External representation will default to the standard CALC output: 32-bit floating-point format with TRUE represented by a value of 1.0 and FALSE represented by a value of 0.0.
- If both operands are of equal length, the result is a Boolean vector of that length.

- If one operand is a scalar, the value is logically replicated to form a vector of length equal to that of the other operand.
- If the operands are vectors of different lengths (but not of length 1), the operation is performed from left to right until the data from the shorter vector is exhausted. The result is not padded.

Table 3–4: Comparison operators (TVS600A models only)

Name	Syntax (<arg> opr <arg>)	Outputs	Description
Less than	<expr> < <expr> <expr> LT <expr>	<boolean>	Returns a vector of the Boolean TRUE/FALSE values for the point-by-point less-than comparison.
Less than or equal	<expr> <= <expr> <expr> LE <expr>	<boolean>	Returns a vector of the Boolean TRUE/FALSE values for the point-by-point less-than-or-equal comparison.
Greater than	<expr> > <expr> <expr> GT <expr>	<boolean>	Returns a vector of the Boolean TRUE/FALSE values for the point-by-point greater-than comparison.
Greater than or equal	<expr> >= <expr> <expr> GE <expr>	<boolean>	Returns a vector of the Boolean TRUE/FALSE values for the point-by-point greater-than-or-equal comparison.
Equal	<expr> == <expr> <expr> EQ <expr>	<boolean>	Returns a vector of the Boolean TRUE/FALSE values for the point-by-point equal comparison.
Not equal	<expr> != <expr> <expr> NE <expr>	<boolean>	Returns a vector of the Boolean TRUE/FALSE values for the point-by-point not-equal comparison.
Inside	<expr> >< <env> <expr> INSide <env>	<boolean>	Returns a vector of the Boolean TRUE/FALSE values for the point-by-point comparison of the source expression to the max:min tuples of the envelope/template.
Outside	<expr> <> <env> <expr> OUTside <env>	<boolean>	Returns a vector of the Boolean TRUE/FALSE values for the point-by-point comparison of the source expression to the max:min tuples of the envelope/template.

This group of operators is used to compare vectors. They generate a vector of Boolean results as follows:

- If the operands are of equal lengths, the result is a vector of that length.
- If one operand is a scalar, it is replicated for the comparison, and the result is a vector of length equal to that of the other operand.
- If the vectors are of differing length (but not of a length equal to one), the comparison is executed from left-to-right until the data from the shorter vector is exhausted. The result is not padded.

Comparison operators lend themselves to testing waveforms against a template. Usually downloaded, templates are envelope waveforms that you must store in a reference. You can then construct expressions that test waveforms against the template.

Given such a template in REF1, the following expression returns a vector of TRUE and FALSE values, with the two out-of-template points denoted as TRUE and all inside-template points denoted as FALSE, given the example shown in Figure 3–15.

```
CALC1:PATH:EXPR "CHAN1 OUTside REF1"
```

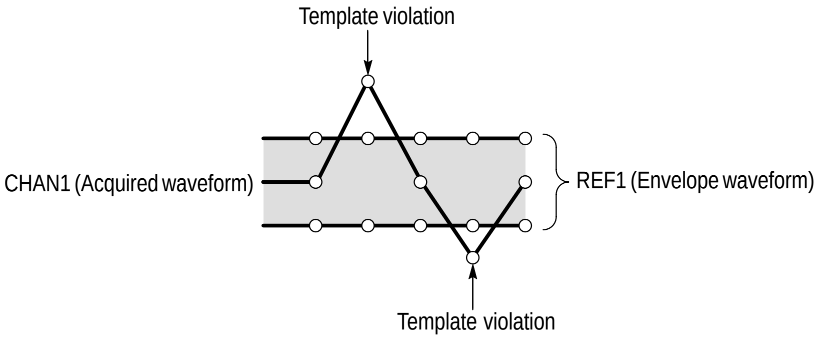


Figure 3–15: Simple template test

Comparison operators also lend themselves to testing waveform measurements against a measurement limit. The following expression will return FALSE (0) if the rise time measured in Ch 1 exceeds 100 μs.

```
CALC1:PATH:EXPR "RTIMe(CHAN1) > 100us"
```

See *Limit Testing Measurements* on page 3–152 and *Template Testing* on page 3–165 to read about limit and template testing.

Table 3–5: Constructor operator (TVS600A models only)

Name	Syntax (<arg> opr <arg>)	Outputs	Description
Range	<scalar> .. <scalar>	<range> (or env)	Returns a tuple consisting of the ordered pair of operands (first scalar, second scalar). The result has the same data type as the sources, but the data is interpreted as type envelope. The two operands must share the same data type.

Constructors are used to assemble vectors or n-tuples. The only constructor operator that the TVS600A Waveform Analyzer supports is range (“..”), which assembles a 2-tuple used by the segment function or in template testing. See the SEGment function on page 3–56 and *Template Testing* on page 3–165.

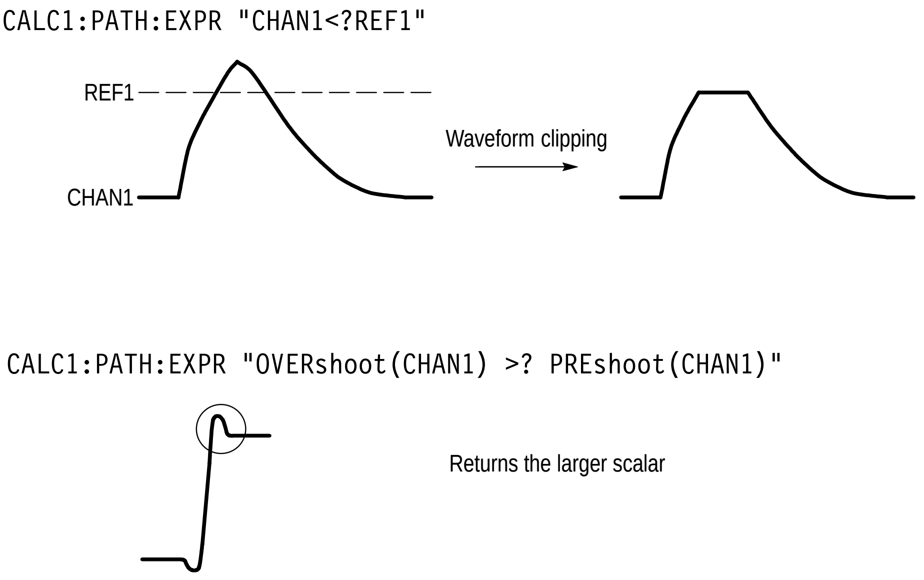
Table 3–6: Min/Max operators (TVS600A models only)

Name	Syntax (<arg> opr <arg>)	Outputs	Description
MIN	<expr> <? <expr>	<vector>	Returns a vector of the minimum values.
MAX	<expr> >? <expr>	<vector>	Returns a vector of the maximum values.

The min/max operators generate a vector of the minimum or maximum values derived by a value-by-value comparison of the source operands.

- If the operands are of equal lengths, the result is a vector of that length, or a scalar if both operands are scalars.
- If one operand is a scalar, it is replicated for the comparison, and the result is a vector of length equal to that of the other operand.
- If the vectors are of differing length (not of a length equal to one), the comparison executes from left to right until the data from the shorter vector is exhausted. The result is not padded.

Min/Max operators are useful for limiting vectors or returning the lesser of two magnitudes. The first expression that follows delivers a vector resulting from the value of REF1 clipping (limiting to minimum) the channel 1 waveform; the second expression delivers the largest of overshoot or undershoot (overshoot in this case).



```
CALC1:PATH:EXPR "CHAN1<?REF1"
```

```
CALC1:PATH:EXPR "OVERshoot(CHAN1) >? PREshoot(CHAN1)"
```

Figure 3–16: Min/max operators with vectors (top) and scalars (bottom)

Table 3–7: Statement termination operator (TVS600A-Models Only)

Name	Syntax (<arg> opr <arg>)	Outputs	Description
Semicolon	<statement>;<statement>	<scalar> null	This operator terminates a statement and pops the results off the stack. If the results of the statement are to be available after the evaluation of the statement they must be assigned to either a local variable or reference trace. Statements are evaluated from left to right. Please note: CALC block expressions that end in a semicolon will result in a null data query response.

The statement termination operator separates statements within a CALC block.

Calculation Functions

This section describes the functions used by the calculation system. The following types of functions are discussed:

- Waveform DSP Functions, which can be used by both calculation models, SCPI and Expression. (Waveform mathematics are an exception; they work only with expressions.) These functions are present in both TVS600 and TVS600A models. These functions perform the following calculations:
  - Waveform Mathematics, which invert, add, subtract, multiply, and divide waveform records (see page 3–44)
  - Waveform Integration, which creates the integral of a waveform record (see page 3–54)
  - Waveform Differentiation, which creates the derivative of a waveform record (see page 3–53)
  - Fast Fourier Transforms (FFT), which produce an amplitude versus frequency waveform record (see page 3–44)
  - Digital filtering, which removes certain frequency components from a waveform record (see page 3–50)
- Waveform Functions, which operate on a vector (usually a waveform) to extract a segment, envelope the waveform, and so on. Waveform Functions are supported by the CALC Expression model, but not by the SCPI model. (See page 3–55.) TVS600A models only.
- Waveform Attribute Functions, which give CALC expressions access to scaling information contained in the dimension block of the waveform preamble. They are not supported in SCPI-model calculations. (See page 3–60.) TVS600A models only.
- Boolean Aperture Test Function (BAT), which collapses a Boolean vector into a scalar TRUE or FALSE value. (See page 3–61.) TVS600A models only.
- Control/Notification Functions, which let you halt operation or generate a trigger when a limit or measurement test passes or fails. (See page 3–62.) TVS600A models only.

***NOTE**. Terminology used to described the data operated on by the functions described in this section is defined under* Expression Operators *on 3–36.*

Waveform DSP Functions

The waveform digital-signal-processing (DSP) functions provide mathematical operations between two or more waveforms or transformations (integrate, differentiate, filter) of a waveform. Waveform math can only be employed using the Expression model; the other DSP functions can be performed using both the Expression and the SCPI models. For more information on the two models, refer to *Calculation System Overview* on page 3–21.

Waveform Math

The waveform analyzer can mathematically manipulate your waveforms. You can add (+), subtract (–), multiply (*), or divide (/) two waveforms or a waveform and a scalar. The scalar can be the result of another measurement such as MEAN in the following example:

```
CALC1:PATH:EXPR "CHAN1 - (MEAN (CHAN1))"
```

You can use the minus operator (–) to negate a scalar value.

You can also use the range operator (..) with most math operations. For example, the following expression creates a +/–5% envelope of channel 1 and assigns it to reference 1:

```
CALC1:PATH:EXPR "REF1:= (CHAN1*(0.95..1.05))"
```

Fast Fourier Transforms (FFT)

The calculation capabilities of the waveform analyzer include taking the Fast Fourier Transform (FFT) of a waveform record. The FFT allows you to transform an amplitude versus time waveform into one that plots the amplitudes of the various discrete frequencies the waveform contains.

The FFT computes the frequency content of a waveform you specify as a CALC-block source. The transform of the source waveform is based on the following equation:

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{\frac{j2\pi nk}{N}} \quad for : k = 0 \ to \ \frac{N}{2}$$

Where: x(n) is a point in the time domain record
X(k) is a point in the frequency domain record
n is the index to the time domain record
k is the index to the frequency domain record
N is the length of the time domain record
j is the square root of –1

The resulting waveform is a record consisting of N/2 complex coefficients representing values in the frequency domain. The horizontal scale for FFT waveforms is expressed in frequency with the first point of the waveform record representing zero frequency (DC).

Setting the Record Format. You can convert the complex data in the resultant transform record into magnitude or phase data using the CALC:FORMat function. The following lines set the waveform analyzer to perform a TRANsform on CH 1 and format the result as phase data:

```
CALC1:FORM PHAS
CALC1:PATH:EXPR "FORM(TRAN(CHAN1))"
```

Zero Phase Reference Point. The zero phase reference point for an FFT phase waveform is in the middle of the FFT time domain record regardless of the waveform record length. Figure 3–17 shows this placement.

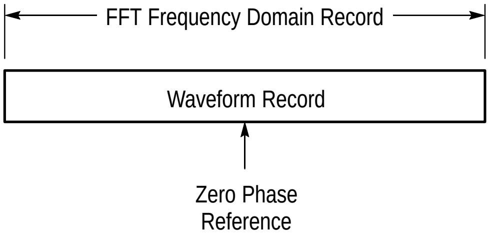


Figure 3–17: Zero phase reference point in FFT phase records

FFT Frequency Range and Resolution. The waveform analyzer can return either the magnitude or phase angle of the FFT frequency domain record. The resolution between the discrete frequencies in this waveform is determined by the following equation:

$$\Delta F = \frac{Sample\ Rate}{Input\ Length}$$

Where:

ΔF is the frequency resolution.
Sample Rate is the sample rate of the source waveform.
Input Length is the length of the source waveform record.

The sample rate also determines the range these frequencies span; they span from 0 to ½ the sample rate of the waveform record. (The value of ½ the sample rate is often referred to as the Nyquist frequency or point.) For example, a sample rate of 20 Megasamples per second would yield an FFT with a range of 0 to 10 MHz.

Undersampling (Aliasing). Aliasing occurs when the waveform analyzer acquires a source waveform with frequency components outside of the frequency range for the current sample rate. In an FFT waveform record, the actual higher frequency components are undersampled, and therefore, they appear as lower frequency aliases that “fold back” around the Nyquist point (½ the sample rate). See Figure 3–18.

- Source waveforms with fast edge transition times create many high frequency harmonics. These harmonics typically decrease in amplitude as their frequency increases.
- Sample the source signal at rates that are at least 2X that of the highest frequency component having significant amplitude.
- If necessary, filter the input to limit its bandwidth to frequencies below the Nyquist frequency.

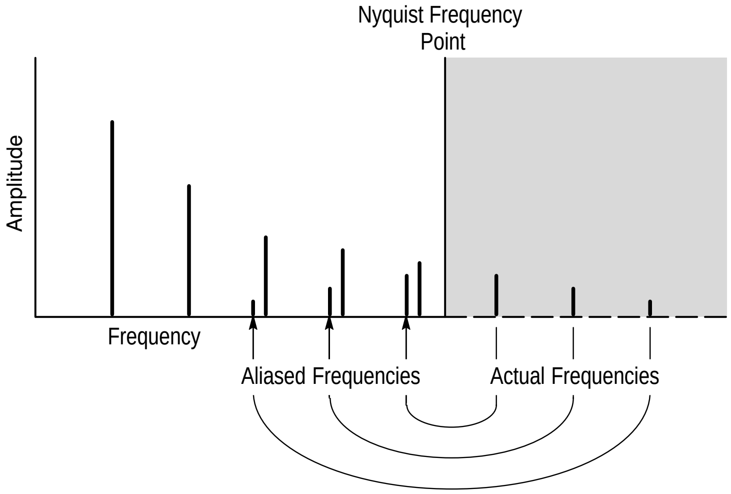


Figure 3–18: How aliased frequencies corrupt an FFT transform

FFT Windows. An FFT window acts like a bandpass filter between the time domain record and the FFT frequency domain record. The shape of the window controls the ability of the FFT to resolve (separate) frequencies and to accurately measure the amplitude of those frequencies. Figure 3–19 shows how a window is used in the transform process.

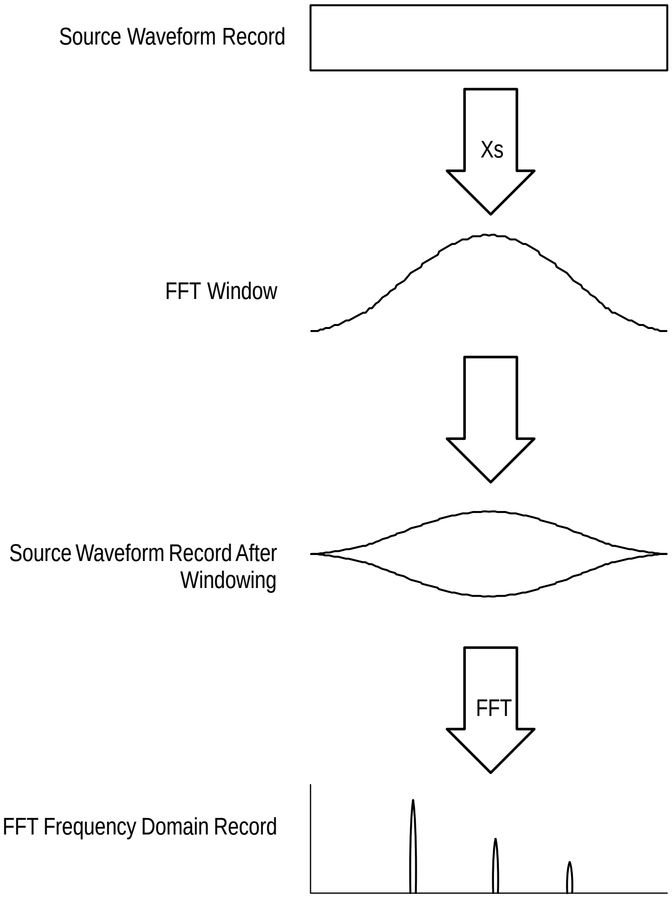

Source Waveform Record
Xs
FFT Window
Source Waveform Record After Windowing
FFT
FFT Frequency Domain Record

Figure 3–19: Windowing the FFT time domain record

You can select from six windows for the transform with the command `CALC:TRAN:FREQ:WIND`. Each window provides benefits and losses. For example, if you select a window to provide better frequency resolution you give up some amplitude accuracy. In general, choose a window that can just resolve the frequencies you want to measure. The available windows and their characteristics are as follows:

- RECTangular. Best type for resolving a narrow band of frequencies but worst for accurate amplitude of those frequencies. Good for measuring nonrepetitive signals and frequency components near DC.
- HAMMing. Very good window for resolving frequencies that are very close to the same value with somewhat improved amplitude accuracy over the rectangular window.
- HANNing. Very good window for measuring amplitude accuracy but degraded for resolving frequencies.

- BHARris. Widest bandwidth and lowest side lobes. Best for viewing a broad spectrum.
- BLACkman. Best window for measuring the amplitude of frequencies but worst at resolving frequencies.
- TRIangular. Window with the least attenuation of side lobes.

Figure 3–20 shows each window, its bandpass characteristic, bandwidth, and highest side lobe.

When choosing a window, consider the following characteristics:

- The narrower the central lobe for a given window, the better it can resolve a frequency.
- The lower the lobes on the side of each central lobe are, the better the amplitude accuracy of the frequency measured in the FFT using that window.
- Hamming, Hanning, Blackman, and BHarris are all bell-shaped windows that taper the waveform record at the ends. The Hanning and BHarris windows taper the data at the end of the record to zero; therefore, they are generally better choices to eliminate leakage. However, be certain to position the most interesting parts of the signal in the center region of the waveform record.
- If the Hanning window merges frequencies, try the Hamming window before settling on the rectangular window. Depending on the distance of the frequencies you are trying to measure from the fundamental, the Hamming window sometimes resolves frequencies better than the Hanning.

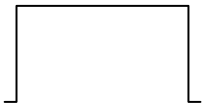
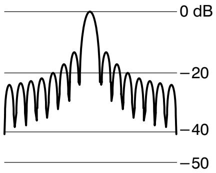
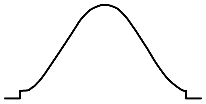
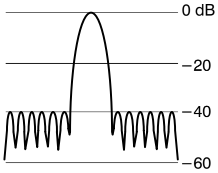
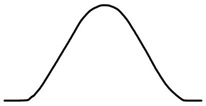
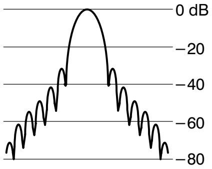
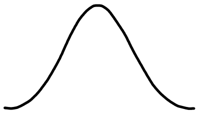
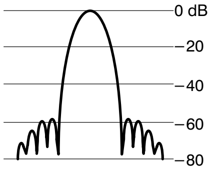
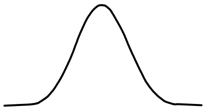
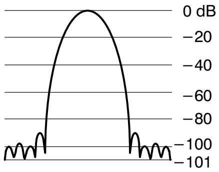
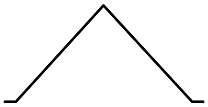
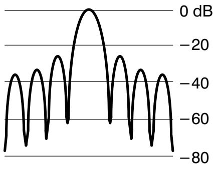
FFT Window Type	Bandpass Filter	–3 dB Bandwidth	Highest Side Lobe
 Rectangular Window		0.89	–13 dB
 Hamming Window		1.30	–43 dB
 Hanning Window		1.44	–32 dB
 Blackman Window		1.68	–58 dB
 BHarris Window		1.90	–92 dB
 Triangle Window		1.28	–27 dB

Figure 3–20: FFT windows and bandpass characteristics

Leakage. Leakage results when the time domain waveform delivered to the TRANsform (FFT) function contains a non-integer number of waveform cycles. Since there are fractions of cycles in such records, there are discontinuities at the ends of the record. These discontinuities cause energy from each discrete frequency to “leak” over on to adjacent frequencies. The result is amplitude error when measuring those frequencies.

Digital Filtering

The waveform analyzer provides a configurable digital filter to perform post-acquisition frequency filtering. There are four types of filters available:

- `BPASs` rejects frequencies outside the defined frequency range.
- `HPASs` rejects frequencies below the defined frequency range.
- `LPASS` rejects frequencies above the defined frequency range.
- `NOTCh` rejects frequencies within the defined frequency range.

Use the command `CALC:FILT:FREQ[:TYPE]` to select a digital filter. Once you have selected a filter and configured it (see the following discussion) you must enable the filter with `CALC:FILT:FREQ:STATe` when using the standard SCPI model.

Filter Parameters. Each digital filter function has a set of parameters that determine its effective frequency range and roll off. The LPASs and HPASs filters have specific cutoff parameters. Other parameters affect the roll off or transition region. Figure 3–21 shows the main parameters that control the four digital filters and the commands used to set them. Note that the parameters set the –6dB point of attenuation.

The NOTCh and BPASs bandpass filters are set with either of two command pairs: CENTer and SPAN or STARt and STOP. Figure 3–22 shows how these command pairs effectively control the same parameters. Setting a parameter of one pair affects the settings of the other pair. For example, you could set the BPAS filter parameter STARt to 50 MHz and STOP to 100 MHz using the following command:

```
CALC1:FILT:FREQ BPAS; FREQ:STAR 50E6; STOP 100E6
```

A query of CENTer and SPAN will return 75 MHz and 50 MHz, respectively. If you then set SPAN to 100 MHz and leave CENTer at 75 MHz, the STARt value changes to 25 MHz and the STOP value changes to 125 MHz. The new SPAN value is spread on either side of CENTer requiring a change in the ends of the pass band.

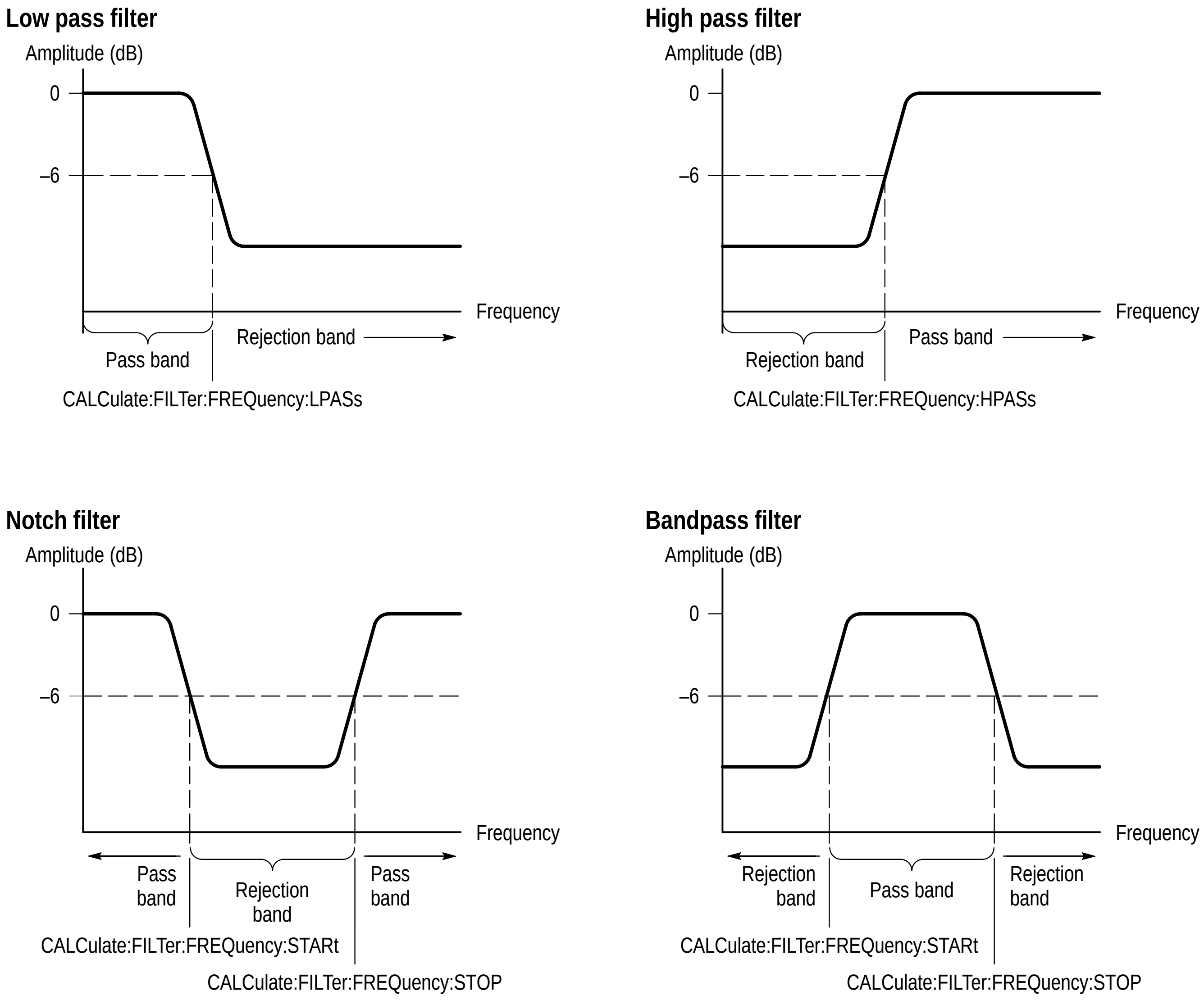


Figure 3–21: Parameters for the four digital filters

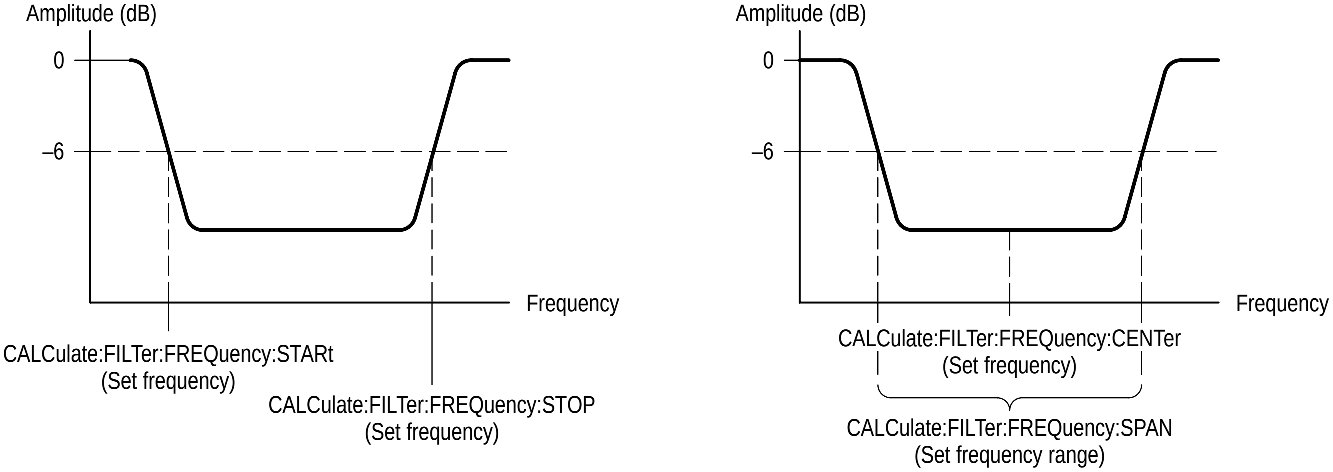

Amplitude (dB)
0
–6
Frequency
CALCulate:FILTer:FREQuency:STARt
(Set frequency)
CALCulate:FILTer:FREQuency:STOP
(Set frequency)
Amplitude (dB)
0
–6
Frequency
CALCulate:FILTer:FREQuency:CENTer
(Set frequency)
CALCulate:FILTer:FREQuency:SPAN
(Set frequency range)

Figure 3–22: Two methods of setting BPASs and NOTCh filters

Transition Settings. The roll off or transition region between the pass band and the rejection band for all filter functions is controlled by the parameters `SREJection` and `TWIDth`. Figure 3–23 shows these parameters. `SREJection` controls the degree of attenuation and is specified in dB. The minimum setting is 15 dB and the maximum is 100 dB. `TWIDth` determines the rate of roll off or transition width. `TWIDth` is specified as a value between 0 and 1 which is derived using the following relation:

`TWIDth` = Transition Frequency * 2 * `SWEep:TINTerval`

The transition frequency is the actual range of frequencies over which the filter transitions from the pass band to the rejection band. `SWEep:TINTerval` is the current sample interval. The smaller the value of `TWIDth` the smaller the transition region and steeper the transition slope. To set `TWIDth` to `0.05` you would use the following command:

```
CALC1:FILT:FREQ:TWID 0.05
```

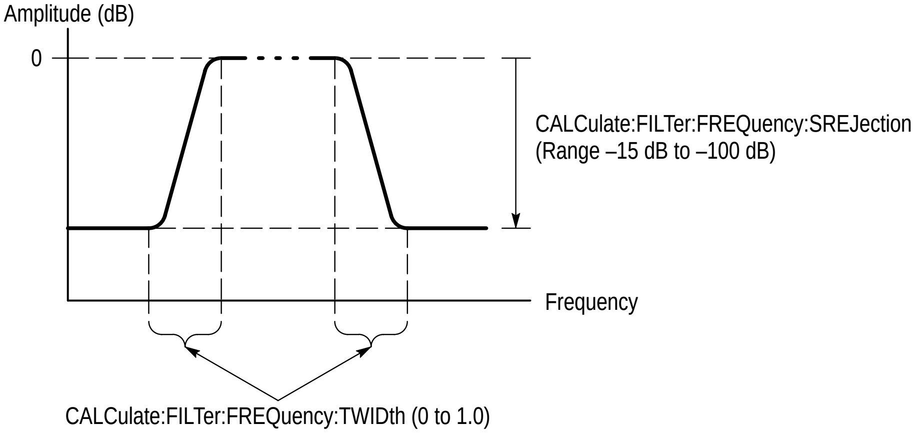

Amplitude (dB)
0
CALCulate:FILTer:FREQuency:SREJection
(Range –15 dB to –100 dB)
Frequency
CALCulate:FILTer:FREQuency:TWIDth (0 to 1.0)

Figure 3–23: Rejection level and transition slope for the digital filter

For more information on the digital filter, refer to the Filter discussion in *Appendix B, Algorithms*.

Waveform Differentiation

The calculation capabilities of the waveform analyzer include waveform differentiation. This capability allows you to compute a derivative math waveform that indicates the instantaneous rate of change of the acquired waveform. Derivative waveforms are useful in the measurement of the slew rate of amplifiers.

Use the command `CALC<n>:DERivative:STATe` to enable differentiation for a particular CALC block. For example, when `CALC2:DER:STATe` is `ON`, the CALC2 block will perform differentiation on the CALC2 source.

The resultant waveform is referred to as `CALC<n>` where `<n>` is the CALC block that performed the differentiation. Each of the four CALC blocks can perform differentiation when enabled. Use `TRACe? CALC<n>` to retrieve the resultant waveform.

The math waveform, derived from the sampled waveform, is computed based on the following equation:

$$Y_n = (X_{(n+1)} - X_n)\frac{1}{T}$$

Where: X is the source waveform
Y is the derivative math waveform
T is the time between samples

Since the resultant math waveform is a derivative waveform, its vertical units are volts/second (its horizontal units are seconds). The source signal is differentiated over its entire record length; therefore, the math waveform record length equals that of the source waveform.

Waveform Integration

The calculation capabilities of the waveform analyzer include waveform integration. This capability allows you to compute an integral math waveform that is an integrated version of the acquired waveform.

Use the command `CALC<n>:INTegral:STATe` to enable integration for a particular CALC block. For example, when `CALC2:INT:STATe` is `ON`, the CALC2 block will perform integration on the CALC2 source.

The resultant waveform is referred to as `CALC<n>` where `<n>` is the CALC block that performed the calculation. Each of the four CALC blocks can perform integration when enabled. Use `TRACe? CALC<n>` to retrieve the resultant waveform record.

The integral waveform record, derived from the acquired waveform, is computed based on the following equation:

$$y(n) = scale \sum_{i=1}^{n} \frac{x(i) + x(i-1)}{2} T$$

Where: x(i) is the source waveform
y(n) is a point in the integral waveform
scale is the output scale factor
T is the time between samples (sample interval)

Since the resultant waveform record is an integral waveform, its vertical units are volt-seconds (its horizontal units are seconds). The source signal is integrated over its entire record length; therefore, the waveform record length equals that of the source waveform.

***NOTE**. Often, an ac-coupled integration is desired. To simulate ac coupling, you can subtract the mean value of a waveform before integrating it, as the following example demonstrates:*

```
CALC1:PATH:EXPR "REF1:=INTEgrate(CHAN1–MEAN(CHAN1))"
```

Waveform Functions (TVS600A Models Only)

Waveform functions operate on a vector of arbitrary values and can only be used within a CALC expression; these functions cannot be used with SCPI-model calculations.

Absolute value

This function returns a vector of the absolute values for the source expression.

Syntax (opr <arg>)	Arguments		Returns
ABSolute(<expr>)	<expr>	An expression reducing to a scalar or a vector	<vector>

Average

The function sums the list of vectors and returns the vector of averages. It may be used in conjunction with the #= operator to accumulate data over multiple acquisition cycles. The members of the list of vectors are not required to be the same length; however, when they are not, the rules for dissimilar vector lengths apply (see page 3–63).

Syntax (opr <arg>)	Arguments		Returns
AVERage(<expr> [, ...])	<expr>	An expression reducing to a vector	<vector>

Envelope

The function takes the list of vectors and returns the vector envelope. It may be used in conjunction with the #= operator to accumulate data over multiple acquisition cycles.

Syntax (opr <arg>)	Arguments		Returns
ENVelope(<expr> [, ...])	<expr>	An expression reducing to vector	<envelope>

Envelope can be used to create templates against which a waveform can be tested. The following command creates a +/– 5% envelope around the channel-1 waveform and stores it in REF1:

```
CALC1:PATH:EXPR "REF1:=ENV (CHAN1+.05*HIGH (CHAN1),
CHAN1-.05*HIGH (CHAN1))"
```

The members of the list of vectors are not required to be the same length; however, when they are not, the following rules apply:

- When enveloping vectors that do not contain tuples, each vector value (a single data point) is enveloped into a 2-tuple, resulting in an output vector that is twice the size of the input vector. In short, there is one two-point tuple in the output vector for each single point in the input vector.
- When enveloping tuple vectors, such as an envelop vector of 2-tuples or a statistics vector of 4-tuples, each tuple in the input vector is is enveloped into a single tuple of the same size, resulting in an output vector equal in size to the input vector. In short, there is one n-point tuple in the output vector for each n-point point tuple in the input vector, where n is the same value for the input vectors and the output vector.

Segment

The segment function takes an expression, which must reduce to a vector, and extracts a segment of the expression.

Syntax (<arg> opr <arg>)	Arguments		Returns

SEGMent(<expr>,<range>)
SEGMent(<expr>,<start> [, <span>])
	<expr>	An expression evaluating to the initial vector	<vector>

	<range>	The start and stop indices[1] separated by the range operator ".."	
	<start>	The start index[1] for spanned segments	
	<span>	The number of points, the percentage, or amount of time spanned from <start>	

[1] Indices are by point, time, or percentage; see text below.

As the syntax above shows, the section extracted by `SEGMent` can be specified using the range operator ".." and declaring lower and upper limits as shown in Figure 3–24 (top); or it can be expressed as a span, with a start index and span value separated by a comma as shown in Figure 3–24 (bottom). The following characteristics apply to `SEGment` and its operators:

- If a range is defined in reverse order (start greater than stop), the segment will be extracted in reverse order while preserving data order within each tuple.
- If a span is defined as negative, the segment is extracted in forward order starting from the point to which span extends and preceding to the start index.

- Unitless arguments imply points; values can also be assigned units (and be scaled) using the standard suffixes `pct`, `s`, `ms`, `us`, `ns`, and `ps`. Also, the suffix `xu` denotes the xunit type that matches the base <`expr`>.
- The default count is the number of points from the start index to the end of the record.
- The segment function acts on tuples as a single data index; that is, SEGMent sees data only at the tuple level and will not disturb data order within tuples.

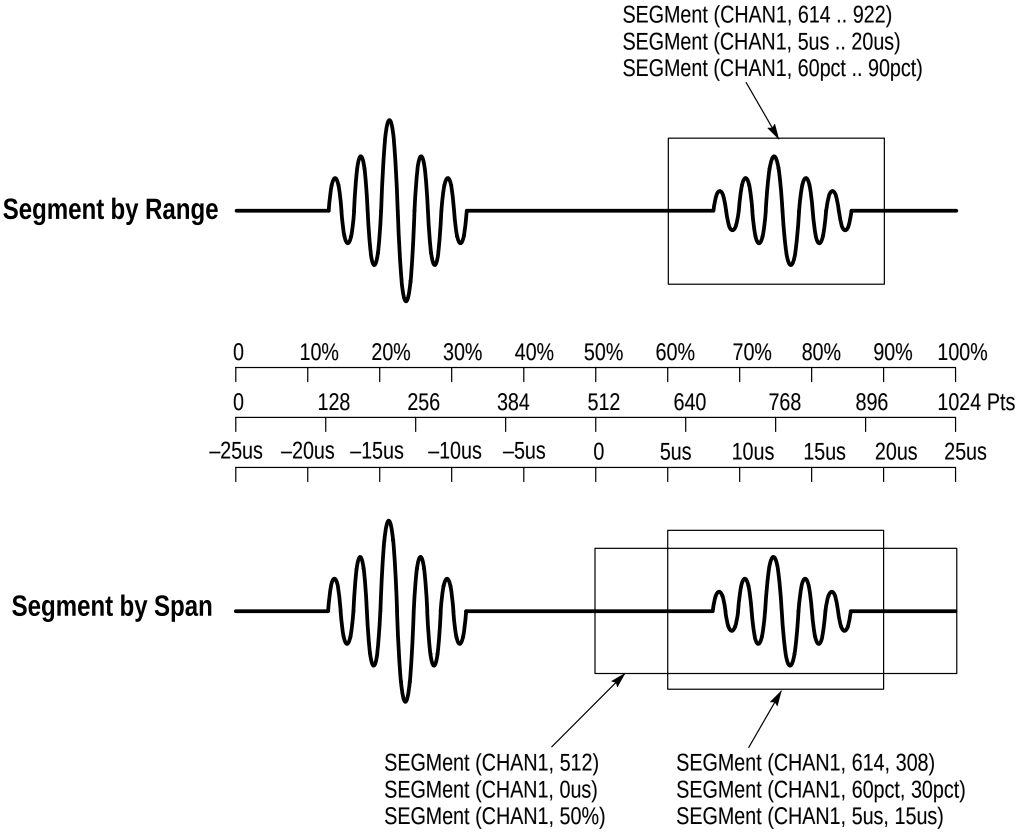


Figure 3–24: Range (top) and Span (bottom) define segments

Note that although the segment statements in Figure 3–24 do not do so, you can mix units when using start and span, for example, the span `SEGMent (CHAN4 60pct, 308)`. (You cannot, however, mix units in a range; the range `SEGMent (CHAN2, 30us .. 90pct)` is invalid.) Also note that the span segments at the left-bottom of Figure 3–24 specify a start index only, causing the segment to default to the end of the waveform record because no span value is specified.

Swapping the order of operands with the range operator swaps the data order in the segment as shown in Figure 3–25 (top). However, note that if the data is a tuple (an ordered set of values) the order of the values within the tuple remains unchanged. (See *Statistics* on page 3–59 for an example of how tuples organize data within vectors.)

You cannot swap the order of the operands when specifying span, but you can specify a negative span value; however, if you do so, the data is not reversed but is gathered starting at the negative span value, before the start index. See Figure 3–25 (bottom).

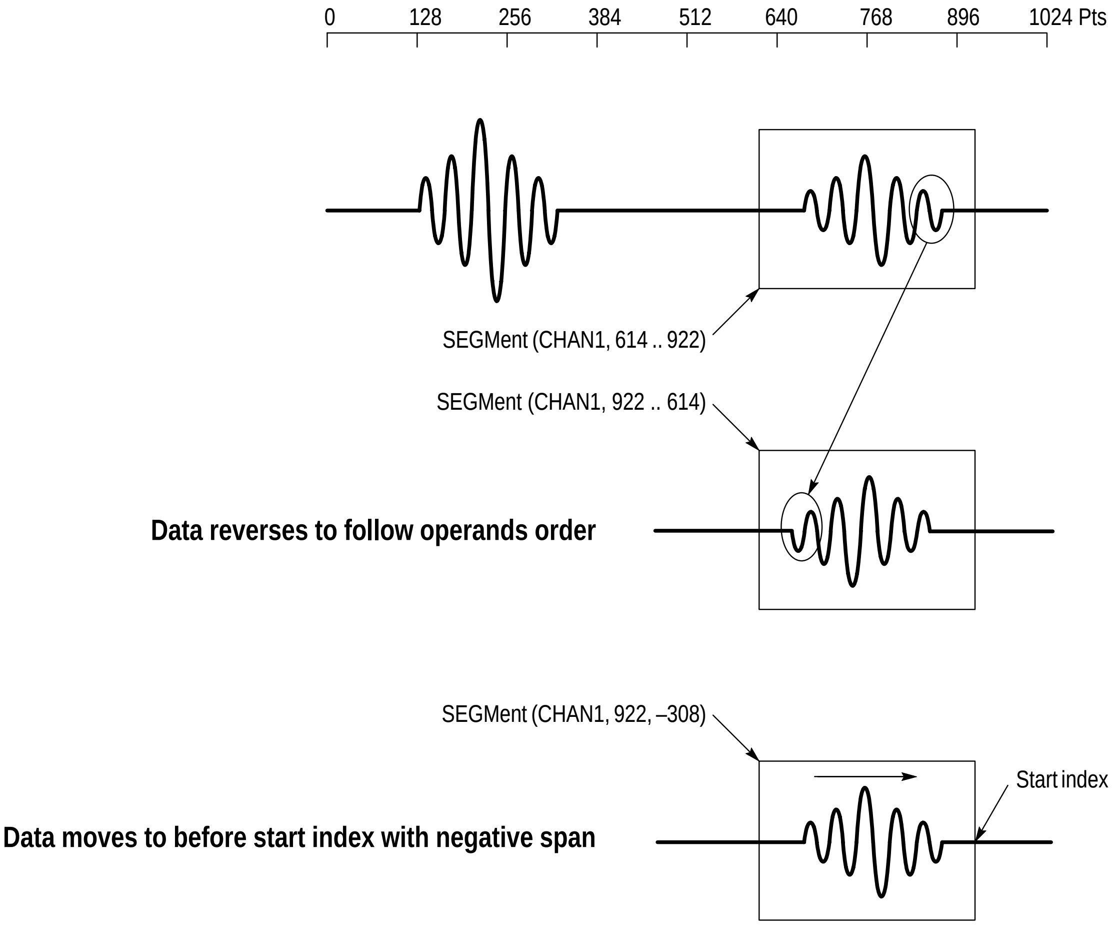

0
128
256
384
512
640
768
896
1024 Pts
SEGMent (CHAN1, 614 .. 922)
SEGMent (CHAN1, 922 .. 614)
Data reverses to follow operands order
SEGMent (CHAN1, 922, –308)
Start index
Data moves to before start index with negative span

Figure 3–25: Reversing range operands (top) and using negative span (bottom)

Vector

The vector function concatenates the content of all the expressions making up the list, returning a single vector. Dimensional information is ignored during the concatenation process; data is simply strung together. An exception occurs when the dimensional information is identical for all data sources in the arguments; if such is the case, the dimensional information is retained in the resulting output vector. This function can be used with the #= operator to accumulate vectors. (See Accumulate operator in Table 3–2 on page 3–37.)

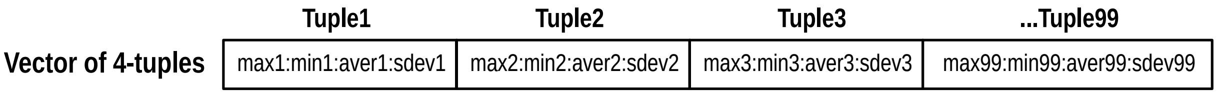
Syntax (opr <arg>)	Arguments	Returns
VECTor(<expr> [, ...])	<expr>	<vector>

Statistics

This function takes the list of vectors and returns a vector of 4-tuples (four datum per tuple). Each 4-tuple orders the data as max:min:average:standard_deviation internally, as shown in Figure 3–26.

Syntax	Arguments	Returns
STATistics(<expr> [, ...])	<expr> An expression ...	<4tuple>[1]

[1] An ordered group of numbers; for example, an envelope waveform is a vector of 2-tuples, where each 2-tuple contains a max:min pair.



	Tuple1	Tuple2	Tuple3	...Tuple99
Vector of 4-tuples	max1:min1:aver1:sdev1	max2:min2:aver2:sdev2	max3:min3:aver3:sdev3	max99:min99:aver99:sdev99

Figure 3–26: STATistics creates a vector of tuples of statistics

STATistics may be used with auto-advance acquisition to collect statistics on multiple acquisitions. For example, the following commands build a vector of 100 frequency measurements and returns a single 4-tuple (min:max:aver:sdev) that characterizes the 100 measurements:

```
CALC1:AAML FREQ
CALC1:PATH:EXPR "STAT(AAML(CHAN1))"
AADV:COUNT 100
INIT
```

STATistics may also be used with the #= operator to accumulate data when not using auto-advance acquisition over multiple acquisition cycles.

The members of the list of vectors are not required to be the same length; however, when they are not, the rules for dissimilar vector lengths apply (see *Dissimilar Vector Length Rules* on page 3–63).

Waveform Attribute Functions (TVS600A Models Only)

Waveform attribute functions give CALC expressions access to scaling information contained in the dimension block of the waveform preamble. These functions cannot be used with SCPI-model calculations.

X Duration

Returns a scalar which is the x-duration value for the vector. Duration is (`XSCale * XSIZe`).

Syntax	Arguments		Returns
XDURation(<expr>)	<expr>	An expression reducing to a vector	<scalar>

X Offset

Returns a scalar which is the X-offset value for the vector.

Syntax	Arguments		Returns
XOFFset(<expr>)	<expr>	An expression reducing to a vector	<scalar>

X Scale

Returns a scalar which is the X-scale value for the vector. For an acquired trace, this will typically equal the sample interval.

Syntax	Arguments		Returns
XSCale(<expr>)	<expr>	An expression reducing to a vector	<scalar>

X Size

Returns a scalar which is the number of elements in the source expression. In the case of raw acquisition data, this scalar equates to the number of points in the record. In the case of an enveloped trace, it equals the number of max:min pairs.

Syntax	Arguments		Returns
XSIZe(<expr>)	<expr>	An expression reducing to a vector	<scalar>

Boolean Aperture/Threshold Measurement Function (TVS600A Models Only)

The CALC system provides a single measurement function that collapses a Boolean vector into a scalar TRUE or FALSE value. The function cannot be used with SCPI-model calculations.

The Boolean Aperture/Threshold (BAT) function returns TRUE if the summation over the moving <aperture> number of consecutive values ever equals or exceeds the specified <threshold> value.

Syntax (<arg> opr <arg>)	Arguments[1]		Returns
BAT(<expr> [,<aperture, <threshold>])	<expr>	An expression containing a vector of Boolean values	<TRUE/FALSE>
	<aperture>	The width[2] of the aperture; defaults to the entire vector	
	<threshold>	The lower limit (inclusive) for the aperture sum which will return TRUE; defaults to 1.0.	

[1] **If you include a value for <threshold>, you must include a value for <aperture>. You can include <aperture> while omitting <threshold>. BAT (<expr>, 5) is valid; BAT (<expr>,,2) is not.**

[2] **The <aperture> parameter is in points if set without units; values may be assigned units (and scaled) using the standard suffixes pct, s, ms, us, ns, and ps. Also, the suffix xu denotes the xunit type of the base <expr>.**

Figure 3–27 illustrates the use of the BAT function, which operates as follows:

1. The BAT function first evaluates the expression, `CHAN 1 OUTside REF1`, reducing it to sequence of Boolean values (a Boolean vector).
2. BAT then slides a window of <aperture> width (4) along the Boolean values.
3. If BAT finds at least as many consecutive TRUE values as specified by <threshold> (also 4 in Figure 3–27), BAT returns TRUE; otherwise it returns FALSE.
4. In this case, since BAT finds 4 consecutive points that evaluate TRUE (that is, where Ch 1 is outside the Ref1 envelope), BAT returns TRUE. If the aperture and threshold were both increased to 5, BAT would return FALSE.

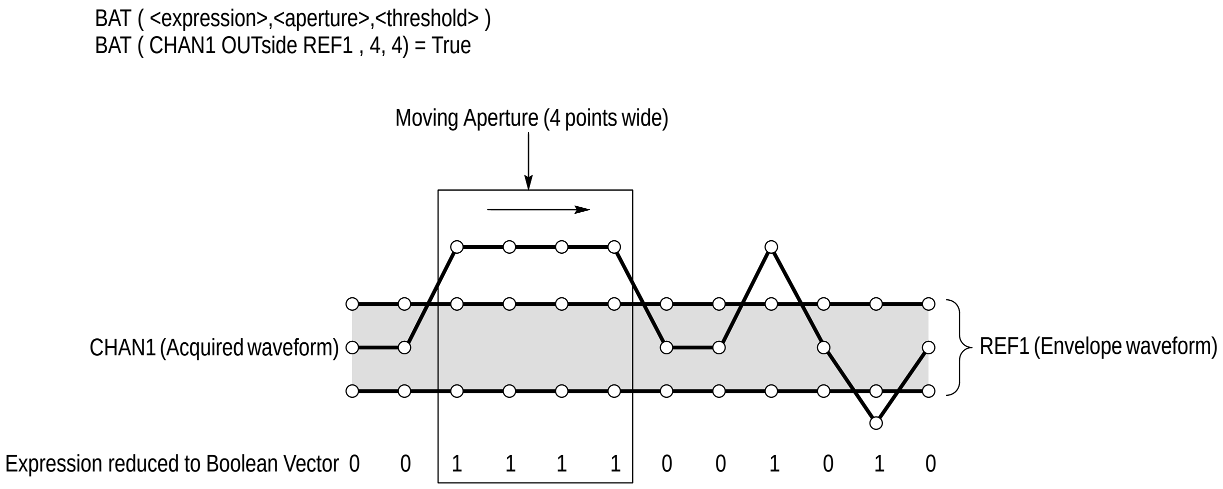

BAT (<expression>,<aperture>,<threshold>)
BAT (CHAN1 OUTside REF1 , 4, 4) = True
Moving Aperture (4 points wide)
CHAN1 (Acquired waveform)
REF1 (Envelope waveform)
Expression reduced to Boolean Vector 0 0 1 1 1 1 0 0 1 0 1 0

Figure 3–27: BAT evaluates a channel-to-reference comparison

In this case, since BAT finds 4 consecutive points that evaluate TRUE (that is, where Ch 1 is outside the Ref1 envelope), BAT returns true.

Control/Notification Functions (TVS600A Models Only)

These functions allow the waveform analyzer to take action based on the results of CALC expressions. (These functions cannot be used with SCPI-model calculations.) The CALC expression must reduce to a number (scalar), where nonzero values evaluate TRUE and zero evaluates FALSE. Control/Notification functions are useful when you want to halt operation or generate a trigger when a limit or measurement test passes or fails.

User Service Request

If the value of <scalar> is not zero, the User Request bit in the Standard Event Status Register will be asserted at the end of the current acquisition cycle. All CALC blocks will be evaluated and implicit outputs (FDC) completed. The function returns the Boolean value (TRUE/FALSE) of its input <scalar>.

Syntax	Arguments	Returns
SRQ(<scalar>)	<scalar> A number.	<TRUE/FALSE>

Event

If the value of <scalar> is not zero, the VXI event associated with the specific CALC block is immediately asserted. The function returns the Boolean value (TRUE/FALSE) of its input <scalar>.

Syntax	Arguments	Returns
EVT(<scalar>)	<scalar> A number.	<TRUE/FALSE>

Halt

If the value of <scalar> is not zero, the acquisition engine halts at the end of the current acquisition cycle. All CALC blocks will be evaluated, and implicit outputs (FDC) completed. The function returns the Boolean value (TRUE/FALSE) of its input <scalar>.

Syntax	Arguments	Returns
HLT(<scalar>)	<scalar> A number.	<TRUE/FALSE>

Trigger

If the value of <scalar> is not zero, the VXI CALC Trigger is immediately asserted. The `OUTPut:ECLTn:SOURce` or `OUTPut:TTLTn:SOURce` command must be set to `CALC` to select this trigger as the source for one of the VXI trigger lines. The function returns the Boolean value (TRUE/FALSE) of its input <scalar>.

Syntax	Arguments	Returns
TRG(<scalar>)	<scalar> A number.	<TRUE/FALSE>

Dissimilar Vector Length Rules

The handling of vectors within the CALC system follows defined, consistent rules. These rules are presented in this section.

Vectors of Different Tuple Size

Generally, operations on vectors ignore tuple organization, operating on vectors as if each vector contain only strings of individual data points, or 1-tuples. Operations on these vectors follow the rules listed for vectors of equal tuple size list below.

The ENVelope function does not ignore tuple organization; see *Envelope* on page 3–55.

Vectors of Equal Tuple Size

For vectors or equal tuple sized, the following rules apply:

- If all vectors are of equal length, the result is a vector of that length.
- If one vector is a scalar (length 1), the value is logically replicated to form a vector of length equal to that of the other vectors before operations are performed.
- If the vectors are of different lengths (but neither is of length 1), the operation is performed from left to right until the data from the shortest vector is exhausted. The result is not padded.

Command Groups

This section lists the waveform-analyzer commands by functional groups, providing an overview of the commands. For the complete alphabetical listing of the programming commands and their details, see the *TVS600 & TVS600A Waveform Analyzers Series Command Reference* (a standard-accessory manual).

A question symbol surrounded by brackets [?] identifies commands that also have a query form.

Auto-Advance Commands

Commands in the AADVance subsystem control how auto-advance acquisition records are acquired and transferred to a VXIbus controller.

Table 3–8: Auto-advance commands

Header	Description
`AADVance[?]`	Sets the state of the auto-advance acquisition mode.
`:COUNt[?]`	Sets the number of records to acquire in the auto-advance acquisition mode.
`:RECord`	
`:COUNt[?]`	Sets the number of auto-advance acquisition records to transfer.
`:STARt[?]`	Sets the number of the first auto-advance acquisition record to transfer.

Abort Commands

Commands in the ABORt subsystem operate with the ARM and TRIGger subsystems to stop signal acquisition.

Table 3–9: Abort commands

Header	Description
`ABORt`	Stops all acquisition and measurements and returns the arm/trigger subsystem to the idle state.

Arm Commands

Commands in the ARM Subsystem operate with the TRIGger, INITiate, and ABORt subsystems to trigger acquisitions.

Table 3–10: Arm commands

Header	Description
ARM	
:DEFine?	Returns the predefined SEQuence1 alias.
:SOURce[?]	Sets the source that will arm the acquisition system.

Average Commands

Commands in the AVERage subsystem control the averaging function.

Table 3–11: Average commands

Header	Description
AVERage[?]	Sets whether the waveform analyzer performs normal acquisition or one of average, peak detect, or envelope modes. (Peak-detect mode is available with TVS600A models only.)
:COUNt[?]	Sets the number of acquisition records to average, peak detect, or envelope.
:TYPE[?]	Sets one of average, peak-detect or envelope modes.

Calculate Commands

Commands in the CALCulate subsystems process and perform measurements on acquisition records.

Table 3–12: Calculate commands

Header	Description
CALCulate<n>	
:AAMList[?]	Sets the list of measurements to perform on auto-advance waveform records.
:STATe[?]	Sets whether to perform waveform measurement(s) on acquisition records captured with Auto Advance acquisition.

Table 3–12: Calculate commands (cont.)

Header	Description
:DATA?	Returns the results of waveform calculations and measurement functions.
:PREamble?	Returns the data preamble for acquisition record calculation or measurement results.
:DERivative	
:STATe[?]	Sets whether to process the acquisition record to produce a derivative of the acquisition record.
:FEED1[?]	Sets the source of data for the specified CALCulate block.
:FEED2[?]	Sets a second source of data for the specified CALCulate block. Used when taking dual-waveform measurements, such as gain or delay.
:CONText[?]	Sets the measurement parameter block (calc_block) used to characterize the feed2 waveform.
:FILTer	
:FREQuency	Sets the type of FREQuency filtering to perform on an acquisition record.
:CENTer[?]	Sets the center frequency to be used by the bandpass or notch filters.
:HPASs[?]	Sets the limit frequency below which the filter attenuates all frequency components.
:LPASs[?]	Sets the limit frequency above which the filter attenuates all frequency components.
:SPAN[?]	Sets the frequency range to be used by the bandpass and notch filters.
:SREJection[?]	Sets the level of rejection or attenuation for frequency components in the defined stop band.
:STARt[?]	Sets the start, or lower limit, frequency of the bandpass and notch filters.
:STATe[?]	Sets whether frequency filtering will be performed on acquisition records.
:STOP[?]	Sets the stop, or upper limit, frequency of the bandpass and notch filters.
:TWIDth[?]	Sets the slope of roll off for the post-acquisition filter.
:FORMat[?]	Sets whether to process the acquisition record to produce a new format.
:IMMediate[?]	Sets the specified CALCulate block to reprocess SENSe data without reacquiring new data.
:INTegral	

Table 3–12: Calculate commands (cont.)

Header	Description
:STATe[?]	Sets whether to process the acquisition record to produce an integral of the acquisition record.
:PATH[?]	Sets a list of CALCulate functions to execute in the order listed.
:EXPRession[?]	Sets an algebraic expression for calculating waveforms and other data.
:SMOothing[?]	Sets whether to perform smoothing on an acquisition record.
:POINts[?]	Sets the number of adjacent points to average in the acquisition record.
:TRANsform	
:FREQuency	
:STATe[?]	Sets whether to perform a Fast Fourier Transform (FFT) on an acquisition record.
:WINDow[?]	Sets the type of data windowing (or shaping) to use prior to the FFT transformation.
:WMList[?]	Sets the list of waveform measurements to perform.
:STATe[?]	Sets whether the waveform measurement list for the specified CALC block will execute after the next acquisition.
:WMParameter	
:EDGE[?]	Set the number of the edge in the waveform record to use for taking delay, cross, ncross, and pcross measurements.
:GATE[?]	TVS600A only. Turns on or off gating of measurements.
:METHod[?]	TVS600A only. Sets the ABSolute or RELative method for gating.
:STARt[?]	TVS600A only. Sets the gating start point on the waveform record.
:STOP[?]	TVS600A only. Sets the gating stop point on the waveform record.
:HIGH[?]	Sets the high (most positive) level used for time and amplitude measurements.
:HMEThod[?]	Sets the method for calculating the high (most positive) level for time and amplitude measurements.
:LOW[?]	Sets the low (most negative) level used for time and amplitude measurements.
:LMEThod[?]	Sets the method for calculating the low (most negative) level for time and amplitude measurements.

Table 3–12: Calculate commands (cont.)

Header	Description
:HFREFerence[?]	Sets the high reference (distal) level in vertical units for time and amplitude measurements.
:RELative[?]	Sets the high reference (distal) level used for time and amplitude measurements.
:LREFerence[?]	Sets the low reference (proximal) level in vertical units for time and amplitude measurements.
:RELative[?]	Sets the low reference (proximal) level used for time and amplitude measurements.
:MREFerence[?]	Sets the middle reference (mesial) level in vertical units for time and amplitude measurements.
:HYSTeresis[?]	Sets the middle reference (mesial) hysteresis used for time and amplitude measurements.
:RELative[?]	Sets the middle reference (mesial) level used for time and amplitude measurements.
:RMEThod[?]	Sets the method for calculating the reference (high, middle, low) levels for time and amplitude measurements.
:SLOPe[?]	Sets the direction, positive or negative, for the waveform edges used in delay measurements.

Calibration Commands

Commands in the CALibration subsystem run the waveform analyzer self-calibration functions.

Table 3–13: Calibration commands

Header	Description
CALibration[?]	Executes all self-calibration functions.
:RESults?	Returns the results code for the last calibration performed.
:VERBose?	Returns an ASCII string describing the results of the last calibration performed.
:PROBe[n][?]	TVS600A only. Calibrate the gain of a level 2 probe connected to channel specified by [n].
:RESults?	TVS600A only. Returns calibration status of probe specified.

Data Commands

Commands in the DATA subsystem provide a means of accessing the data produced by the sense functions.

Table 3–14: Data commands

Header	Description
DATA?	Returns the results for the specified function or for all sense functions that are enabled.
:PREamble?	Returns the data preamble for the specified function or for all enabled sense functions.

Format Commands

Commands in the FORMat subsystem set the format of acquisition record data and measurement data transferred out of or into the waveform analyzer.

Table 3–15: Format commands

Header	Description
FORMat[?]	Sets the type of encoding used to transfer acquisition records acquired with the SENSe subsystem.
:BORDer[?]	Sets the byte order used to transfer binary data.
:CALCulate[n][?]	Sets the type of data format used to transfer CALCulate data.
:TRACe	
:AATS[?]	Sets the type of data format used to transfer AATS data.
:REF[?]	Sets the type of data format used to transfer REF data.
:DINTerchange[?]	TVS600A only. Sets the data interchange format to enabled or disabled.

Function Commands

Commands in the FUNCtion subsystem control sense functions. The sense functions for the waveform analyzer are its input channels.

Table 3–16: Function commands

Header	Description
`FUNCtion[?]`	Sets which sense functions are enabled (which channels acquire).
`:ALL`	Enables all sense functions.
`:COUNt?`	Returns the number of enabled sense functions.
`:OFF[?]`	Sets which sense functions are disabled.
`:ALL`	Disables all sense functions at once.
`:COUNt?`	Returns the number of disabled sense functions.
`:CONCurrent[?]`	Sets whether more than one sense function can be enabled at a time.
`:STATe[?]`	Sets the state of a specified sense function.

Initiate Commands

Commands in the INITiate subsystem operate with the ARM and TRIGger subsystems to start signal acquisition.

Table 3–17: Initiate commands

Header	Description
`INITiate`	Starts waveform analyzer acquisitions and measurements.
`:CONTinuous[?]`	Sets whether the acquisition loop repeats continuously.
`:COUNt[?]`	Sets the number of times to repeat the arm/trigger acquisition loop.

Input Commands

Commands in the INPut subsystem control input parameters that include coupling, filtering, impedance, and protection.

Table 3–18: Input commands

Header	Description
INPut<n>	
:COUPling[?]	Sets the type of signal coupling for the specified input channel.
:FILTer[?]	Sets whether the low pass filter is on or off.
:FREQuency[?]	Sets the frequency limit of the low pass filter.
:IMPedance[?]	Sets the input impedance.
:PROBe	
:ATTenuation[?]	TVS600A only. Returns the input attenuation factor of the attached level 2 probe.
:IDENtification[?]	TVS600A only. Returns the identification of the attached level 2 probe. (ID is <probe type>, <serial number>.)
:OFFSet[?]	TVS600A only. Returns the input offset factor of the attached level 2 probe.
:PROTection	
:STATe[?]	Sets the state of the input protection circuitry for all input channels.

Memory Commands

Commands in the MEMory subsystem store and retrieve instrument settings.

Table 3–19: Memory commands

Header	Description
MEMory	
:DATA[?]	Sets the instrument settings (or state) for the ten on-board nonvolatile memory locations.
:NSTates?	Returns the number of instrument settings (states) that can be stored in the waveform analyzer.
:STATe	
:CATalog?	Returns the list of predefined names for the on-board stored settings.
:DEFine?	Returns the register number of a specified instrument settings location in the waveform analyzer.

Output Commands

Commands in the OUTPut subsystem route signals to the VXIbus trigger lines and enable the probe compensation and reference signals.

Table 3–20: Output commands

Header	Description
`OUTPut`	
`:ECLTrg<n>[?]`	Sets whether the instrument should drive (source) the specified VXIbus ECL trigger line when a trigger event occurs.
`:POLarity[?]`	Sets the drive polarity for each VXIbus ECL trigger line.
`:SOURce[?]`	Sets the drive source for each VXIbus ECL trigger line.
`:PCOMpensate[:STATE][?]`	Sets whether the probe compensation signal is output on the connector PROBE COMPENSATION.
`:FUNCtion[?]`	Sets which compensate signal is output on the connector PROBE COMPENSATION.
`:REFerence[:STATE][?]`	Sets whether a reference signal is output on the front-panel connector REFERENCE OUTPUT.
`:FUNCtion[?]`	Sets which reference signal is output on the connector REFERENCE OUTPUT.
`:TTLTrg<n>[?]`	Sets whether the instrument should drive (source) the specified VXIbus TTL trigger lines.
`:POLarity[?]`	Sets the drive polarity for each VXIbus TTL trigger line.
`:SOURce?`	Sets the drive source for each VXIbus TTL trigger line.

Roscillator Commands

Commands in the ROSCillator subsystem control the source of the reference oscillator (clock) for the :SWEep subsystem.

Table 3–21: Reference Oscillator commands

Header	Description
`ROSCillator`	
`:SOURce[?]`	Sets the source of the 10 MHz clock reference for the acquisition system.

Sense Commands

For a listing of the SENSe commands refer to their root level names in this section:

- AADVance, page 3–65
- AVERage, page 3–66
- DATA, page 3–70
- FUNCtion, page 3–71
- ROSCillator, page 3–73
- SWEep, page 3–75
- VOLTage, page 3–82

Status Commands

Commands in the STATus subsystem, along with several IEEE 488.2 Common Commands, control the status and event reporting system.

Table 3–22: Status commands

Header	Description
STATus	
:OPERation?	Returns the contents of the Operation Status Register as a decimal number.
:CONDition?	Returns the contents of the Operation Status Condition Register (OSCR).
:ENABle[?]	Sets the contents of the Operation Status Enable Register (OSER).
:NTRansition[?]	Sets the contents of the Operation Negative Transition Register (ONTR).
:PTRansition[?]	Sets the contents of the Operation Positive Transition Register (OPTR).
:QENable	
:NTRansition[?]	Sets the contents of the Negative Transition Queue Enable Register (NTQER) for the Operation Status Register.
:PTRansition[?]	Sets the contents of the Positive Transition Queue Enable Register (PTQER) for the Operation Status Register.

Table 3–22: Status commands (cont.)

Header	Description
:PRESet	Presets the SCPI Enable and Transition registers, and the Status Queue enable registers.
:QUEStionable?	Returns the contents of the Questionable Status Register.
:CONDition?	Returns the contents of the Questionable Status Condition Register (QSCR).
:ENABle[?]	Sets the contents of the Questionable Status Enable Register (QSER).
:NTRansition[?]	Sets the contents of the Questionable Negative Transition Register (QNTR).
:PTRansition[?]	Sets the contents of the Questionable Positive Transition Register (QPTR).
:QENable	
:NTRansition[?]	Sets the contents of the Negative Transition Queue Enable Register (NTQER) for the Questionable Status Register.
:PTRansition[?]	Sets the contents of the Positive Transition Queue Enable Register (PTQER) for the Questionable Status Register.
:SESR	
:QENable	Sets the contents of the Event Status Enable Register (ESER).

Sweep Commands

Commands in the SWEep subsystem control the acquisition timebase for all VOLTage[n] acquisitions.

Table 3–23: Sweep commands

Header	Description
SWEep	
:OFFSet	
:POINts[?]	Sets the position in points of the acquisition record relative to the trigger point.
:TIME[?]	Sets the position in time (seconds) of the acquisition record relative to the trigger point.
:OREFerence	

Table 3–23: Sweep commands (cont.)

Header	Description
:LOCation[?]	Sets the location of the reference point in an acquisition record.
:POINts[?]	Sets the number of data points in an acquisition record.
:TIME[?]	Sets the time span or duration of the acquisition record.
:TINTerval[?]	Sets the time interval between acquired data points.

System Commands

Commands in the SYSTem subsystem program utility functions and return version information about the waveform analyzer.

Table 3–24: System commands

Header	Description
SYSTem	
:AUToset	TVS600A only.
:VOLTage	TVS600A only. Sets VOLTage:RANGe:UPPer and :LOWer on the specified channel so that the incoming signal fills the center 90% of the vertical window.
:SWEep	TVS600A only. Sets SWEep:TINTerval to the nearest value required to acquire 2.5 cycles of the incoming signal on the specified channel.
:TRIGger	TVS600A only. Sets the TRIGger[:A]:SOURe to the channel specified and the TRIGger:LEVel to 50% of the peak to peak value of the trigger signal.
:BDATE[?]	TVS600A only. Sets the date of installation of the battery supplying the nonvolatile memory.
:CDATE[?]	TVS600A only. Sets the date of the last factory service/calibration in nonvolatile memory.
:COMMunicate	
:SERial	
:BAUD?	Sets the baud rate of the front panel RS-232 port.
:CONTrol	
:DCD[?]	Sets whether the instrument is sensitive to the DCD line.
:RTS[?]	Sets the operation of the RTS and CTS lines.
:ECHO[?]	Sets whether incoming characters are echoed back.
:ERESponse[?]	Sets whether error messages are automatically returned.

Table 3–24: System commands (cont.)

Header	Description
:LBUFfer[?]	Sets the state of the character buffer.
:PACE[?]	Sets whether software flow control (XON/XOFF) is enabled.
:PARity[?]	Sets the type of parity for the front panel RS-232 port.
:PRESet	
[:ALL]	Configures RS-232 port parameters to default values.
:RAW	Configures the RS-232 port parameters for use with a computer.
:TERMinal	Configures the RS-232 port parameters for use with a terminal.
:SBITs[?]	Sets the number of stop bits sent with each character.
:ERRor?	Returns the next entry from the waveform analyzer Status Queue.
:ALL?	Returns the list of all events stored in the waveform analyzer Status Queue.
:CODE?	Returns the next event code stored in the waveform analyzer Status Queue.
:ALL?	Returns the list of all event codes stored in the waveform analyzer Status Queue.
:COUNt?	Returns the number of unread events in the Status Queue.
:PROTect[?]	Sets whether protection for a group of sensitive instrument commands is enabled.
:SECurity	
:IMMediate	Immediately destroys all measurement and reference data and stored instrument settings.
:SET[?]	Sets the internal state of the instrument as a binary data block.
:VERSion?	Returns the SCPI version supported by the waveform analyzer.

Test Commands

Commands in the TEST subsystem execute the internal self-tests of the waveform analyzer module.

Table 3–25: Test commands

Header	Description
TEST	Executes all internal self-tests once.
[:ALL][?]	Executes all internal self-tests once. The query returns the test results.
:RESults?	Returns the failure code for the last self-test command that was executed.
:VERBose?	Returns a failure code as a string describing the last executed self-test command and the test results.
:STOP	TVS600A only. Aborts an active test sequence when the test that's currently executing test completes.

Trace Commands

Commands in the TRACe subsystem store and retrieve acquisition and measurement results.

Table 3–26: Trace commands

Header	Description
TRACe?	Transfers acquisition records or measurement results to your VXIbus controller.
:PREamble?	Transfers the data preamble for acquisition records or measurement results to your VXIbus controller.
:CATalog?	Returns list of the predefined trace names in your waveform analyzer.
:COPY	Copies acquisition or measurement data to the outgoing Fast Data Channel (FDC) or to a REF trace.
:DELete	TVS600A only.
[:NAME]	TVS600A only. Deletes all data in the specified REF trace.
:DELete:ALL	TVS600A only. Deletes all data in all ten REF traces.
:FEED?	Returns the source of data for pre-defined trace names.
:LIST[?]	Sets the list of traces to transfer using FDC.
:POINts?	Returns the number of sample points in the acquisition record or CALCulate block record.

Trigger Commands

Commands in the TRIGger subsystem operate with the ARM, INITiate and ABORt subsystems to trigger acquisitions.

Table 3–27: Trigger commands

Header	Description
TRIGger	
:ATRigger[:STATe][?]	Sets whether to generate an automatic trigger.
:COUPling[?]	Sets the source of the A trigger to AC or DC coupled.
:<preset>	Sets trigger coupling, filtering, and hysteresis.
:DELay[?]	Sets the trigger delay for the trigger A circuit.
:FILTer	Sets the state of the 50 kHz low pass trigger filter.
:HPASs[:STATe][?]	Sets the state of the 50 kHz high pass trigger filter.
:HOLDoff	
:TIME[?]	Sets the trigger holdoff time.
:HYSTeresis	
:SELect[?]	Sets how far the trigger A signal must fall below or rise above TRIGger:LEVel to detect an edge.
:LEVel[?]	Sets the trigger level for the TRIGger subsystem.
:METastable	TVS600A only.
:STATe[?]	TVS600A only. Sets whether data acquired on a metastable trigger is rejected.
:SLOPe[?]	Sets whether triggering occurs on the positive-going or negative-going edge of the trigger source.
:SOURce[?]	Sets the source of the trigger signal for the trigger A circuit.
:TYPE[?]	Sets the type of triggering to use for the next acquisition.
:DEFine?	Returns the predefined SEQuence1 alias, A.
:B	
:COUPling[?]	Sets the source of the B trigger to AC or DC coupled.
:<preset>	Sets B trigger coupling, filtering, and hysteresis.
:DELay[?]	Sets the trigger delay for the trigger B circuit.
:ECOunt[?]	Sets the number of B trigger events to count before starting acquisition.

Table 3–27: Trigger commands (cont.)

Header	Description
:FILTer	Sets the state of the 50 kHz, low-pass filter for the B trigger circuit.
:HPASs[:STATe][?]	Sets the state of the 50 kHz high pass filter for the B trigger circuit.
:HYSTeresis[?]	
:SELect[?]	Sets how far the trigger B signal must fall below or rise above TRIGger:B:LEVel to detect an edge.
:LEVel[?]	Sets the trigger level for the TRIGger:B subsystem.
:SLOPe[?]	Sets whether triggering occurs on the positive-going or negative-going edge of the trigger B source.
:SOURce[?]	Sets the source of the trigger signal for the trigger B circuit.
:SEQuence2	
:DEFine?	Returns the predefined SEQuence2 alias, B.
:LOGic	
:CLASs[?]	TVS600A only. Sets the class of logic triggering to use for the next acquisition.
:CONDition[?]	TVS600A only. Sets the bit-pattern for state or pattern triggering.
:FUNCtion[?]	TVS600A only. Sets the Boolean logic function applied to the bit-pattern.
:PATTern	TVS600A only.
:QUALify[?]	TVS600A only. Sets time qualification to GT (TRUE Greater Than), LT (TRUE Less Than), or OFF (no time qualification).
:WIDTh[?]	TVS600A only. Sets the time for time qualification.
:STATe	TVS600A four-channel models only.
:SLOPe[?]	TVS600A four-channel models only. Selects the edge of clock signal (from CH4).
:THReshold[?]	TVS600A only. Sets the High/Low thresholds for logic inputs.
:PULSe	
:CLASs[?]	Sets the class of pulse triggering to use for the next acquisition.
:GLITch	
:POLarity[?]	Sets the polarity of the event pulse for pulse glitch triggering.

Table 3–27: Trigger commands (cont.)

Header	Description
:QUALify[?]	Sets the type of time qualification for pulse glitch triggering.
:WIDth[?]	Sets the width of the pulse used for pulse glitch triggering.
:SOURce[?]	Sets the source of the trigger signal used for all pulse triggering.
:THReshold[?]	Sets the voltage threshold used for pulse triggering.
:TIMEout	TVS600A only.
:POLarity[?]	TVS600A only. Sets the polarity of the event pulse for pulse timeout triggering.
:WIDth[?]	TVS600A only. Sets the width of the pulse used for pulse timeout triggering.
:WIDth	
:HLIMit[?]	Sets the upper or maximum valid pulse width to qualify for pulse triggering.
:LLIMit[?]	Sets the lower or minimum valid pulse width to qualify for pulse triggering.
:POLarity[?]	Sets the polarity of the pulse used for pulse width triggering.
:QUALify[?]	Sets the type of time qualification for pulse width triggering.
:SHOLDtime	TVS600A only.
:CLOCk	TVS600A only.
:POLarity[?]	TVS600A only. Sets the polarity of the the clock edge.
:SOURce[?]	TVS600A only. Selects the channel containing the clock (the clock source).
:THReshold[?]	TVS600A only. Sets the threshold for recognition of the clock edge.
:DATA	TVS600A only.
:SOURce[?]	TVS600A only. Selects the channel containing the data monitored for set & hold time violations (the data source).
:THReshold[?]	TVS600A only. Sets the threshold for recognition of a data transition
:HTIMe[?]	TVS600A only. Sets the time data must be valid after the clock edge.

Table 3–27: Trigger commands (cont.)

Header	Description
:STIMe[?]	TVS600A only. Sets the time data must be valid before the clock edge.
:TRANsition	TVS600A only.
:CLASs[?]	TVS600A only. Sets the class, which determines whether to trigger on runt pulses or on pulse slew rates.
:RUNT	TVS600A only.
:QUALify[?]	TVS600A only. Sets the time-qualification type to GT (Greater Than) or LT (Less Than).
:SLOPe[?]	TVS600A only. Sets the polarity of the pulse used for runt triggering.
:SLEW	TVS600A only.
:QUALify[?]	TVS600A only. Sets the time-qualification type to GT (Greater Than) or LT (Less Than) for slew-rate triggering.
:SLOPe[?]	TVS600A only. Sets the polarity of the pulse edge used for slew-rate triggering.
:SOURce[?]	TVS600A only. Selects the channel to serve as the trigger source.
:THReshold	TVS600A only.
:HIGH[?]	TVS600A only. Sets the high level for a pulse transition (edge).
:LOW[?]	TVS600A only. Sets the high level for a pulse transition (edge).
:TIME[?]	TVS600A only. Sets the time used when time qualification is enabled for runt or slew-rate triggering.

Voltage Commands

Commands in the VOLTage subsystem control parameters that relate to the input voltage range of the waveform analyzer.

Table 3–28: Voltage commands

Header	Description
VOLTage<n>	
:RANGe	

Table 3–28: Voltage commands (cont.)

Header	Description
[:UPPer][?]	Sets the most positive end of the amplifier voltage range.
:LOWer[?]	Sets the most negative end of the amplifier voltage range.
:OFFSet[?]	Sets the voltage offset.
:PTPeak[?]	Sets the peak-to-peak (full-scale) voltage range.

IEEE 488.2 Commands

The waveform analyzer supports the following IEEE 488.2 common commands.

Table 3–29: IEEE 488.2 Common commands

Header	Description
*CAL?	Initiates internal calibration and returns a status code.
*CLS	Clears the SCPI and IEEE 488.2 event registers and the Status Queue.
*ESE[?]	Sets the Event Status Enable Register (ESER).
*ESR?	Returns the contents of the Standard Event Status Register.
*IDN?	Returns the waveform analyzer identification message.
*LRN?	Returns the current state of the waveform analyzer as a sequence of ASCII settings. (Sequence is resendable.)
*OPC[?]	Synchronizes command execution with the controller.
*OPT?	Returns the options installed in the instrument.
*PUD[?]	Sets the protected user data stored in the waveform analyzer.
*RCL	Recalls the specified instrument setting from non-volatile memory.
*RST	Resets instrument settings to a default state.
*SAV	Saves the current instrument settings in the non-volatile memory.
*SRE[?]	Sets the Service Request Enable Register (SRER).
*STB?	Returns the contents of the Status Byte Register.
*TST?	Initiates an internal self-test and returns a status code.
*WAI?	Synchronizes command execution with the system controller.

Command Syntax

This section contains information on the Standard Commands for Programmable Instruments (SCPI) and IEEE 488.2 Common Commands that you can use to program your waveform analyzer.

SCPI Commands and Queries

SCPI is a standard created by a consortium that provides guidelines for remote programming of instruments. These guidelines provide a consistent programming environment for instrument control and data transfer. This environment uses defined programming messages, instrument responses, and data format across all SCPI instruments, regardless of manufacturer. The waveform analyzer uses a command language based on the SCPI standard.

The SCPI language is based on a hierarchical or tree structure (see Figure 3–28) that represents a subsystem. The top level of the tree is the root node; it is followed by one or more lower-level nodes.

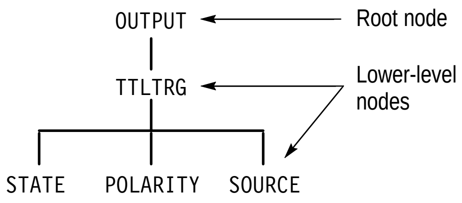


Figure 3–28: Example of SCPI-subsystem hierarchy tree

You can create commands and queries from these subsystem hierarchy trees. Commands specify actions for the instrument to perform. Queries return measurement data and information about parameter settings.

Creating Commands

SCPI command headers are created by stringing together the nodes of a subsystem hierarchy and separating each node by a colon.

In Figure 3–28, OUTPUT is the root node and TTLTRG, STATE, POLARITY, and SOURCE are lower-level nodes. To create a SCPI command, start with the root node OUTPUT and move down the tree structure adding nodes until you reach the end of a branch. Most commands and some queries have parameters; you must include a value for each of these parameters. If you specify a parameter value that is out of range, the parameter will be generally set to a default value and return an error (if error return is enabled). The *TVS600A Command Reference* lists the valid values for all parameters.

For example, `OUTPUT:TTLTRG1:STATE ON` is a valid SCPI command created from the hierarchy tree in Figure 3–28.

Creating Queries

To create a query, start at the root node of a tree structure, move down to the end of a branch, and add a question mark. `OUTPUT:TTLTrg:STATe?` is an example of a valid SCPI query using the hierarchy tree in Figure 3–28.

Parameter Types

Every parameter in the command and query descriptions is of a specified type. The parameters are enclosed in brackets, such as <pattern>. The parameter type is listed after the parameter and is enclosed in parentheses, for example, (discrete). Some parameter types are defined specifically for the waveform analyzer command set;some are defined by ANSI/IEEE 488.2-1992 (see Table 3–30).

Table 3–30: Parameter types used in syntax descriptions

Parameter Type	Description	Example
binary	Binary numbers	#B0110
binary block[1]	A specified length of binary data	#512234xxxxx . . . where 5 indicates that the following 5 digits (12234) specify the length of the data in bits; xxxxx ... indicates the binary data
boolean	Boolean numbers or values	ON or 1 OFF or 0
discrete	A list of specific values	HIGH, LOW, MID
hexadecimal[2]	Hexadecimal numbers (0–9, A, B, C, D, E, F)	#HAA, #H1
NR1[2,3] numeric	Integers	0, 1, 15, –1
NR2[2] numeric	Decimal numbers	1.2, 3.141516, –6.5
NR3[2] numeric	Floating point numbers	3.1415E–9, –16.1E5
NRf[2] numeric	Flexible decimal number that may be type NR1, NR2 or NR3	See NR1, NR2, NR3 examples
string[4]	Alphanumeric characters (must be within quotation marks)	“Testing 1, 2, 3”

[1] **Defined in ANSI/IEEE 488.2 as “Definite Length Arbitrary Block Response Data.”**

[2] **An ANSI/IEEE 488.2–1992-defined parameter type.**

[3] **Some commands and queries will accept a hexadecimal value even though the parameter type is defined as NR1.**

[4] **Defined in ANSI/IEEE 488.2 as “String Response Data.”**

Abbreviating Commands, Queries, and Parameters

You can abbreviate most SCPI commands, queries, and parameters to an accepted short form. This manual shows these short forms as a combination of upper and lower case letters. The upper case letters indicate the accepted short form of a command. As shown in Figure 3–29, you can create a short form by using only the upper case letters. The accepted short form and the long form are equivalent and request the same action of the instrument. Intermediate forms are not accepted; for example `OUTPu`.

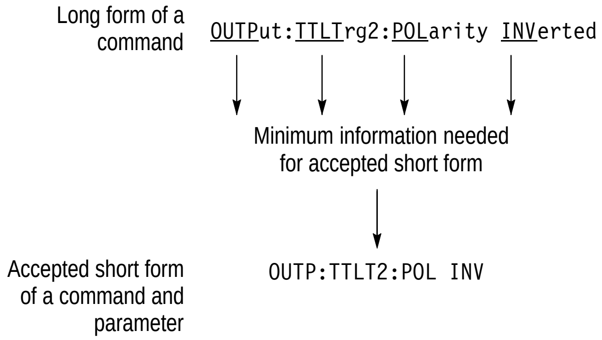
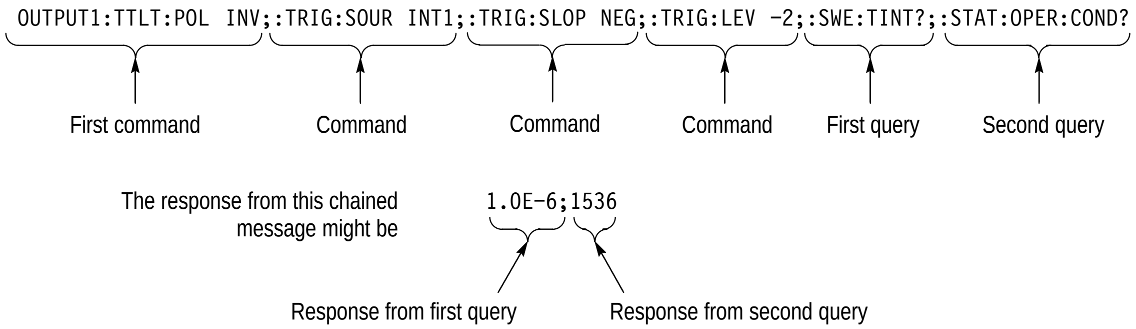

Long form of a command
OUTPut:TTLTrg2:POLarity INVerted
Minimum information needed for accepted short form
Accepted short form of a command and parameter
OUTP:TTLT2:POL INV

Figure 3–29: Example of abbreviating a command

NOTE*. The numeric part of a command or query must always be included in the accepted short form. In Figure 3–29, the "2" of "TTLTRG2" is always included in the short form.*

Chaining Commands and Queries

You can chain several commands or queries together into a single message. To create a chained message, first create a command or query, add a semicolon (;), and then add more commands or queries and semicolons until you are done. If the command following a semicolon is a root node, precede it with a colon (:). Figure 3–30 illustrates a chained message consisting of several commands and queries. A semicolon is not required after the final command or query in a chained message. Responses to any queries in your message are separated by semicolons and returned in the order sent.



```
OUTPUT1:TTLT:POL INV;:TRIG:SOUR INT1;:TRIG:SLOP NEG;:TRIG:LEV -2;:SWE:TINT?;:STAT:OPER:COND?
```

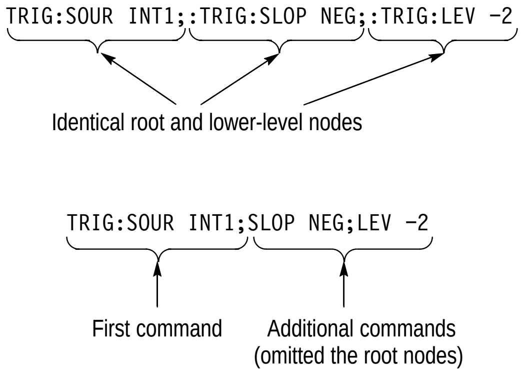
First command
Command
Command
Command
First query
Second query
The response from this chained message might be

```
1.0E-6;1536
```

Response from first query
Response from second query

Figure 3–30: Example of chaining commands and queries

If a command or query has the same root and lower-level nodes as the previous command or query, you can omit these nodes. In Figure 3–31, the second command has the same root node (TRIG) as the first command, so these nodes can be omitted.



```
TRIG:SOUR INT1;:TRIG:SLOP NEG;:TRIG:LEV -2
```

Identical root and lower-level nodes

```
TRIG:SOUR INT1;SLOP NEG;LEV -2
```

First command
Additional commands (omitted the root nodes)

Figure 3–31: Example of omitting root and lower-level nodes in chained message

General Rules

Here are some general rules for using SCPI commands, queries, and parameters:

- You can use single (‘ ’) or double (“ ”) quotation marks for quoted strings, but you cannot use both types of quotation marks to delimit the same string.

  correct: “This string uses quotation marks correctly.”

  correct: ‘This string also uses quotation marks correctly.’

  incorrect: “This string does not use quotation marks correctly.’

- You can use upper case, lower case, or a mixture of both cases for all commands, queries, and parameters.

  ```
  OUTPUT1:TTLTRG:POLARITY INVERTED
  ```

  is the same as

  ```
  output1:ttltrg:polarity inverted
  ```

  and

  ```
  OUTPUT1:ttltrg:polarity INVERTED
  ```

- No embedded spaces are allowed between or within nodes.

  correct: `OUTPUT1:TTLTRG:POLARITY INVERTED`

  incorrect: `OUTPUT1: TTLTRG: POLARITY INVERTED`

IEEE 488.2 Common Commands

Description

ANSI/IEEE Standard 488.2 defines the codes, formats, protocols, and usage of common commands and queries used on the interface between the controller and the instruments. The waveform analyzer complies with this standard.

Command and Query Structure

The syntax for an IEEE 488.2 common command is an asterisk (*) followed by a three-character mnemonic and, optionally, a space and parameter value. The syntax for an IEEE 488.2 common query is an asterisk (*) followed by a three-character mnemonic and a question mark. All of the common commands and queries are listed in the last part of the *Syntax and Commands* section. The following are examples of common commands:

- `*ESE 16`
- `*CLS`

The following are examples of common queries:

- `*ESR?`
- `*IDN?`

Backus-Naur Form Definition

This manual may describe commands and queries using the Backus-Naur Form (BNF) notation. Table 3–31 defines the standard BNF symbols:

Table 3–31: BNF symbols and meanings

Symbol	Meaning
< >	Defined element
::=	Is defined as
|	Exclusive OR
{ }	Group; one element is required
[ ]	Optional; can be omitted
. . .	Previous element(s) may be repeated
( )	Comment

To see the TVS600A expression syntax in BNF notation, see *Expression Syntax* on page 3–34.

Message Terminators

This manual uses `<EOM>` (End of message) to represent a message terminator.

Symbol	Meaning
<EOM>	Message terminator

The end-of-message terminator may be the END message (EOI asserted concurrently with the last data byte), the ASCII code for line feed (LF) sent as the last data byte, or both. The waveform analyzer always terminates messages with LF and EOI. It allows white space before the terminator.

Constructed Mnemonics

Some header mnemonics specify one of a range of mnemonics. For example, a channel mnemonic can be either `INP1`, `INP2`, `INP3`, or `INP4`. You use these mnemonics in the command just as you do any other mnemonic. For example, there is a `INP1:FILT` command, and there is also an `INP2:FILT` command. In the command descriptions, this list of choices is abbreviated as `INP<n>`.

Block Arguments

Several waveform-analyzer commands use a block argument form:

Symbol	Meaning
`<NZDig>`	A non-zero digit character, in the range 1–9
`<Dig>`	A digit character, in the range 0–9
`<DChar>`	A character with the hex equivalent of 00 through FF hexadecimal (0 through 255 decimal)

`<Block>`	A block of data bytes, defined as:
	`<Block> ::=`
	`{ #<NZDig><Dig>[<Dig>...][<DChar>...]`
	`| #0[<DChar>...]<terminator> }`

`<NZDig>` specifies the number of `<Dig>` elements that follow. Taken together, the `<Dig>` elements form a decimal integer that specifies how many `<DChar>` elements follow. The #0 format is for blocks of indefinite length and an END message termination is required.

Hardware Interfaces

The waveform analyzer provides VXIbus and RS-232C communication ports for instrument control. Additionally, you can use a GPIB controller and GPIB Slot 0 card to control VXIbus instruments.

This section describes the VXIbus and the RS-232C control interfaces.

VXIbus Interface

The waveform analyzer complies with the VXIbus System Specification in the following ways:

- The waveform analyzer supports VXIbus System Specification revision 1.4.
- The waveform analyzer is a Message-Based Servant, which supports VXIbus configuration and communication registers.
- The waveform analyzer supports Word Serial Protocol and Fast Data Channel data transfers. (For information on the Fast Data Channel, refer to the *Fast Data Channel* discussion on page 3–126.) The waveform analyzer responds to the SCPI and IEEE 488.2 Common Commands listed in the *Command Groups* on page 3–65.
- The waveform analyzer is a programmable interrupter for levels 1–7, capable of asserting interrupts and performing interrupt acknowledge sequences.

TTL and ECL Trigger Buses

The waveform analyzer uses the VXIbus TTLTRG and ECLTRG buses to export and import signals to and from other modules in the system. The waveform analyzer supports the SYNC trigger protocol; exported signals are broadcast on a TTLTRG or ECLTRG line and do not require acknowledgement from an acceptor module.

The available waveform analyzer sources for TTLTRG and ECLTRG lines are are as follows:

- `ARM`. A valid ARM event occurs which enables the TRIG:A circuit.
- `ATR`. A valid trigger event from the TRIGger A subsystem.
- `BTR`. A valid trigger event from the TRIGger B subsystem.
- `OPC`. The signal indicating the active command is complete. It is derived from the Operation Complete bit in the Standard Event Status Register.

- CALC. A valid trigger event occurring when the TRG control/notification function evaluates to TRUE in a CALC expression. (See *Trigger* on page 3–63.)

Table 3–32 shows the TTLTRG and ECLTRG lines and their default assignments.

Table 3–32: Trigger output lines and their default assignments

TTLTRg line number	Default assignments
TTLTRG0	ARM signal
TTLTRG1	TRIGger:A event
TTLTRG2	TRIGger:B event
TTLTRG3	OPC
TTLTRG4	ARM signal
TTLTRG5	TRIGger:A event
TTLTRG6	TRIGger:B event
TTLTRG7	OPC
ECLTRG0	TRIGger:B event
ECLTRG1	TRIGger:A event

When importing triggers, the waveform analyzer can enable any one of the TTLTRG or ECLTRG lines to be the trigger input. You can select rising- or falling-edge polarity.

VXIbus Pin Out

The VXIbus connectors are shown in Figure 3–32. Tables 3–33 through 3–35 list the pin assignments for each connector. Connectors are identified (left or right) when viewing the waveform analyzer from the front panel. Refer to Figure 3–32 for connector locations. For detailed information regarding the signals on the pins, refer to the VMEbus and VXIbus standards referenced above.

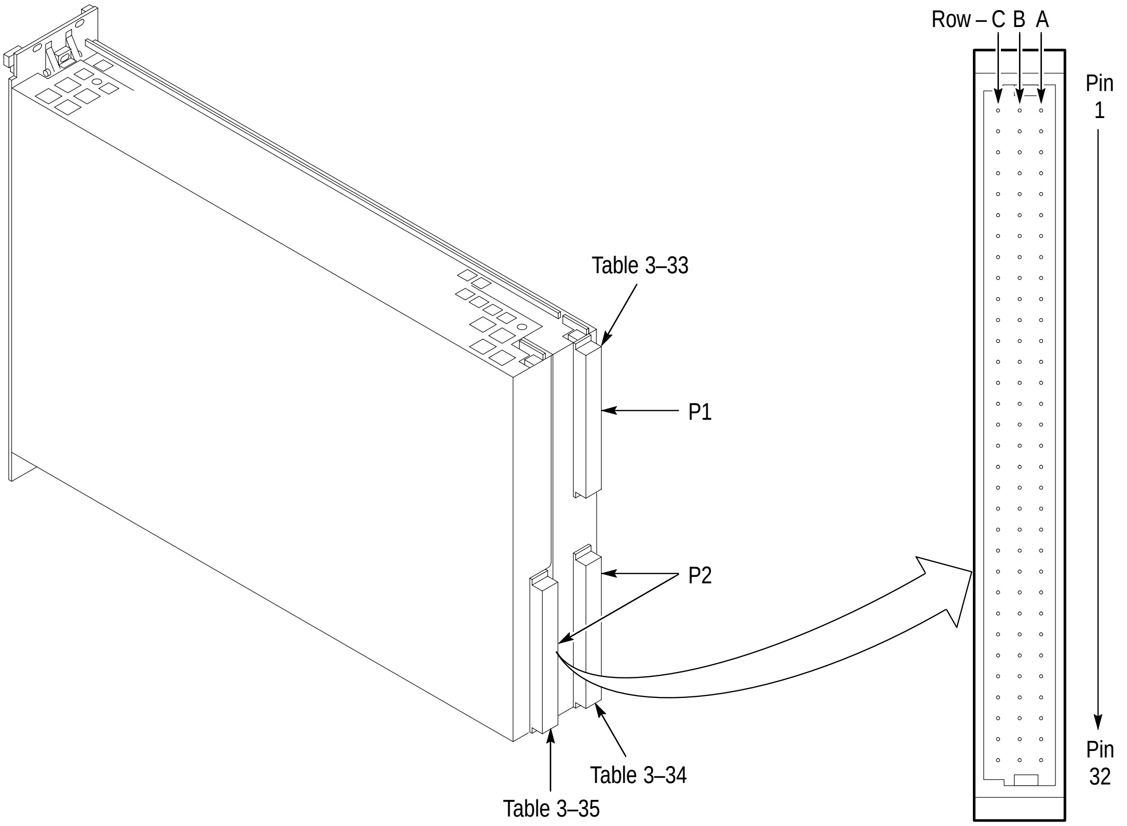


Figure 3–32: VXIbus connectors P1 and P2

***NOTE**. In the following tables, "NC" means "No Connection."*

Table 3–33: Left slot P1 pin-out

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	D00	NC	D08
2	D01	NC	D09
3	D02	ACFAIL	D10
4	D03	BG0IN	D11
5	D04	BG0OUT	D12
6	D05	BG1IN	D13
7	D06	BG1OUT	D14
8	D07	BG2IN	D15
9	GND	BG2OUT	GND
10	NC	BG3IN	SYSFAIL~
11	GND	BG3OUT	BERR~
12	DS1	NC	SYSRESET~
13	DS0	NC	NC
14	WRITE	NC	AM5
15	GND	NC	A23
16	DTACK~	AM0	A22
17	GND	AM1	A21
18	AS	NC	A20
19	GND	AM3	A19
20	IACK	GND	A18
21	IACKIN	NC	A17
22	IACKOUT~	NC	A16
23	AM4	GND	A15
24	A07	IRQ7	A14
25	A06	IRQ6	A13
26	A05	IRQ5~	A12
27	A04	IRQ4~	A11
28	A03	IRQ3~	A10
29	A02	IRQ2~	A09
30	A01	IRQ1~	A08
31	–12 V	+5 VSTDBY	+12 V
32	+5 V	+5 V	+5 V

Table 3–34: Left slot P2 pin-out

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	NC	+5 V	NC
2	–2 V	GND	NC
3	NC	NC	GND
4	GND	NC	–5.2 V
5	NC	NC	NC
6	NC	NC	NC
7	–5.2 V	NC	GND
8	NC	NC	NC
9	NC	NC	NC
10	GND	NC	GND
11	NC	NC	NC
12	NC	GND	NC
13	–5.2 V	+5 V	–2 V
14	NC	NC	NC
15	NC	NC	NC
16	GND	NC	GND
17	NC	NC	NC
18	NC	NC	NC
19	–5.2 V	NC	–5.2 V
20	NC	NC	NC
21	NC	NC	NC
22	GND	GND	GND
23	NC	NC	NC
24	NC	NC	NC
25	+5 V	NC	GND
26	NC	NC	NC
27	NC	NC	NC
28	GND	NC	GND
29	NC	NC	NC
30	VMOD1D	NC	GND
31	GND	GND	NC
32	NC	+5 V	NC

Table 3–35: Right slot P2 pin-out

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	ECLTRG0	+5 V	CLK10+
2	NC	GND	CLK10–
3	ECLTRG1	NC	GND
4	GND	NC	NC
5	NC	NC	NC
6	NC	NC	NC
7	NC	NC	GND
8	NC	NC	NC
9	NC	NC	NC
10	GND	NC	GND
11	NC	NC	NC
12	NC	GND	NC
13	NC	+5 V	NC
14	NC	NC	NC
15	NC	NC	NC
16	GND	NC	GND
17	NC	NC	NC
18	NC	NC	NC
19	NC	NC	NC
20	NC	NC	NC
21	NC	NC	NC
22	GND	GND	GND
23	TTLTRG0~	NC	TTLTRG1~
24	TTLTRG2~	NC	TTLTRG3~
25	+5 V	NC	GND
26	TTLTRG4~	NC	TTLTRG5~
27	TTLTRG6~	NC	TTLTRG7~
28	GND	NC	GND
29	NC	NC	NC
30	NC	NC	GND
31	GND	GND	NC
32	NC	+5 V	NC

RS-232C Port

The SERIAL INTERFACE located on the front panel is an RS-232C port. You can use the RS-232C port to control the waveform analyzer. All commands and queries are accepted over this serial interface. The command interpreter responds to the interface that issues a query, whether it is the VXIbus or the SERIAL INTERFACE. The RS-232C interface uses a standard 9-pin D type connector. Figure 3–33 shows the pin numbers for the connector and the signals assigned to each pin.

Serial Interface Commands

The serial interface commands are in the SYSTem:COMMunicate:SERial subsystem.

Three sets of settings, available with the SYSTem:COMMunicate:SERial:PRESet command, support standard uses such as a display terminal or computer connection.

- ALL sets baud to 9600, RTS control ON, and echo, parity and pace (XON/XOFF) handshaking off.
- RAW sets echo and pace (XON/XOFF) handshaking off. This mode is appropriate for a computer interface.
- TERMinal sets echo on and automatically sends all status messages to the serial interface. This mode is appropriate for a basic display terminal.

For more information, refer to the serial-interface command definitions in the group *System Commands* starting on page 3–76.

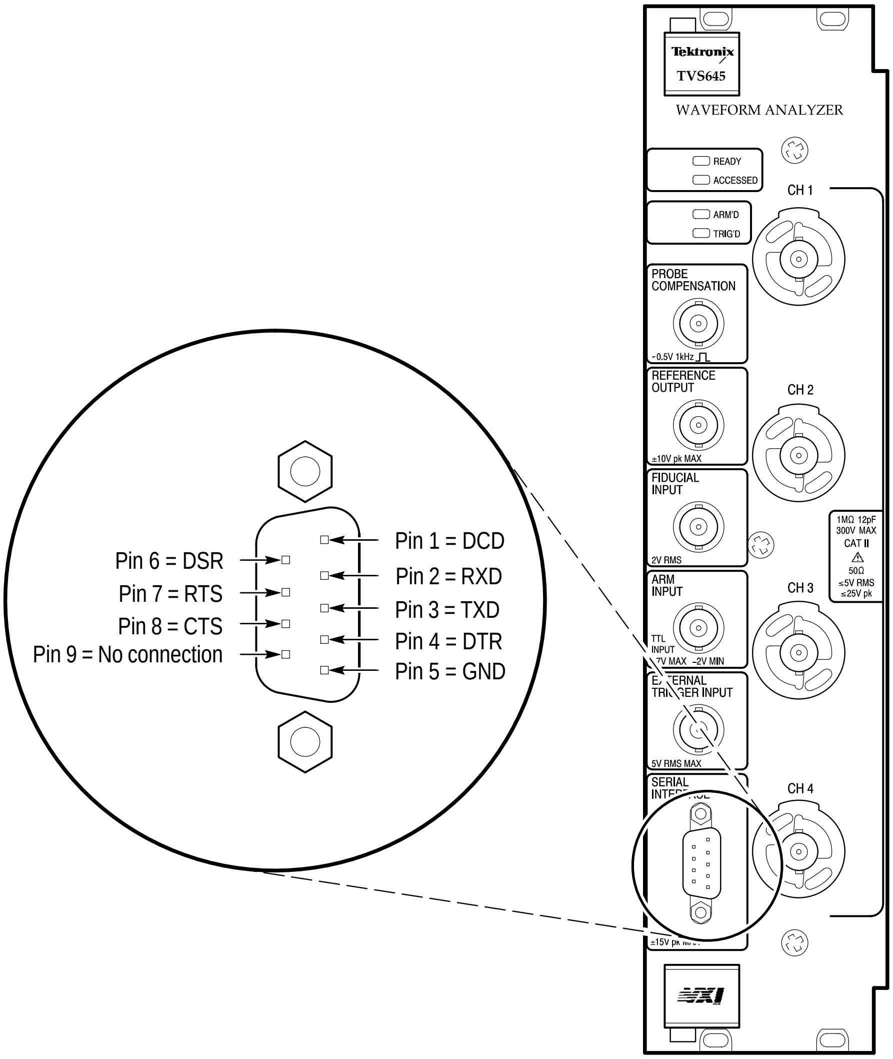


Figure 3–33: Pin assignments for the SERIAL INTERFACE (RS-232) connector

Input Signal Conditioning

To acquire and measure an input signal, you must connect the signal to an input channel and set up the waveform analyzer to acquire the waveform data you want. This section describes the following tasks:

- How to couple waveforms to the waveform-analyzer channels
- How to enable channels and set their individual vertical (coupling and bandwidth) parameters
- How to size and offset a vertical window for the input signal for each channel in use
- How to size and offset a horizontal window to apply to the input signals in all active channels in use

Set up of Probe, Input Coupling, and Input Bandwidth

The waveform analyzer comes equipped with the following features for coupling your input signals:

- TVS641A or TVS645A models provide four input channels, Ch 1 through Ch 4; TVS621A and TVS625A provide two channels, Ch 1 and Ch 2.
- Each channel supports TEKPROBE level 1 and level 2 probes. For level 2 probes, SCPI commands support probe identification, setting of offset and attenuation factor, and probe calibration.
- Each channel can be set for 1-MΩ or 50-Ω input impedance and for DC, AC, and Ground coupling. The Ground setting connects the internal amplifier to ground but presents an open circuit impedance (≥ 500 kΩ) to the input signal.
- Each channel has automatic input protection, switching from 50-Ω to 1 M-Ω coupling automatically when an input signal exceeds the input range of the channel.

Why Use?

You set up the input parameters of the channel to control which components of the input signal couple to the waveform analyzer and to choose the input impedance presented to the probe or cable that connects to the input signal.

To Use

Use the following procedure to set up the probe and each input channel in which you intend to acquire:

1. Set input protection on for all channels: first send `SYSTem:PROTect 1`; then send `INPut:PROTection:STATe 1`.

WARNING. *Setting input protection off exposes the waveform analyzer to serious damage if input signals exceed the maximum ratings for the input channels. Leave input protection on unless data acquisition is compromised and then only if you know that the input signals applied will not exceed input ratings.*

2. Connect a probe or cable to an input channel. (This procedure assumes a probe connection.)

3. Turn on the channel or channels you intend to use:

   - To turn one channel on, send `FUNCtion:ON CHAN<n>`, where `CHAN<n>` is one of `CHAN1` through `CHAN4`.
   - To turn more than one channel on, specify the channel arguments as comma-separated arguments; for example, to turn on channels 1, 3, and 4, send `FUNCtion:ON CHAN1,CHAN3,CHAN4`.
   - To turn on all channels, send `FUNCtion:ON:ALL`
   - To turn off channel(s), substitute `:OFF` for `:ON` in the commands just listed.

   To turn on more than one channel at a time, `FUNCtion:CONCurrent ON` must be sent. See your Command Reference for more information on function commands.

4. You should compensate the passive probe or calibrate the active probe you connect (see *Probe Calibration* on page 3–159). If you have installed a Tektronix level 2 active voltage probe, you can check its calibration status: send the query `CALibration:PROBe[1]?`, where the return of:

   - 0 (zero) indicates a calibrated probe.
   - –1 indicates a probe calibration failed.
   - –2 indicates probe calibration failed due to the probe not being connected to the calibration source.
   - –3 indicates a calibration failed because an unsupported probe or no probe was connected.

5. Set up the input coupling: send `INPut<n>:COUPling <arg>`, where `<n>` is the channel used and `<arg>` is one of `AC`, `DC`, or `GROund`.

6. If desired, set an upper limit for bandwidth to be passed: send `INPut<n>:FILTer[:LPASs]:FREQuency <arg>`, where `<n>` is the channel used and `<arg>` is one of `20.0E+6` or `250.0E+6`.

   Then send `INPut<n>:FILTer[:LPASs][:STATe] 1` to turn the bandwidth filter on. (Sending 0 turns the bandwidth filter off.)

7. Set up the common horizontal parameters (see procedure on page 3–113).

8. Set up triggering requirements (see procedures in *Trigger Types*, starting on page 3–193) and acquisition modes (see procedure *To Use* under *Acquisition Modes* on page 3–12).

9. Initiate the acquisition: send `INITiate`.

Commands

The commands and functions to set up the input-signal coupling and connection follow:

Used to:	SCPI Commands[1]	Driver Functions[2]	SFP?[3]

turn on or off one or more channels	`FUNCtion:ON CHAN<n>`
	`FUNCtion:ON:ALL`
		`tktvs600_setVert`	Yes

set AC, DC or Ground coupling for Ch <n>	`INPut<n>:COUPling`	`tktvs600_setVert`	Yes

turn on the low pass filter	`INPut<n>:FILTer[:LPASs]`
	`[:STATe]`
			Yes

set the low pass filter upper limit frequency	`INPut<n>:FILTer[:LPASs]`
	`:FREQuency`
			Yes

set 50-Ω or 1 M-Ω input Z for Ch <n>	`INPut<n>:IMPedance`		Yes
set automatic overvoltage protection for all channels	`INPut:PROTection:STATe`	`tktvs600_setInputProtection`	Always on

return information on probe or probe cal status	`INPut:PROBe<n>:ATTenuation?`
	`:OFFSet?`
	`:IDENtification?`
		`tktvs600_getPROBe`	Yes

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Functions Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application.**

Usage Notes

Some usage notes follow:

- The SCPI model (shown on page 2–9) applies to the input commands. Each channel has its own input block (see Figure 3–34) that is set up with

INPut<n> commands, where the block number <n> corresponds to the channel set up. (INPut1:COUPling AC sets channel 1 coupling, and so on.)

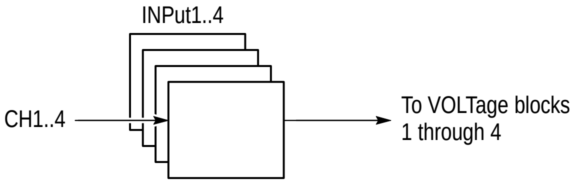


Figure 3–34: Channel numbers carry through to Input blocks

- The probe-attenuation factor reduces the input voltage applied to the input channel.

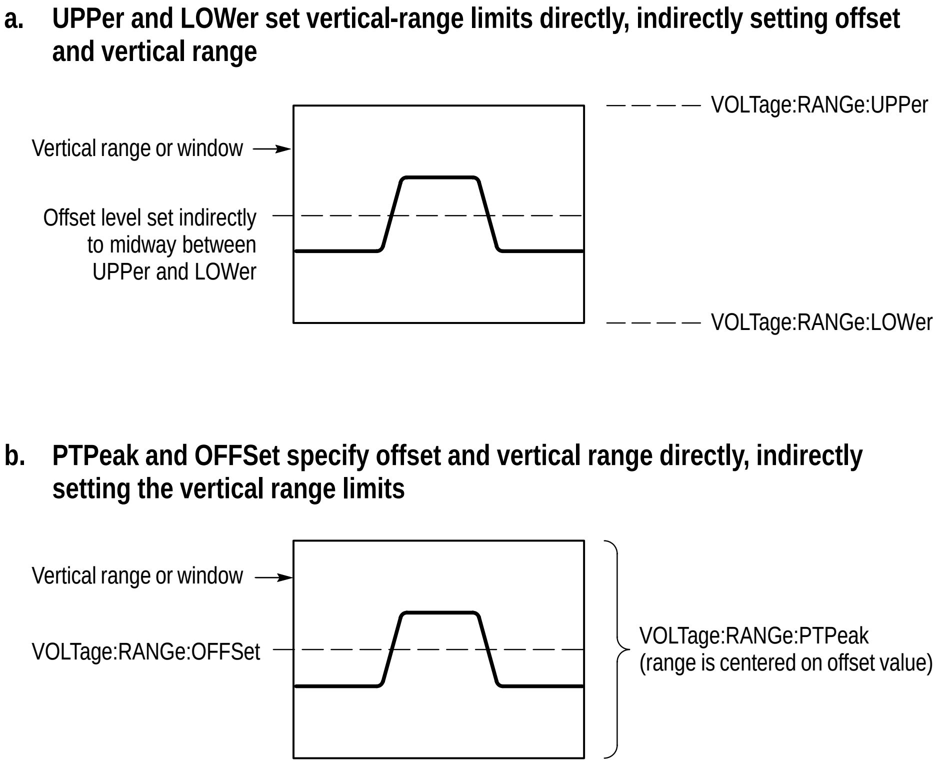
Vertical Scaling and Offset of Waveforms

The waveform-analyzer VOLTage commands let you set the vertical range and offset of each channel independently of the other channels.

Vertical Window

To set the vertical scale and position for an input channel, you define the vertical window for that channel, where:

- The vertical range you set determines the vertical size of the window, allowing you to scale it to contain all of a waveform amplitude or only part. Figure 3–35 shows the commands for setting the vertical range. (Two methods are shown.)
- The vertical offset you set is always locked to (is) the voltage level at middle of the vertical window. (As Figure 3–35 shows, the vertical range (window) is centered around the offset value.)
- The vertical offset you set determines the vertical position of the waveform within the vertical window (waveform smaller than window) or the vertical position of the window on the waveform (waveform larger than window). In the later case, vertical position determines what portion of the waveform amplitude the window contains. Figure 3–36 shows how offset positions the vertical window.



a. UPPer and LOWer set vertical-range limits directly, indirectly setting offset and vertical range

b. PTPeak and OFFSet specify offset and vertical range directly, indirectly setting the vertical range limits

Figure 3–35: Setting vertical range and offset of input channels

Vertical Range and Offset

The waveform analyzer provides two methods of setting the vertical range and offset: PTPeak and OFFSet, and UPPer and LOWer. Figure 3–35 shows how these two methods are used; Figure 3–36 shows how PTPeak and OFFSet can move the vertical window. Note the following characteristics:

- PTPeak and OFFSet specify the offset directly and the upper and lower window limits indirectly, while UPPer and LOWer specify the limits directly and the range and offset indirectly.
- It is the input vertical range, or window, of the digitizer that is moved by vertical OFFset and not the DC level of the input signal. Applying a negative OFFset moves the vertical range down. Likewise, applying a positive OFFset moves the vertical range up. See Figure 3–36.
- You can use the UPPer and LOWer parameters to set the range to absolute vertical values if you prefer.

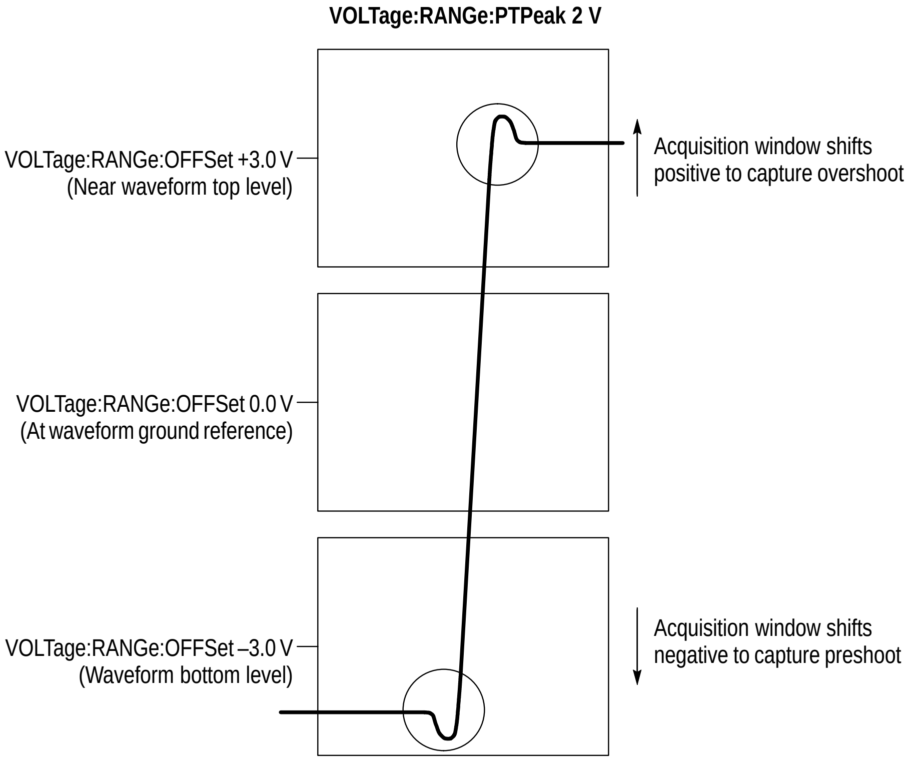

VOLTage:RANGe:PTPeak 2 V
VOLTage:RANGe:OFFSet +3.0 V
(Near waveform top level)
Acquisition window shifts positive to capture overshoot
VOLTage:RANGe:OFFSet 0.0 V
(At waveform ground reference)
VOLTage:RANGe:OFFSet –3.0 V
(Waveform bottom level)
Acquisition window shifts negative to capture preshoot

Figure 3–36: Varying offset positions vertical window on waveform amplitude

Why Use? You set input parameters to control the size of the vertical window in order to capture as much as you want, part or all, of the vertical amplitude of the input signal. For those signals that you only want to capture part of the vertical amplitude, you set vertical offset to scroll the vertical window vertically on the signal amplitude to capture the portion you want.

To Use Use the following procedure to set the size of the vertical window:

1. Set up probe and input coupling as described in the procedure on page 3–102.

2. Set the vertical window range and offset using either step 3 or step 4.

3. Set the range limits directly and the offset indirectly: send the following commands:

   ■ `[SENSe:]VOLTage<n>[:DC]:RANGe[:UPPer] <upper>`, where `<n>` is the channel used and `<upper>` is the most positive level of the vertical window.

- `[SENSe:]VOLTage<n>[:DC]:RANGe:LOWer <lower>`, where `<n>` is the channel used and <`lower`> the the most negative level of the vertical window.

If you use this step, offset is set indirectly to the level half way between the upper and lower levels you set.

4. Set the offset directly and the vertical window range limits indirectly: send the following commands:

- `[SENSe:]VOLTage<n>[:DC]:RANGe:PTPeak <range>`, where `<n>` is the channel used and <`range`> is a value between 10 mV and 100 V, which must be set in increments as follows:

10 mV– 20 mV	(100 µV steps)	1 V – 2 V	(10 mV steps)
20 mV – 50 mV	(200 µV steps)	2 V – 5 V	(20 mV steps)
50 mV – 100 mV	(500 µV steps)	5 V – 10 V	(50 mV steps)
100 mV – 200 mV	(1 mV steps)	10 V – 20 V	(100 mV steps)
200 mV – 500 mV	(2 mV steps)	20 V – 50 V	(200 mV steps)
500 mV – 1 V	(5 mV steps)	50 V – 100 V	(500 mV steps)

For example, <range> can be set

- `[SENSe:]VOLTage<n>[:DC]:RANGe:OFFSet <offset>`, where `<n>` is the channel used and `<offset>` is one of the values corresponding to the PTPeak range that you set above:

<offset>	**PTPeak <range>**
±1.0 (in 1 mV steps)	10 mV – 1 V
±10.0 (in 10 mV steps)	1.01 V – 10 V
±100.0 (in 100 mV steps)	10.1 V – 100 V
MINimum MAXimum	Any (sets the minimum or maximum offset allowed by PTPeak range)

If you use this step, range limits are set indirectly to levels that result in half of the vertical window above and half below the offset level you set.

5. Set up triggering requirements (see procedures in *Trigger Types,* starting on page 3–193) and acquisition modes (see procedure under *Acquisition Modes* on page 3–12).

6. Set up the common horizontal parameters (see procedure on page 3–113).

7. Init the acquisition: send `INITiate`.

Commands

The commands and functions to set up vertical window size and offset follow:

Used to:	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
set upper limit of vert. window for ch <n>	`VOLTage<n>[:DC]:RANGe[:UPPer]`	`tktvs600_setVert`	No
set lower limit of vert. window for ch <n>	`VOLTage<n>[:DC]:RANGe:LOWer`		No
set pk-to-pk size of vert. window ch <n>	`VOLTage<n>[:DC]:RANGe:PTPeak`		Yes
set DC offset of vert. window for ch <n>	`VOLTage<n>[:DC]:RANGe:OFFSet`		Yes

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Functions Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application**

Usage Notes

Some usage notes follow:

- The SCPI model (shown on page 2–9) applies to the VOLTage commands. Each channel has its own input block and voltage block (see Figure 3–37), the latter of which is set up with VOLTage<n> commands, where the block number <N> corresponds to the channel set up. (For example, `VOLTage1:RANGe:PTPeak 5` sets channel 1 vertical window to 5 volts peak-to-peak.)

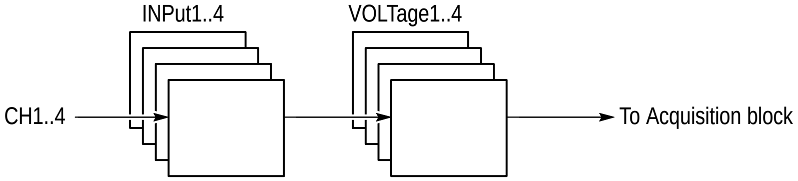


Figure 3–37: Channel numbers carry through to Input and Voltage blocks

- There are also four CALC blocks numbered 1–4 used by the CALCulation system. Unlike the INPut and VOLTage blocks that always match the channel number they apply to, any number CALC block can be associated (fed) any channel. See *Calculation Overview* on page 3–21.

- Valid waveform data are in the range +32256, for waveform values that equal `VERTical:RANGe UPPer`, to –32256, for waveform values that equal `VERTical:RANGe LOWer`. The waveform preamble contains the scaling information to convert these values to voltages; see *I/O of Waveforms* on page 3–117 for more information.

- Parts of a waveform that exceed the vertical-window range are clipped. Parts that exceed the most positive range limit (UPPer) are acquired as over-range points and are assigned the value +32767. Parts that exceed the most negative range limit (LOWer) are acquired as under-range points and are assigned the value –32767.

Horizontal Scaling and Offset of Waveforms

The waveform analyzer lets you set several parameters that determine the segment of an incoming signal that becomes the waveform record acquired. (For background, please read *Waveform Record* on page 3–5.) These common parameters specify a common horizontal window that is applied to all channels in parallel. (See *A Shared Window* on page 3–112.)

Horizontal Window

To set the horizontal scale (size) for the horizontal window and its position on a incoming waveform, you must define the horizontal window:

- The sweep interval and record length you set determines the horizontal size of the window relative to any waveform, allowing you to scale it to contain a waveform edge, a cycle, or several cycles.
- The sweep offset you set determines how the horizontal window is positioned horizontally relative to the waveform. Figure 3–39 shows how the horizontal range (window) is located.

Sweep Interval, Sweep Time, and Record Length

Three parameters describe the size or duration of the horizontal window relative to any waveform; you can use any two of these parameters to define the window:

Window duration = Sample Interval * Record Length

Expressed in terms of the SWEep commands that control these parameters:

```
SWEep:TIME = SWEep:TINTerval*SWEep:POINts
```

Discussions of each of these three parameters follow.

Sample Interval. This parameter specifies precise time between sample points taken during acquisition. Figure 3–38 shows the sample interval between two waveform samples. The sample interval is the reciprocal (1/x) of the sample rate.

Set a short sample interval to acquire more detail for a particular waveform feature. Set a longer sample interval if you wish to capture more cycles of a waveform. Use the command `SWEep:TINTerval` to set the sample interval.

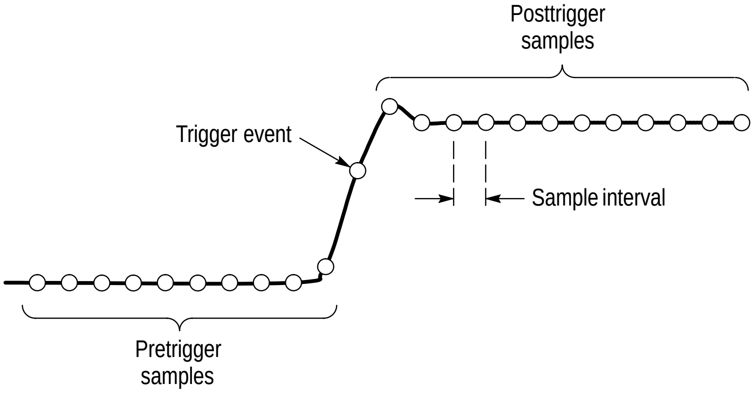


Figure 3–38: Waveform record showing the sample interval, trigger event, and pretrigger samples

Record Length. This parameter specifies the record length by setting the number of samples required to fill a waveform record. When multiplied by the sample interval, it sets the size of the horizontal window.

All channels share the record length setting. You can set the record length from 256 to 15,000 points for real-time acquisition. A record length of 30,000 points is available in the extended real-time mode. Use the command SWeep:POINts to set or query the number of samples in the waveform record.

Waveform Duration. This parameter specifies the record length in time rather than points. More specifically, it is the product of the sweep interval and record length in points. You can set the time-duration of the waveform record with `SWEep:TIME`. (The sweep interval will change to the closest setting that allows the SWeep:TIME you set.)

Waveform Record Offset and Trigger Point

In addition to sizing the horizontal window, you also must position the window on the input signal. Two parameters combine to determine where in the input signal the horizontal window is placed and the waveform record is taken: the trigger point and the record offset.

Trigger Point. This point marks time zero in a waveform record. All waveform samples are located in time with respect to the trigger point.

The trigger point is based on triggering requirements you set up using trigger commands. You can define the trigger point to occur on a simple level and slope crossing or upon more complex requirements, such as upon detection of a defined type of pulse. Figure 3–38 shows a simple level- and slope-based trigger point. (The trigger circuit, with its event count, time delay, and holdoff capabilities, is controlled independently from the record-positioning functions. For information on the trigger system, refer to the discussion *Triggering* on page 3–181.)

Record Offset. You can offset or position the horizontal window relative to the trigger point to control where the waveform record is taken. The record can be shifted so the trigger point occurs anywhere within the waveform record (see Figure 3–39). All acquisitions in all channels share the record-offset settings.

Figure 3–39 shows two commands for offsetting the horizontal window relative to the trigger point, `SWEep:OFFSet` and `SWEep:OREFerence`. Offset is effected using either of the two following processes:

- You set the `SWEep:OFFSet` to zero so that the `SWEep:OREFerence` and the trigger point always occur at the same sample in the waveform. Then the `SWEep:OREFerence` is adjusted to the desired position (sample) in the waveform record.

  See Figure 3–39, left, which shows a fixed offset (0) with OReference varied from 0 to .5 to 1 to shift the horizontal window.

- You set the `SWEep:OREFerence` to zero so that its always anchored to the first sample in the waveform record. Then the `SWEep:OFFSet` is adjusted so that `SWEep:OREFerence`, and the start of the waveform record, positions the waveform record relative to the trigger point. See Figure 3–39, right.

  See Figure 3–39, right, which shows a fixed OReference (0, the first sample) with offset varied from 0 to –1/2 to –1 record length to shift the horizontal window.

You can set `SWEep:OFFSet` as a number of record points (:POINts) or as a period of time (:TIME). For example, to acquire 50% of a 1024 point waveform record as pretrigger sample points, set `:OREFence:LOCation` to zero and set `:OFFSet:POINts` to –512. Now the trigger is horizontally centered in the record; half the samples are in the pretrigger region and half in the posttrigger region.

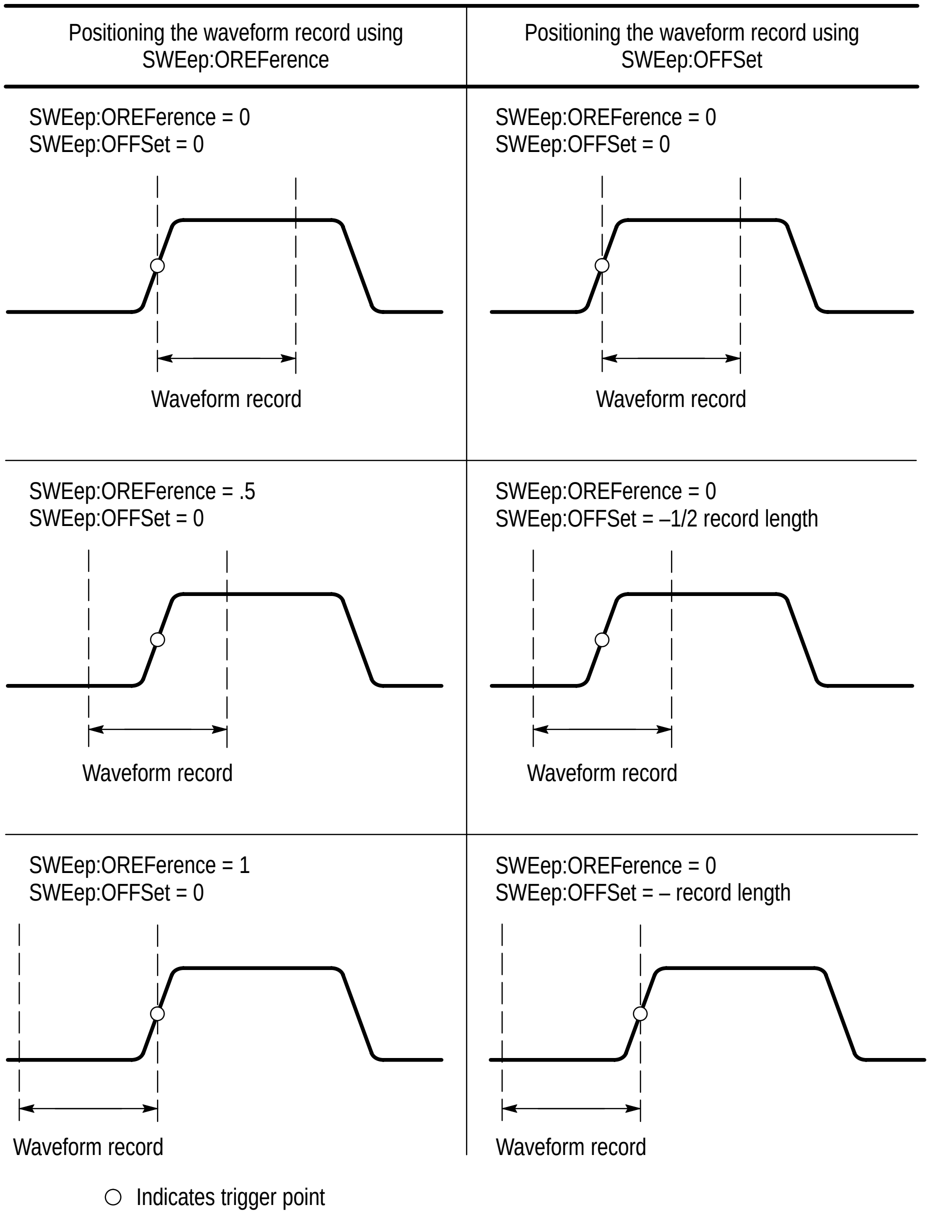


Figure 3–39: Positioning the waveform record relative to the trigger point

A Shared Window

The waveform analyzer applies the same horizontal window to all channels from which it acquires data. Unlike the vertical window that you size and offset independently for each channel, the same record length and sweep offset (from the trigger point) apply to all channels simultaneously. In other words, one trigger, from a single trigger source, will locate a common horizontal window on all active channels, which you can shift in parallel using the sweep commands.

The horizontal window determines the waveform records extracted from all signals present at all active channels. You can think of the horizontal window as cutting across any input signals present in the input channels to extract the same slice of time into waveform records. See Figure 3–40.

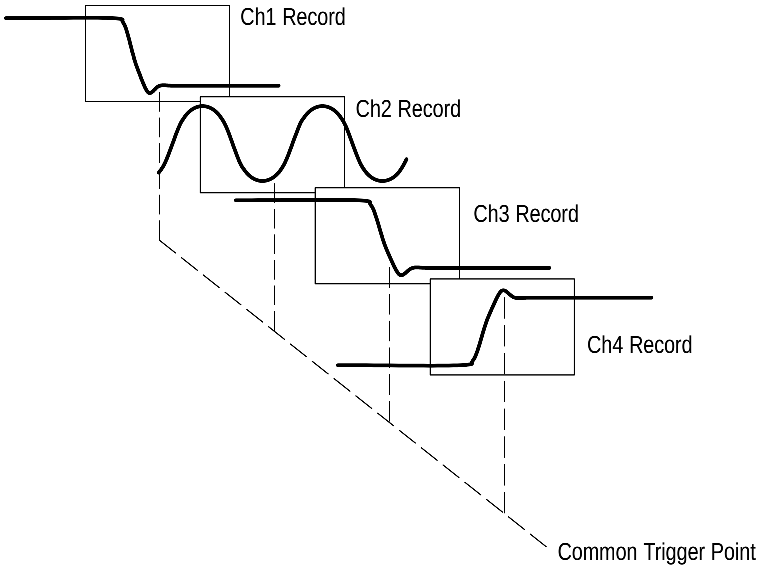


Figure 3–40: Common trigger, record length, and acquisition rate for all channels

Why Use?

You set common parameters to control the size of the horizontal window in order to capture as much as you want, part or all, of the input signal(s). You do so by horizontally scrolling the window relative to a common trigger to capture the waveform portion you want.

To Use

Use the following procedure to set the duration and position of the horizontal window:

1. Set up probe and input coupling as described in the procedure on page 3–102.

2. Set the vertical window range and offset as described in the procedure on page 3–106.

3. Set up triggering requirements (see procedures in *Trigger Types*, starting on page 3–193) and acquisition modes (see procedure *To Use* under *Acquisition Modes* on page 3–12).

4. Setting the horizontal window duration requires that you set two of the three parameters defined by the relationship:

   ```
   SWEep:TIME = SWEep:TINTerval*SWEep:POINts
   ```

Set the duration by performing any two of the following three steps (steps 5 – 7):

5. Set the sweep interval: send `[SENSe:]SWEep:TINTerval<arg>`, where `<arg>` is one of the values that follow:

    200E–12 (TVS625A, TVS645A only)
    400E–12 (TVS625A, TVS645A only)
    1E–9
    2E–9
    4E–9
    10E–9
    20E–9
    40E–9
    100E–9
    .
    .
    .
    200E–3

    MINimum
    MAXimum

    The default multiplier for `<arg>` is `S` for seconds. You can also use the multipliers `MS` for milliseconds, `US` for microseconds, `NS` for nanoseconds, and `PS` for picoseconds.

    If using `SWEep:TINTerval`, send it first, because it may modify `SWEep:TIME`. See step 4 for the relationship between interval and time.

6. Set the number of sweep points: send `[SENSe:]SWEep:POINts <arg>`, where `<arg>` is one of the values that follow:

    256
    512
    1024
    2048
    4096
    8192
    15000
    30000 (ERT mode only)
    MINimum
    MAXimum RT mode
    MAXimum ERT mode

    If using `SWEep:POINts` and `SWEep:TIME`, send `SWEep:POINts` first because `SWEep:POINts` may modify `SWEep:TIME`. Issuing these two commands sets the sample interval (`SWEep:TINTerval`) to the nearest legal value.

7. Set the sweep time: send `[SENSe:]SWEep:TIME <arg>`, where `<arg>` is `MINimum`, `MAXimum`, or a number within the range set by `MINimum` and `MAXimum`: `51.2E–9 ≤ n ≤6000` (seconds).

8. Positioning the horizontal window, relative to the trigger point, can be done in two ways:

    - Set horizontal offset constant at zero to lock the trigger point and horizontal-offset-reference point together. Then vary the horizontal-offset-reference point location and the trigger point in lock-step within the waveform record (see Figure 3–39 on page 3–112).

    - Set the horizontal-offset-reference point location to a single sample point (say the first point) in the waveform record. Then vary the horizontal offset to move the reference point and the waveform record relative to the trigger point.

Steps 9 – 10 illustrate positioning and explain the first method just described.

9. Set the horizontal offset: send one of two following commands:

`[SENSe:]SWEep:OFFSet:POINts <arg>`,where `<arg>` is either `MIN`, `MAX`, or number between `MIN` and `MAX`. `MIN` and `MAX` are determined as follows:

- `MINimum  (SWE:OREF:LOC X SWEep:POINts) – SWEep:POINts`
- `MAXimum  (SWE:OREF:LOC X SWEep:POINts)`

`[SENSe:]SWEep:OFFSet:TIME <arg>`,where `<arg>` is either `MIN`, `MAX`, or a number between min and max. `MIN` and `MAX` are determined as follows:

- `MINimum  (SWEep:OREFerence:LOC X SWEep:TIME) – (SWEep:TIME)`
- `MAXimum  (SWEep:OREFerence:LOC X SWEep:TIME)`

Setting offset to `MIN` (no offset) would lock trigger point and offset reference point together, allowing you set their common location in the waveform record in step 10.

10. Set the location of horizontal-offset-reference point as a portion of the waveform record: send `[SENSe:]SWEep:OREFerence:LOCation <arg>` where:

- `<arg>` is unitless and either `MIN` (0.0), `MAX` (1.0), or a number between MIN and MAX, or...
- `<arg>` is in PCT (percent) units and either `MIN` (1), `MAX` (100), or a number between MIN and MAX

Setting 0.5 (no units) or 50 PCT and would locate 1/2 the waveform record immediately in front of and 1/2 immediately after the trigger point, assuming you set horizontal offset to minimum in step 9.

11. Init the acquisition: send `INITiate`.

Commands

The commands and functions to set up horizontal window size and offset follow:

Used to:	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
set the record length in data points	`SWEep:POINts`	`tktvs600_setHoriz`	Yes
set the record length in time	`SWEep:TIME`		No
set the time between sample points (1/sample rate)	`SWEep:TINTerval`	`tktvs600_setVert`	Sets sample rate
set the number of points between trigger point and horizontal-offset-reference point	`SWEep:OFFSet:POINts`		No
set the time between trigger point and horizontal-offset-reference point	`SWEep:OFFSet:TIME`		No
set position in record of the horizontal-offset-reference point	`SWEep:OREFerence:LOCation`		Yes

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Functions Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application.**

Usage Notes

Some usage notes follow:

- The minimum record length is 256 data points; the maximum is 15,000 when sampling in real time (RT) mode and 30,000 when sampling in extended real time (ERT) mode. All active channels share the same record length setting.

- If `SWEep:POINts` is set to 30000 (extended-real time sampling only), changing `SWEep:TINTerval` such that the waveform analyzer enters real-time sampling mode causes `SWEep:POINts` to be set to 15000.

- The point at which the instrument transitions from real time (RT) sampling to extended real time (ERT) sampling mode is independent of waveform-analyzer configuration or settings.

  RT < 100 ns (per point) ≥ ERT

- The following equation defines the first point of the waveform record:

  $PT1_{time}$ = `SWEep:OFFSet`:TIME – (`SWEep:OREFerence`:LOCation * `SWEep:TIME`)

I/O of Waveforms

This section describes the transfer of waveforms and other data; the following transfers are covered:

- *Data Uploads* describes transfers from the waveform analyzer to the controller
- *Data Downloads* describes transfers from the controller to the waveform analyzer
- *Internal Transfers* describes transfers to references from other references, from channels, or from CALC blocks.
- *Fast Data Channel* describes FDC-transfers from the waveform analyzer to the controller

Following these discussions and procedures, *Data Interchange Format* on page 3–131 provides reference information on DIF block construction and data format issues. (All waveform-analyzer preamble uploads are encoded in DIF expressions, and other transfers of data can be encoded in DIF expressions.)

Appendix E: Supported Preambles contains an example of each type of preamble that the waveform analyzer can output and brief descriptions of the key words within those preambles.

Data Uploads

This section surveys the general process used to format data and upload it to a controller. (Detailed information on data format is found on page 3–138.) You can upload the following types of data from the waveform analyzer:

- Acquired waveforms stored in acquisition memory
- Reference waveforms stored in any of REF1 through REF10
- Calculation results, including measurements, stored in acquisition memory

All three transfer types usually require two transfers: one of the data block and one of the preamble block that characterizes the data. Data and preambles are returned to the interface, serial or VXI bus, over which the data or preamble query is sent.

TVS600 and TVS600A models always send preambles formatted as a DIF expression. Preamble DIF expressions have null data blocks: `(DATA(CURV(CTYP NONE)))`. Data, when queried, is returned as binary-block or ASCII numeric data (not in DIF expressions) as described under *Data Formats* on page 3–138.

TVS600A models (not TVS600 models) can combine data and preamble into a single DIF block. Just send the command FORMat:DINTerchange ON. Thereafter, a data query (one of those listed in step 3 in the procedure that follows) will return the data and the preamble to the requesting interface wrapped in a single DIF expression; you do not have to fetch the preamble separately.

Combined transfers remain in effect until you send FORMat:DINTerchange OFF. See the command FORMat:DINTerchange in your *TVS600A Command Reference* manual for more information.

Why Use?

Waveform, calculations, and measurement results may be uploaded as required by user-applications.

To Use

To transfer a waveform to a controller, you must format the data and preamble (or accept the default formats) and then query the waveform analyzer to fetch them. The following steps outline this process:

1. Specify the formats that you will use to transfer your data by sending the FORMat command appropriate for the source of the data you intend to transfer:

- To transfer data from channels, send:
  FORMat[:DATA] <type>[,<length>], where <type>[,<length>] is either ASCII,0, to format in ASCII, or INT,16, to format in 16-bit binary-integer format.

- To transfer data from references, send:
  FORMat:TRACe:REF <type>[,<length>], where <type>[,<length>] is either ASCII,0, to format in ASCII, or REAL,32, to format in 32-bit floating-point binary format.

- To transfer the auto-advance timestamp (from channels), send:
  FORMat:TRACe:AATS <type>[,<length>], where <type>[,<length>] is either ASCII,0, to format in ASCII, or REAL,32, to format in 32-bit floating-point binary format.

- To transfer data resulting from calculations, send:
  FORMat[:DATA]:CALCulate[n] <type>[,<length>], where <n> is the number of the CALC block sourcing the data and <type>[,<length>] is either ASCII,0, to format in ASCII or REAL,32, to format in 32-bit floating-point binary format.

Note that you set a format for all channels collectively. References are also set collectively, but calculations are set independently, each by its CALC block number.

2. If returning auto-advance acquisitions, do the following substeps; if not, skip to step 3:

a. Set the starting index from which the count (set in step b) proceeds: send `AADVance:RECord:STARt <arg>`, where `<arg>` is one of the following values:

0	Last record
1 to MAX	Record number from first record to last acquired
–1 to –(MAX + 1)	Record number from last record to first acquired
MINimum	First record acquired
MAXimum	Last record acquired

b. Set the number of auto-advance records to return: send `AADVance:RECord:COUNt <arg>`, where `<arg>` is one of the following values:

0	Transfer all records based on current acquisition settings
1 to MAX	Transfer specified number of records
MINimum	Transfer one record specified in step a.
MAXimum	Transfer all records based on initial acquisition settings

3. Query the data and preamble based on the data source: send a data query and a preamble query selected from Table 3–36, where <n> corresponds to the data source used (for example, CHAN1, REF10, or CALC3):

Table 3–36: Data and preamble queries

Data Source	Data Query Forms	Preamble Query Forms

Channel data, normal and auto-advance cycles	`TRACe:DATA? CHAN<n>`[1]
	`DATA? CHAN<n>`[1,2]
		`TRACe:PREamble? CHAN<n>`
		`DATA:PREamble? CHAN<n>`

Auto-advance time stamps	`TRACe:DATA? AATS`	`TRACe:PREamble? AATS`
Reference data	`TRACe:DATA? REF<n>`	`TRACe:PREamble? REF<n>`

CALC Data	`CALC<n>:DATA?`[3]
	`TRACe:DATA? CALC<n>`[3]
		`CALC<n>:DATA:PREamble?`
		`TRACe:PREamble? CALC<n>`

[1] **When auto-advance is on, returns specified count of records starting at index.**

[2] **When no data source (channel) is given, returns the records for all channels, starting with the record of the lowest-numbered channel and following in ascending order of channels. Returns auto-advance records if auto-advance is on.**

[3] **When auto-advance is on, calculation results depend on the calculation specified. See *Auto Advance Uploads* on page 3–120.**

4. Once uploaded, reconstruct your data. Data records can be reconstructed from information in the DIF preamble and data block. See *Waveform Reconstruction* on page 3–121.

Commands

The commands and functions to upload data from the waveform analyzer follow:

Used to:	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
return calculation data	`CALC:DATA?`	`tktvs600_fetchCalculation`	Yes
return preamble-characterizing calc data	`CALC:DATA:PREamble?`	`tktvs600_getPreamble`	No
toggle combined data and preamble transfers on and off	`FORMat:DINTerchange`	`tktvs600_setFormatOptions`	Automatic
return sense (CHAN) data	`DATA?`	`tktvs600_fetchWfmData`	Yes
return preamble that characterizes the sense data	`DATA:PREamble?`	`tktvs600_getPreamble`	Automatic
set format for all channel transfers	`FORMat[:DATA]`	`tktvs600_setFormat`	Automatic
set format for CALCulation <n>	`FORMat[:DATA]:CALC<n>`		Automatic
set format for all reference or AATs transfers	`FORMat:TRACe:AATS` `FORMat:TRACe:REF`		Automatic
return REF, CHAN, CALC, or AATS data	`TRACe:DATA?`	`tktvs600_fetchWfmData`	Yes, except AATS
return REF, CHAN, CALC, or AATS preambles	`TRACe:PREamble?`	`tktvs600_getPreamble`	Automatic

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Functions Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application. Automatic means that capability occurs in the course of performing an SFP operation. For example, transferring a waveform sets data format and sends preamble automatically.**

Auto-Advance Uploads

The procedure just listed returns both normal-cycle acquisitions and acquisition sequences acquired using the auto-advance cycle. For a description of the auto-advance cycle, see *Acquisition Advance Acquisition* on page 3–13; some details pertinent to retrieving auto-advance records follow:

- Queries returning auto-advance records wait for the acquisition of all records specified by the command `AADVance:COUNt` to complete.
- The number of auto-advance records returned is controlled by `AADVance:RECord:STARt` and `AADVance:RECord:COUNt`. See footnotes to Table 3–36 on page 3–119 and your *TVS600A Command Reference.*
- Queries returning auto-advance preambles return one preamble per channel, since the parameter is the same for all records for that channel in the auto-advance sequence.
- Queries returning one or more measurements specified using the `CALCulate:AAMList` return a measured value for each auto-advance record.

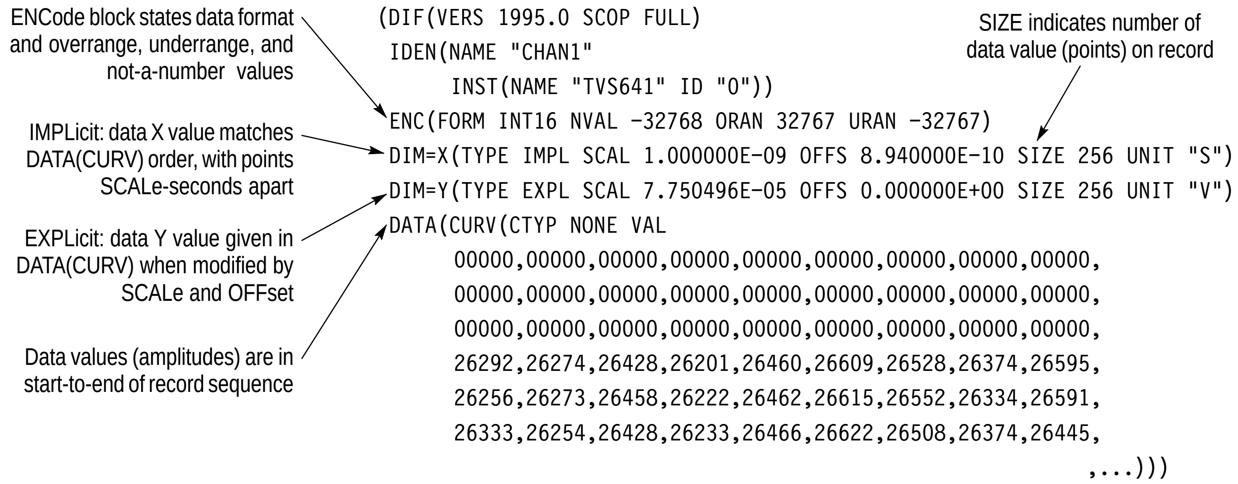
The records used are in the sequence specified by the AADVance:REC:STAR and AADVance:REC:COUN commands.

This last point reveals an important requirement: you must use the AAMList to take measurements over a sequence of auto-advance acquisitions. Any measurement specified using the WMList, or as an individual measurement in a CALC expression, returns only the measurement for the auto-advance record specified by AADVance:REC:STARt. Also, any calculation other than AAMList measurements returns the results for the auto-advance record specified by AADVance :REC:STARt. In a CALC expression, you can override the auto-advance record specified by AADVance:REC:STARt; see *Expressions for Auto-Advance Records* on page 3–36.

Waveform Reconstruction

Data that you upload from calculations (all models) or from reference memories (TVS600A models only) will have values already in volts (SCALE will be one and offset will be zero). Waveforms uploaded directly from channels as they are acquired must be reconstructed with the correct scale and offset.

To reconstruct amplitude vs. time (Y–T) waveforms that you upload, use the scaling and offset data in the DIMension block of the waveform preamble. The DIF waveforms transferred by the waveform analyzer contain explicit values used to reconstruct amplitudes (y axis) and implicit values from which you can extract times (x or T axis). A common example, based on the partial DIF block shown in Figure 3–41, follows.



```
(DIF(VERS 1995.0 SCOP FULL)
 IDEN(NAME "CHAN1"
      INST(NAME "TVS641" ID "0"))
 ENC(FORM INT16 NVAL -32768 ORAN 32767 URAN -32767)
 DIM=X(TYPE IMPL SCAL 1.000000E-09 OFFS 8.940000E-10 SIZE 256 UNIT "S")
 DIM=Y(TYPE EXPL SCAL 7.750496E-05 OFFS 0.000000E+00 SIZE 256 UNIT "V")
 DATA(CURV(CTYP NONE VAL
      00000,00000,00000,00000,00000,00000,00000,00000,00000,
      00000,00000,00000,00000,00000,00000,00000,00000,00000,
      00000,00000,00000,00000,00000,00000,00000,00000,00000,
      26292,26274,26428,26201,26460,26609,26528,26374,26595,
      26256,26273,26458,26222,26462,26615,26552,26334,26591,
      26333,26254,26428,26233,26466,26622,26508,26374,26445,
                                                            ,...)))
```

ENCode block states data format and overrange, underrange, and not-a-number values

IMPLicit: data X value matches DATA(CURV) order, with points SCALe-seconds apart

EXPLicit: data Y value given in DATA(CURV) when modified by SCALe and OFFset

Data values (amplitudes) are in start-to-end of record sequence

SIZE indicates number of data value (points) on record

Figure 3–41: DIF subblocks allow waveform reconstruction

1. First, note the data format (in this case, 16-bit integers) and the y boundaries (in this case data clips at +/–32767 values). Note that SIZE indicates the number of data points (256). Write your program routine or set up your application to handle the data accordingly.

2. Y data is explicit; to properly scale and offset each data point, your application should perform the following calculations for each point:

Y_i = SCALe $*$ datavalue$_i$ + OFFSet, where *i* is 1 .. SIZE
Y_{28} = 7.750496E–05 $*$ 26292 + 0
= 2.04 UNITS (volts)

3. X data is implicit; that is, to properly position each point in the data record, your application should accept the order of values within the data block as the order for reconstruction of the data record:

X*i* = i * SCALe + OFFset, where *i* is 1 .. SIZE
X_{28} = 28 * 1.0E–9 + 8.94E–10
= 28.89E–9 UNITS (seconds)

You may have noted that, in Figure 3–41, DIM=X lists an OFFset; it denotes the position of the trigger point from the start of the record and may be useful for aligning this record with other data records you process.

The data you upload from the waveform analyzer can be more complex than the Y-T curve just described. The waveform analyzer can return envelope waveforms, or it can return waveforms containing time, magnitude and phase information, with magnitude and phase in complex or polar form. For information useful in reconstructing this data, see the following sections and title:

- *Appendix E: Supported Preambles* for a list of all the data preambles that the waveform analyzer can generate.
- *Data Interchange Format* on page 3–131 for a description the TVS600A implementation of DIF.
- *Vol. 3: Data Interchange Format* of the *Standard Commands for Programmable Instruments,* published by the SCPI Consortium, for the complete description of DIF.

NOTE. *To obtain your channel waveform in voltage values instead of values that require reconstructing as described above, include the channel in a "no operation" expression. For example,* `send CALC1:PATH:EXPR "CHAN1"`, *and then query the calculation with* `CALC1:DATA?` *This method trades a higher waveform throughput for obtaining the voltage values directly (and ties up a Calc block), so be aware that it will slow the return of data when you use it.*

Data Downloads (TVS600A Models Only)

This section provides an overview of the general process used to download data from a controller; detailed information on data format is found on page 3–138. You can download the following types of data:

- Waveforms previously acquired by, and uploaded from, the waveform analyzer
- Envelope templates such as those provided in the TVS600A VXI*plug&play* Software (see *Template Sources* on page 3–166)
- Any data that meets TVS600A data interchange requirements for format of data and preambles

Rules for Download

All data that you download must download into one of the ten references, REF1 through REF10. References are locations in volatile memory that can store data as long as the waveform analyzer remains powered on.

You can send only the data block or only the preamble block, or you can send both blocks; the rules described in Table 3–37 cover all cases:

Table 3–37: Rules for downloads

Download contains:	Destination REF not empty	Destination REF empty
Preamble and data	Overwrites existing preamble Overwrites existing data	Writes new preamble Writes new data
Preamble, no data	Overwrites existing preamble Restructures existing data based on new preamble[1]	Writes new preamble Creates new data values initiated to zero
No preamble, data only	Existing preamble has its size adjusted to fit incoming data Overwrites existing data	Creates preamble with default scale (1.0), offset (0.0), and size (adjusted to fit incoming data) Writes new data

[1] **Size (record length) of existing reference data record will be truncated or zero-padded as necessary to match incoming preamble.**

Why Use?

Downloaded waveforms can be used in calculations; in fact, reference waveforms can be used in calculations just like waveforms acquired in channels. See the section *Template Testing* starting on page 3–165 for examples of REF waveforms used to test incoming waveforms against a downloaded template.

To Use

The following process describes in a general sense what's required to send a waveform or other data to the waveform analyzer:

1. Ensure that the data you intend to download meets the subset of the Data Interchange Format (DIF) standard that this product supports (see *Data Interchange Format* on page 3–131).

2. Use the Soft Front Panel or another message-based talker/listener application or write a program routine to send and feed the DIF block that contains your data to the following command:

   `TRACe REF<n>,<dif_block>`, where `<n>` is the number for one of `REF1` through `REF10` and <dif_block> contains the data and preamble you wish to download to the waveform analyzer.

You can also send a data and preamble separately, each in its own DIF block. As long as both DIF blocks meet requirements, it does not matter which you send first as long as they are both sent to the same destination REF.

Commands

The commands and functions to download data to the waveform analyzer follow:

Used to:	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
download data into one of REF1 – REF10	`TRACe REF<n>,<dif_block>`	`tktvs600_setRefTrace`	Yes

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Functions Reference.* Functions listed may available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application.**

Internal transfers

This section describes the general process used to temporarily save waveform-analyzer acquisitions and calculations internally, for use in subsequent processing. You can save from the following data sources:

- Channel waveforms
- Calculation results
- Reference waveforms or data

Rules for Transfer

All data that you save or transfer must use one of the ten references, REF1 through REF10, as a target. References are locations in volatile memory that can store data as long as the waveform analyzer remains powered on. Internal transfers save both data and preamble into the reference at the same time.

Why Use?

You may wish to save acquired waveforms for use in later calculations or for uploading later. Waveforms saved to a reference may be used as arguments in CALC expressions just as acquisition channels are used. Typically, a saved waveform is used to compare against recently acquired waveforms.

To Use

The following two processes result in internal transfers of data to references:

- Send the command: `TRACe:COPY <destination>,<source>`, where `<destination>` is any one of `REF 1` through `REF10` and `<source>` is one of `CHAN1` through `CHAN4`, `REF1` through `REF10`, or `CALC1` through `CALC4`. For example:

  ```
  TRACe:COPy REF6, CHAN1      copies CHAN1 to REF6
  TRACe:COPy REF1, REF3       copies REF3 contents to REF1
  TRACe:COPy REF1, CALC2      copies CALC2 results to REF1
  ```

  Channels copied must have been acquired; calculations copied must have been completed (either by sending the command `INITiate` or the command `CALC:IMMediate`).

- Create a calculation that assigns results to a reference: send the command `CALC1:PATH:EXPR "REF1 := <expression>"` where `<expression>` is any allowed expression. For example:

  ```
  CALC1:PATH:EXPR "REF1 := MEAN(CHAN1 + CHAN2)"
  ```

  stores the mean of the sum of channel 1 and channel 2 in reference 1.

Commands

The commands and functions to copy data to the onboard references within the waveform analyzer follow:

Used to:	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
copy channels, references, and calculation results into references	`TRACe:COPy REF<n> <source>`	`tktvs600_copyTrace`	Yes

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Functions Reference.* Functions listed may available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application.**

Fast Data Channel

The Fast Data Channel (FDC) provides a fast transfer protocol for moving waveform records and other data between the waveform analyzer and a VXIbus controller. The waveform analyzer includes an FDC driver that provides the instrument-side of the protocol.

Why Use?

Use FDC to enhance throughput for waveform and measurement transfers.

To Use

To use FDC, you must do the following tasks:

- Ensure that the controller you use provides support for the FDC protocol. (The VXI*plug&play* VISA I/O interface provides FDC support; the GPIB-VXI interfaces do not.)
- Open two sessions with the waveform analyzer, one for sending ASCII commands and another for FDC data transfer.
- Configure the FDC session to use the FDC protocol (by setting VISA attributes;see step 10 in the example on page 3–127).
- Use the ASCII session to set up the waveform analyzer, acquire the data, and initiate its transfer.

When configuring the ASCII and FDC sessions, you must perform setup steps in proper order:

- The number of active channels and the record length must be set before you enter the number of waveform records to acquire with `AADVance:COUNt`. The waveform analyzer automatically computes the maximum (MAX) value based on the number of active channels and the record length. (This rule applies to all auto-advance acquisitions.)
- You must establish all acquisition settings and configure the FDC session before issuing the INITiate command to begin acquisition.

The following procedure below shows the steps necessary to set up an auto-advance acquisition with FDC data transfer, observing the principles listed above. (Auto-advance and FDC match up to help increase throughput.) The procedure assumes you want to acquire CH1 and CH2.

1. Enable the channels to acquire with the command `FUNC:ON "XTIM:VOLT 1","XTIM:VOLT 2"`.
2. Set the individual channel parameters, such as voltage range.
3. Set the trigger parameters.

4. Set the waveform record length with `SWE:POIN` and the sample interval with `SWE:TINT`.

5. Enable auto-advance acquisition with the command `AADV:STAT ON`.

6. Specify the number of waveform records to acquire with the command `AADV:COUN`. The value `MAX` acquires enough waveform records to fill DSP memory.

7. Specify the number of waveform records to transfer over the FDC with the command `AADV:REC:COUN`. The value `MAX` selects all acquisitions, from `:STARt` on, for transfer.

8. Specify the list of signal sources to transfer with the command `TRAC:LIST`. You may have an active channel but not select it for transfer. In this example, use `TRAC:LIST CHAN1,CHAN2`. The value `AATS` selects the timestamp record for the auto-advance waveform records that you will acquire.

9. You can choose to skip the transfer of the first acquisition records for both channels if, for example, you think the device under test might initially be unstable. To skip just the first two waveform records, use the command `AADV:REC:STAR 3`, which skips over records 1 and 2 to begin FDC transfers on record 3.

10. Configure the FDC session to use the FDC protocol. Use the `viSetAttribute` function to set the VISA attributes as follows:

    ```
    viSetAttribute (vi_session, attr_name, attr_value)

    Attribute Name
    Attribute Value

    VI_ATTR_FDC_CHNL
    1

    VI_ATTR_FDC_MODE
    VI_FDC_STREAM

    VI_ATTR_IO_PROT
    VI_FDC
    ```

11. Start acquisition on the waveform analyzer with the `INITiate` command. Acquisition begins.

12. Use the VISA function `viRead (fdc_session, buf, buf_size, return_count)` to transfer the acquisition from the waveform analyzer to the character array buffer of the controller.

Commands

The commands and functions to set FDC follow:

Used to:	Commands[1]	Driver Functions[2]	SFP?[3]

set VISA attributes	`viSetAttribute (vi_session, attr_name, attr_value)`	`tktvs600_setFDCTraceList`
		`tktvs600_copyTrace`
		`tktvs600_acqInitiateFDC`
		`tktvs600_readFDC`
		`tktvs600_initiateFDC`
		`tktvs600_fetchFDCBlock`
		`tktvs600_abortFDC`
			Yes

copy acquired channels to FDC channel	`TRACe:COPy FDC1,<source>`		No
send list after acquiring	`TRACe:LIST <source>`		No

[1] **The first command listed is a VISA attribute command; the remaining commands listed are SCPI commands.**

[2] **Look up in the online *TVS600A Functions Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the TVS600A Soft Front Panel application**

Usage Notes

Some usage notes follow:

- You can obtain the time stamp of the each acquisition record; see the `TRACe` commands in the *TVS600 & TVS600A Series Waveform Analyzers Command Reference*.
- For more information, refer to *Acquisition Cycle* on page 3–6 and *Auto-Advance Acquisition* on page 3–13. Additionally, refer to the command descriptions in the *TVS600 & TVS600A Series Waveform Analyzers Command Reference*.

FDC Example

The following C-code example sets up an FDC channel, configures the TVS600A, transfers a 512-point waveform, and closes the FDC channel. The program assumes the controller uses the VXI*plug&play* Interface to control the FDC session; users of other interfaces supplying FDC support should modify the program to support those interfaces.

The program simply exits if an error occurs. You should add statements to the program so it closes all sessions before exiting the program after an error.

The transferred waveform data is stored in a buffer (waveformBuf) as binary bytes (2 bytes/waveform sample). For a 512-point record, the received data includes 1024 bytes plus a few bytes of header information. The example program ends with the appropriate statements to close all sessions.

***NOTE**. This program was written in standard ANSI C.*

*The VXI*plug&play *VISA I/O interface provides FDC support; the GPIB-VXI interfaces do not.*

*For another example of FDC operations that make use of driver functions, see the TVS600A VXI*plug&play *software disk that comes with this product.*

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "c:\vxipnp\win\include\visa.h"

#define WAVEFORM_BUF_SIZE 2048 */Buffer sized for 512 pt record */

void main() {
   ViStatus   status; /* Return value from VISA.DLL functions */
   ViSession  visaRM; /* VISA resource manager session handle */
   ViSession  tvs600; /* Session handle for command channel */
   ViSession  fdc;    /* Session handle for FDC channel */
   ViChar tvs600Desc[256]; /* Descriptor used to address TVS600 */
   ViChar waveformBuf[WAVEFORM_BUF_SIZE]; /* Local waveform memory */
   ViUInt32   retCnt; /* Returns actual number of bytes transferred */

   /****************************************************************/
   /*                                                              */
   /*  Establish Communication with the TVS600A                    */
   /*                                                              */
   /****************************************************************/

   /* Set the TVS600 descriptor */
   /* Customized following line with the logical address of TVS600. */
   /* The address here is 2. */
   strcpy(tvs600Desc, "VXI::2::INSTR");

   /* Open session with the VISA default resource manager */
   status = viOpenDefaultRM(&visaRM);
   if (status) return;

   /* Open a session with the tvs600 to send ASCII commands */
   status = viOpen(visaRM, tvs600Desc, VI_NULL, VI_NULL, &tvs600);
   if (status != VI_SUCCESS) return;

   /* Open a session with the tvs600 to receive FDC data */
   status = viOpen(visaRM, tvs600Desc, VI_NULL, VI_NULL, &fdc);
   if (status != VI_SUCCESS) return;
```

Example continued on following page.

Example continued from previous page

```
/****************************************************************/
/*                                                              */
/*  Set up the FDC session attributes                           */
/*                                                              */
/****************************************************************/
/* Selects FDC channel 1. */
/* The TVS600 outputs data on FDC channel 1.  */
    status = viSetAttribute(fdc, VI_ATTR_FDC_CHNL, 1);
    if (status != VI_SUCCESS) return;

/* The TVS600 uses a stream channel. A stream channel can */
/* send a series of waveforms more efficiently.              */
status = viSetAttribute(fdc, VI_ATTR_FDC_MODE, VI_FDC_STREAM);
if (status != VI_SUCCESS) return;

/* Select the FDC protocol for this session */
status = viSetAttribute(fdc, VI_ATTR_IO_PROT, VI_FDC);
if (status != VI_SUCCESS) return;

/* Set the FDC session timeout to 10 seconds */
status = viSetAttribute(fdc, VI_ATTR_TMO_VALUE, 10000);
if (status != VI_SUCCESS) return;

/****************************************************************/
/*                                                              */
/*  Set up the TVS600 acquisition                               */
/*                                                              */
/****************************************************************/
/* Initialize the TVS600 to a known state */
if (viPrintf(tvs600, "ABORT\n") != VI_SUCCESS) return;
if (viPrintf(tvs600, "*RST\n")  != VI_SUCCESS) return;

/* Set up channel 1 for the acquisition: */
/* Ch 1 50 ohm, 512 pts record, 100 MS/s, transfer Ch 1 only */
if (viPrintf(tvs600, "INP:IMP 50\n")      != VI_SUCCESS) return;
if (viPrintf(tvs600, "VOLT1:RANG 0.5\n")  != VI_SUCCESS) return;
if (viPrintf(tvs600, "FUNC:ON CHAN1\n")   != VI_SUCCESS) return;
if (viPrintf(tvs600, "FUNC:OFF CHAN2\n")  != VI_SUCCESS) return;
if (viPrintf(tvs600, "DATA:LIST CHAN1\n") != VI_SUCCESS) return;
if (viPrintf(tvs600, "SWE:POIN 512\n")    != VI_SUCCESS) return;
if (viPrintf(tvs600, "SWE:TINT 10nS\n")   != VI_SUCCESS) return;
if (viPrintf(tvs600, "INIT\n")            != VI_SUCCESS) return;
```

Example continues on following page

Example continued from previous page

```
      /****************************************************************/
      /*                                                              */
      /*  Start transfer of TVS600 acquired data using FDC.          */
      /*                                                              */
      /****************************************************************/
      status = viRead(fdc, waveformBuf, WAVEFORM_BUF_SIZE, &retCnt);
      if (status != VI_SUCCESS) return;

      /****************************************************************/
      /*                                                              */
      /*  Close all sessions to deallocate resources and memory      */
      /*                                                              */
      /****************************************************************/
      /* Close the command session */
      status = viClose(tvs600);
      if (status != VI_SUCCESS) return;
      /* Close the FDC session */
      status = viClose(fdc);
      if (status != VI_SUCCESS) return;
      /* Close the VISA resource manager */
      status = viClose(visaRM);
      if (status != VI_SUCCESS) return;
   }
```

Data Interchange Format

To use the data that you upload and to ensure the data you download is usable by the waveform analyzer, you need to understand how the data is organized and how scaling and other information is encoded. The following sections describe data encoding and format:

- *DIF Block Composition* describes the make up of the preamble; that is, the body of information that lets you, after uploading, or lets the waveform analyzer, after downloading, reconstruct the raw data with proper vertical scaling and timing.

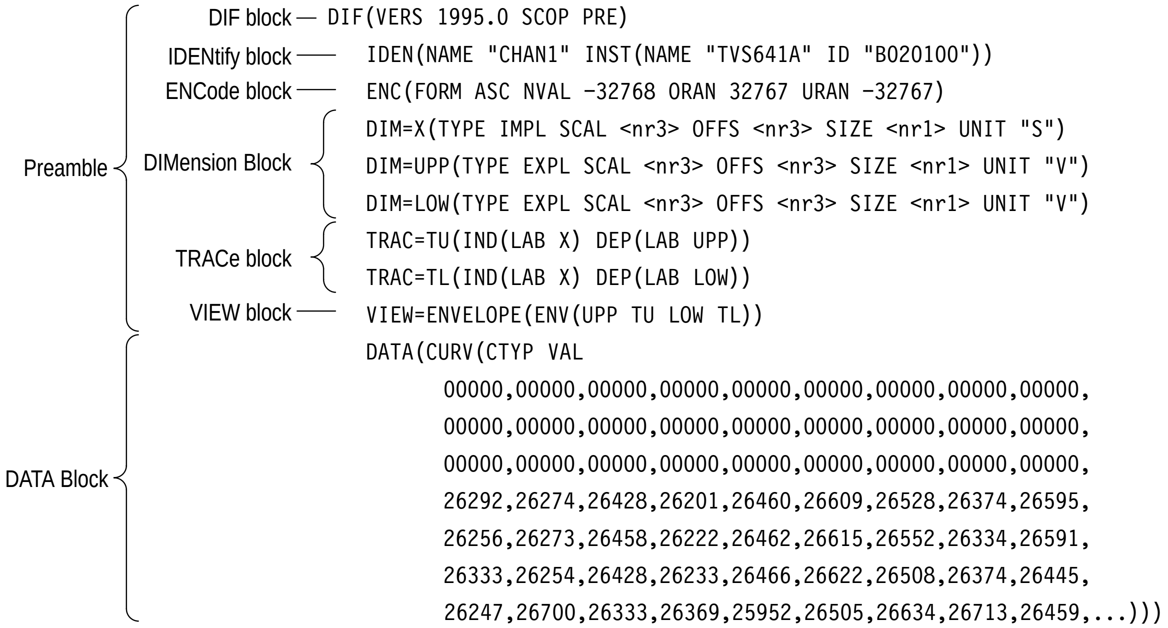
- *Data Formats*, on page 3–138, describes the formats for encoding the raw data.

The waveform analyzer accepts and outputs both the data and the preamble using the Data Interchange Format (DIF) standard. This section briefly describes the TVS600A implementation of DIF; the complete description of the standard is *Vol. 3: Data Interchange Format* of the *Standard Commands for Programmable Instruments*, published by the SCPI Consortium.

DIF Block Composition

Figure 3–42 shows the composition of a complete DIF block consisting of data and preamble (and its subblocks). The subblocks are then briefly described; the syntax for all blocks is as per the SCPI 1995 standard just referenced.

In their descriptions, subblocks that must be present in any download to the waveform analyzer are so indicated. Subblocks that are present in uploaded waveform-analyzer waveforms, calculations, and measurements are also so indicated.



```
DIF(VERS 1995.0 SCOP PRE)
   IDEN(NAME "CHAN1" INST(NAME "TVS641A" ID "B020100"))
   ENC(FORM ASC NVAL -32768 ORAN 32767 URAN -32767)
   DIM=X(TYPE IMPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "S")
   DIM=UPP(TYPE EXPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "V")
   DIM=LOW(TYPE EXPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "V")
   TRAC=TU(IND(LAB X) DEP(LAB UPP))
   TRAC=TL(IND(LAB X) DEP(LAB LOW))
   VIEW=ENVELOPE(ENV(UPP TU LOW TL))
   DATA(CURV(CTYP VAL
         00000,00000,00000,00000,00000,00000,00000,00000,00000,
         00000,00000,00000,00000,00000,00000,00000,00000,00000,
         00000,00000,00000,00000,00000,00000,00000,00000,00000,
         26292,26274,26428,26201,26460,26609,26528,26374,26595,
         26256,26273,26458,26222,26462,26615,26552,26334,26591,
         26333,26254,26428,26233,26466,26622,26508,26374,26445,
         26247,26700,26333,26369,25952,26505,26634,26713,26459,...)))
```

Figure 3–42: Anatomy of DIF block

DIF block. This block must be the first received and is not optional. This block is used to identify the expression as a DIF expression. The waveform analyzer accepts information and outputs information in a DIF expression.

Arguments: `VERS, SCOPe`

This block, with the argument as shown, will be present on uploaded data; all arguments are ignored and are not retained when part of a download to the waveform analyzer.

REMark block. (Not shown.) This block is optional, but if included, it must precede any IDENtify block.

This block will not be present on uploaded data; it is ignored and not retained when part of a download to the waveform analyzer.

IDENtify. This block is optional, but if included ,it must precede any ENCode block. It includes information intended to identify the data.

This block will be present with uploaded data and will include the data-source (channel, CALC block, or reference) and the instrument name and ID; it is ignored and not retained if part of a download to the waveform analyzer unless the following special argument is included:

Arguments: `SETup <string> {<string>*}`

The special SETup argument contains information needed to set up the waveform analyzer for limit testing against templates as described in *Template Anatomy* on page 3–167. It is not used by the waveform analyzer; rather it is included template waveforms so it can be processed later by a host program or application doing template testing.

The SETup argument must be processed from the template file on the system, not the downloaded template file, since the waveform analyzer discards the SETup information. SETup is not a SCPI standard argument.

This block will not be present on uploaded data; it is retained in the preamble but otherwise ignored if it is part of a download to the waveform analyzer.

ENCode. This block is optional, but if included it must precede any DIMension block. It specifies the format which encodes the data found in the data block.

Arguments: The following data formats can be downloaded:

ASCII			
INT8	INT16	INT32	INT64
UINT8	UINT16	UINT32	UINT64
	SINT16	SINT32	SINT64
	SUINT16	SUINT32	SUINT64
		IFP32	IFP64
		SFP32	SFP64

The allowed formats for uploads are listed in Table 3–38 on page 3–138.

This block will be present on uploaded data; it is needed to help reconstruct the data when part of a download to the waveform analyzer.

DIMension. This block is required and must precede any TRACe block. It provides the information needed to reconstruct the data with the correct vertical and horizontal scale and offset.

Arguments: TYPE, SCALE, OFFSET, SIZE, ENCODE, and UNITS.

This block will be present on uploaded data; it is needed to help reconstruct the data and must be part of a download to the waveform analyzer.

The waveform analyzer produces (in uploaded data) and handles (in downloaded data) a limited number of data-dimension configurations. When it downloads an unrecognized configuration, the waveform analyzer issues an event to the sender, reconstructs the data by tuple order with default attributes (unity scale, 0 offset, size = data), and stores it in the specified reference.

(*Dimension Block Details* on page 3–135 provides more extensive information on the arguments used in DIMension blocks.)

ORDer. (Not shown.) This block is optional, but if included, it must precede any TRACe block. It specifies the data-ordering method for data labeled as EXPLicit in the dimension block.

Arguments: TUPLe or DIMension

Order must be by tuple or dimension. Briefly, the tuple method orders explicit data in a row orientation and the dimension method would order explicit data in a column orientation. For example, an envelope of max/min pairs would be ordered as follows:

Tuple (Default)	Dimension
max value$_0$, min value$_0$,	max value$_0$, max value$_1$,
max value$_1$, min value$_1$,	... , max value$_n$,
...	min value$_0$, min value$_1$,
max value$_n$, min value$_n$	... , min value$_n$

Tuple is the default method, and when data is single valued (often the case), each tuple is a 1-tuple, meaning the data is simply a string of encoded data points ordered in the same order as acquired (a Y-T curve). For an example of reconstructing such data, see *Waveform Reconstruction* on page 3–121.

TRACe. This block is optional, but if included, it must precede any VIEW block. It describes relationships between dimensions in the dimension block and affects data order in the data block.

Arguments: INDEPENDENT, START, STOP, DEPENDENT, and LABEL.

If the TRACe block is omitted in downloaded data, data is assumed to be a simple Y–T curve (vector) and must contain only one implicit and one explicit dimension; all other organizations of data will cause the waveform analyzer to issue a warning event to the sender, reconstruct the data by tuple order with default attributes (unity scale, 0 offset, and size = data), and store the reconstructed data in the specified reference.

VIEW Block. (Not shown.) This block is optional, but if included, it must precede any DATA block. VIEW and TRACe blocks are used together to define ENVelope and COMPlex curve data. VIEW affects data order in the data block.

Arguments: `ENVELOPE`, `UPPER`, `LOWER`, `RCOMPLEX`, `REAL`, `IMAGINARY`, `PCOMPLEX`, `MAGNITUDE`, and `PHASE`.

If this block is present in downloads, it must contain only one implicit and two explicit dimensions; any other organization of the data will cause the waveform analyzer to issue an event to the sender, reconstruct the data by tuple order with default attributes (unity scale, 0 offset, and size = data), and store it in the specified reference.

DATA Block. This block is required and must be the last block. The DATA block contains the waveform data ordered as described under ORDer, TRACe, and VIEW blocks.

Arguments: `CURVE`, `CTYPE`, `VALUES`, and `CSUM`.

Waveform-analyzer data may be a string of single values, or 1-tuples, such as for an acquired waveform with only amplitude and time information (a Y–T curve). Or it may be a sequence of multi-values, such as for an envelope waveform, or for a waveform containing time, magnitude and phase information, with the later two values in complex or polar form. See Table D–1 *TVS600A Preambles* on page D–1 for a list of DIF blocks shown with the various preambles that characterize the possible data cases.

When the waveform analyzer responds to preamble queries, it returns a null data block in the DIF expression; if you send a preamble without data to the waveform analyzer, the DIF expression must contain a null data block. A null data block looks like `(DATA(CURV(CTYP NONE))`.

Dimension Block Details

The following information on the dimension block is adapted with permission from *Volume 3: Data Interchange Format* of the SCPI standard. Consult that standard for further information.

Dimension blocks provide implicit and explicit information needed to reconstruct the data you upload so it retains vertical and horizontal scaling and offset information.

A DIMension block is required for each dimension of the data (implicit and explicit). The DIMension block is defined as follows:

```
dimension -> DIMension = <Label>(
   [NOTE     <String>]
   [NAME     <String>]
   TYPE      {IMPLicit | EXPLicit}
   [SCALe    <Numeric>]
   [OFFSet   <Numeric>]
   [SIZE     <+NR1>]
   UNITs     <String>
   [encode]
)
```

Each dimension is uniquely distinguished by its required block modifier <Label> value. No two DIMension blocks may have the same <Label> value. Dimension labels should be kept short, yet meaningful.

NOTE. This is arbitrary text. Value type is <String>.

Example: NOTE “DS1C at test point 12”

NAME. This is an arbitrary name or description of the dimension.

Example: NAME “DS1C Voltage/Amplitude”

TYPE. This indicates if the dimension is implicit or explicit.

TYPE takes a specified enumerated set of values. TYPE is required, may occur only once, and takes one of the two following enumerated set values:

IMPLicit — Values for this dimension are not in DATA(CURVe), but are derived from a linear function, y = mx + b.

EXPLicit — Values for this dimension are present in DATA(CURVe).

Example: TYPE EXPL

SCALe. Scaling factor to be applied to the dimension’s values in DATA(CURVe(VALues)).

Value type is <Numeric>. Default value is 1.0. Only one occurrence of SCALe is allowed and only one value is allowed.

SCALe is always applied to DATA(CURVe) data.

For implicit dimensions:

X’ = (SCALe * *i*) + OFFSET *i* = 1 to SIZE

For explicit dimensions:

X’ = (SCALe * X) + OFFSet

where X is a value from DATA(CURVe(VALues)). Also see *ORDer* on page 3–134.

Example: SCALe 0.5E–2

OFFSet. Offset factor to be applied to the dimension's values in DATA(CURVe(VALues)).

Value type is <Numeric>. Default value is zero. Only one occurrence of OFFSet is allowed and only one value is allowed.

OFFSet is always applied to DATA(CURVe) data.

For implicit dimensions:

X' = (SCALe * *i*) + OFFSet, *i* = 1 to SIZE

For explicit dimensions:

X' = (SCALe * X) + OFFSet

X is a value from DATA(CURVe(VALues)).

Example: OFFSet –1.0E–2

SIZE. This is the number of points or values in the dimension.

Value is type <+NR1>. Only one occurrence of SIZE is allowed and only one value is allowed.

The following two invariants hold:

1. All explicit dimensions must have the same SIZE value.
2. The product of the SIZE values of the implicit dimensions must equal the SIZE value of any explicit dimension.

If SIZE is not specified for a dimension, it is assumed to be consistent with the above variants. It is an error if there is insufficient information to determine a dimension's SIZE. For example, SIZE cannot be determined if two or more implicit dimensions do not specify a SIZE, nor can it be determined if two or more explicit dimensions specify different SIZE values.

It is strongly recommended that SIZE be specified for all dimensions whenever possible.

Example: UNITs "V"

Data Formats

The waveform analyzer encodes waveform data in your choice of either signed-integer ASCII or 16-bit binary formats. It encodes calculated data as 32-bit, floating point values: either ASCII or binary. Table 3–38 summarizes the formats supported by FORMat commands of the waveform analyzer:

Table 3–38: Data formats

Data Type	Format	Data values range	Over/Underrange values[1]	NULL values[2]
Waveform	ASCII (ASCII,0)	–32767 to 32767	+32767/–32767	–32768
	16-bit binary (INT,16)	–32767 to 32767	+32767/–32767	–32768
Calculated	ASCII integer (ASCII,0)	–9.9e+37 to +9.9e+37	+9.9e+37/–9.9e+37	+9.91e+37/–9.91e+37
	32-bit REAL (REAL,32)	–9.9e+37 to +9.9e+37	+ ∞ /– ∞	NaN (Not a Number)
ATTS	Same as for *Calculated*	Same as for *Calculated*	Same as for *Calculated*	Same as for *Calculated*

[1] **This value indicates out of range data, such as when acquired data exceeds the range of a channel; if this value is defined in IEEE 754–1985, the value listed matches the defined value.**

[2] **This value indicates invalid data or no data for a data point; if this value is defined in IEEE 754–1985, the value listed matches the defined value.**

ASCII Waveform Data. ASCII waveform data encodes each data point as a signed integer value. Each data value requires two to seven characters: one to five characters that represent the value, a character that represents a minus sign for negative values, and a comma that separates data points. An example of an ASCII waveform record follows:

```
0,-23,-64,-47,-15,2,87,324,989,1952,29847, ...
```

The number of data bytes returned is determined by the length of the waveform record and the number of characters in each data value.

Binary Waveform Data. Binary waveform data is represented by 16-bit integer values and is used for normal trace acquisition. The data ranges from –32767 to 32767. Every data value is represented by two data bytes as shown in the example in Figure 3–43.

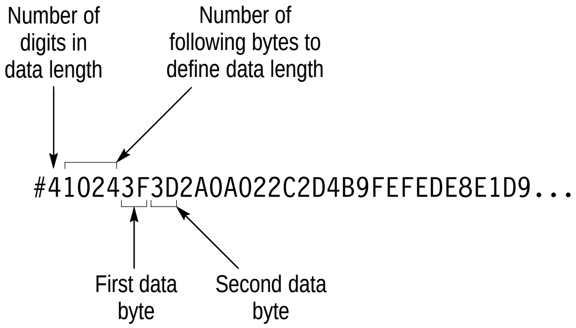

Number of digits in data length
Number of following bytes to define data length
#410243F3D2A0A022C2D4B9FEFEDE8E1D9...
First data byte
Second data byte

Figure 3–43: Binary transfer format

The binary formats send the high byte (most significant) followed by lower bytes by default. You can change the byte order with FORMat:BORDer to either NORM (high byte/low byte) or SWAP (low byte/high byte).

ASCII CALC Data. Data from a CALC block and the AATS-timestamp data can be transferred in ASCII (format ASC,0). Each data value includes a minus sign (if negative), a fraction, and an exponent. Commas separate the data points.

ASCII waveform record may appear as follows:

```
2E–6,5.34E–6,2.21E–4,4.65E–2,5.6E1,7.91E2,. . .
```

The number of data bytes returned is determined by the length of the waveform record and the number of characters in each data value.

You specify the data format for the four CALC blocks with the `FORMat:CALC<n>` commands and the time stamp format with the `FORMat:TRAC:AATS` command.

Binary CALC Data. Data from a CALC block and the AATS-timestamp data can also be transferred as IEEE Standard 32-bit, floating-point binary values (format REAL,32).

A 32-bit binary, floating-point number is formatted as follows:

<sign 1 bit><exponent 8 bits><fraction 23 bits>

Figure 3–43 shows the basic binary transfer format. The 32-bit numbers are sent as four bytes. The first or high byte will contain the sign bit plus 7 exponent bits. The second byte contains the remaining 1 exponent bit and 7 fraction bits.

You can reverse the order of the bytes with the command `FORMat:BORDer`. However, note that you can not swap bytes when transferring 16-bit words as when transferring waveforms.

You specify the data format for the four CALC blocks with the `FORMat:CALC<n>` commands and the time stamp format with the `FORMat:TRAC:AATS` command.

Measurements

The waveform analyzer can automatically take voltage, time, area, and statistical waveform measurements. This section describes the measurements and how to use them.

Automatic measurements have the following characteristics:

- Measurements may be taken over the entire waveform or limited to a segment on the waveform you specify (localizing of measurements).
- Measurements may be compared against limits you define (limit testing).

Measurements available for your use are listed in Table 3–39 on page 3–154.

Read *Calculation System Overview* on page 3–21 for background on how the CALCulate commands work. Measurements are derived from the CALC system and must be set up using CALC blocks. Because the CALC system supports the two models for calculations, SCPI and Expression, both SCPI-based and expression-based measurements can be made.

This section first explains the measurement parameters that affect measurement results, and then describes and lists the procedure for setting up and taking measurements. Topics that describe localizing measurements to zones on waveforms and testing measurements against limits follow, beginning on page 3–148.

Measurement Parameters

To make automated measurements, the waveform analyzer uses a CALC block that, in addition to specifying which measurement is taken (for example, rise time) also specifies methods for determining the measurement parameters. You can accept the default values and methods for these parameters or change them using `CALCulate:WMParameter` commands.

Waveform Measurement Parameter Block (WMP)

Figure 3–44 introduces the Waveform Measurement Parameter block (WMP), which contains the parameter values of each CALC block.

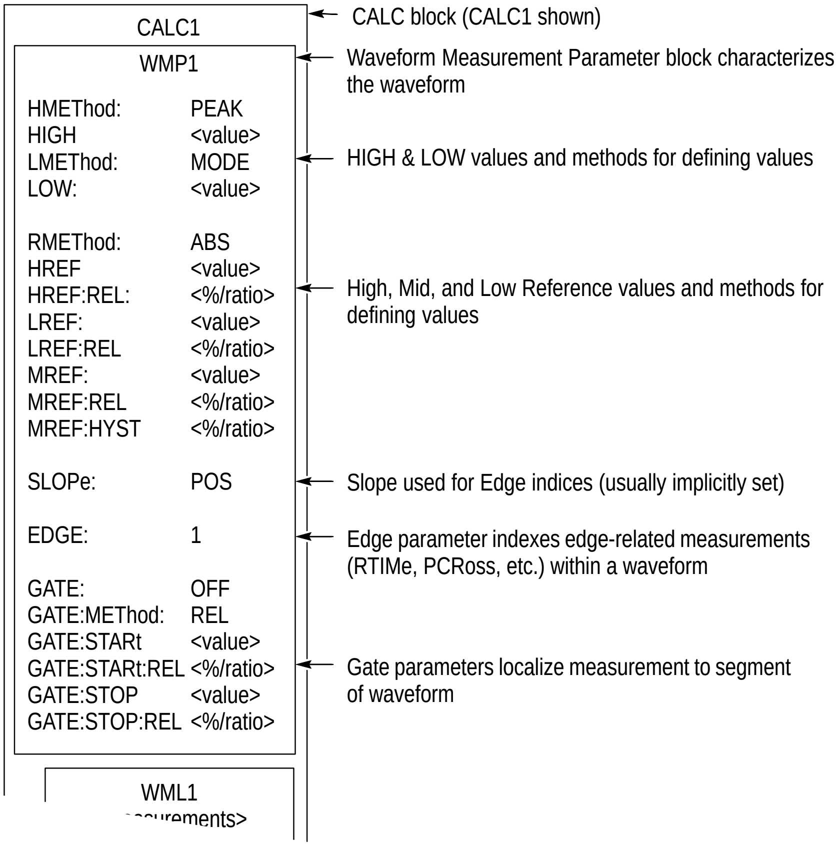

CALC1
WMP1
HMEThod: PEAK
HIGH <value>
LMEThod: MODE
LOW: <value>
RMEThod: ABS
HREF <value>
HREF:REL: <%/ratio>
LREF: <value>
LREF:REL <%/ratio>
MREF: <value>
MREF:REL <%/ratio>
MREF:HYST <%/ratio>
SLOPe: POS
EDGE: 1
GATE: OFF
GATE:METhod: REL
GATE:STARt <value>
GATE:STARt:REL <%/ratio>
GATE:STOP <value>
GATE:STOP:REL <%/ratio>
WML1
<measurements>
CALC block (CALC1 shown)
Waveform Measurement Parameter block characterizes the waveform
HIGH & LOW values and methods for defining values
High, Mid, and Low Reference values and methods for defining values
Slope used for Edge indices (usually implicitly set)
Edge parameter indexes edge-related measurements (RTIMe, PCRoss, etc.) within a waveform
Gate parameters localize measurement to segment of waveform

Figure 3–44: Calculation showing waveform parameters in WMP block

HIGH and LOW. HMEThod and LMEThod specify the methods used to determine the HIGH and LOW levels. To make most measurements, the waveform analyzer must determine the top of the waveform, or the HIGH level, and the bottom, or LOW level. You can select from the following four methods:

- `ABSolute`. Specifies that HIGH and LOW are set to absolute amplitude levels with `CALCulate:WMParameter:HIGH` and `CALCulate:WMParameter:LOW`.
- `AUTO`. Selects the MODE method of setting HIGH and LOW when the histogram function is able to detect a consistent level above and below MID. Otherwise, the PEAK method is used. AUTO method is effective when you are not certain what type of waveform to expect.
- `MODE`. Selects the levels for HIGH and LOW based on a peak histogram function which looks for the most common value above and below MID. (LMEThod is set to MODE in Figure 3–44.) This method ignores short term aberrations, such as overshoot and ringing, on a digital logic waveform.

- PEAK. Specifies that HIGH and LOW are set to the highest and lowest values in the waveform record. (HMEThod is set to PEAK in Figure 3–44.) This method is useful for a sine wave or triangle waveform but not for digital pulses.

For more information on HIGH and LOW and setting specific values for these parameters, refer to Table 3–40 *Measurement Parameters* on page 3–157 and to *Appendix B: Algorithms*.

LREF, MREF, and HREF. RMETHod specifies the method used to determine the LREF, MREF, and HREF levels. To make most edge-related measurements, such as rise time or period, the waveform analyzer must determine the location of these reference levels on the waveform. You can select from the following two methods:

- ABSolute lets you set the reference levels as absolute vertical levels.
- RELative lets you set them as a ratio or percentage of the difference between the HIGH and LOW measurement parameters.

You choose between these methods using the command CALC:WMP:RMEThod.

For more information on these reference levels, refer to *Appendix B: Algorithms*.

SLOPe. Determines the slope of any edges used in DELay measurements. All edge measurements, except DELay, determine the slope of the edge implicitly from the measurement specified. For example, FTIMe (falltime) implies a negative slope. See *EDGE*. (DELay measurements require that you specify slope.)

EDGE. Specifies the edge index. All measurements of edges, FTIMe, RTIMe (rise time), PCRoss, NCRoss, CROSs (positive-, negative-, and either-polarity crossings), DELay (delay), and COPulse (center of pulse) use the edge parameter as the index. Edge counting proceeds from the start of the waveform record for positive-edge values and from the end for zero and negative-edge values. Only the appropriate edges for measurement are counted; for example, positive-going edges are counted for measurements that are implicitly positive, like RTIMe or PCRoss. See *Measurement Zone and Edge Selection* in *Appendix B: Algorithms.*

GATE. All measurements can be taken based on the entire waveform record (GATE:OFF) or on a gated area or zone on the record that you specify (GATE:ON). See the procedures that follow in this section for more information.

Parameters List

All the measurement parameters that you may set are described in Table 3–40 on page 3–157.

Specifying Measurements

In general, a measurement is a specific calculation, which is a post-acquisition process that occurs after a waveform record is acquired and stored in acquisition memory. (TVS600A models can also perform calculations on waveforms stored locally in references.) The generic process for calculating a measurement follows; refer to Figure 3–45 as you read the following overview process:

1. You set up the waveform analyzer to acquire the waveform(s) you want to measure.
2. You send CALC commands to define your measurement using a CALC block. You define the CALC block (see *The CALC Blocks* on page 3–22) somewhat differently depending on whether you use the SCPI or Expression calculation model, as is indicated in the following steps:
   - Define the WMP block, setting waveform-characterization parameter methods and values (from Table 3–40).
   - Define the WML block. Add the measurements to the list.
   - Define the Feed. Select the channel or reference to be measured. (SCPI only).
   - Set the WML:STATe On (SCPI only).
   - Define any gating requirements (TVS600A models only).
   - Define the function list (SCPI model) or the expression (Expression model) to include the measurement to be taken.
3. You initiate an acquisition.
4. You send the `TRACE:DATA`? or `CALC:DATA?` query to fetch your measurement results.

The high-level process just described should be modified when you recalculate or perform new measurements on data already acquired or stored locally in references. In such cases, skip step 1 and replace step 3 with a `CALCulate:IMMediate` command.

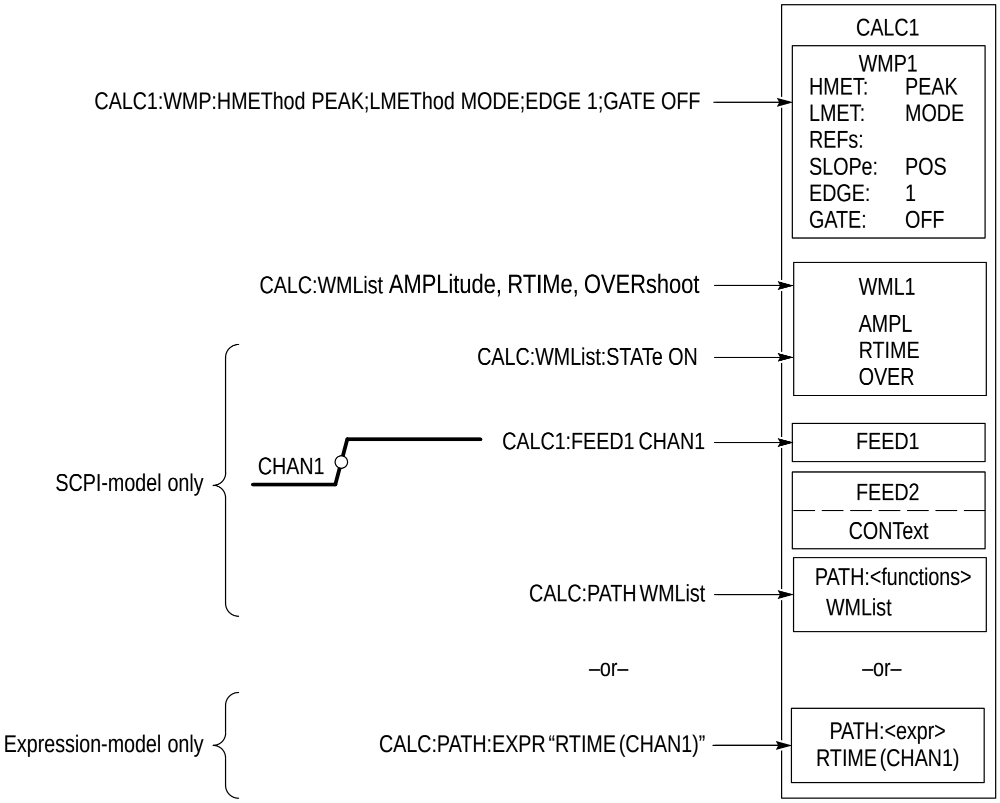


Figure 3–45: Setup of CALC block to take measurements

Two Models

Both SCPI and Expression models are covered in the procedures that follow; choose the model you use based on the capabilities of the model:

- SCPI model. This model processes measurements as one function in a linear list of functions applied to the data source. SCPI-measurement calculations are best suited for running multiple measurements on a waveform, perhaps after filtering data before passing it to the WML (Waveform Measurements List) function. (Functions must be able to run linearly, in sequence.)

- Expression model. This model processes measurements in the course of processing a user-defined expression. The expression calculation can return a single measurement, a vector of measurements or of statistics on measurements, or the sums, differences, products, and ratios of measurements, all depending on the expression you define. Measurements can also be tested against limits using the expression model.

Why Use?

You use automated measurements to easily extract parameters on your test waveform, perhaps for comparison against limits in a test program. You use gating to limit the measurements to a segment on the waveform record.

To Use (SCPI Calc Model)

Use the following procedure to set up a CALC block specifying a measurement calculation:

1. Set up the probe and input coupling and the vertical and horizontal windows. (See procedures in *Input Signal Conditioning* starting on page 3–101.)

2. Set up triggering to meet your requirements. (See procedures in *Trigger Types* starting on page 3–193.)

3. Set up the WMP block: send `CALC<n>:WMP:<para1> <value1>;<para2> <value2>;<paran> <valuen>`, where `CALC<n>` is one of CALC block 1 through 4 and `<para>` and `<value>` are the parameters and values selected from Table 3–40 on page 3–157.

   For example, to set absolute values for HIGH and LOW and turn gating off:

   ```
   CALC1:WMP:HMEThod ABSolute;
   CALC1:WMP:LMEThod ABSolute;
   CALC1:WMP:GATE OFF
   CALC1:WMP:HIGH 5;LOW 0
   ```

4. Select the data source: send `CALC<n>:FEED1 CHAN<n>` or `REF<n>`, where `CALC<n>` is one of CALC block 1 through 4, `CHAN<n>` is one of Ch1 through Ch 4, and `REF<n>` is one of Ref 1 through Ref 10.

5. Select one or more measurements to be calculated: send `CALC<n>:WMList <arg1>, <agr2>, ... <argn>`, where `CALC<n>` is one of CALC block 1 through 4, and `<arg 1>` through `<argn>` are measurements selected from those listed in Table 3–39 *Measurement Definitions* on page 3–154.

   For example, to measure `AMPLitude, RTIMe, OVERshoot`, send: `CALC<n>:WMList AMPLitude, RTIMe, OVERshoot`.

6. If defining a dual-waveform measurement (DELay, GAIN, or PHAse), set a second data source and the WMP block used to characterize it:

   - Send `CALC<n>:FEED2 CHAN<n>` or `REF<n>`, where `CALC<n>` is one of CALC block 1 through 4, `CHAN<n>` is one of Ch1 through Ch 4, and `REF<n>` is one of Ref 1 through Ref 10.

   - Send `CALC<n>:FEED2:CONText WMP<n>`, where `CALC<n>` is one of CALC block 1 through 4 and `WMP<n>` is one of the WMP blocks 1 through 4.

   Note that while CONTEXT defaults to match CALC-block number, you can redefine it to specify any WMP block; it need not match the CALC block in which it is defined.

7. Enable the measurement list: send `CALC<n>:WMList:STATE ON.`

8. Include the measurement list with your CALC functions; send: `CALC<n>:PATH WMList`, where `CALC<n>` is the CALC-block number.

9. Use the appropriate commands to return your measurement:

   - If you have specified a channel not yet acquired as the FEED, send the command `INITiate` and then the query `CALC<n>:DATA?` to first acquire your data source and then to return your calculation.

   - If you have specified a channel already acquired as the FEED (as when you are recalculating the same data) or if the feed is a reference waveform, send the command `CALC<n>:IMMediate` and then query `CALC:DATA?` to calculate without acquiring and to return your calculation. (Only TVS600A models come equipped with References.)

***NOTE**. You can combine both a recalculation of data and a fetch of the results by sending the query form CALCulate:IMMediate? query instead of the CALCulate:IMMediate command, followed by the CALC:DATA?query.*

To Use (Expression CALC Model)

The procedure just described can be used to perform measurements using calculation expressions; just call out your data sources in the expression and ignore the state ON of WML. (If you use WML in an expression, you must include any measurements you want to take in the measurement list; use of individual measurements in an expression do not have to be in the WML.)

1. Perform steps 1 through 3 of the SCPI-based procedure just listed.

2. Specify an expression, including data sources, functions and operations: send `CALCulate<n>:PATH:EXPR <expr>`, where <n> is the CALC-block number, `1–4`, and `<expr>` is any algebraic expression that evaluates to the form <measurement><data source>.

   For example, to measure the RMS value of channel 1 using CALC3, send `CALCulate3:PATH:EXPR "RMS (CHAN1)"`.

   To measure the RMS value of the difference between channel 1 and channel 2, using CALC4 send: `CALCulate4:PATH:EXPR "RMS (CHAN1–CHAN2)"`.

3. Use the appropriate commands to return your calculation:

   - If you have specified any channels in your expression and need to acquire those channels, send the command INITiate and then the query `CALC<n>:DATA?` to first acquire your data sources and then return your calculation.

   - If you have specified any channels and have already acquired them (as when recalculating the same data) or if the data sources are all references, send the command CALC:IMMediate and then the query CALC<n>:DATA? to calculate without acquiring and to return your calculation. (Only TVS600A-models come equipped with References.)

Expression syntax is defined by the BNF description for expression syntax on page 3–34.

Expressions may use the input channels, CHAN1 through CHAN4, as data sources for use with measurements. With TVS600A-models only, expressions can also use references, REF1through REF10, and scratch-pad variables %1 through %9 (see page 3–37) as data sources.

Commands

The commands and functions to setup automated measurements follow:

Used to:	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
set the primary data source for CALC<n> (SCPI calc model only)	CALCulate<n>:FEED[1]	None	No
set the secondary data source for CALC<n> for dual-source measurements	CALCulate<n>:FEED2		No
set the WMP block used to characterize the FEED2 source to WMP<n>	CALCulate<n>:FEED2:CONText		No
force CALCulation<n> without reacquiring	CALCulate<n>:IMMediate	tktvs600_initiateCalculation	No
specify the WMList as a function to calculate	CALCulate<n>:PATH WMList	None	No
specify measurements and source explicitly (Expression calc model only)	CALCulate<n>:PATH:EXPR <expr>	tktvs600_setCalcExpression	Yes[4]
select the measurements available to the CALC block and turns them on or off	CALCulate<N>:WMList	tktvs600_setWMList	Yes
set the waveform parameters for a calculation (HIGH, LOW, GATING, and so on)	CALCulate<n>:WMParameter	tktvs600_setCalcWMPLevel tktvs600_setCalcWMPRef tktvs600_setCalcWMPEdge	No

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Functions Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application.**

[4] **Returns only results that evaluate to a single number (scalars, not vectors such as waveforms).**

Localizing Measurements

Measurements are generally taken over the entire waveform; that is, the measurement algorithm is fed the entire waveform. You can can localize a measurement to a particular area or edge on the waveform using three methods: gating, searching for edges (crossings), and segmenting.

Searching (Edge)

Some measurements require that the user specify the waveform edge to be measured. The following measurements use the edge specification: CROSs, PCRoss, and NCRoss; RTIMe and FTIMe; DELay, and COPulse.

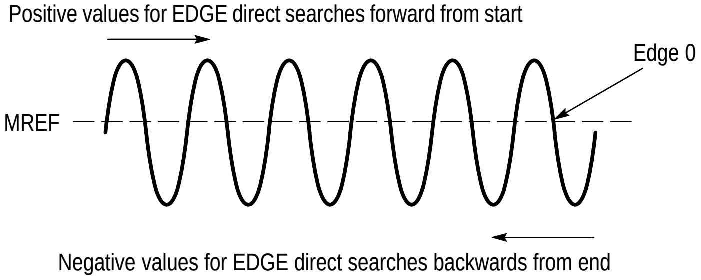

Positive values for EDGE direct searches forward from start
Edge 0
MREF
Negative values for EDGE direct searches backwards from end

Figure 3–46: Measurement localized using edge searching

Edge is specified as follows:

- The `CALCulate:WMParameter:EDGE` command sets or queries the index of the edge that the measurement is to use. It takes a negative, positive, or zero integer as an argument.
- Positive arguments search forward from the start of the waveform record; negative arguments search backward from the end.
- Zero specifies the last edge in the waveform record.
- The measurement determines implicitly the slope of the edges that it searches. FTIME and NCRoss, for example, only count negative-going edges when searching; RTIME and PCRoss count only positive. CROSs and COPulse count both negative- and positive-polarity edges when searching.

Edge searching works with both the SCPI CALC model and the Expression CALC model. Also, when using the Expression model, edge searching can also be used with the segment function (see page 3–150) to localize your measurement by first finding a specific edge and then by extracting a zone around that edge (TVS600A models only).

Gating

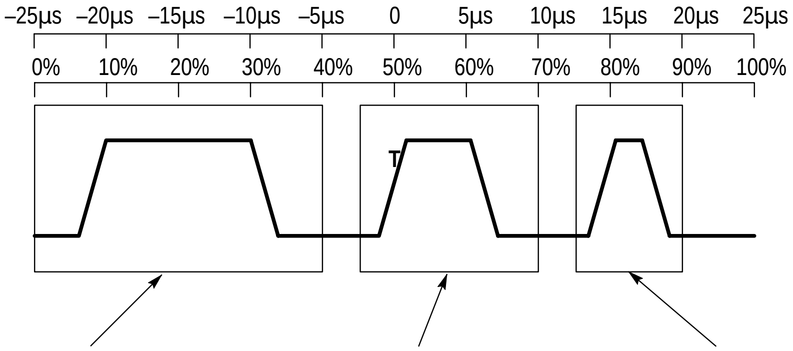
TVS600A models only. The waveform analyzer can take measurements over the entire waveform or over a user-specified measurement zone. For example, you might wish to gate your measurement to return the pulse width of any one of the three pulses shown in Figure 3–47. Gating allows you to restrict your measurement to any portion of the waveform that you want.

The gate parameters are part of the the Waveform Measurement Parameter subblock (WMP) of each CALC block (see Figure 3–8 on page 3–23). You can locate the zone using either of two methods, which are set as follows:

- `CALC:WMP:GATE:METHod ABSolute`. Requires that you set gate start and stop limits in time relative to the trigger point, which is defined as time zero (see Figure 3–47).
- `CALC:WMP:GATE:METHod RELative`. Requires that you set gate start and stop limits as a percentage of the whole waveform record, where the record starts and stops at 0% and 100%, respectively.

To use gating, send the command corresponding to the method you want to use, and then set the limits that correspond to the method that you choose, as shown in Figure 3–47. Finally, Send `CALC:WMP:GATE ON` to enable gating.

Once enabled for a given CALC block, all subsequent measurements you take with that CALC block will be gated accordingly until you turn gating off. Gating works with both SCPI and Expression Calculation models.



Method = ABSolute	CALC:WMP:GATE:START:ABS –25 us CALC:WMP:GATE:STOP:ABS –5 us	CALC:WMP:GATE:START:ABS –2.5 us CALC:WMP:GATE:STOP:ABS 10 us	CALC:WMP:GATE:START:ABS 12.5 us CALC:WMP:GATE:STOP:ABS 20 us
Method = RELative	CALC:WMP:GATE:START:REL 0 pct CALC:WMP:GATE:STOP:REL 40 pct	CALC:WMP:GATE:START:REL 45 pct CALC:WMP:GATE:STOP:REL 70 pct	CALC:WMP:GATE:START:REL 75 pct CALC:WMP:GATE:STOP:REL 90 pct

Figure 3–47: Measurement localized using gating (two methods)

Segmenting

TVS600A models only. When using the Expression calculation model to specify your measurement, you can use the segment function to extract a segment of the results of any expression that reduces to a vector. *Segment* on page 3–56 defines the segment function; this section will give a few applications.

Segment can be used with edge-searching (see page 3–149) to localize your measurement first to an specific edge and then to a zone around that edge.

The following commands measure peak-to-peak amplitude of the first (leftmost) burst shown in Figure 3–48:

```
CALC1:WMP:MREF 0.2
CALC1:WMP:EDGE 1
CALC1:PATH:EXPR "PTP(SEGM(CHAN1, CROss(CHAN1)–2us, 35us))"
INIT
```

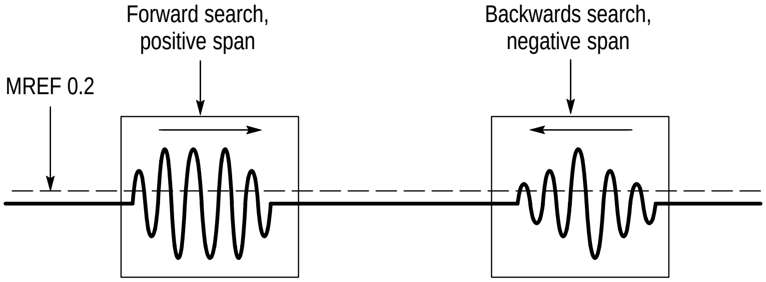


Figure 3–48: Measurement localized using edge search and the SEGMent function

The technique just described can be used with NCRoss, PCRoss, and CROSs to locate and extract almost any waveform feature you can relate to an edge. For example, the following commands would catch the third falling edge at MREF of, for example, a pulse train and a span of 100 ns after.

```
CALC1:WMP:EDGE 3
CALC1:PATH:EXPR "FTIMe(SEGM(CHAN1, NCRoss(CHAN1), 100ns))"
```

Gating and Segment Interaction

When using measurement zones and the segment function in a single CALC block, each function evaluates in the same order as the expression is parsed. When analyzing such a CALC block and its expression, apply the rules listed below as you encounter each CALC function:

- Gating is defined using the waveform measurement parameter block and, therefore, usually applies to a waveform object that is being measured. Gating limits the area of interest to a subpart or segment of a waveform.
- SEGMent is a function applied to a subpart of the expression in which it is used. That subpart must reduce to a vector and may be a waveform or other calculated data. SEGMent, in effect, creates a new waveform.
- If GATE:METHod is RELative, gating is applied relative to the start of a vector which can be the result of a segment operation.
- GATE:METHod ABSolute identifies a zone using absolute time coordinates that may or may not coincide with any given segment specified in a CALC expression.

- In cases where gating is on and a segment is measured, only the portion of the waveform common to both gate zone and segment is measured. If the two regions are not common, the result is Not a Number (NaN).
- In the case of a complex expression, with measurement functions and segment functions nested within each other, the order of evaluation is left to right for the comma and semicolon list, with the deepest nesting first. The comma is the argument separator and the semicolon in the statement separator.

Limit Testing Measurements (TVS600A Models Only)

All measurements return a single value (scalar) that can be tested against limits you define. This section shows examples of:

- Limit testing an individual measurement using comparison operators.
- Combining several measurements into a GO/NO GO test using Boolean operators.
- Controlling operation and sending notice to the controller based on test results.

CALC expressions (not SCPI calculations) must be used to do measurement-limit testing. A limit-test expression uses a comparison operator to compare a measurement against a limit. The expression then reduces to a Boolean result, which, in effect, is the outcome of a GO/NO GO test.

Figure 3–49 shows three measurements with nominal values and desired limits. The following commands will test each of the three measurements:

```
CALC1:PATH:EXPR "PWDITH(CHAN1)<> 0.9us..1.1us"
CALC2:PATH:EXPR "HIGH(CHAN1)<> 4.9..5.1"
CALC3:PATH:EXPR "LOW(CHAN1)<> –0.9..0.1"
INITIATE
CALC1:DATA?;:CALC2:DATA?;:CALC3:DATA?
```

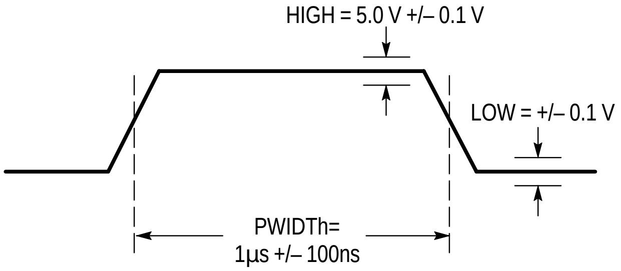


Figure 3–49: Measurements can be tested against limits

Table 3–4 *Comparison Operators* on page 3–39 lists the comparison operators that you can use to test your measurements results.

Multiple Tests

The Boolean operators can logically combine your measurement test into a single test. The following commands return a single Boolean value—FALSE if any test fails, TRUE if all pass:

```
CALC1:PATH:EXPR "!(PWIDTH(CHAN1)<> 0.9us..1.1us|HIGH(CHAN1)
                <> 4.9..5.1|LOW(CHAN1)<> –0.9..0.1)"
INITIATE
CALC1:DATA?
```

Note that “|” is the Boolean OR operator and “!” is the unary NOT operator. NOT is used to “flip” the results of the test so that if all three tests are inside limits, a TRUE (1) is returned.

Table 3–3 *Boolean Operators* on page 3–38 lists the Boolean operators that you can use with measurement tests.

Action Based on Test Result

Any expression that can reduce to a Boolean scaler value can be used with Control and Notification functions. Measurement tests can be so reduced; the following commands halt acquisition and issue an SRQ if pulse width is outside limits:

```
CALC1:PATH:EXPR "HLT(SRQ(PWDITH(CHAN1)<> 0.9us..1.1us))"
INITIATE:CONTinuous
```

The following commands issue a VXI backplane trigger, instead of an SRQ, if the pulse width is outside limits. The issue of the trigger does not interrupt acquisition and test:

```
OUTPut:TTLTrg0:STATe ON
OUTPut:TTLTrg0:POLarity NORMal
OUTPut:TTLTrg0:SOURce CALC
CALC1:PATH:EXPR "TRG(PWDITH(CHAN1)<> 0.9us..1.1us)"
INITIATE:CONTinuous
```

Control/Notification Functions on page 3–62 lists the functions you can use to take action based on measurement test results.

Measurements Tables

Table 3–39 lists all the measurements that you can use in CALC functions statements or expressions. To use them, see *Specifying Measurements* on page 3–144. For more information on how they measure parameters, see *Appendix B:Algorithms* on page B–1.

As you read the table, note that measurements defined as measured over "the entire waveform" or "the first cycle in the entire waveform" assume that measurement gating is off; if it is on, measurements are taken over the gated subsegment of the entire waveform.

Table 3–39: Measurement definitions

Measurement Command	Definition
AMPLitude	Voltage measurement. The High value minus the Low value. This result is not usually equal to the peak-to-peak value because of the way High and Low are determined. *AMPLitude = HIGH – LOW*
AREA	Voltage over time measurement. The area over the entire waveform[1] in volt-seconds. Area measured above ground is positive; area below ground is negative. Compare with CARea.
CARea (Cycle area)	Voltage over time measurement. The area over the first cycle in the entire waveform in volt-seconds. Area measured above ground is positive; area below ground is negative. Compare with AREA.
CMEan (Cycle mean)	Voltage measurement. The arithmetic mean over the first cycle in the entire waveform. Compare with MEAN.
COPulse	Time measurement. Returns the average of the times, relative to the trigger point, of the three measurement reference levels on the pulse leading edge that contains MCRoss1 and the three levels on the pulse trailing edge containing MCRoss2.
CPARea **(Cycle Positive Area)**	Amplitude (voltage) measurement. The area over the absolute value of one waveform cycle. Compare with PARea.
CRMS (Cycle RMS)	Voltage measurement. The true Root Mean Square voltage over the first cycle in the entire waveform.
CROSs	Time measurement. The time of the Nth positive- or negative-going crossing of the MREF (mid-reference) level, where N is set by CALC<n>:WMP:EDGE. Time is relative to the trigger point of the acquisition.
DELay	Time measurement. The time between the MREF crossings of a reference and a target waveform.
FREQuency	Time measurement. Frequency for the first complete cycle in the entire waveform. The reciprocal of the PERiod. Measured in Hertz (Hz).
FTIMe (Fall time)	Time measurement. Time taken for the falling edge of the selected pulse to fall from the HREF value (default = 90%) to the LREF value (default =10%).
GAIN	Ratio measurement. The AMPL measurement of a target waveform divided by the AMPL measurement of a reference waveform.

Table 3–39: Measurement definitions (cont.)

Measurement Command	Definition
HIGH	The value used as 100% whenever HREF, MREF, and LREF values are needed (as in fall time and rise time measurements). Set to absolute value or calculated as the peak value or histogram, which finds the most common value. See *HIGH and LOW* on page 3–142.
LOW	The value used as 0% whenever HREF, MREF, and LREF values are needed (as in fall time and rise time measurements). Set to absolute value or calculated as the peak value or histogram, which finds the most common value. See *HIGH and LOW* on page 3–142.
MAXimum	Voltage measurement. The most positive peak voltage measured over the entire waveform.
MEAN or DC	Voltage measurement. The arithmetic mean over the entire waveform[1]. Use MEAN or DC to select this measurement.
MID	Voltage measurement. The value halfway between MAXimum and MINimum values. $$MID = \frac{MAXimum - MINimum}{2}$$
MINimum	Voltage measurement. The most negative peak voltage measured over the entire waveform.
NCRoss	Time measurement. The time of the Nth negative-going crossing of the MREF (mid-reference) level, where N is set by CALC<n>:WMP:EDGE. Time is relative to the trigger point of the acquisition.
NDUTycycle **(Negative Duty Cycle)**	Time measurement of the first cycle in the entire waveform. The ratio of the negative pulse width to the signal period, expressed as a percentage. $$NDUTycycle = \frac{NegativeWidth}{Period} \times 100\%$$
NWIDth (Negative Width)	Time measurement of the first pulse in the entire waveform. The time between MREF (default 50%) amplitude points of a negative pulse.
OVERshoot	Amplitude (voltage) measurement. The ratio of the difference between maximum and high levels to the amplitude. $$OVERshoot = \frac{Maximum\text{–}High}{Amplitude} \times 100\%$$
PARea (Positive Area)	Amplitude (voltage) measurement. The arithmetic area over the absolute value of the entire waveform[1]. Compare with CPARea.
PDUTycycle **(Positive Duty Cycle)**	Time measurement of the first cycle in the entire waveform. The ratio of the positive pulse width to the signal period, expressed as a percentage. $$PDUTycycle = \frac{PositiveWidth}{Period} \times 100\%$$
PERiod	Time measurement. Duration of the first complete signal cycle in the entire waveform. The reciprocal of frequency. Measured in seconds.
PCRoss	Time measurement. The time of the Nth positive-going crossing of the MREF (mid-reference) level, where N is set by CALC<n>:WMP:EDGE. Time is relative to the trigger point of the acquisition.
PHAse	Time measurement. The amount one waveform leads or lags another in time. Expressed in degrees, where 360° comprise one waveform cycle.

Table 3–39: Measurement definitions (cont.)

Measurement Command	Definition
PREShoot	Amplitude (voltage) measurement. The ratio of the difference between low and high levels to the amplitude. $$PREShoot = \frac{Low–High}{Amplitude} \times 100\%$$
PTPeak (Peak to Peak)	Voltage measurement. The absolute difference between the MAXimum and MINimum amplitude in the entire waveform.
PWIDth (Positive Width)	Time measurement of the first pulse in the entire waveform. The time between MREF (default 50%) amplitude points of a positive pulse.
RMS or AC	Voltage measurement. The true Root Mean Square voltage over the entire waveform[1]. Use RMS or AC to select this measurement.
RTIMe (Rise time)	Time measurement. Time for the leading edge of the selected pulse to rise from the LREF value (default = 10%) to the HREF value (default = 90%).
σ **SDEViation (Standard deviation)**	Deviation from the MEAN measurement. The square root of the arithmetic mean of the squares of each measurement deviation from the measured MEAN. Result is in current vertical units.
TTTRig (Time between Triggers)	Time measurement. The time between the main and the delay triggers. Value returned is independent of channel number. Value returned is valid only when the delay trigger source is not set to immediate.

[1] **AREA, PARea, PARea, MEAN, and RMS measure the entire waveform record (or the gated area). If you do not want to measure the entire record, you might want to choose the cycle-based measurement instead.**

For example, an RMS measurement of a waveform record containing 5 1/2 cycles will accurately reflect the rms value of the waveform record, but not of the waveform, because it measures the nonintegral number of cycles (5 1/2). Cycle RMS will accurately characterize the rms value of the waveform, assuming the waveform is periodic and has a constant amplitude, because it measures the first cycle (if present) in the record.

Table 3–40 lists all the measurement parameters that you can use in CALC functions or expressions. To use them, see *Measurement Parameters* on page 3–141 and *Specifying Measurements* on page 3–144.

You can set these parameters using the `CALCulate:WMParameter` commands or use their default (reset) values, which are also listed in the table.

Table 3–40: Measurement parameters

Measurement Parameter	Reset Value	Description
EDGE	1	Sets the ordinal number of the edge used for CROSs, NCRoss, PCRoss, DELay, RTIMe, FTIMe, and COPulse measurements.
HIGH	0.0	Sets the high or most positive level in current vertical units. Used for amplitude and time measurements when CALC:WMP:HMEThod is set to ABSolute.
LOW	0.0	Sets the low or most minimal level in current vertical units. Used for amplitude and time measurements when CALC:WMP:LMEThod is set to ABSolute.
HREFerence	0.0	Sets the high reference or distal level in current vertical units. Used for rise time (RTIM) and fall time (FTIM) measurements when CALC:WMP:RMEThod is set to ABSolute.
HREFerence:RELative	0.9 (90%)	Sets the high reference or distal level as a percentage of the current value for AMPLitude. Used for rise time (RTIM) and fall time (FTIM) measurements when CALC:WMP:RMEThod is set to RELative.
LREFerence	0.0	Sets the low reference or proximal level in current vertical units. Used for rise time (RTIM) and fall time (FTIM) measurements when CALC:WMP:RMEThod is set to ABSolute.
LREFerence:RELative	0.1 (10%)	Sets the low reference or proximal level as a percentage of the current value for AMPLitude. Used for rise time (RTIM) and fall time (FTIM) measurements when CALC:WMP:RMEThod is set to RELative.
MREFerence	0.0	Sets the middle reference or mesial level in current vertical units. Used for timing measurements when CALC:WMP:RMEThod is set to ABSolute.
MREFerence:HYSTeresis	0.05 (5%)	Sets the hysteresis value as a percentage of the current value for AMPLitude. Reduces effects of noise on measurements.

Table 3–40: Measurement parameters (cont.)

Measurement Parameter	Reset Value	Description
MREFerence:RELative	0.5 (50%)	Sets the middle reference or mesial level as a percentage of the current value for AMPLitude. Used for timing measurements when CALC:WMP:RMEThod is set to RELative.
SLOPe	POS	Sets the direction, positive- or negative-going, of the edge(s) used for DELay measurements.

Probe Calibration

This section describes how to compensate passive probes and calibrate active voltage probes.

Compensation of Passive Probes

If you connect a passive voltage probe to an input channel, you should compensate the passive probe to match the input to which it is connected. Compensation is a manual procedure best accomplished interactively while viewing the waveform on a display.

The procedure in this section calls for using the TVS600A VXI*plug&play* Soft Front Panel (SFP) to perform probe compensation. The TVS600A VXI*plug&play* software that you installed with this product includes the SFP. (See *Software Installation* on page 1–12 for information on installing the SFP.)

Why Use?

Uncompensated passive probes can introduce low-frequency distortion to waveform measurements. Perform a probe compensation anytime you want to optimize measurement accuracy for the probe you use. It is recommended that you compensate a passive probe anytime you install the probe to a channel input for which it is not currently compensated.

To Compensate

The commands needed to compensate a passive voltage probe follow:

1. Power on the waveform analyzer and install the passive probe on the input channel to be used with it.
2. Start the TVS600A Soft Front Panel.
3. Connect the probe tip to the PROBE COMPENSATION output connector through a probe-to-BNC connector (see your probe manual for the part to use).
4. Reset the SFP and waveform analyzer: click RESET in the menu bar and choose RESET TVS600A from the RESET menu.
5. Click the ACQUISITION SINGLE/CONT. button on the SFP to toggle it to CONT. Then click the ACQUISITION RUN/ACQ’G button to toggle it to ACQ’G (acquiring).
6. Click in the VERTICAL Scale control on the SFP and select 100 mV/div in the pop-up list of selections. Click in the VERTICAL Offset control and type in `0.1`.

7. Click in the ACQUISITION Samp. Rate control on the SFP and select `1 Msa/div` (1 mSample/second) in the pop-up list of selections.

8. Click in the TRIGGER Level control on the SFP and type in `0.25`.

9. Click the ACQUISITION menu button on the SFP. In the dialog box that pops up, set the Acquisition Mode to Average and the Number of Averages to `5`.

10. Using a low-capacitance probe adjustment tool, adjust the probe compensation box on the probe (see Figure 3–50) for the best flatness of the top and bottom of waveform (see Figure 3–51).

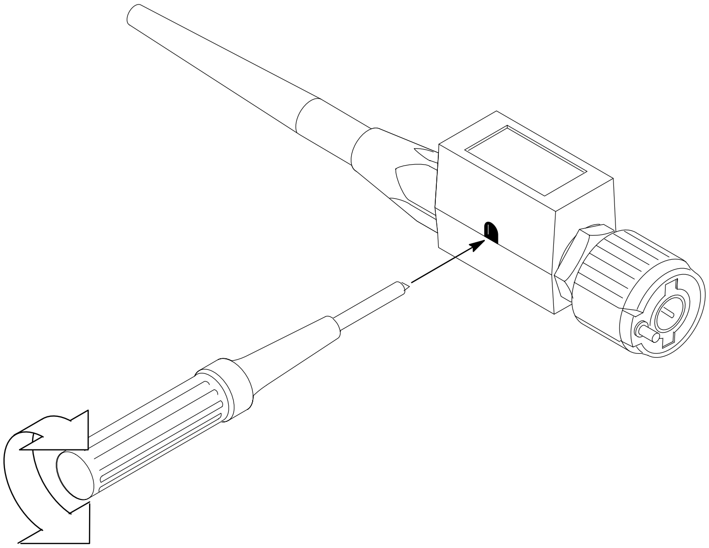


Figure 3–50: Passive probe adjustment

Figure 3–51 shows compensated and uncompensated waveforms.

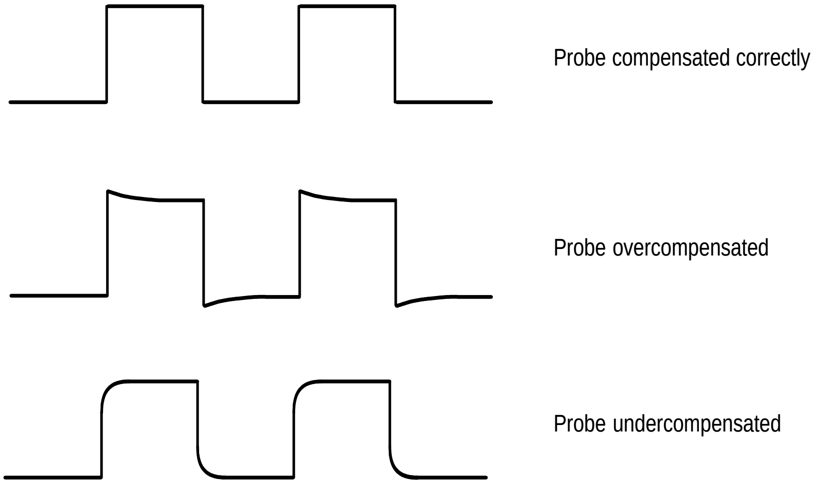


Figure 3–51: How probe compensation affects signals

Calibration of Active Probes (TVS600A Models Only)

The waveform analyzer channel input comes equipped with a Level 2 TekProbe interface. If you connect an active voltage probe to an input channel, the waveform analyzer can adjust the probe gain to match the channel to which it is connected. Only active voltage probes can be calibrated.

A successful probe calibration records the probe identification number and serial number of the probe it calibrates, along with the probe calibration data, in nonvolatile memory. The waveform analyzer will use the probe calibration as long as the identification and serial numbers match the attached probe.

Why Use?

Perform a probe calibration anytime you want to optimize measurement accuracy for the probe you use. It is recommended that you calibrate the probe anytime you install an active probe to an input channel for which it is not currently calibrated.

To Calibrate

The commands needed to calibrate an active voltage probe follow:

1. Power on the waveform analyzer and install the active probe on the input channel on which it will be used.

2. Connect the probe tip to the PROBE COMPENSATION output connector through a probe-to-BNC connector (see your probe manual for the correct connector to use).

3. Start the probe calibration: send `CALibration:PROBe<n>`, where <n> is the channel to which the probe under calibration is attached.The waveform analyzer will save its current setup and execute a probe calibration.

   This command/query sets the CAL_OPC pending flag. *WAI, *OPC, or *OPC? may be used to synchronize the command.

4. A successful calibration will not return any value, but you can check the calibration status: send the query `CALibration:PROBe[1]?`, where the return of:

   - 0 (zero) indicates probe calibration passed.
   - –1 indicates probe calibration failed.
   - –2 indicates probe calibration failed due to the probe not being connected to the calibration source.
   - –3 indicates a calibration failed because an unsupported probe or no probe was connected to the input.

5. Unsuccessful calibrations may return an execution error as follows:

   - Execution Error –240, “Hardware error”. Calibration failed. CAL:PROB? will return –1.
   - Execution Error –241, “Hardware missing”. Calibration attempted with no level 2 probe attached.
   - Execution Error –241, “Hardware missing”. Calibration attempted for PROBe3 (channel 3) and PROBe4 (channel 4) on a two-channel instrument.

A successful probe calibration records the probe-identification number and probe-serial number and the probe calibration data in nonvolatile memory. The waveform analyzer will use the probe calibration it records as long as the identification and serial numbers match the attached probe.

Commands

The commands and functions to calibrate an active voltage probe attached to the waveform analyzer follow:

Use to...	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
calibrate probe<n> and to query the results	`CALibration:PROBe<n>`	`tktvs600_calibrateProbe`	No

query the results or status of a probe cal for channel <n>	`CALibration:PROBe<n>? or`
	`CALibration:PROBe<n>:RESults?`
			No

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Functions Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application.**

Template Testing (TVS600A Only)

The TVS600A Waveform Analyzer can test waveforms against templates stored locally, in internal references. This topic covers template testing. For basic background on using the CALC engine, which is essential to understand the information that follows, read *CALCulation System Overview* on page 3–21.

Template testing is a function of the CALCulate system's expression model. In other words, there are no SCPI commands dedicated to setting up template tests; rather, you must construct your tests as an expression, using operations and functions accessed through the CALC commands.

This section begins with an overview of template tests and follows with an example procedure that demonstrates a basic template test. Methods for obtaining suitable templates and variations (expansions) on tests follow the procedure.

Overview

Template tests compare incoming waveforms (or other vectors) against a template waveform or vector. The template is an envelope that defines the limits against which a waveform is tested (see Figure 3–52).

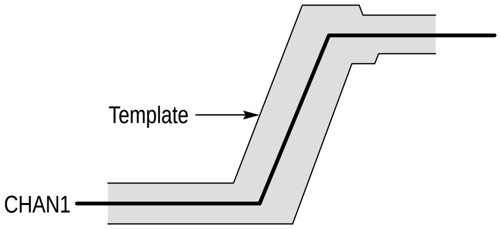


Figure 3–52: Comparing a waveform to a limit template

When a template test finds such a waveform that moves outside the template, the action taken depends on how the test is constructed. Tests return Boolean values of TRUE or FALSE; you can construct your test so that the waveform analyzer stops acquisition and waits for user (or program) input and/or notifies the controller of a pass or fail.

Comparison templates consist of max/min pairs (an envelope) for each sample in a Y–T waveform. They are characterized as follows:

- Templates may be constructed externally and imported using TRACe:DATA.

- Templates may be created using the CALC engine to envelope a waveform.
- Templates must be loaded as TVS600A reference waveforms. There are 10 locations for storing reference waveforms; they are identified as REF1 through REF10.

Basic Test

The basic template test uses the INside or OUTside comparison operators in a CALC expression that compares a waveform against an envelope, or template. Such expressions return a vector of Boolean values, where each value indicates whether a waveform data point passed or failed the comparison (see Figure 3–55 on page 3–173). *Template Test* on page 3–170 provides a procedure for constructing a basic test.

You can expand the basic test using CALC operators and functions to scale and align templates and to specify GO/NOGO tests, total and consecutive points out qualifications, and so on. See the topics *Template Operations* on page 3–176 and *Variations on Template Tests* on page 3–173 describe how to build on the basic test.

Template Sources

Template testing requires a template; here are three possible sources:

- TVS600A VXI*plug&play* Software. The software disk included with this product contains standard templates that you can import, either using SCPI commands or using the TVS600A Soft Front Panel.
- TVS600A Soft Front Panel (SFP). This application contains the Golden Waveform feature, which provides for creating a template based on an ideal (golden) waveform that you input. See the SFP online help for more information.
- Envelope function. This function can be used to envelope an ideal waveform, returning an envelope of that waveform. See *Envelope* under *Waveform Functions* on page 3–55.

Table 3–41 lists the templates included in the TVS600A VXI*plug&play* software.

Table 3–41: Supplied templates—TVS600A VXI*plug&play* software

Name	Bit Rate	Sample Interval	Vertical Range[1]	Length (points)
DS–0 Sgl	64 kBytes/s	40 ns	1.87 V	390
DS–0 Dbl	64 kBytes/s	40 ns	1.87 V	487
DS–0 Data Contra	64 kBytes/s	100 ns	1.87 V	332
DS–0 Timing	64 kBytes/s	40 ns	1.87 V	415
E1 Sym	2.048 MBytes/s	2 ns	5.6 V	244
E1 Coax	2.048 MBytes/s	2 ns	5.6 V	244
E2	8.44 MBytes/s	400 ps	4.5 V	295
E3	34.368 MBytes/s	200 ps	2.35 V	145
DS1	1.544 MBytes/s	4 ns	Scale Vertically	312
DS1A	2.048 MBytes/s	4 ns	Scale Vertically	235
DS1C	3.152 MBytes/s	2 ns	Scale Vertically	237
DS2	6.312 MBytes/s	1 ns	Scale Vertically	356
DS3	44.736 MBytes/s	200 ps	Scale Vertically	251
FDDI Halt	125MBytes/s	200 ps	Scale Vertically	399

[1] **Value for vertical range indicates a template that should not be scaled; “Scale Vertically” indicates a scalable template. See 3–169.**

Template Anatomy

Knowledge of the construction of a template is important because you must fit (align and match scale) between your template and the incoming waveform you intend to acquire and test. Two cases can occur:

- You have imported a standard template and need to set up the input channel and acquisition system to acquire the waveform to fit the template.
- You have imported or created a template and need to fit it to the incoming waveforms you wish to test.

Figure 3–53 shows a template with possible references for edge or center-of-pulse alignment.

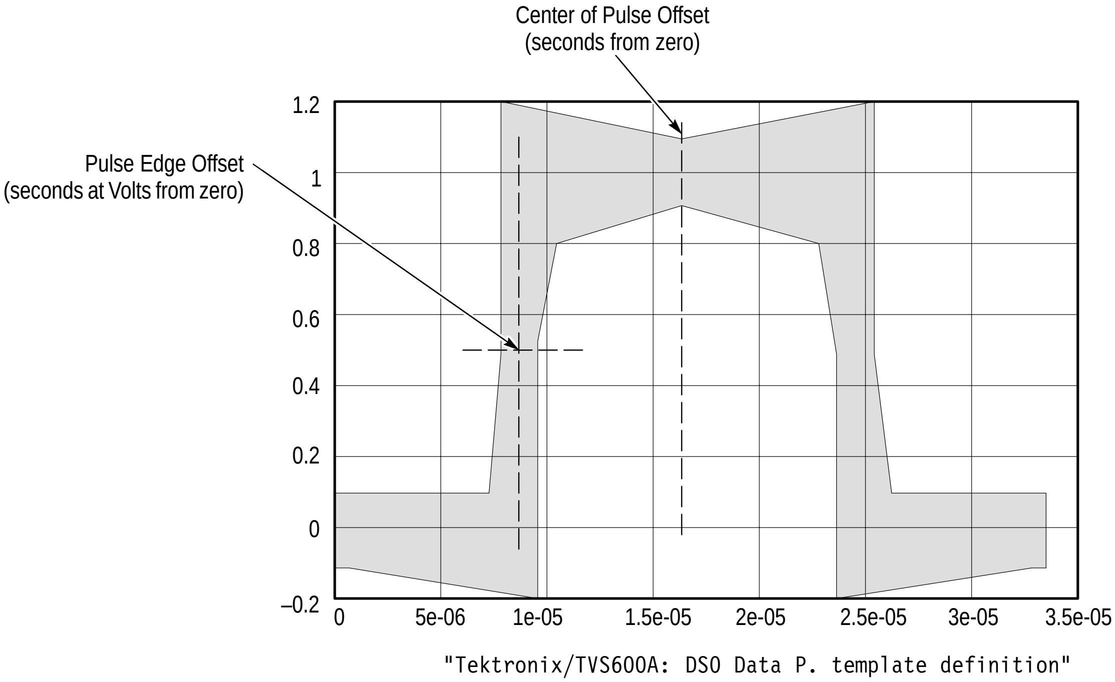
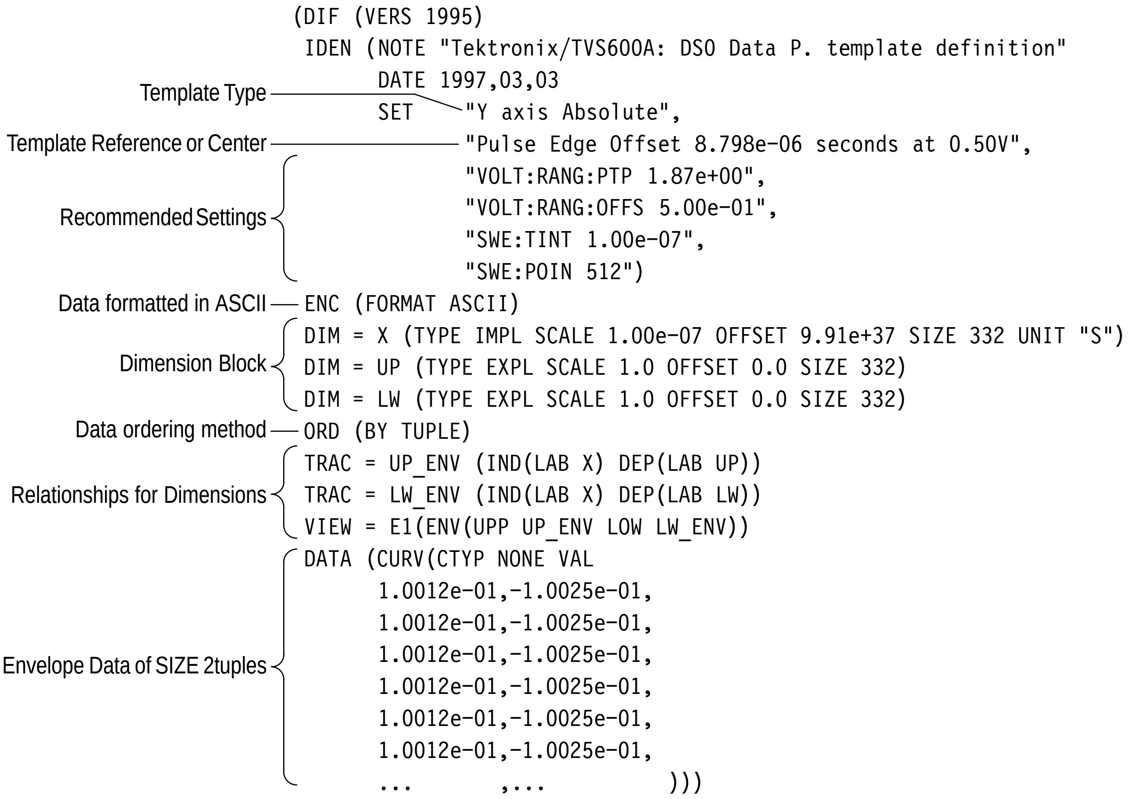


Figure 3–53: Possible template references (offset time is from zero)

Figure 3–54 shows the preamble of a standard template included in the software shipped with this product. The templates shipped with the TVS600A software are in DIF format, each in its own file. Some of these standards specify voltages that require you to fit the waveform to be tested to the template; others specify scaling of the template. You can examine the preamble to determine how to handle the fitting between template and waveform.

***NOTE**. The SETup argument (to IDENTify) shown in the Figure 3–54 is not included in the SCPI standard**.** The waveform analyzer accepts SETup and its arguments in downloads so that templates can include the setup information shown (template type, template reference, and recommended waveform-analyzer settings), needed to fit template and waveform.*

It is the responsibility of the host application or program to read the data that SETup provides (there are driver functions for reading data, see page 3–172) and effect the setup. The waveform analyzer does not use nor retain SETup arguments for application use.



```
(DIF (VERS 1995)
 IDEN (NOTE "Tektronix/TVS600A: DSO Data P. template definition"
       DATE 1997,03,03
       SET    "Y axis Absolute",
              "Pulse Edge Offset 8.798e-06 seconds at 0.50V",
              "VOLT:RANG:PTP 1.87e+00",
              "VOLT:RANG:OFFS 5.00e-01",
              "SWE:TINT 1.00e-07",
              "SWE:POIN 512")
 ENC (FORMAT ASCII)
 DIM = X (TYPE IMPL SCALE 1.00e-07 OFFSET 9.91e+37 SIZE 332 UNIT "S")
 DIM = UP (TYPE EXPL SCALE 1.0 OFFSET 0.0 SIZE 332)
 DIM = LW (TYPE EXPL SCALE 1.0 OFFSET 0.0 SIZE 332)
 ORD (BY TUPLE)
 TRAC = UP_ENV (IND(LAB X) DEP(LAB UP))
 TRAC = LW_ENV (IND(LAB X) DEP(LAB LW))
 VIEW = E1(ENV(UPP UP_ENV LOW LW_ENV))
 DATA (CURV(CTYP NONE VAL
       1.0012e-01,-1.0025e-01,
       1.0012e-01,-1.0025e-01,
       1.0012e-01,-1.0025e-01,
       1.0012e-01,-1.0025e-01,
       1.0012e-01,-1.0025e-01,
       1.0012e-01,-1.0025e-01,
       ...       ,...         )))
```

Figure 3–54: Template preamble and partial data

Template Type. Indicates whether or not the template is to be scaled vertically. *Y axis Absolute* flags a template not intended to be scaled; *Y axis Relative* flags a template meant to be scaled. To conduct a valid test as defined by the template, you must not scale templates that are Y axis absolute.

Template Reference. Locates the template reference. The reference is one of the two possible:

- Pulse Edge Offset. Indicates the offset of the first edge of the pulse from the template start at a given voltage level.
- Center of Pulse Offset. Indicates the center of the pulse from the template start; no voltage level is given.

Recommended Settings. Specifies the setup that will scale and offset the vertical and horizontal window to best acquire a waveform that will fit the template.

Data Format. Specifies the data-encoding format (ASCII for all supplied templates).

Dimension Block. Provides vertical and horizontal scaling and offset information required by the waveform analyzer to reconstruct the template properly. The waveform analyzer applies the information to scale and offset the data that follows the preamble. Horizontal template scale must be (1.0), and offset must be (0.0) or the results will be indeterminate. The number for SIZE indicates the number of 2-tuples, or max/min pairs, that the template contains.

Templates labeled *Y axis Absolute* require specific voltages at specific places on the templates; the waveform analyzer reconstructs these templates using scaled Y values at a fixed sample interval selected for compatibility with the standard being tested and the TVS600A.

Templates labeled *Y axis Relative* require fitting to the test signals; therefore, these templates are provided in normalized units as a ratio of the acquired data, where +1 defines the ideal HIGH level of the test waveform and 0 the ideal LOW. This approach allows you to scale the template using the method described in *Vertical Scaling* on page 3–178. These templates come with sample intervals that have been selected for compatibility with the standard they test and the TVS600A.

Data Ordering Method. Specifies the data-ordering method; all supplied templates order by tuples of max/min pairs (as opposed to a sequence of all of the max values followed by all of the min values).

TRACe and VIEW Blocks. Define the relationships between the dimensions in the dimension block that form envelope waveforms, including templates.

Data Block. Contains the waveform data.

Template Test

In its most basic form, a template test requires that you do the following steps:

1. Obtain and store the template.
2. Set up the channel to acquire the source of the data to be compared, adjusting its vertical and horizontal window to acquire the data so it fits the template conditions.
3. Define the comparison using INSide and OUTside operations.
4. Initiate the acquisition and fetch the test results.

The procedure that follows embodies the process just outlined. Once you have reviewed this basic procedure, explore the topics *Template Operations* and *Variations on Template Tests.* These topics describe use of CALC expressions to register test waveforms to templates and how to tailor the comparisons to match your test needs (for example, how to form GO/NOGO and points-out tests). The notification of the controller based on test results is also covered.

Why Use?

Use template testing when you want to qualify a waveform point-by-point against maximum and minimum values for each point. You can also set the test noise tolerance, so it has no more than a threshold number of points outside limits. You can test the following waveforms:

- waveforms acquired normally or in average, envelope, or peak-detect modes.
- waveforms resulting from post-acquisition processing: math, integrated or differentiated, FFTs, Boolean-vector data—any waveform processed by the CALC engine.

To Use

The following procedure demonstrates a basic template test that assumes the use of the template described previously under *Template Anatomy* on page 3–167. This template is shown in Figure 3–53 on page 3–168, and its preamble is shown in Figure 3–54 on page 3–169.

1. Download the template: use your message-based talker/listener application or write a program routine to send and feed the DIF block containing your data to the following command:

   `TRACe REF<n>,<dif_block>`, where `<n>` is the number for one of `REF1` through `REF10` and <dif_block> is contains the template you wish to download.

2. Set up to acquire the test waveform; for example, to set up channel 1 so that the acquired waveform fits the template described in Figure 3–54 on page 3–169, send the following commands:

```
Vertical window set to fit ideal test waveform  { VOLTage1:DC:RANGe:PTPeak 1.87e+00
                                                { VOLTage1:DC:RANGe:OFFSet 5.00e-01
Record length set to at least as long as template { SWEep:TINTerval 1.00e-07
                                                { SWEep:POINts 512)
Trigger offset to template reference            { SWEep:OFFSet:TIME -8.798e-06
                                                { SWEep:OREFerence:LOCation MINimum
Trigger level and slope set to match            { TRIGger:SLOPe POSitive
                          template reference    { TRIGger:LEVel 0.50V
```

   Note that `SWEep:OFFSet:TIME` must be sent as a negative quantity (it sets pretrigger amount) even though it appears positive in the template preamble.

3. Specify a CALC expression of the form (or that reduces to the form) `<expr> OUTside <envelope>`; for example, assuming an envelope stored in REF1, send: `CALCulate1:PATH:EXPR "CHAN1 OUTside REF1"`

- Test waveforms may be the input channels, CHAN1 – CHAN4, the references, REF1 through REF10, and scratch-pad variables, %1 through %9 (page 3–37).
- Templates must be envelopes stored in one of the references, REF1–REF10.
- Expressions may be much more complex as long as the reduce to a test vector that registers (fits) the envelope template.
- Expression evaluations can depend on proper setup of the waveform parameter list block (WMP). For general instruction on using CALC expressions, see *Calculation System Overview* on page 3–21.

4. Acquire your test waveform and return the test results; send `INITiate;:CALC:DATA?`

A vector of Boolean TRUE/FALSE (1/0) test values will return; where each point in the test waveform outside the REF1 template is a TRUE (1) value.

The procedure just listed depends on the template preamble for waveform-analyzer setup information. Setting the waveform analyzer as suggested in the template preamble captures each test waveform so that the 0.5-volt level of its rising edge is offset from the start of the waveform record an amount that matches that of the template reference point from the start of the template.

Commands

The commands and functions to set up a template test follow:

Used to:	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
download the template for testing against	`TRACe REF<N>,<dif_block>`	`tktvs600_setRefTrace`	Yes
test the vector defined by <expr> against template in <REFn>	`CALC<n>:PATH:EXPR "<expr> OUTside <REFn>`	`tktvs600_doBasicTemplateTest`	Yes
force calculation reacquiring	`CALC<n>:IMMediate`	`tktvs600_initiateCalculation`	Yes
set waveform parameters used to characterize data sources	`CALC<n>:WMParameter`	`tktvs600_setWMList`	Yes
reduce <test> to a Boolean outcome	`CALC<n>:PATH:EXPR "BAT(<test>) "`	`tktvs600_setCalcExpression`	Yes
extract test waveform segments that match up with test templates	`CALC<n>:PATH:EXPR "SEGMent (<start>, <span> <test>)"`	`tktvs600_setCalcExpression`	Yes

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Functions Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application.**

Variations on Template Tests

This section summarizes the basic template test as outlined in the basic procedure under *Template Test* on page 3–170; the topics that follow describe variations that increase the capability of the basic test.

Basic Test

The basic test is to use the INSide and OUTside operators to perform logical (Boolean) point-by-point comparison to an envelope of max/min values that returns a vector reflecting the results. For example, consider Figure 3–55. The following commands test channel 1 against the reference 1 envelope:

```
CALC1:PATH:EXPR "CHAN1 OUTside REF1"
INIT
CALC1:DATA?
```

The vector shown at the bottom of Figure 3–55 would return.

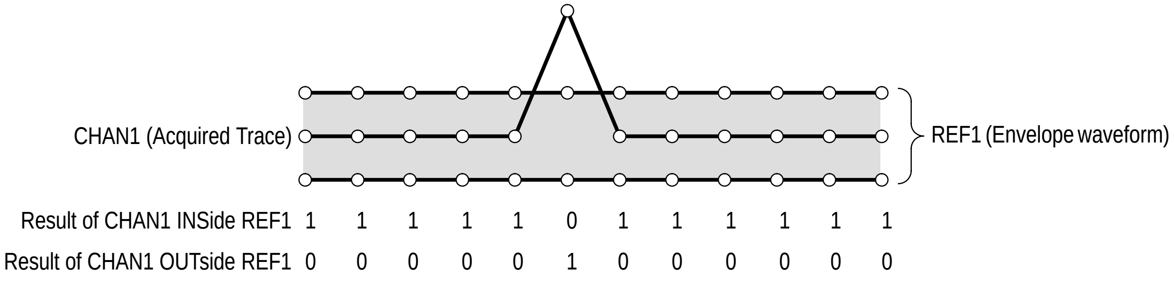

CHAN1 (Acquired Trace)
REF1 (Envelope waveform)
Result of CHAN1 INSide REF1 1 1 1 1 1 0 1 1 1 1 1 1
Result of CHAN1 OUTside REF1 0 0 0 0 0 1 0 0 0 0 0 0

Figure 3–55: Basic test returns a vector of point-for-point results

GO/NO GO Testing

To obtain pass/fail results from your test, you can use the BAT (Boolean Aperture Test) function to collapse the Boolean vector delivered by the basic test just described to a Boolean scalar. The BAT function returns TRUE (1) if the summation over the moving <aperture> number of consecutive values ever equals or exceeds the specified <threshold> value. The BAT-function form is `BAT (<expression>,<aperture>,<threshold>)` and is fully described on page 3–61.

To use BAT over the entire record, looking for any failures, omit the width of <aperture> and the number of points for <threshold> in the form above. The following commands return the Boolean scalar-value TRUE, rather than the vector shown at the bottom of Figure 3–55 for the basic test example:

```
CALC1:PATH:EXPR "BAT(CHAN1 OUTside REF1)"
INIT
CALC1:DATA?
```

Qualification by Consecutive Points Out

To obtain pass/fail results based on how many consecutive points violate the template, you can add aperture and threshold values to the GO/NOGO test just described. The following commands returns TRUE (1) if any sequence of four or more consecutive points are outside the template, as is the case shown in Figure 3–56:

```
CALC1:PATH:EXPR "BAT (CHAN1 OUTside REF1,4,4)"
INIT
CALC1:DATA?
```

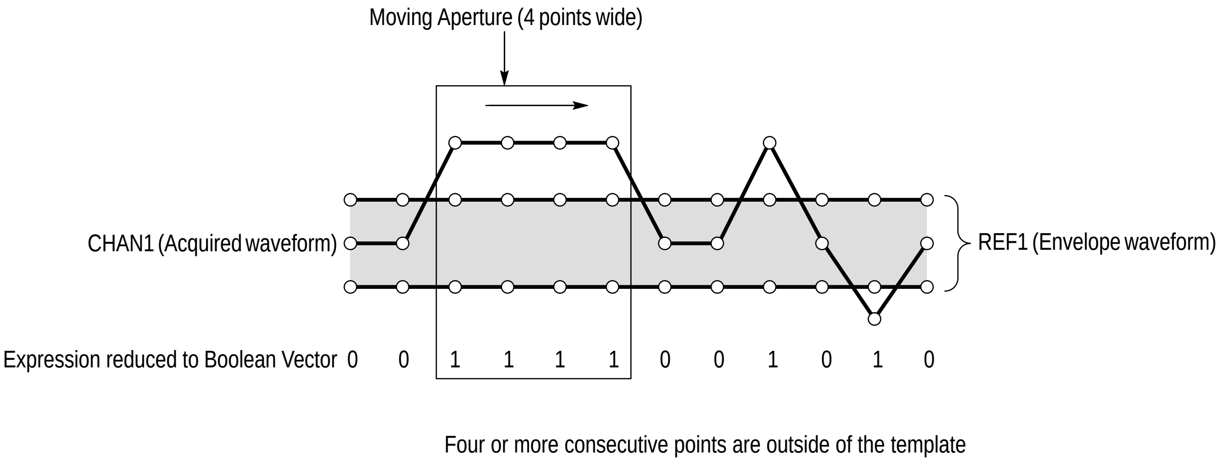

Moving Aperture (4 points wide)
CHAN1 (Acquired waveform)
REF1 (Envelope waveform)
Expression reduced to Boolean Vector 0 0 1 1 1 1 0 0 1 0 1 0
Four or more consecutive points are outside of the template

Figure 3–56: BAT tests for consecutive points out

Qualification by Total Points Out

To obtain pass/fail results based on the total number, rather than the consecutive, number of points that violate the template, you can add aperture and threshold values to the GO/NOGO test just described. The following commands return TRUE (1) if three or more points in any order are outside the template, as is the case for Figure 3–57:

```
CALC1:PATH:EXPR "BAT (CHAN1 OUTside REF1,XSIZE(CHAN1),3)"
INIT
CALC1:DATA?
```

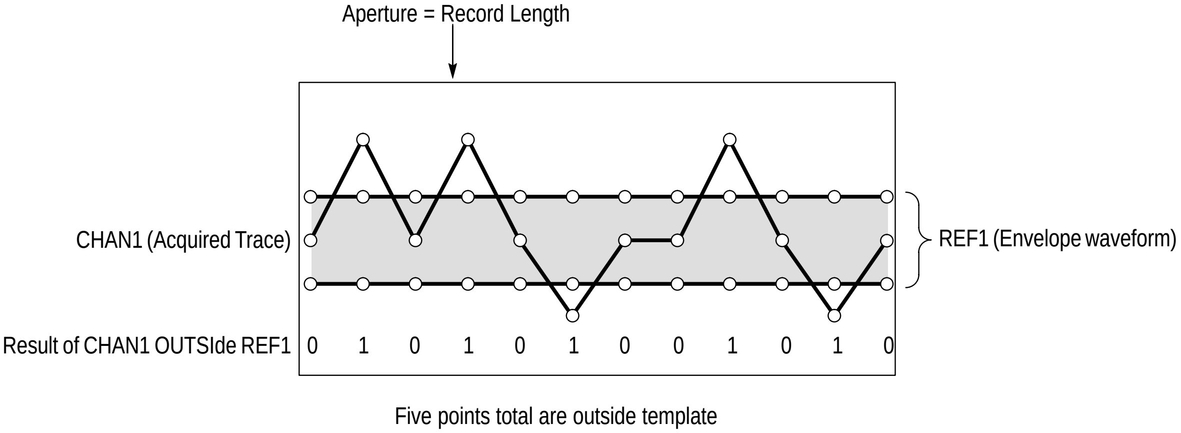


Figure 3–57: BAT tests for total points out

Note the use of the XSIZE function to set the aperture to the entire record length of channel 1. If you include a value for <threshold>, you must include a value for <aperture>; `BAT (<expr>,,2)` is not valid. Use `XSIZE (CHAN<n>)` or `100PCT` when you want the aperture to default to the waveform record length, but still want to specify a value for threshold.

Action Based on Test Result

Any expression that can reduce to a Boolean scalar value can be used with the control and notification functions. Template tests that reduce to a Boolean value using the BAT function certainly qualify; the following commands halt acquisition and issue an SRQ if four or more consecutive points are outside the limits of the REF1 template:

```
CALC1:PATH:EXPR "HLT(SRQ(BAT(CHAN1 OUTside REF1,4,4))"
INITiate:COUNt 45
```

The following commands issue a VXI backplane trigger instead if the waveform is outside limits:

```
OUTPut:TTLTrg0:STATe ON
OUTPut:TTLTrg0:POLarity NORMal
OUTPut:TTLTrg0:SOURce CALC
CALC1:PATH:EXPR "HLT(TRG(BAT(CHAN1 OUTside REF1,4,4))"
INITiate:CONTinuous
```

Control/Notification Functions on page 3–62 lists the functions you can use to take action based on measurement test results.

Template Operations

This section describes operations and methods that register, or fit, your template to waveforms under test. Like the method illustrated in the *Template Test* on page 3–170, these methods require that you know key characteristics about the template you want to use, either because you designed them or because you ascertained what they are.

The *Template-Test* procedure sets up the waveform analyzer based on the offset and level of a key reference point in the template, and the template length and its amplitude. The setup that results acquires the waveform record so it matches the length, amplitude, and reference point of the template. In that way, the waveform is acquired so it aligns with the template (same record length, same horizontal offset from start of record to point of reference, same nominal amplitudes).

The methods in this section take a different approach: these methods assume you will acquire your waveforms so that their records contain the feature to be tested, but their record length may exceed the template length and that their amplitude may not fit within the template. The methods use edge alignment, center alignment, and SEGMent to cut and fit the waveform to the template or vertical scaling to match the template to the waveform.

Edge Alignment

Properly aligning your incoming waveforms to the template that tests them requires that you know where a key reference point in the template is located. For example, consider the progression shown in Figure 3–58. The goal is to extract the segment of the acquired waveform that fits the template:

- Determine the offset template reference from the template start in points (see Figure 3–58, center)
- Use the PCRoss function to find the location on the test waveform that corresponds to the template reference. Then back up, towards the start of the waveform, a value equal to the template offset times the test-waveform sample interval (see Figure 3–58, left)
- Use the SEGMent function to extract a segment from the waveform equal to the template length in points, starting from the point backed up (see Figure 3–58, right)

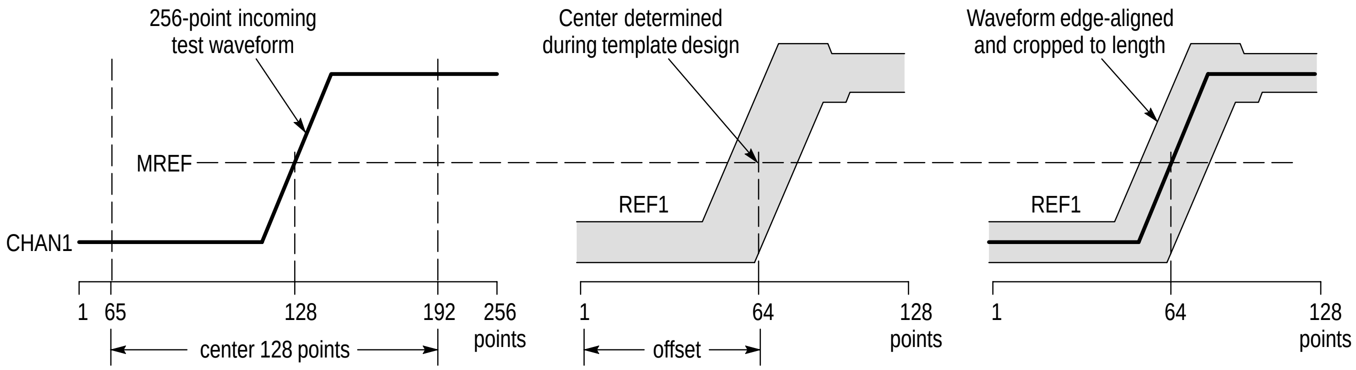

256-point incoming test waveform
Center determined during template design
Waveform edge-aligned and cropped to length
MREF
CHAN1
REF1
REF1
1
65
128
192
256 points
center 128 points
1
64
128 points
offset
1
64
128 points

Figure 3–58: Edge alignment, waveform to template

The SEGMent function, defined on page 3–56, is used as just described to extract the segment in the following example. In the case shown in Figure 3–58, you know the template is 128 points long with a center reference at point 64 at midlevel, as is shown in Figure 3–58, center. Based on the 63-point offset (64 –1 points), the following expression uses PCRoss to locate the reference on the test waveform, backs up a time equal to 63 points, and uses SEGMent to cut out 63 points before and 64 points after the reference point, as shown in Figure 3–58, left.

```
(SEGMent (CHAN1, PCRross(CHAN1)–63*XSCale(CHAN1), 128) OUTside REF1)
```

The resulting segment of channel 1 properly aligns within the REF1 template; the SEGMent expression could then be wrapped into a test expression, using BAT, as follows:

```
(BAT (SEGM (CHAN1, PCRross(CHAN1)–63*XSCale(CHAN1), 128) OUT REF1))
```

Center Alignment

A method similar to edge alignment can be used to align a template designed around template center as shown in Figure 3–59, center. Center Alignment uses the center-of-pulse (COP) function to locate the point on the test waveform that matches the reference point on the template. The test with SEGMent is as follows:

```
(BAT (SEGM (CHAN1, COP (CHAN1)–63*XSCale(CHAN1), 128) OUT REF1))
```

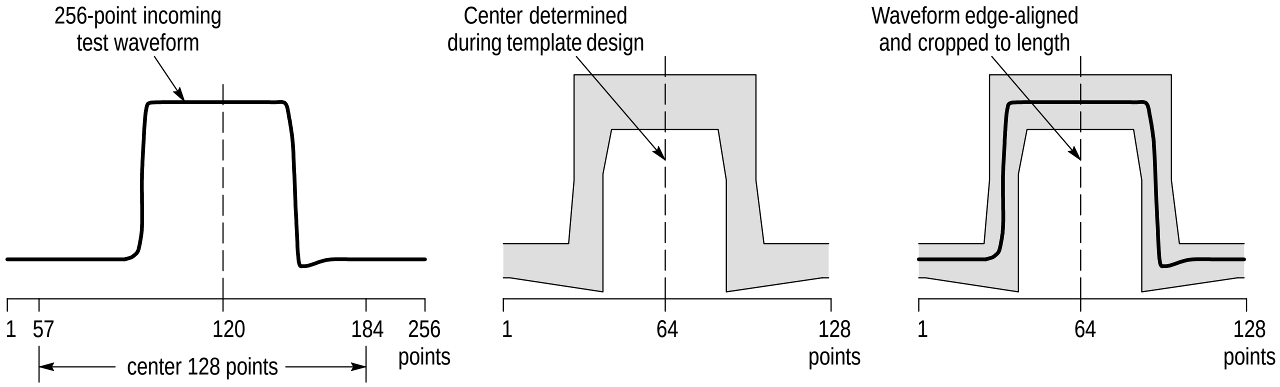

256-point incoming test waveform
1
57
120
184
256 points
center 128 points
Center determined during template design
1
64
128 points
Waveform edge-aligned and cropped to length
1
64
128 points

Figure 3–59: Center-of-pulse alignment, waveform to template

Vertical Scaling

You also can scale templates to match incoming waveforms. The following example makes the following assumptions:

- A template is specified in normalized coordinates such that the nominal low value is 0.0 and the nominal high value is 1.0, with a +/– 5% tolerance around these nominal values (see Figure 3–60, center).
- Incoming test waveforms will have ideal values of HIGH = 1.6 and ideal low values of LOW = 0.8, (see Figure 3–60, left).

The following expression scales such a template, stored in REF1, and stores the scaled version in REF2 (see Figure 3–60, right).

```
CALC1:PATH:EXPR "REF2:=REF1*AMPL(CHAN1)+LOW(CHAN1)"
CALC1:IMMediate?
```

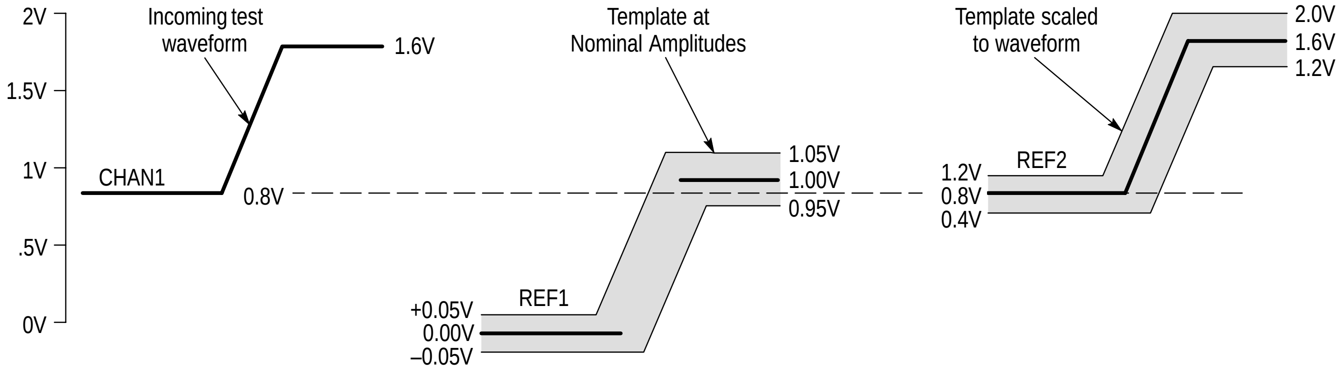


Figure 3–60: Vertical scaling, template to waveform

Note that the expression scales REF1 so that REF2 nominal values and template limits are properly scaled:

- REF2 LOW nominal = 0.00(1.6 – 0.8) + 0.8 = 0.8
- REF2 HIGH nominal = 1.00(1.6 – 0.8) + 0.8 = 1.6
- Template limits: +/–5% x (1.6 – 0.8) = +/–0.4 V

***NOTE**. In cases where it is preferable to scale the waveform to match nominal template values of LOW = 0 to HIGH =1, the following expression provides the conversion:*

```
CALC1:PATH:EXPR "REF2:=(CHAN1–LOW(CHAN1))/AMPL(CHAN1)"
```

Triggering Overview

To properly acquire data, that is, to use the waveform analyzer to sample a signal and digitize it into a waveform record that you want to measure or otherwise process, you need to set up its trigger conditions. Toward that end, this overview provides background the basic elements of TVS600A triggering: type, source, coupling, holdoff, mode, and so on.

After covering these basics, see *Trigger Types*, on page 3–193 for details on how to trigger using the various trigger types provided: edge, pulse, (all TVS models) and logic, setup & hold, and transition (TVS600A models only).

This section concludes with details about using the *Delayed Runs After* and *Delayed trigger* modes to delay the acquisition of a waveform relative to a trigger event. (See *Delayed Acquisitions* on page 3–189.)

Triggering

Given that INITiate and ARM conditions are first met, triggering determines where in the data stream (the input signal) the waveform analyzer acquires a waveform record. When not triggered, the waveform analyzer does not acquire data (normal trigger mode) or, in the case of repetitive acquisitions in automatic trigger mode, the waveform record is acquired, but at different places in the data stream. See Figure 3–61; trigger mode details are on page 3–188.

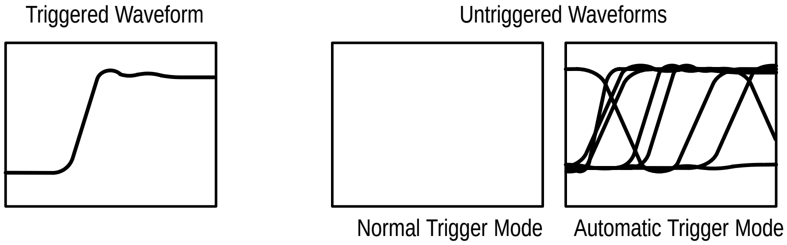


Figure 3–61: Triggered versus untriggered acquisitions

Initiate/Arm/Trigger Cycle

INITiate and ARM conditions must be met before triggering can occur. Figure 3–62 and 3–63 show two views of the waveform-analyzer arm/trigger cycle that all acquisitions follow. (Figure 3–62 illustrates that the cycle follows the SCPI standard for triggering; Figure 3–63 better shows the signal sources controlling the cycle.) Note the following points:

- The waveform analyzer idles unless INITiate is on.

- With INITiate on, ARM event detection must be satisfied (usually by selecting IMMediate, effectively bypassing the ARM function) or the waveform analyzer idles.
- With ARM event detection satisfied and no delay or B triggering wanted, the A Trigger system can be configured to trigger immediately upon satisfying the A triggering requirements. To so configure, TRIGger:B:SOURce must be set to IMMediate and TRIGger:A:Delay or TRIGger:B:Delay set to zero, effectively bypassing these functions.

To use the DELay and TRIGger:B capabilities shown in Figure 3–62, see *Delayed Acquisitions* on page 3–189.

The Trigger Event

The trigger event establishes the time-zero point in the waveform record. All points in the record are located in time with respect to that point. The waveform analyzer continuously samples the signal, and shifts the resulting data stream through internal registers, retaining enough sample points to fill the pretrigger portion of the waveform record (that part of the waveform that occurs *before* the triggering event in the waveform record). See Figure 3–4 on page 3–6.

When a trigger event occurs, the waveform analyzer continues acquiring samples to build the posttrigger portion of the waveform record (occurs *after* the trigger event). See Figure 3–4 on page 3–6. Once it recognizes a trigger, the waveform analyzer ignores triggers until the acquisition completes and holdoff expires.

The trigger system parameters, such as trigger level and slope, determine at what point on a waveform the trigger event occurs.

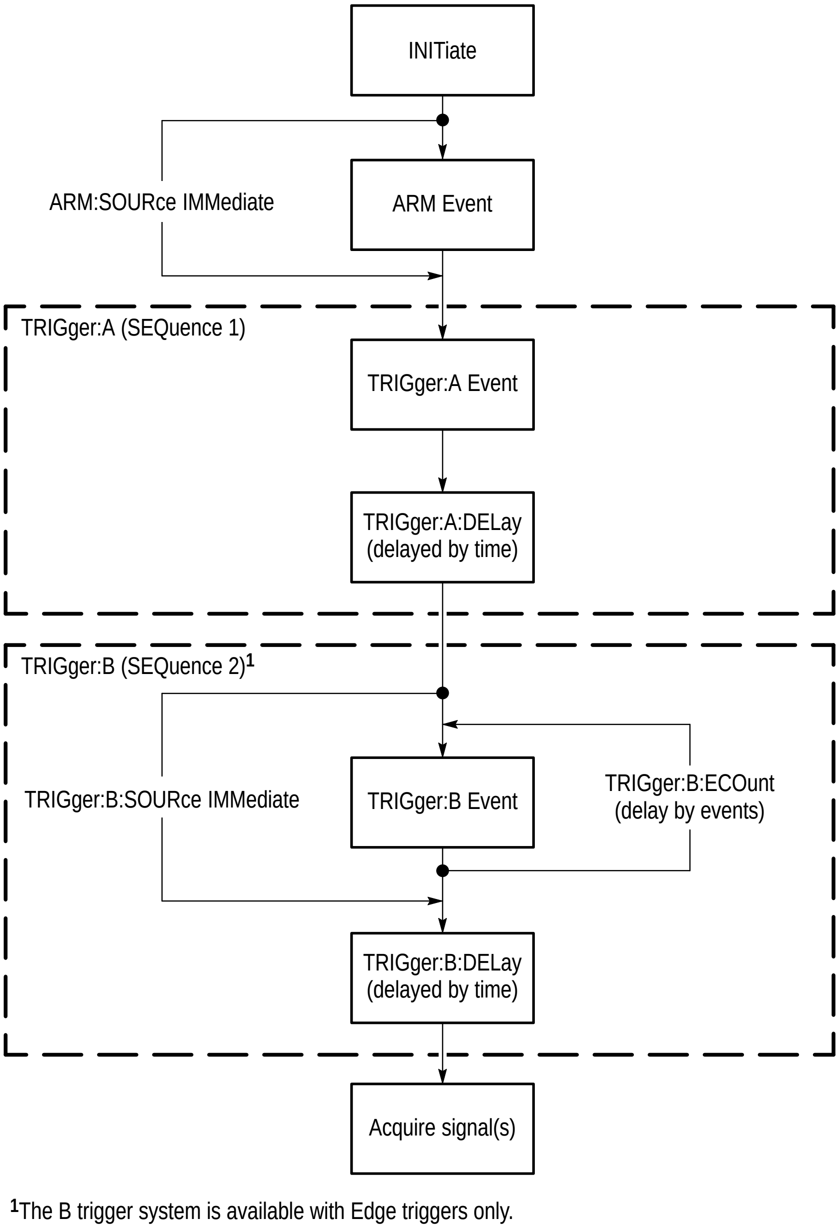
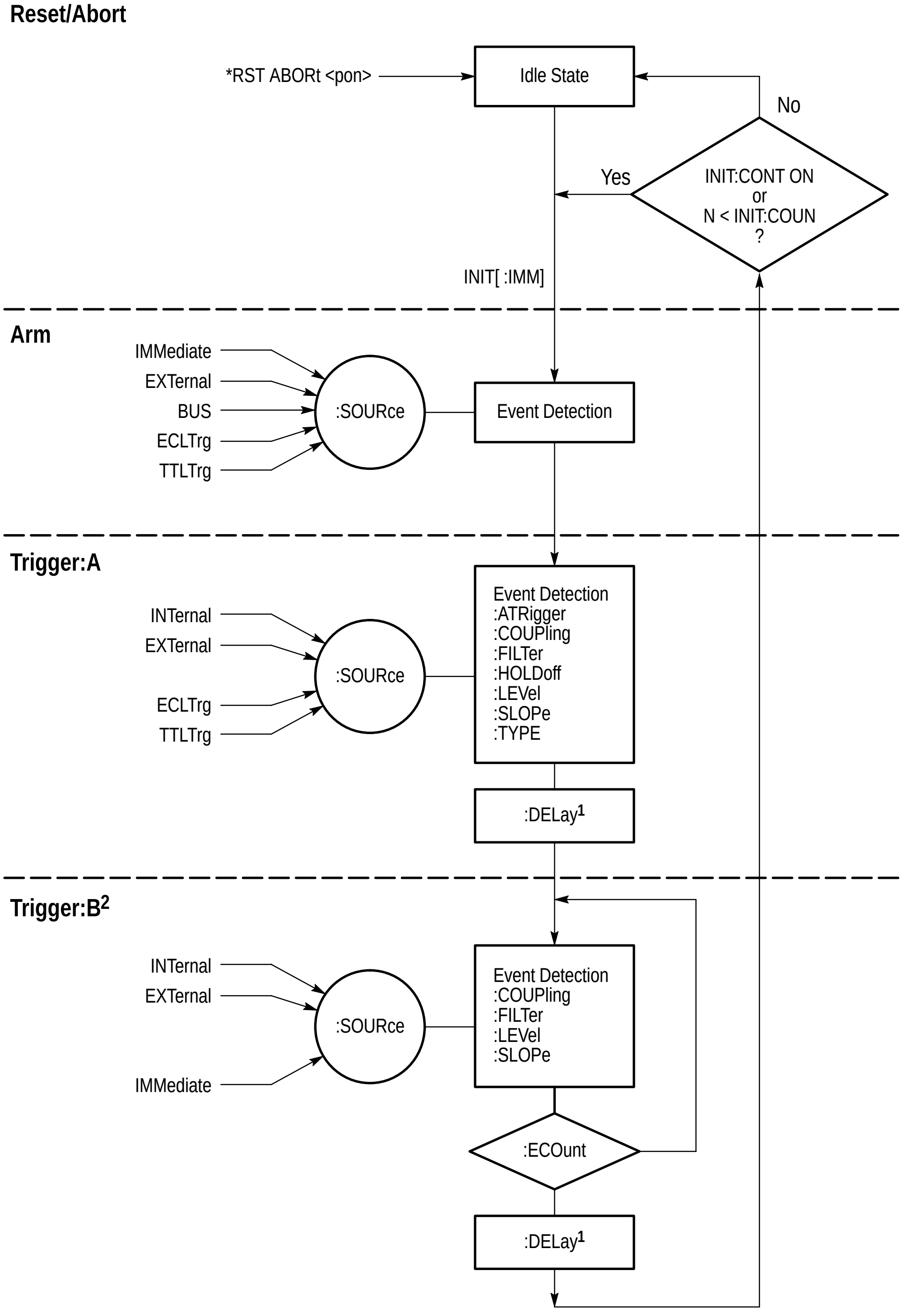

INITiate
ARM:SOURce IMMediate
ARM Event
TRIGger:A (SEQuence 1)
TRIGger:A Event
TRIGger:A:DELay (delayed by time)
TRIGger:B (SEQuence 2)1
TRIGger:B:SOURce IMMediate
TRIGger:B Event
TRIGger:B:ECOunt (delay by events)
TRIGger:B:DELay (delayed by time)
Acquire signal(s)
1The B trigger system is available with Edge triggers only.

Figure 3–62: The initiate/arm/trigger cycle (SCPI model)



[1] DELay can be set for either the A or B trigger systems, but not for both simultaneously.

[2] The B trigger system is available with Edge triggers only.

Figure 3–63: The initiate/arm/trigger cycle (sources model)

Trigger Types

The waveform analyzer provides various trigger types, selectable with the SCPI command `TRIGger:TYPE`. A brief definition of each type follows. All models include these two trigger types:

- `EDGE` is the default trigger type. An edge trigger event occurs when the trigger source passes through a specified voltage level in the specified direction (the trigger slope). (Level and slope are trigger parameters.)
- `PULSe` is a special-purpose trigger based on the shape and duration of pulses on the trigger signal. The two classes of pulse triggers are glitch and width; TVS600A models only add a third class, timeout. Pulse triggering uses trigger parameters that are independent of those used with edge triggering.

The TVS600A models add the following trigger types:

- `LOGic` is a special-purpose trigger based on the logic pattern found at the input channels of the waveform analyzer. The two classes of logic triggers are pattern and state. Logic triggering uses parameters that are independent of those used with edge triggering.
- `SHOLdtime` (Setup & Hold) is a special-purpose trigger based on a data state change within user-specified setup and hold times relative to a clock. Setup & hold triggering uses parameters that are independent of those used with edge triggering.
- `TRANsition` is a special-purpose trigger based on how a pulse transitions from one level to another. The two classes of transition triggers are runt and slew rate. Transition triggering uses parameters that are independent of those used with edge triggering.

The basic A-trigger-system parameters for delay time, holdoff, and auto trigger (trigger mode) are shared by all trigger types. Otherwise, each trigger type may be programmed independently and will not interact with any other.

Only the selected trigger type is active at any one time: the waveform analyzer uses the selected trigger type to configure the underlying triggering hardware and ignores settings for all other trigger types.

Trigger Sources

The trigger source is the signal monitored for a trigger event. Table 3–42 lists the possible trigger sources and their compatibility with each trigger type.

Table 3–42: Trigger sources vs. trigger types

		Trigger Type				
Trigger Source	Description	Edge TRIG:A	EDGE TRIG:B	TRIG :PULSe	TRIG :LOGIC	TRIG :TRAN
Input channels CH1, CH2, CH3 CH4 (depending on model)	■ Provides triggering from signals connected to the input channels ■ Parameters include coupling (AC/DC), level, slope, and trigger filters ■ Trigger filters (LPASs, HPASs, and NREJ) available to eliminate low or high frequency elements ■ The channel selected as a trigger source need not be acquired	✓	✓	✓	✓	✓
External Trigger	■ Provides triggering on the signal at the input connector EXTERNAL TRIGGER INPUT on the waveform analyzer front panel ■ Parameters include slope and level ■ Provides DC coupling and a maximum input range of ±1 V	✓	✓			
VXIbus Triggers	■ Provides triggering on digital trigger signals from the VXI bus (are available to all VXI modules) ■ Provides triggering off any of eight TTL logic trigger lines (TTLTrg0 – TTLTrg7) and of two ECL logic trigger lines (ECLTrg0 and ECLTrg1) ■ The TTLTrg lines have a 12.5-Mhz clock speed and the ECLTrg lines have a 62.5 Mhz clock speed. ■ The waveform analyzer can source the TTLTrg lines with any valid trigger source. Refer to the OUTPut:TTLTrg commands in your *Command Reference*.	✓				
IMMediate	Effectively bypasses the TRIGger:B event detector.		✓			

Trigger Coupling

Trigger coupling determines which frequency components of an analog trigger signal are passed to the trigger system. The VXIbus trigger sources are digital signals and do not provide coupling selections. The External trigger provides DC coupling only.

The TRIGger:A:COUPling and TRIGger:B:COUPling commands can set basic AC or DC coupling; TRIGger:A:FILTer and TRIGger:B:FILTer can effect more sophisticated filtering. The coupling/filtering selections are as follows:

- DC passes all of the input signal. In other words, it passes both AC and DC components to the trigger circuit.
- AC passes only the alternating components of an input signal. It removes the DC component from the trigger signal.

- HFReject removes the high frequency portion of the triggering signal. Only the low frequency components pass on to the triggering system. High frequency rejection attenuates signals above 50 kHz.
- LFReject removes the low frequency portion of the triggering signal. Only the high frequency components pass on to the triggering system. Low frequency rejection attenuates signals below 50 kHz.
- ACNReject provides noise rejection while rejecting the DC portion of the signal.
- DCNReject provides noise rejection but passes all other portions of the signal.

Slope and Level

The slope and level settings determine the analog parameters for the trigger point for edge triggering. Figure 3–64 shows the commands that control each parameter. The trigger level must be within the signal range to cause a trigger event. The slope setting allows you to capture the rising or falling edge of a signal.

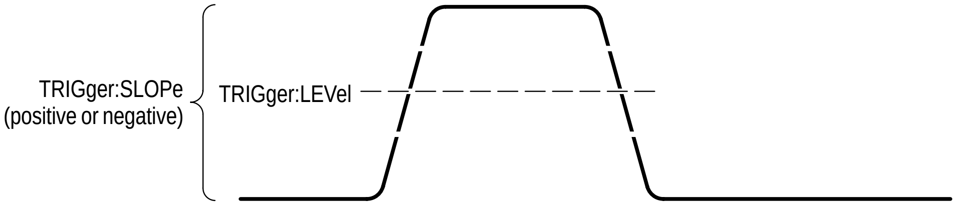


Figure 3–64: Slope and level define the trigger event

Trigger Position

The trigger position defines where on the waveform record the trigger occurs. It lets you properly align and measure data within records. The part of the record that occurs before the trigger is the pretrigger portion. The part that occurs after the trigger is the posttrigger portion.

For information on setting the trigger position, refer to *Record Offset* on page 3–111.

Trigger Holdoff

When the waveform analyzer recognizes a trigger event, it disables the trigger system until acquisition and trigger holdoff complete. Trigger holdoff starts when the trigger event occurs as shown in Figure 3–65. As shown in the illustration, you can set the holdoff time to skip signal pulses that would cause false triggering. Without trigger holdoff, pulses two, three, and four would be valid trigger events assuming sufficient time for acquisition and rearming. The desired trigger event is the first rising edge of each group of pulses. Note that the waveform record has 50% of its samples before the trigger point.

The holdoff range is from 250 ns (minimum) to 12 seconds (maximum).

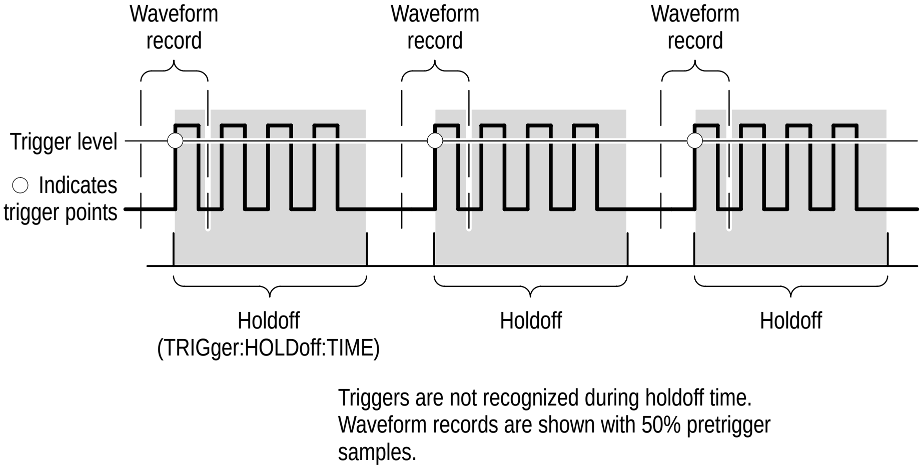

Waveform record
Waveform record
Waveform record
Trigger level
○ Indicates trigger points
Holdoff (TRIGger:HOLDoff:TIME)
Holdoff
Holdoff
Triggers are not recognized during holdoff time. Waveform records are shown with 50% pretrigger samples.

Figure 3–65: Trigger holdoff time ensures valid triggering

Trigger Mode

The trigger mode determines how the waveform analyzer behaves in the absence of a trigger event. The waveform analyzer provides two trigger modes, normal and automatic.

- Normal. This trigger mode enables the waveform analyzer to acquire a waveform only when it is triggered. If no trigger occurs, the waveform analyzer does not acquire a waveform.
- Automatic. This trigger mode enables the waveform analyzer to acquire a waveform even if a trigger event does not occur. Auto mode uses a 500 ms timer that starts after INIT starts acquisition. If the trigger circuit does not detect a trigger event before the time expires, the waveform analyzer forces a trigger.

Use the command TRIGger:ATRigger to enable automatic triggering

Trigger Status Lights

To quickly determine trigger status, check the two status lights, TRIG’D and ARM’D, on the instrument front panel.

- When ARM ’D is lighted, it means the waveform analyzer can accept a valid trigger event and the waveform analyzer is waiting for that event to occur.
- When TRIG’D is lighted, it means the waveform analyzer has recognized a valid trigger and is filling the posttrigger portion of the waveform.
- When ARM’D and TRIG’D are both off, the digitizer is stopped.

Delayed Acquisitions

The waveform analyzer has two modes for delaying the acquisition of waveforms, delayed runs after main and delayed triggerable:

Mode	Trigger System Used	Provides:
Delayed Runs After Main	TRIGger:A	Time delay[1]
Delayed Triggerable	TRIGger:B	Events and/or time delay[1]

[1] **Specify Time Delay for TRIGger:A or :B systems, but not for both at the same time.**

Delayed Runs After Main

The time delay specifies a period to wait after the trigger event to start acquisition. Delayed runs after main mode looks for a TRIGger:A event, then waits a user-defined time, and then acquires the waveform. See Figure 3–66.

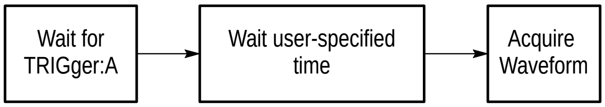


Figure 3–66: Delayed runs after main

Delayed Triggerable

Delayed triggerable mode looks for a TRIGger:A event and then makes one of the three types of delayed triggerable acquisitions. Figure 3–67 shows the three delayed-triggerable types; Figure 3–68 shows how both delayed-runs-after and delayed-triggerable modes work.

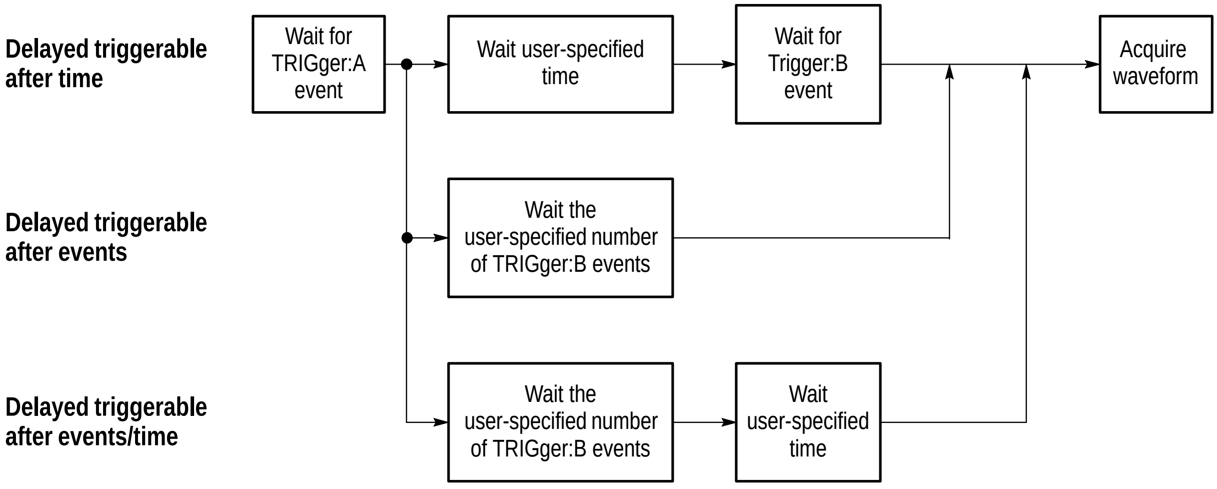

Delayed triggerable after time
Delayed triggerable after events
Delayed triggerable after events/time
Wait for TRIGger:A event
Wait user-specified time
Wait for Trigger:B event
Acquire waveform
Wait the user-specified number of TRIGger:B events
Wait the user-specified number of TRIGger:B events
Wait user-specified time

Figure 3–67: Delayed triggerable types

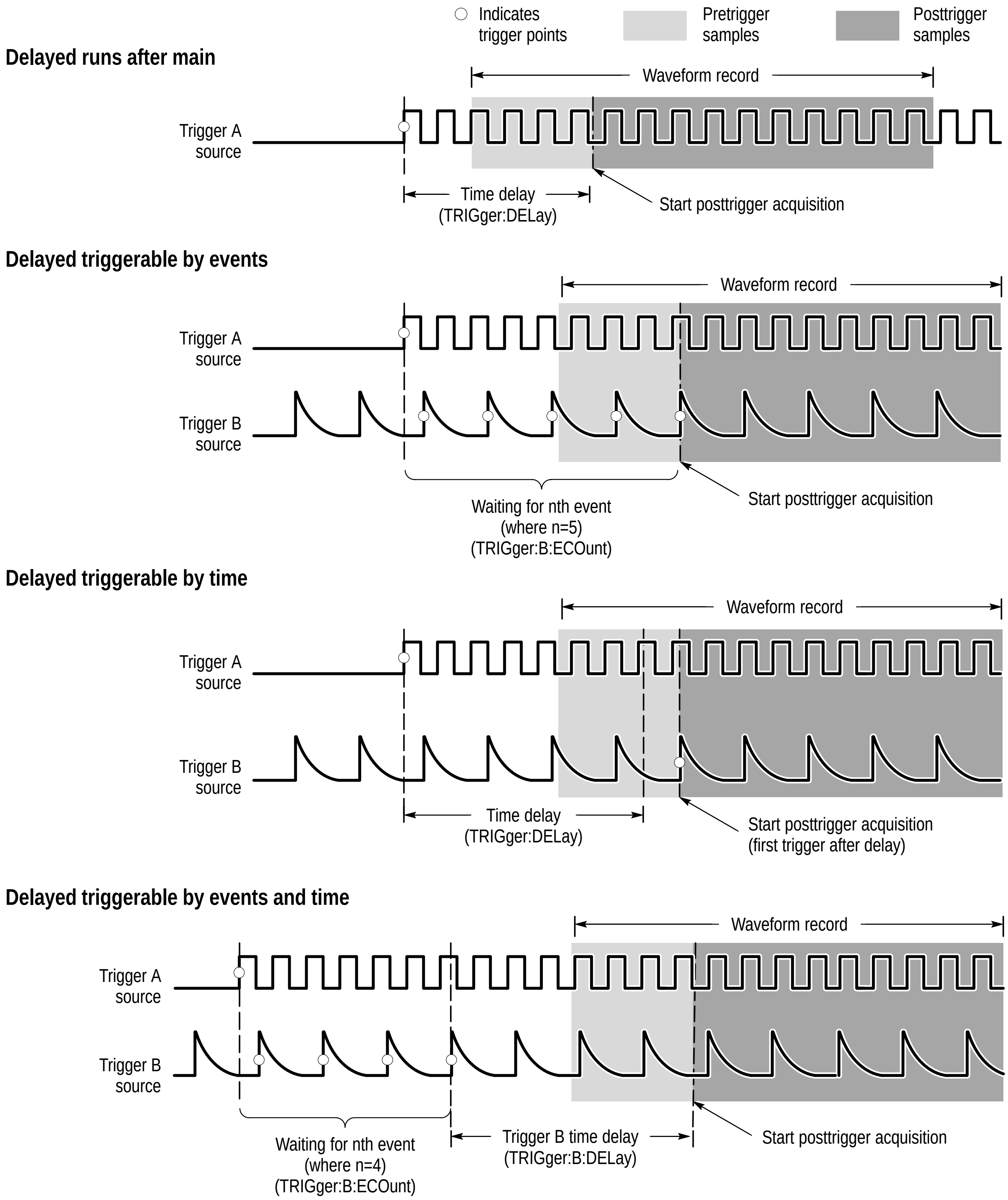

○ Indicates trigger points
Pretrigger samples
Posttrigger samples
Delayed runs after main
Waveform record
Trigger A source
Time delay (TRIGger:DELay)
Start posttrigger acquisition
Delayed triggerable by events
Waveform record
Trigger A source
Trigger B source
Waiting for nth event (where n=5) (TRIGger:B:ECOunt)
Start posttrigger acquisition
Delayed triggerable by time
Waveform record
Trigger A source
Trigger B source
Time delay (TRIGger:DELay)
Start posttrigger acquisition (first trigger after delay)
Delayed triggerable by events and time
Waveform record
Trigger A source
Trigger B source
Waiting for nth event (where n=4) (TRIGger:B:ECOunt)
Trigger B time delay (TRIGger:B:DELay)
Start posttrigger acquisition

Figure 3–68: How the delayed triggers work

***NOTE**. In the case of time delay, you can specify delay for either the TRIGger:A or the TRIGger:B system, but not both at the same time.*

Interaction Between Delay and Holdoff

Trigger delay determines how long after a trigger event to start acquisition. Trigger holdoff sets how long after one trigger before another event can occur. Figure 3–69 shows the relationship between trigger delay and trigger holdoff. Note that trigger holdoff must include the trigger delay time to be effective.

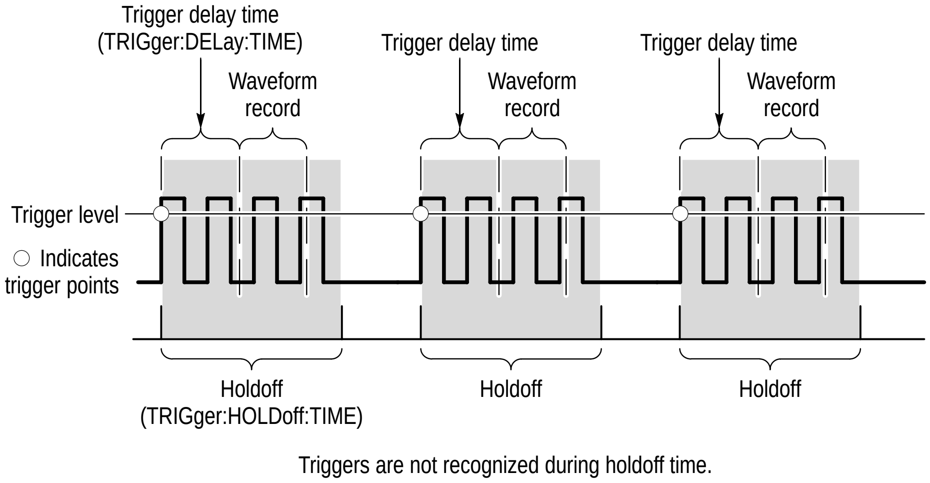


Figure 3–69: Trigger holdoff time with trigger delay time

Trigger Types

This section describes how to trigger using the various trigger types provided by the trigger system: edge, pulse, logic, setup & hold, and transition.

- To use the general-purpose trigger type, edge, see *Edge Triggering* on page 3–193.
- To pulse trigger based on pulse type (glitch) or parameters (width, timeout), see *Pulse Triggering* on page 3–199. Timeout triggers are available with TVS600A only.
- To logic trigger based on an input pattern or state, see *Logic Triggering* on page 3–195. TVS600A only.
- To setup & hold trigger based on violation of specified setup and hold times relative to a clock, see *Setup & Hold Triggering* on page 3–205. TVS600A only:
- To transition trigger based on how a pulse transitions between levels, see *Transition Triggering* on page 3–209. TVS600A only.

This section concludes with details about and instructions for using the Delayed time base and Delayed trigger system to delay the acquisition of a waveform relative to a trigger event. (See *Delayed Acquisitions* on page 3–189.)

Edge Triggering

The waveform analyzer can trigger on the rising or falling edge of a waveform. An edge trigger event occurs when the trigger source passes through a specified voltage level in a specified direction (the trigger slope). (See Figure 3–64 on page 3–187.) Although this section uses the TRIGger:A system in describing how to use edge triggers, both TRIGger:A and TRIGger:B systems can use edge triggers.

Why Use?

Edge triggers are simple to set up and will trigger on a large variety of waveforms. Use edge triggers unless you need the more discriminating triggering capability of the pulse, logic, setup-&-hold, or transition trigger types.

To Use

When you use a pattern trigger, you define the source, slope, level, and coupling parameters for the trigger signal.

Set up Edge Triggering. The following procedure shows how to edge trigger on an internal (channel) or external source. (Note that the steps following step 4 only apply to internal trigger sources.)

1. Select edge triggers: send `TRIGger:TYPE EDGE`.
2. Select the trigger source: send `TRIGger:SOURce INT<n>`, where <n> is the number of the channel providing the internal trigger, or `EXT` to select external trigger.
3. Select the edge of trigger signal for triggering: send `TRIGger:SLOPe POSitive` or `TRIGger:SLOPe NEGative`.
4. Set the level: send `TRIGger:LEVel <arg>`, where `<arg>` is an allowed value. (See your *TVS600A Command Reference* for range of values..
5. Select the coupling mode: send `TRIGger:COUPling AC` or `TRIGger:COUPling DC`. Or you can use the following step.
6. If desired, use `TRIGger:FILTer:` to select coupling and filtering mode as follows:
   - `NREJect ON` to provide noise rejection using the coupling mode that you selected in step 5.
   - `LPASs ON` to set AC coupling and attenuate trigger signal components above 50 kHz.
   - `HPASs ON` to set DC coupling and to attenuate trigger signal components below 50 kHz.

Set Mode and Holdoff. You can change the holdoff time and select the trigger mode for any type of trigger; to do so, do the following steps:

1. Select automatic (on) or normal (off) triggering (see *Usage Notes* below): send `TRIGger:ATRigger ON` or `TRIGger:ATRigger OFF`.
2. Use `TRIGger:HOLDoff:TIME <arg>`, where `<arg>` is `MINimum`, `MAXimum`, or a number within the range set by `MINimum` and `MAXimum`.

The holdoff range is from 250 ns (minimum) to 12 seconds (maximum).

Commands

The commands and functions that select edge triggering and the related parameters follow:

Used to:	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
set trigger type to edge for A triggers	`TRIGger:TYPE:EDGE`	`tktvs600_setTrigMain`	Yes
select edge slope, positive or negative	`TRIGger:SLOPe`		Yes
set the level for triggering	`TRIGger:LEVel`		Yes
set DC or AC coupling	`TRIGger:COUPling`		Yes
set filtering to be used in coupling trigger	`TRIGger:FILTer`		Yes
select normal or automatic	`TRIGger:ATRigger`	`tktvs600_setTrigOptions`	Yes
set the holdoff time between triggers	`TRIGger:HOLDoff:Time`		Yes

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Driver Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application**

Usage Notes

Some usage notes follow:

- For triggering to occur, the INITiate/ARM part of the cycle must occur. See *Initiate/Arm/Trigger Cycle* on page 3–181.
- Trigger type edge is the only type available for both the TRIGger:A and TRIGger:B systems.
- Trigger source EXT is always DC coupled.
- Trigger holdoff, mode (normal or automatic), and time delay are common parameters affecting all trigger types.
- Trigger sources can be digital signals, such as those available from the VXI backplane. These sources ignore the commands setting level, coupling, filtering, and so on; see TRIGger:SOURce in your *Command Reference.*
- Trigger levels that can be entered are affected by the source selected, probe used to connect the trigger signal, the input scaling, and so on. See TRIGger:LEVel in your *Command Reference.*

Logic Triggering (TVS600A Models Only)

The TVS600A Waveform Analyzer can trigger on a logic (or binary) pattern or on the state of a logic pattern at the time it is clocked. (Only four-channel TVS600A models have state triggers.) This section describes how to use the two classes of logic triggering: pattern and state.

Classes

There are two classes of logic triggering:

- Pattern. A trigger occurs when the logic inputs to the logic function you select cause the function to become TRUE.
- State. A trigger occurs when the logic inputs to the logic function cause the function to be TRUE *at the time* the clock input changes state.

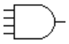
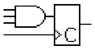
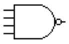
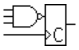
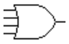
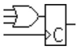
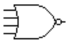
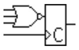
For pattern triggering, the waveform analyzer waits until the end of trigger holdoff and then samples the inputs from all the selected channels. The waveform analyzer then triggers if the conditions defined in Table 3–43 are met.

For state triggering, the waveform analyzer waits until the end of trigger holdoff and then waits until the edge of channel 4 transitions in the specified direction. At that point, the waveform analyzer samples the inputs from the other selected channels and triggers if the conditions defined in Table 3–43 are met.

Logic Functions

Both pattern and state triggers apply Boolean logic functions to the logic inputs. Table 3–43 defines these four logic functions.

Table 3–43: Pattern and state logic

Pattern	State	Definition[1]
 **AND**	 **Clocked AND**	If *all* the preconditions selected for the logic inputs[2] are TRUE[3], then the waveform analyzer triggers.
 **NAND**	 **Clocked NAND**	If *not all* of the preconditions selected for the logic inputs[2] are TRUE[3], then the waveform analyzer triggers.
 **OR**	 **Clocked OR**	If *any* of the preconditions selected for the logic inputs[2] are TRUE[3], then the waveform analyzer triggers.
 **NOR**	 **Clocked NOR**	If *none* of the preconditions selected for the logic inputs[2] are TRUE[3], then the waveform analyzer triggers.

[1] **Note that for state class triggers, the definition must be met at the time the clock input changes state.**

[2] **The logic inputs are channels 1, 2, 3, and 4 when using pattern logic triggers (1 and 2 for two-channel models). For State Logic Triggers, channel 4 becomes the clock input, leaving the remaining channels as logic inputs (no state triggers available for two-channel models).**

[3] **If time qualification is on, the input must be TRUE for a time greater or less (you specify) than the time limit you specify.**

Why Use?

Use pattern triggering when you need to trigger based on the Boolean behavior of several digital sources, such as when testing complex logic circuits. Use state triggering when you need a clocked pattern trigger.

To Use

When you use a pattern trigger, you define:

- The precondition for each logic input—logic high, low, or do not care (the logic inputs are channels 1, 2, 3, and 4)
- The Boolean logic function—select from AND, NAND, OR, and NOR
- Whether the TRUE condition is time qualified and how

When you use a state trigger, you define:

- The precondition for each logic input, channels 1, 2, and 3
- The direction of the state change for the clock input, channel 4
- The Boolean logic function—select from clocked AND, NAND, OR, and NOR

Set up Logic Triggering. You can set up logic triggering as follows:

1. To select logic triggers, send `TRIGger:TYPE LOGic`.
2. To select pattern or state triggering, send `TRIGger:LOGic:CLASs PATTern` or `TRIGger:LOGic:CLASs STATe` respectively. (State triggering is not available for two-channel TVS models).
3. If you selected `STATe` in step 2, send `TRIGger:LOGic:STATe:SLOPe POSitive` or `TRIGger:LOGic:STATe:SLOPe NEGative` to select the edge of clock signal (from CH4).
4. To set logic thresholds for highs and lows, send `TRIGger:LOGic:THReshold[1] <level>` to first set the threshold for the channel 1 logic input. Then repeat for thresholds 2 through 4.
   - The minimum and maximum values for threshold are `VOLT:RANG:OFFS` +/– `VOLT:RANG:PTP` (in `0.002` x `VOLT:RANG:PTP` steps).
   - Attempting to program `THReshold3` or `THReshold4` with two channel models generates `Execution Error –241, "Hardware missing"`.

5. To define the input pattern, send `TRIGger:LOGic:CONDition <arg>` to set the pattern bits, channels 1 – 4, to their Boolean levels. Select from `:LC0000` to `LC1111` as `<arg>`, where:

   - The digits in the argument refer to the four channels of the instrument in a left to right syntax corresponding to channel 4 through channel 1.
   - A `"1"` indicates a high condition is selected, a `"0"` indicates a low condition is selected, and an `"X"` indicates a Don't care is selected.
   - Pattern bits for channels 4 and 3 are ignored by two-channel waveform analyzers.
   - Pattern bit or channel 4 is ignored when `TRIGger:LOGic:CLASs` is set to `STATe`.

6. Use `TRIGger:LOGic:FUNCtion <arg>` to select the Boolean logic function to apply to the input pattern just defined. Select one of `AND`, `NAND`, `NOR`, or `OR` for `<arg>`.

Set Mode and Holdoff. The procedure for setting mode and holdoff is the same for all trigger types; see page 3–194. To learn more about trigger mode and holdoff, see the descriptions *Trigger Modes* and *Trigger Holdoff* on page 3–187.

Time Qualify Pattern Triggering. If you selected :PATTern in step 2 above, you can require a pattern be true for a specific duration; to do so, do the following steps (in addition to the steps above):

1. To select time qualification, send `TRIGger:LOGic:PATTern:QUALify <arg>`, `where <arg>` is either `GT` (TRUE Greater Than), `LT` (TRUE Less Than), or `OFF` (no time qualification).
2. To set the time for qualification, send `TRIGger:LOGic:PATTern:WIDTh <arg>`, `where <arg>` is either `MIN (1.0E–9)`, `MAX (1.0E+0)`, or number representing a time between min and max.

Commands

The commands and functions that set up logic triggering and related parameters follow:

Used to...	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
set trigger type to logic	`TRIGger:TYPE:LOGic`	Use the function below that matches class (pattern or state) used.	Yes
select Boolean: AND, OR, NAND, NOR	`TRIGger:LOGic:FUNCtion`		Yes
select bit pattern used for triggering	`TRIGger:LOGic:CONDition`		Yes
set logic thresholds for ch's in bit pattern	`TRIGger:LOGic:THReshold`		Yes
select pattern triggering	`TRIGger:LOGic:CLASs:PATTern`	`tktvs600_setLogicPatternTrigger`	Yes
set qualification to GT, LT, or off	`TRIGger:LOGic:PATTern:QUALify`		Yes
set width (time) to qualify against	`TRIGger:LOGic:PATTern:WIDTh`		Yes
select state triggering	`TRIGger:LOGic:CLASs STATe`	`tktvs600_setLogicStateTrigger`	Yes
select state clock slope, + or –	`TRIGger:LOGic:STATe:Slope`		Yes
select normal or automatic trigger mode	`TRIGger:ATRigger`	`tktvs600_setTrigOptions`	Yes
set the holdoff time between triggers	`TRIGger:HOLDoff:TIME`		Yes

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Driver Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application**

Usage Notes

Some usage notes follow:

- For triggering to occur, the INITiate/ARM part of the cycle must occur. See *Initiate/Arm/Trigger Cycle* on page 3–181.
- The TRIGger:B system cannot use LOGic type triggers (B system uses EDGe only).
- Only four-channel TVS600A models have state triggers.
- Two-channel TVS600A models have pattern triggers, but the pattern bits for channels 3 and 4 are ignored.
- Trigger source EXT is ignored for logic triggering.

Pulse Triggering

The waveform analyzer can trigger on a glitch pulse, on a pulse based on its width, or on a missing pulse in a steady stream of pulses. This section describes how to use each of the three classes of pulse triggers: glitch, width, and timeout.

Classes

There are three classes of pulse triggering:

- A `GLITch` trigger occurs when the trigger source detects a pulse narrower (or wider) in width than some specified time. It can trigger on glitches of either polarity.
- A `WIDTh` trigger occurs when the trigger source detects a pulse that is inside or, optionally, outside some specified time range (defined by the upper limit and lower limit). The waveform analyzer can trigger on positive or negative width pulses.
- A `TIMEout` trigger occurs when the trigger source does *not* detect a pulse edge when it expected to. (TVS600A only)

Qualifying Parameters

All three classes have unique parameters except for the level parameter, TRIGger:PULSe:THReshold, which they share.

The unique parameters listed in Table 3–44 determine whether an incoming pulse coupled to the triggering system triggers an acquisition. Study the table below and the figures that follow to understand how these parameters, and the commands used to set them, work.

Table 3–44: Pulse trigger qualifying parameters

Class (PULSe)	POLarity[1]	WIDth[2]	QUALify[3]
`:GLITch` Figure 3–70	Set `:POSitive`, `:NEGative`, or `:Both` (either polarity accepted)	Sets a pulse width used to qualify pulse	GT for greater than; LT for less than WIDth parameter.
:WIDth Figure 3–71	Set :POSitive or :Negative	Sets a range of pulse widths (using :WIDth:HLIMit and :LLIMit) used to qualify pulse	INside or OUTside the region defined by the WIDth parameters at left.
:Timeout Figure 3–72	Set POSitive or Negative	Set pulse width used to qualify pulse	No qualification selections[4]

[1] **POLarity determines which polarity pulses will be evaluated for possible triggering.**

[2] **WIDth sets a comparison pulse width (or range of widths) used to qualify the incoming pulses being evaluated for triggering. The THReshold level helps determine the measured width of the pulses so evaluated.**

[3] **QUALify specifies how the width of the incoming pulse must compare with the WIDth parameter(s) (a single width or range of widths).**

[4] **Time out qualification is fixed: if pulse width is less than the WIDth parameter, triggering occurs at the trailing edge of the pulse. If pulse width is greater, triggering is forced at time out: one “WIDth’s-time” after the pulse leading edge.**

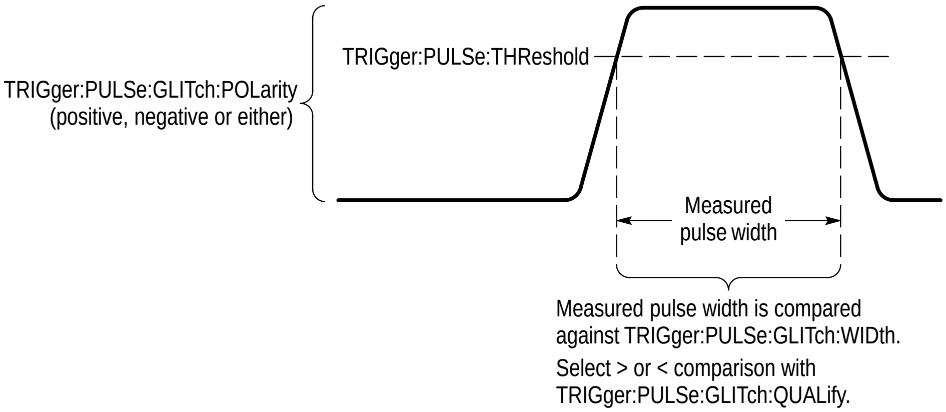

TRIGger:PULSe:THReshold
TRIGger:PULSe:GLITch:POLarity
(positive, negative or either)
Measured
pulse width
Measured pulse width is compared
against TRIGger:PULSe:GLITch:WIDth.
Select > or < comparison with
TRIGger:PULSe:GLITch:QUALify.

Figure 3–70: Glitch triggering (TRIGger:PULSe:CLASs:GLITch)

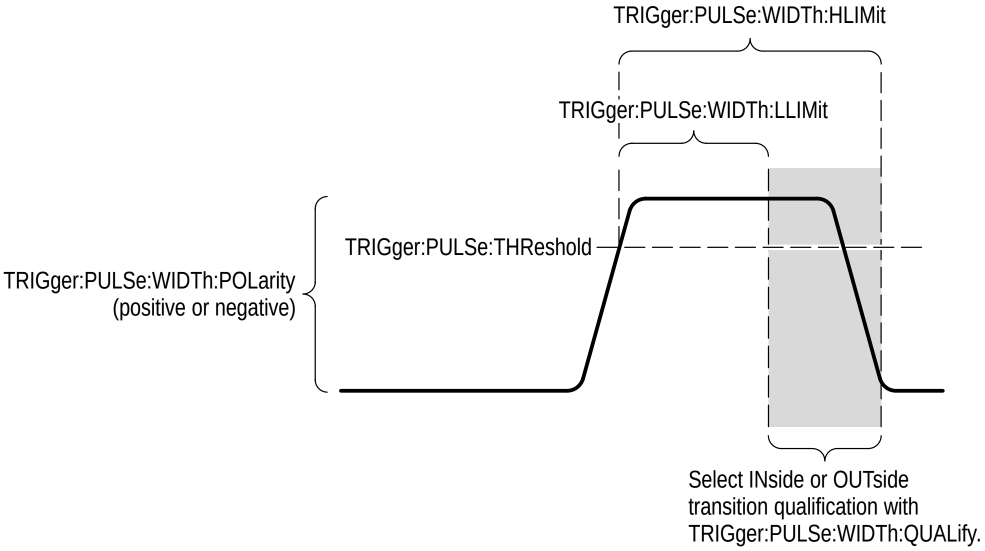

TRIGger:PULSe:WIDTh:HLIMit
TRIGger:PULSe:WIDTh:LLIMit
TRIGger:PULSe:THReshold
TRIGger:PULSe:WIDTh:POLarity
(positive or negative)
Select INside or OUTside
transition qualification with
TRIGger:PULSe:WIDTh:QUALify.

Figure 3–71: Width triggering (TRIGger:PULSe:CLASs:WIDTh)

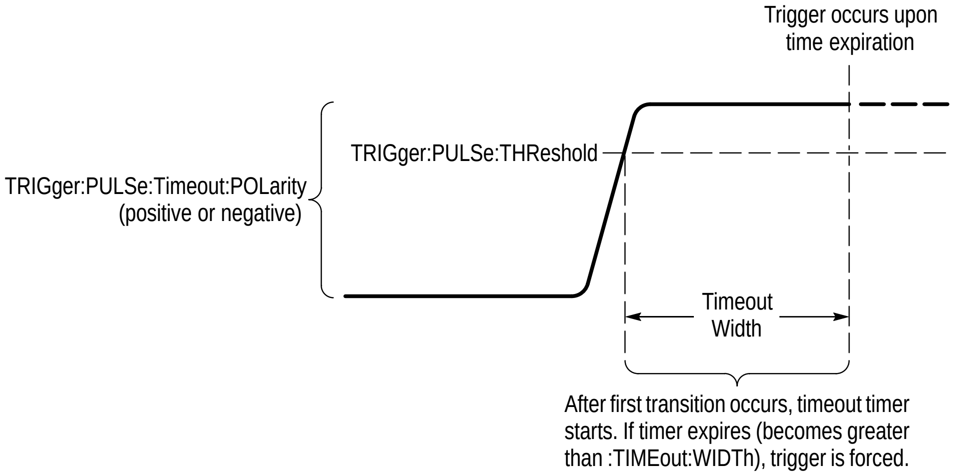

Trigger occurs upon time expiration
TRIGger:PULSe:THReshold
TRIGger:PULSe:Timeout:POLarity (positive or negative)
Timeout Width
After first transition occurs, timeout timer starts. If timer expires (becomes greater than :TIMEout:WIDTh), trigger is forced.

Figure 3–72: Timeout triggering (TRIGger:PULSe:CLASs:TIMEout) TVS600A only

Why Use?

These triggering types make the waveform analyzer suitable for such tasks as unattended monitoring for, and capturing of, a power supply glitch or for GO/NO GO slew-rate testing of operational amplifiers.

To Use

When you use pulse triggering, you define:

- Whether to trigger on glitches, pulse widths, or time-outs.
- Polarity and threshold (at which their width is measured) required for incoming pulses to be recognized as a trigger.
- Qualification times (widths) and relations for incoming pulses to be recognized as a trigger.

You can set up pulse triggering as follows:

1. To select pulse triggers, send `TRIGger:TYPE PULSe.`
2. To select glitch, width, or timeout pulse triggering, send `TRIGger:PULSe:CLASs <arg>`, where `<arg>` is `GLITch`, `WIDTh`, or `TIMEout`, respectively.
3. To specify the level on the pulse that determines pulse width, send `TRIGger:PULSe:THReshold[1] <arg>`, where `<arg>` is `MINimum`, `MAXimum`, or a number representing a voltage level within minimum and maximum.

   The minimum and maximum values for threshold are `VOLT:RANG:OFFS` +/– `VOLT:RANG:PTP` (in `0.002` x `VOLT:RANG:PTP` steps).

Set up Glitch Trigger. If you specified :GLITch in step 2, continue:

1. To specify the polarity of the pulse to be triggered on, send `TRIGger:PULSe:GLITch:POLarity <arg>`, where `<arg>` is one of `EITHer` (look at either polarity pulses), `NEGative` (only), or `POSitive` (only).
2. To specify the width of the pulse to be triggered on, send `TRIGger:PULSe:GLITch:WIDTh <arg>`, where `<arg>` is either `MIN` (1.0E–9), `MAX` (1.0E+0), or a number representing a a time between min and max.
3. To trigger on a pulse of a width greater than or less than that just specified in step 2, send `TRIGger:PULSe:GLITch:QUALify <arg>`, `where <arg>` is either `GT` (Greater Than) or `LT` (Less Than).

Set up WIDTh Trigger. If you specified :WIDTh in step 2, continue:

1. To specify the polarity of the pulse to be triggered on, send `TRIGger:PULSe:WIDTh:POLarity <arg>`, where `<arg>` is either `NEGative` (look at negative pulses only), or `POSitive` (positive pulses only).
2. To specify the lower limit for pulse-width range, send `TRIGger:PULSe:WIDTh:LLIMit <arg>`, where `<arg>` is either `MIN` (`1.0E–9`), `MAX` (`1.0E+0`), or a number representing a time between min and max.
3. To specify the higher limit for pulse-width range, send `TRIGger:PULSe:WIDTh:HLIMit <arg>`, where `<arg>` is either `MIN` (`1.0E–9`), `MAX` (`1.0E+0`), or or a number representing a time between min and max.
4. To trigger on a pulse the width of which is within the range just specified, send `TRIGger:PULSe:WIDTh:QUALify <arg>`, `where <arg>` is either `IN` (included in range) or `OUT` (falls outside range).

Set up Timeout Trigger. If you specified :TIMEout in step 2, continue:

1. To specify the polarity of the pulse to be triggered on, send `TRIGger:PULSe:TIMEout:POLarity <arg>`, where `<arg>` is either `NEGative` (look at negative pulses only), or `POSitive` (positive pulses only).
2. To specify the expected width of the pulse to be triggered on, send `TRIGger:PULSe:TIMEout:WIDTh <arg>`, where `<arg>` is either `MIN` (1.0E–9), `MAX` (1.0E+0), or `NRf` (a time between min and max).

   Triggering occurs in the case where:

   - The first transition of a pulse of the polarity defined in step 1 occurs *and...*

- The second transition of the pulse fails to occur before TIMEout:WIDth expires, resulting in a trigger at the point the expiration point.

Set Mode and Holdoff. The procedure for setting mode and holdoff is the same for all trigger types; see page 3–194. To learn more about trigger mode and holdoff, see the descriptions *Trigger Modes* and *Trigger Holdoff* on page 3–187.

Commands

The commands and functions that set up pulse triggering and the related triggering parameters follow:

Used to...	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
set trigger type to pulse	`TRIGger:TYPE PULSe`	Use the function below that matches class (glitch, width or time out) used.	No
set level on pulse that its width is measured	`TRIGger:PULSe:THReshold`		No
set source for incoming trigger pulses	`TRIGger:PULSe:SOURce`		No
select width triggering	`TRIGger:PULSe:CLASs WIDTth`	`tvs600setPulseWidthTrigger`	No
set high limit for pulse-width range	`TRIGger:PULSe:WIDTth:HLIMit`		No
set low limit for pulse-width range	`TRIGger:PULSe:WIDTth:LLIMit`		No
select pulse slope, positive or negative	`TRIGger:PULSe:WIDTth:POLar-ity`		No
set qualification to inside or outside range	`TRIGger:PULSe:WIDTth:QUALi-fy`		No
select glitch triggering	`TRIGger:PULSe:CLASs:GLITch`	`tvs600setGlitchTrigger`	No
set limit for pulse width	`TRIGger:PULSe:GLITch:WIDTh`		No
select pulse slope, positive. negative, or both	`TRIGger:PULSe:GLITch:POLar-ity`		No
set qualification to greater or less than pulse width	`TRIGger:PULSe:GLITch:QUALi-fy`		No
select timeout triggering	`TRIGger:PULSe:CLASs:TIMEout`	`tvs600setTimeoutTrigger`	No
select pulse slope, positive or negative	`TRIGger:PULSe:TIMEout:PO-Larity`		No
set timeout width after which trigger is forced	`TRIGger:PULSe:TIMEout:WIDTh`		No
select normal or automatic	`TRIGger:ATRigger`	`tvs600setTrigOptions`	No
set the holdoff time between triggers	`TRIGger:HOLDoff:TIME`	`tvs600setTrigOptions`	No

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Driver Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application.**

Usage Notes Some usage notes follow:

- For triggering to occur, the INITiate/ARM part of the cycle must occur. See *Initiate/Arm/Trigger Cycle* on page 3–181.
- The TRIGger:B system cannot use PULSe type triggers (B system uses EDGe only).

Setup and Hold Triggering (TVS600A Models Only)

The TVS600A Waveform Analyzer can trigger when a logic input changes state inside of user-specified setup and hold times relative to a clock. This section describes how to use setup/hold triggering.

Times Define the Zone. Setup/hold triggering uses the setup and hold times to define a "setup/hold violation zone" relative to the clock. Data that changes state within this zone triggers the waveform analyzer. Figure 3–73 shows how the setup and hold times you choose positions this zone relative to the clock.

- Positive settings for both setup and hold times (the most common application) locate the setup/hold violation zone so it *spans* the clocking edge. (See the top waveform in Figure 3–73.) The waveform analyzer detects and triggers on data that does not become stable long enough before the clock (setup time violation) or that does not stay stable long enough after the clock (hold time violation).
- Negative settings for setup or hold times skew the setup/hold violation zone to locate it before or after the clocking edge. (See the bottom and center waveforms of Figure 3–73.) The waveform analyzer can then detect and trigger on violations of a time *range* that occurs *before* or one that occurs *after* the clock.

***NOTE**. Keep hold time to at least 2 ns less than the clock period or the waveform analyzer cannot trigger.*

Zone Violations Define the Trigger. Setup/hold triggering uses the setup/hold violation zone to detect when data is unstable too near the time it is clocked. Each time trigger holdoff ends, the waveform analyzer monitors the data and clock sources. When a clock edge occurs, the waveform analyzer checks the data stream it is monitoring the data source) for transitions occurring within the setup/hold violation zone. If any occur, the waveform analyzer triggers *with the trigger point located at the clock edge*.

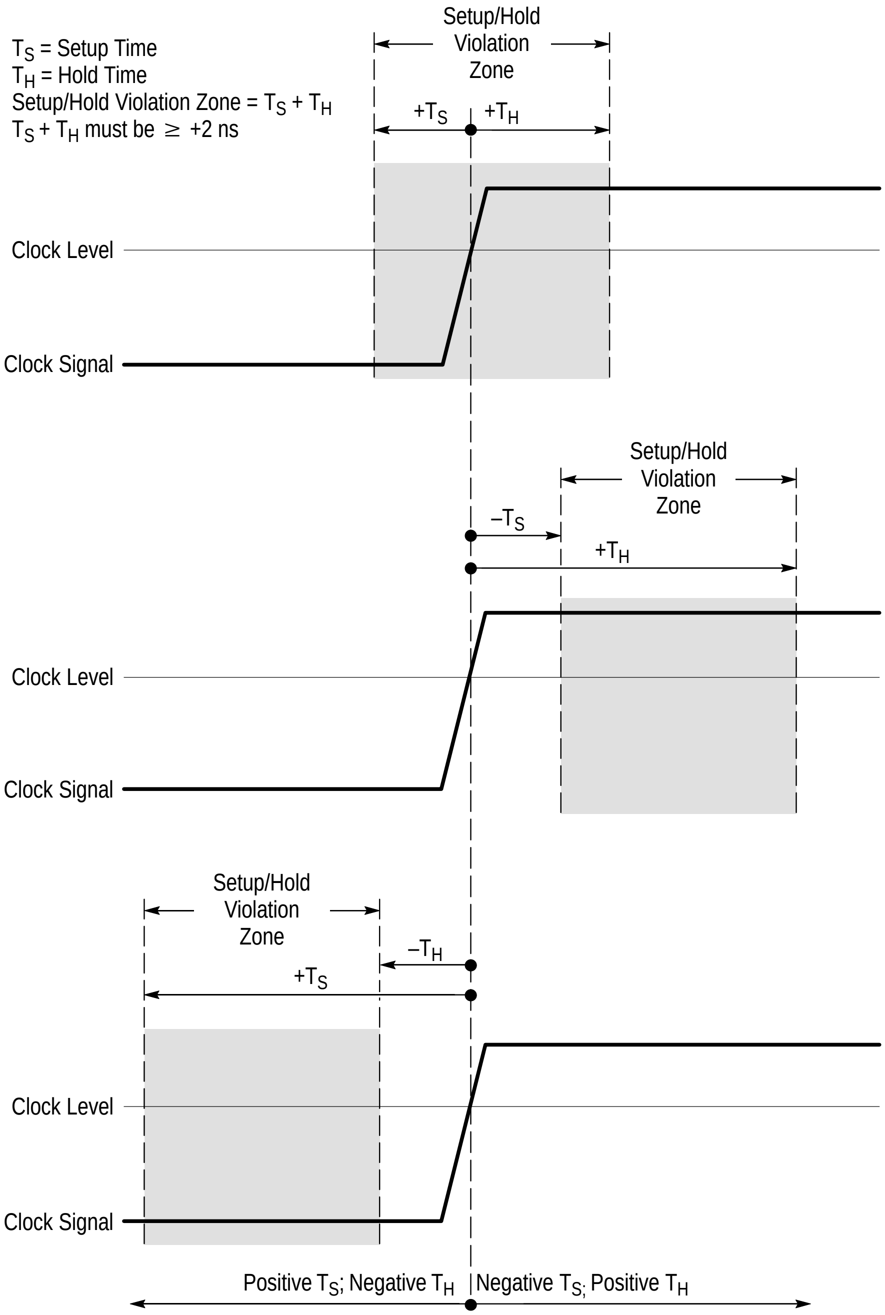

T_S = Setup Time
T_H = Hold Time
Setup/Hold Violation Zone = $T_S + T_H$
$T_S + T_H$ must be ≥ +2 ns
Setup/Hold Violation Zone
$+T_S$
$+T_H$
Clock Level
Clock Signal
Setup/Hold Violation Zone
$-T_S$
$+T_H$
Clock Level
Clock Signal
Setup/Hold Violation Zone
$-T_H$
$+T_S$
Clock Level
Clock Signal
Positive T_S; Negative T_H
Negative T_S; Positive T_H

Figure 3–73: Violation zones for Setup/Hold triggering

Why Use? Use setup/hold triggering to detect when data is not asserted long enough to be considered valid.

To Use

When you use setup/hold triggering, you define:

- The channel containing the logic input (the data source) and the channel containing the clock (the clock source)
- The direction of the clock edge to use
- The clocking level and data level that the waveform analyzer uses to determine if a clock or data transition has occurred
- The setup and hold times that together define a time range (violation zone) relative to the clock

Setup & Hold Triggering. You can set up for this type of triggering as follows:

1. To select setup & hold triggers, send `TRIGger:TYPE SHOLdtime.`
2. To select the data-signal source, send `TRIGger:SHOLdtime:DATA :SOURce INTernal<1–4>`, where `<1–4>` is the digit corresponding to the input channel containing the data signal.
3. To select the threshold of the data-signal source, send `TRIGger:SHOLdtime:DATA:THReshold {MIN|MAX|<nr3>}`, where:
   - `MIN` and `MAX` are the minimum and maximum values for threshold: `VOLT:RANG:OFFS +/– VOLT:RANG:PTP` (in `0.002` x `VOLT:RANG:PTP` steps).
   - `<nr3>` is a number representing a value between `MIN` and `MAX`.
   - Attempting to set threshold to an illegal value will generate Execution Error `–222, "Data out of range"`.
4. To select the clock source, send `TRIGger:SHOLdtime:CLOCk :SOURce INTernal<1–4>`, where `<1–4>` is the digit corresponding to the input channel containing the data signal.
5. To select the threshold of the clock source, send `TRIGger:SHOLdtime:CLOCk:THReshold {MIN|MAX|<nr3>`, where:
   - `MIN` and `MAX` are the minimum and maximum values for threshold: `VOLT:RANG:OFFS +/– VOLT:RANG:PTP` (in `0.002` x `VOLT:RANG:PTP` steps).
   - `<nr3>` is a number representing a value between MIN and MAX.
   - Attempting to set threshold to an illegal value will generate Execution Error `–222, "Data out of range"`.
6. To select the polarity of the clock edge, send `TRIGger:SHOLdtime:CLOCk:POLarity <arg>`, where `<arg>` is one of `NEGative` (look only

tat positive polarity pulses) or `POSitive` (look only at negative polarity pulses).

7. To specify the time data must be valid before and after the clock edge, set the set and hold times; send `TRIGger:SHOLdtime:HTIMe <arg>;STIME <arg>`, where `<arg>` is one of:

   - `MIN` and `MAX` are the minimum and maximum values for setup and for hold times. `MIN` is `–100.E–9` for both; setup time `MAX` is `100.0E–9` and hold time `MAX` is `102.0E–9`.

   - `<nr3>` is a number representing a value between MIN and MAX.

Commands

The commands and functions that set the setup and hold triggering and the related triggering parameters follow:

SCPI Commands[1]	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
set trigger type to logic	`TRIGger:TYPE SHOLdtime`	`tktvs600_setSetupHoldTrigger`	Yes
select source for data monitored for violations	`TRIGger:SHOLdtime:DATA:SOURce`		Yes
set threshold for data to change state	`TRIGger:SHOLdtime:DATA:THReshold`		Yes
select source for clock	`TRIGger:SHOLdtime:CLOCk:SOURce`		Yes
select threshold for clock	`TRIGger:SHOLdtime:CLOCk:THReshold`		Yes
set polarity of clock, pos. or neg.	`TRIGger:SHOLdtime:CLOCk:POLarity`		Yes
set setup time re clock (1st end of zone)	`TRIGger:SHOLdtime:STIMe`		Yes
set hold time re clock (2nd end of zone)	`TRIGger:SHOLdtime:HTIMe`		Yes
select normal or automatic	`TRIGger:ATRigger`	`tktvs600_setTrigOptions`	Yes
set the holdoff time between triggers	`TRIGger:HOLDoff:TIME`	`tktvs600_setTrigOptions`	Yes

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Driver Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application.**

Usage Notes

Some usage notes follow:

- For triggering to occur, the INITiate/ARM part of the cycle must occur. See *Initiate/Arm/Trigger Cycle* on page 3–181.

- The TRIGger:B system cannot use SHOLdtime type triggers (B system uses EDGe only).
- Trigger source EXT is ignored for setup & hold triggering.

Transition Triggering

The TVS600A Waveform Analyzer can trigger based on the how a pulse transitions across user-specified thresholds. This section describes how to use the two classes of transition triggering: runt and slewrate.

A runt trigger occurs when the trigger circuit detects a short pulse that crosses one threshold but fails to cross a second threshold before recrossing the first. (See Figure 3–74.)

A slew rate trigger occurs when the trigger circuit detects a pulse edge that traverses (slews) between two amplitude levels at a rate faster than or slower than you specify. The waveform analyzer can trigger on positive or negative slew rates. You can also think of slew rate triggering as triggering based on the slope (change in voltage/change in time) of a pulse edge. (See Figure 3–75.)

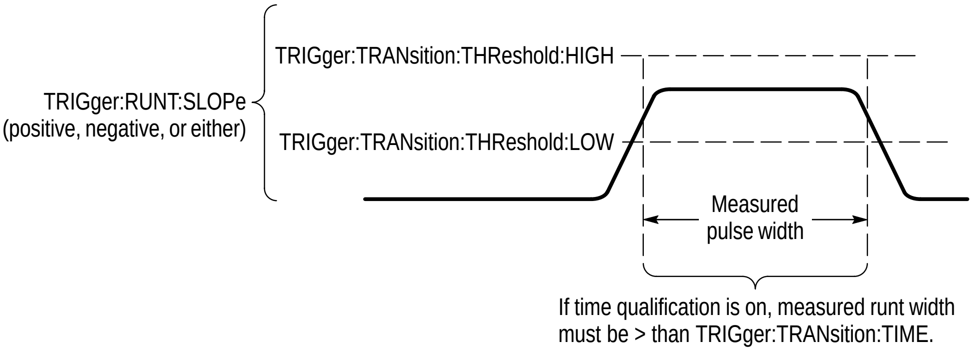

TRIGger:RUNT:SLOPe
(positive, negative, or either)
TRIGger:TRANsition:THReshold:HIGH
TRIGger:TRANsition:THReshold:LOW
Measured
pulse width
If time qualification is on, measured runt width
must be > than TRIGger:TRANsition:TIME.

Figure 3–74: Runt triggering (TRIGger:TYPE:TRANsition:CLASs RUNT)

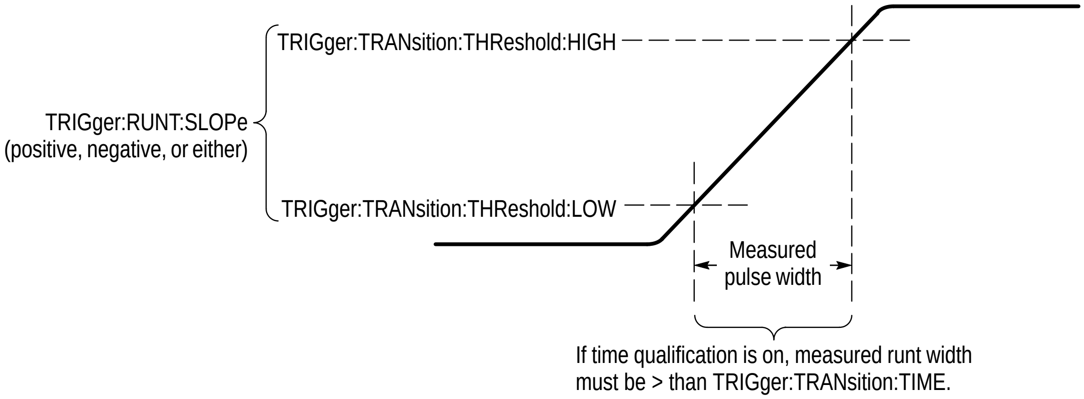

TRIGger:TRANsition:THReshold:HIGH
TRIGger:RUNT:SLOPe
(positive, negative, or either)
TRIGger:TRANsition:THReshold:LOW
Measured
pulse width
If time qualification is on, measured runt width
must be > than TRIGger:TRANsition:TIME.

Figure 3–75: Slew rate triggering (TRIGger:TYPE:TRANsition:CLASs SLEW)

Why Use?

Use transition triggering for such tasks as unattended monitoring for, and capturing of, a power supply glitch or for GO/NO GO slew rate testing of operational amplifiers.

To Use

When you use transition triggering, you define:

- The transition class: whether to trigger on runt pulses or on pulse slew rates
- The polarity of the pulse or its transition to use
- The high and low levels used to discriminated runt pulses or to measure slew rates between
- Time qualifications for triggering: runts with widths greater than or less than you specify or slew rates greater than or less than a time you specify

You can set up transition triggering as follows:

1. To select transition triggers, send `TRIGger:TYPE TRANsition`.
2. To select runt or slewrate triggering, send `TRIGger:TRANsition:CLASs <arg>`, where `<arg>` is `RUNT` or `SLEWrate`, respectively.
3. Use `TRIGger:TRANsition:SOURce INT` <n>, where <n> is the number of the channel providing the internal trigger (you can't use external triggering).
4. To specify the levels that determine crossings by a pulse transition, , send `TRIGger:TRANsition:THReshold:HIGH <level> and :LOW <level>`, where `<level>` is `MINimum`, `MAXimum`, or `<NRf>`, a voltage lying within minimum and maximum.

The minimum and maximum values for level are `VOLT:RANG:OFFS and VOLT:RANG:PTP`

5. To specify the transition qualification time of the pulse to be triggered on, send `TRIGger:TRANsition:TIME <arg>`, where `<arg>` is either `MIN` (1.0E–9), `MAX` (1.0E+0), or `NRf` (a time between min and max).

Set up RUNT Trigger. If you specified `:RUNT` in step 2, continue:

1. To specify the polarity of the runt pulse to be triggered on, send `TRIGger:TRANsition:RUNT:SLOPe <arg>`, where `<arg>` is one of `EITHer` (look at either polarity pulses), `NEGative` (only), or `POSitive` (only).
2. To trigger on all runts or only on runts wider than the time just specified in step 5, send `TRIGger:TRANsition:RUNT:QUALify <arg>`, where `<arg>` is either `OFF` (all runts) or `GT` (Greater Than).

Set up SLEWrate Trigger. If you specified `:SLEWrate` in step 2, continue:

1. To specify the polarity of the pulse edge to be triggered on, send `TRIGger:SLEW:SLOPe <arg>`, where `<arg>` is either `NEGative` (look at negative pulses only) or `POSitive` (positive pulses only).
2. To trigger on pulse-edge transitions within a time less than or greater than the time just specified in step 5, send `TRIGger:TRANsition:SLEW:QUALify <arg>`, where `<arg>` is either `GT` (Greater Than) or `LT` (Less Than).

Commands

The commands and functions that set up runt- or slew-rate triggering and their related triggering parameters follow:

Used to:	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
set trigger type to transition	`TRIGger:TYPE TRANsition`	Use the function below that matches class (runt or slew rate) used.	No
set source for incoming trigger pulse transitions	`TRIGger:TRANsition:SOURce`		No
set transition qualification time	`TRIGger:TRANsition:TIME`		No
set high (most positive) crossing level	`TRIGger:TRANsition:THReshold:HIGH`		No
set low (most negative) crossing level	`TRIGger:TRANsition:THReshold:LOW`		No
select runt triggering	`TRIGger:TRANsition:CLASS RUNT`	`tktvs600_setRuntTrigger`	No
select triggering on pos., neg., or any runt	`TRIGger:TRANsition:RUNT:SLOPe`		No
select triggering on runts with widths greater or less than qualification time	`TRIGger:TRANsition:RUNT:QUALify`		No
select slewrate triggering	`TRIGger:TRANsition:CLASs:SLEWrate`	`tktvs600_setSlewRateTrigger`	No
select triggering on pos. or neg. transitions	`TRIGger:TRANsition:SLEWrate:SLOPe`		No
select triggering on transitions slewing at rates greater or less than qualification time	`TRIGger:TRANsition:SLEWrate:QUALify`		No
select normal or automatic	`TRIGger:ATRigger`	`tktvs600_setTrigOptions`	No
set the holdoff time between triggers	`TRIGger:HOLDoff:TIME`		No

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Driver Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application.**

Usage Notes

Some usage notes follow:

- For triggering to occur, the INITiate/ARM part of the cycle must occur. See *Initiate/Arm/Trigger Cycle* on page 3–181.
- The TRIGger:B system cannot use TRANsition type triggers (B system uses EDGe only).
- Trigger source EXT is ignored for transition triggering.

System (VXIbus) Triggering

The waveform analyzer can trigger on the VXIbus TTL and ECL trigger lines. It can also drive these trigger lines, providing a trigger source. This section describes how to trigger on these signals; for information on driving (sourcing) the system trigger lines, see the commands OUTPut:TTLTrg and :ECLTrg in your *Command Reference.*

The system trigger lines TTLTrg and ECLTrg are digital signals carried by the P2 backplane and have the following characteristics:

- They may be generated by any VXIbus module in the system, including the waveform analyzer.
- The TTL signals conform to the TTL logic standard. Likewise, the ECL signals conform to the ECL logic standards.
- The TTLTrg lines have a 12.5 MHz clock speed and the ECLTrg lines 62.5 MHz. No coupling or filtering settings apply due to the digital nature of these signals.

Use the command TRIGger:[A]:SOURce to select from the signals TTLTrg0 to TTLTrg7 and ECLTrg0 to ECLTrg1.

Why Use?

VXIbus triggers provide an easy way to trigger several modules from one VXI source. Use VXIbus triggers when you need to synchronize several VXI modules to a single trigger line or want to reduce front-panel cabling requirements.

To Use

When you use a system trigger, you simply define an edge trigger, specifying an VXI bus line as the source (see *Edge Triggering* on page 3–193.)

Commands

The commands and functions that set up VXIbus triggering follow:

Used to...	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
set trigger type to edge (only)	`TRIGger:TYPE EDGE`	None	NA

set trigger source to one of two ECL lines or of eight TTL lines	`TRIGger:SOURce ECLTrg[0,1]`
	`TRIGger:SOURce TTLTrg[0–7]`
			No

select normal or automatic	`TRIGger:ATRigger`	`tktvs600_setTrigOptions`	Yes
set the holdoff time between triggers	`TRIGger:HOLDoff:TIME`		No

[1] Look up in the *TVS600A Command Reference.*

[2] Look up in the online *TVS600A Driver Reference.* Functions listed may be available with TVS600A models only; consult the online reference.

[3] If so indicated, feature can be set using the Soft Front Panel application

Usage Notes

Some usage notes follow:

- For triggering to occur, the INITiate/ARM part of the cycle must occur. See *Initiate/Arm/Trigger Cycle* on page 3–181.
- The TRIGger:B system cannot use system type triggers (B system uses EDGe only).
- Because of variation in the VXI power supplies, VXI module-to-VXI module delay cannot be specified for the waveform analyzer.

Delayed Triggering

The waveform analyzer can delay acquisition by time, by a number of trigger events, or by both.

Please read *Delayed Acquisitions* on page 3–189 for background on how the delayed acquisitions work before reading the following procedures. Also see Figure 3–63 on page 3–184 to become familiar with the sequential relationship between events recognition, delay time counting, and events counting in the A and B trigger systems.

Note that there is no command for setting B-trigger type; the B-trigger system supports edge triggering only (for a description, see *Edge Triggering* on page 3–193).

Why Use?

Delay is useful in acquiring events that occur with known delay from another event. Delay also is useful for disabling acquisition for a known delay, in time, event, or both, from an event.

To Use

When you use a system trigger, you set up the A-trigger system as usual, using any of the trigger types (edge, pulse, logic, an so on). You then set delay either using the TRIGger:A:DELay command or using the B-trigger system commands.

Simple Delay. After receiving the A-trigger, you can delay a time you specify and then force an acquisition. You use either the A-trigger system or B-trigger system to set the delay (but not both):

1. Set up using the TRIGger[:A] commands to trigger the A-trigger system on the event from which delay will occur. Use the procedures outlined throughout this section to set the A-trigger type, source, coupling, hold off, and other settings that you want.

2. Send either TRIGger[:A]:DELay <arg> or TRIGger:B:DELay <arg>, where 16 ns ≤ arg ≤ 250 s (in 4 ns steps) or <arg> is MINimum (0.0E+0) or MAXimum (250.0E+0).

   - See *Delayed runs after main* in Figure 3–68 on page 3–190, which shows how this type of delayed acquisition occurs. Note that acquisition is forced after delay expires.
   - Use SWEep:OFFSet:TIME to acquire pretrigger data. (See *Waveform Record Offset and Trigger Point* on page 3–110 for discussion of offset.)
   - Setting either delay, A or B, sets the other delay off (to minimum).

3. Send TRIGger:B:SOURce IMMediate to bypass events detection in the B-trigger system.

4. Initiate the acquisition and query the channel used.

Time-Delayed Trigger. You can also wait for the first B-trigger event to occur after a delay time expires; in this case, you must use the A-trigger system to specify the delay time and enable events recognition in the B-trigger system:

1. Do steps 1 and 2 of the procedure *Simple Delay*, setting the delay time in the B-trigger system, not the A, in step 2.

2. Set to trigger on the first B trigger after the delay: send TRIGger:B:ECOunt 1.

3. Select the trigger source: send TRIGger:B:SOURce INT <n>, where <n> is the number of the channel providing the internal trigger, or EXT to select external trigger. (This step enables events recognition in the B-trigger system.)

4. Select the edge of trigger signal for B-triggering: send TRIGger:B:SLOPe POSitive or TRIGger:SLOPe NEGative.

5. Set the level: send TRIGger:B:LEVel <arg>, where <arg> is an allowed value. (See your *TVS600A Command Reference* for range of values.)

6. Select the coupling mode: send TRIGger:B:COUPling AC or TRIGgerB:COUPling DC. Or you can use the following step.

7. If desired, use TRIGger:FILTer: to select coupling and filtering mode as follows:

   - NREJect ON to provide noise rejection using the coupling mode that you selected in step 5.
   - LPASs ON to set AC coupling and attenuate trigger signal components above 50 kHz.

- HPASs ON to set DC coupling and to attenuate trigger signal components below 50 kHz.

8. Initiate the acquisition and query the channel used.

Events-Delayed Acquisition. You can also wait for the nth B-trigger event to occur before triggering; in this case you must use the A-trigger system to specify the delay time, and enable the events recognition in the B-trigger system:

1. Set up as for the previous procedure, *Time-Delayed Trigger*, except as follows:

- Set TRIGger[:A]:DELay and TRIGger:B:DELay to zero.
- Set the B events count to the number of trigger events to wait; send: TRIGger:B:ECOunt <arg>, where 1 ≤ arg ≤ 10,000,000 or <arg> is MINIMUM (1) or MAXIMUM (10000000).

2. Initiate the acquisition and query the channel used.

Delayed triggerable by events in Figure 3–68 on page 3–190 shows how this type of delayed acquisition occurs.

Events- and Time-Delayed Acquisition. You can also use delay time with B events counting:

1. Set up as for the previous procedure, *Events-Delayed Acquisition,* except as follows: set a value other than zero for either TRIGger[:A]:DELay or TRIGger:B:DELay (but not for both) .

2. Initiate the acquisition and query the channel used.

If you set TRIGger[:A]:DELay, the waveform analyzer recognizes the A trigger, waits the delay time, counts to the specified nth B-trigger event, and triggers on the nth B-trigger event.

If you set TRIGger:B:DELay, the waveform analyzer recognizes the A trigger, counts to the specified nth B-trigger event, waits the delay time and forces an acquisition. *Delayed triggerable by time* in Figure 3–68 on page 3–190 shows how this type of delayed acquisition occurs.

Commands The commands and functions that set up delayed acquisition follow:

Used to...	SCPI Commands[1]	Driver Functions[2]	SFP?[3]
set A trigger delay[4]	`TRIGger[:A]:DELay`	`tktvs_setDelayedEdgeTrigger`	No
set B trigger delay[4]	`TRIGger:B:DELay`		No
set B trigger-delay events count	`TRIGger:B:ECOunts`		No
set B trigger events source	`TRIGger:B:SOURce`		No
select edge slope, positive or negative	`TRIGger:B:SLOPe`		No
set the level for triggering B events	`TRIGger:B:LEVel`		No
set DC or AC coupling for B trigger	`TRIGger:B:COUPling`		No
set filtering to be used in coupling B trigger	`TRIGger:B:FILTer`		No

[1] **Look up in the *TVS600A Command Reference.***

[2] **Look up in the online *TVS600A Driver Reference.* Functions listed may be available with TVS600A models only; consult the online reference.**

[3] **If so indicated, feature can be set using the Soft Front Panel application**

[4] **Setting one of these delays automatically sets the other to minimum; they cannot both be used at the same time.**

Status and Events

The Status and Event reporting system reports asynchronous events and errors that occur in the TVS600A. This system consists of status registers and message queues that you access through the command language. Use these status registers and message queues to determine when an error has occurred or when a function completes.

This section describes each status register and message queue, the Status and Event reporting process, and how to synchronize programming. This section ends with a list of the system messages.

Registers

The registers in the Status and Event reporting system fall into three functional groups:

- Status Registers contain information about errors and normal operations in the waveform analyzer. The status registers latch their event status and are cleared when read.
- Enable Registers determine whether selected types of events are reported to the Status Registers and the Event Queue. The enable registers must be cleared by setting with the appropriate values.
- Condition Registers provide access to the event status lines before they enter the Enable and Status Registers. Only the OPERation and QUEStionable SCPI registers have Condition registers.

A set of commands for each register allow you to read its status and set the enable register. Figure 4–1 illustrates how the registers are connected and gives the commands that control them.

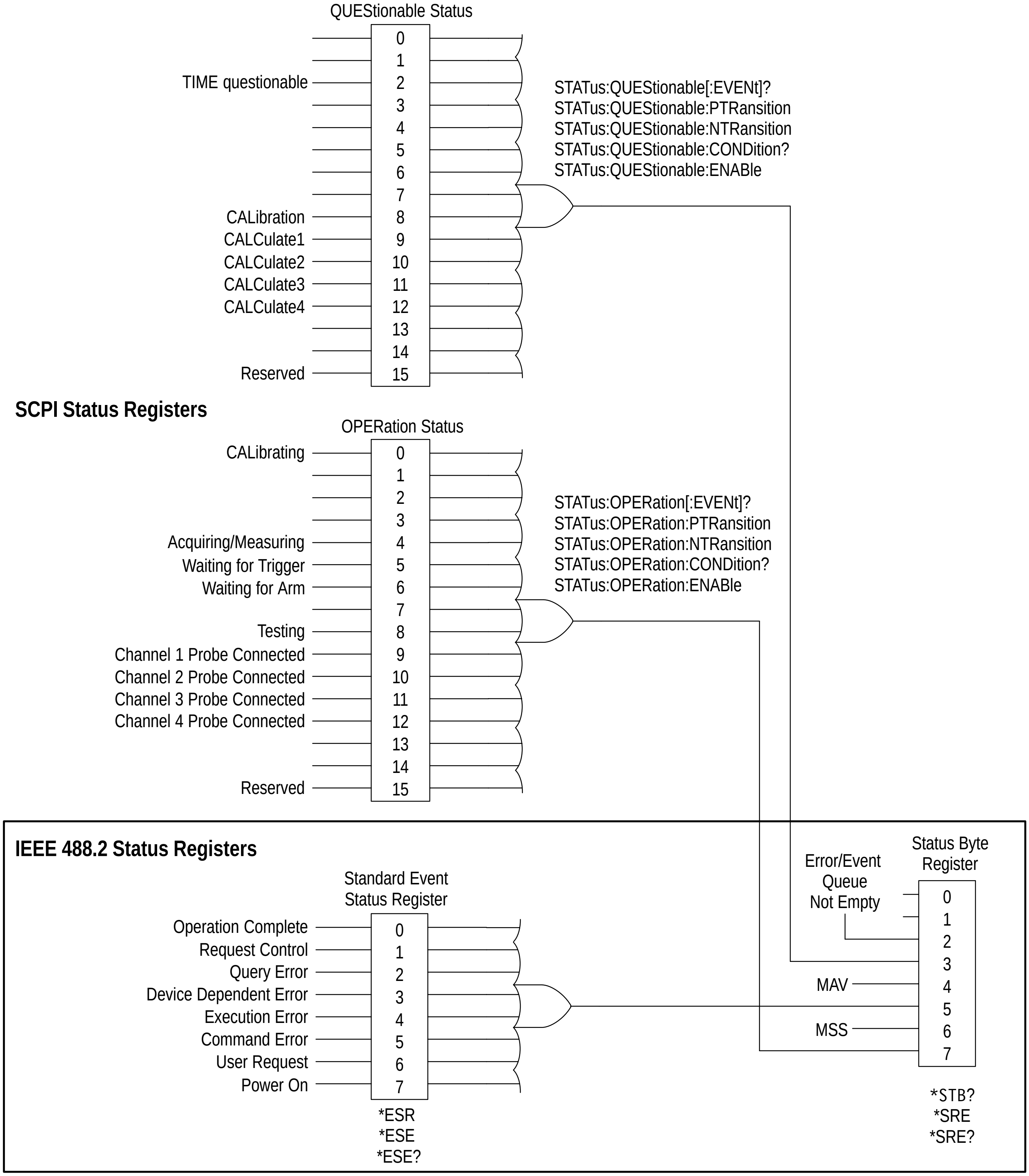


Figure 4–1: SCPI & IEEE 488.2 Status and Event Registers

Status Byte Register

The Status Byte Register, defined in Table 4–1, summarizes information from other registers and indicates when message data is in the Status Byte or Message queue. Refer to Figure 4–1 to see how the other status registers connect to the Status Byte Register. To read the contents of the Status Byte Register, use the *STB? query. The response is the sum of the decimal values for all true (1) bits. The *STB? query *does not* clear the Status Byte Register.

Table 4–1: The Status Byte Register

Bit	Decimal Value	Function
0–1	–	Not used.
2	4	**Error/Event Queue not empty** indicates that the error/event queue contains information and is waiting to be read.
3	8	**Questionable Event Status** indicates that the quality of result data or of an operation is questionable.
4	16	**Message Available (MAV)** shows that data is available in the Output Queue.
5	32	**Event Status Bit** indicates that one or more events have occurred in the Standard Event Status Register.
6	64	**MSS** (Master Summary Status) is a summary bit that indicates other bits in the Status Byte Register are set.
7 (MSB)	128	**Operation Event Register** indicates that the waveform analyzer is busy performing a normal operation such as acquiring a waveform.

Service Request Enable Register

The Service Request Enable Register (SRER) controls which bits in the Status Byte Register generate a service request. The bits in the SRER correspond to those in the Status Byte Register defined in Table 4–1. To set bits in the Service Request Enable Register, use the *SRE command. Set a SRER bit off (0) to disable its event from generating an interrupt. To see which bits are disabled, use the *SRE? query. The *SRE query returns the decimal sum for all set bits.

If, for example, the *SRE? query returns a value of 48, bits 4 and 5 are set in the Service Request Enable Register. All other bits are zero (0) and their events are disabled. Any event that sets the Message Available bit 4 or the Event Status bit 5 in the Status Byte Register will generate an interrupt. Other masked events, such as Questionable Status events, are prevented from generating an interrupt. To disable all interrupts except the Event Status bit 5, you would use the command *SRE 32.

Standard Event Status Register

The Standard Event Status Register,defined in Table 4–2, records eight types of events that may occur in the waveform analyzer. To read the contents of the Standard Event Status Register, use the *ESR? query. The response is the sum of the decimal values for all true (1) bits. The *ESR? query clears all bits in the

Standard Event Status Register. The Event Status Enable Register allows you to disable bits for specific events. Additionally, you can select which events cause an event message in the Status Queue with the command `STATus:SESR:QENable`.

Table 4–2: The Standard Event Status Register

Bit	Decimal Value	Function
0	1	**Operation Complete** shows that the operation is complete. This bit is set when all pending operations complete following an *OPC command.
1	2	**Request Control** (not used)
2	4	**Query Error** shows that the waveform analyzer attempted to read the Output Queue when no data was present or pending, or that data in the Output Queue was lost.
3	8	**Device Dependent Error** shows that a device error occurred. Table 4–10 on page 4–14 lists the device error messages.
4	16	**Execution Error** shows that an error occurred while the waveform analyzer was executing a command or query. Table 3–31 on page 4–13 lists the execution error messages.
5	32	**Command Error** shows that an error occurred while the waveform analyzer was parsing a command or query. Table 4–8 on page 4–12 lists the command error messages.
6	64	**User Request** indicates that a probe ID button was pressed.
7	128	**Power On** shows that the waveform analyzer was powered on.

The following example assumes that all bits have been enabled using the Event Status Enable Register (see the next topic). If an `*ESR?` query returns a value of 128, bit 7 is set (1) which indicates the instrument is in the initial power-on state and no other event bits are set.

Event Status Enable Register

The Event Status Enable Register (ESER) controls which events are summarized in bit 5 of the Status Byte Register. The bits in the ESER correspond to those in the Standard Event Status Register, which are defined in Table 4–2. Use the `*ESE` command to set bits in the Event Status Enable Register. Set an ESER bit off (0) to disable its event in the Standard Event Status Register. To see which bits are disabled, use the `*ESE?` query. The `*ESE?` query returns the decimal sum for all set bits.

For example, if the `*ESE?` query returns a value of 255, all bits are true (1) indicating that all events set the event status bit (bit 5) of the Status Byte Register. To disable Execution Errors, you could send the command `*ESE 239`, which disables only bit 4, Execution Errors.

Operation Status Register

The Operation Status Register (OSR), defined in Table 4–3, identifies normal waveform analyzer events that are still in progress, such as acquisition in progress and waiting for a trigger. When an Operation event sets a bit true, the summary output sets bit 7 true in the Status Byte Register.

To read the contents of the OSR, use the `STATus:OPERation?` query. The response is the sum of the decimal values for all set (1) bits. Reading the OSR clears all bits.

Table 4–3: The Operation Status Register

Bit	Decimal Value	Function
0	1	**Calibrating** shows that a calibration routine is in progress.
1–3		Not used.
4	16	**Measuring/Acquiring** shows that measurement or acquisition is in progress.
5	32	**Waiting for Trigger** shows that the acquisition system is armed and waiting for a trigger event.
6	64	**Waiting for Arm** shows that the acquisition system has been initialized with INIT and is waiting to be armed.
7		Not used.
8	256	**Testing** shows that a self test routine is in progress.
9	512	**CH 1 Probe** shows that a probe is attached to the Channel 1 input.
10	1024	**CH 2 Probe** shows that a probe is attached to the Channel 2 input.
11	2048	**CH 3 Probe** shows that a probe is attached to the Channel 3 input.
12	4096	**CH 4 Probe** shows that a probe is attached to the Channel 4 input.
13–15		Not used.

Table 4–4 describes the control registers for the OSR. The control registers allow you to determine what is reported with the `STATus:OPERation?` query and what event can set the Operation bit (bit 7) in the Status Byte Register.

Table 4–4: Control registers for the Operation Status Register

Control Register	Description	Control Commands	Affects of STATus:PRESet
Operation Status	Records the status of normal operating events. The Positive and Negative Transition registers will affect which events are stored in the Operation Status register.	STATus:OPERation?	None
Operation Condition	Provides the current state of Operation event lines prior to the Transition registers.	STATus:OPERation:CONDition?	None
Positive Transition	Reports an event in the Operation Status Register when the Condition Register event changes from false to true (0 to 1).	STATus:OPERation:PTRansition STATus:OPERation:PTRansition?	All 1's (reports all events on positive transitions)
Negative Transition	Reports an event in the Operation Status Register when the Condition Register event changes from true to false (1 to 0).	STATus:OPERation:NTRansition STATus:OPERation:NTRansition?	All 0's (reports no events on negative transitions)
Operation Enable	Disables events in the Operation Status Register from setting the summary bit 7 in the Status Byte Register. Setting a bit to one (1) enables that event, and setting it to zero (0) disables it.	STATus:OPERation:ENABle STATus:OPERation:ENABle?	All 0's (all events are disabled)
Status Queue Enable Positive Transition	Determines if transitions from false to true (0 to 1) on event lines in the Operation Condition register will generate a message in the Status Queue.	STATus:OPERation: QENable:PTRansition STATus:OPERation: QENable::PTRansition?	All 0's (all events are disabled)
Status Queue Enable Negative Transition	Determines if transitions from true to false (1 to 0) on event lines in the Operation Condition register will generate a message in the Status Queue.	STATus:OPERation: QENable:NTRansition STATus:OPERation: QENable::NTRansition?	All 0's (all events are disabled)

Questionable Status Register

The Questionable Status Register, (QSR) defined in Table 4–5, identifies operations whose results are questionable. To read the contents of the QSR, use the `STATus:QUEStionable?` query. The response is the sum of the decimal values for all set (1) bits. Reading the QSR clears all bits. Use the Questionable Status Enable Register to enable or disable specific events. In addition, the Questionable Status Condition Register provides access to the current state of these events.

An example of a questionable condition is when the waveform analyzer is due for self calibration because of changes in the ambient temperature. Because the waveform analyzer is in need of calibration, any data you acquire will be of questionable quality.

Table 4–5: The Questionable Status Register

Bit	Decimal Value	Function
0,1,3–7		Not used.
2	4	TVS600 models: Not used. TVS600A models: **Time questionable** indicates metastable trigger status.
8	256	**Calibration** indicates that calibration is required due to the change in ambient temperature.
9	512	**Calculate1** indicates that source data contained a value that was overrange or underrange, making the results of the CALC1 block questionable.
10	1024	**Calculate2** indicates that source data contained a value that was overrange or underrange, making the results of the CALC2 block questionable.
11	2048	**Calculate3** indicates that source data contained a value that was overrange or underrange, making the results of the CALC3 block questionable.
12	4096	**Calculate4** indicates that source data contained a value that was overrange or underrange, making the results of the CALC4 block questionable.
13–15		Not used.

Table 4–6 describes the control registers for the QSR. The control registers allow you to determine what is reported with the `STATus:QUEStionable?` query and what event sets the Questionable bit (bit 3) in the Status Byte Register.

Table 4–6: Control registers for the Questionable Status Register

Control Register	Description	Control Commands	Affects of STATus:PRESet
Questionable Status	Records the status of normal operating events. The Positive and Negative Transition registers will affect which events are stored in the Questionable Status Register.	STATus:QUEStionable?	None
Questionable Condition	Provides the current state of Questionable event lines prior to the Transition registers.	STATus:QUEStionable:CONDition?	None
Positive Transition	Reports an event in the Questionable Status Register when the Condition Register event changes from false to true (0 to 1).	STATus:QUEStionable:PTRansition STATus:QUEStionable:PTRansition?	All 1's (reports all events on positive transitions)
Negative Transition	Reports an event in the Questionable Status Register when the Condition Register event changes from true to false (1 to 0).	STATus:QUEStionable:NTRansition STATus:QUEStionable:NTRansition?	All 0's (reports no events on negative transitions)

Table 4–6: Control registers for the Questionable Status Register (cont.)

Control Register	Description	Control Commands	Affects of STATus:PRESet
Questionable Enable	Disables events in the Questionable Status Register from setting the summary bit 3 in the Status Byte Register. Setting a bit to one (1) enables that event, and setting it to zero (0) disables it.	STATus:QUEStionable:ENABle STATus:QUEStionable:ENABle?	All 0's (all events are disabled)
Status Queue Enable Positive Transition	Determines if transitions from false to true (0 to 1) on event lines in the Questionable Condition register will generate a message in the Status Queue.	STATus:QUEStionable: QENable:PTRansition STATus:QUEStionable: QENable::PTRansition?	All 0's (all events are disabled)
Status Queue Enable Negative Transition	Determines if transitions from true to false (1 to 0) on event lines in the Questionable Condition register will generate a message in the Status Queue.	STATus:QUEStionable: QENable:NTRansition STATus:QUEStionable: QENable::NTRansition?	All 0's (all events are disabled)

Queues

The waveform analyzer has two message queues. The Status Queue stores error and event messages that result from incorrect commands or waveform analyzer operations. When you send the `STATus:QUEue?` query, messages are stored temporarily in the second message queue, the Output Queue. The Output queue sets the Message Available (MAV) bit four in the Status Byte Register and waits for a VXIbus read command.

The Status Queue

The Status and Event reporting system stores event messages in the Status Queue. The Status Queue stores events and errors until the queue memory is filled. Events are stored in first-in, first-out order. When the queue overflows, the last event message is replaced with the device specific error

```
–350, "Queue overflow".
```

To retrieve the first (and oldest) event message, send the command `SYSTem:ERRor?`. The returned event message contains the event number and a text description of the event such as error –350 above. Reading an event removes it from the queue. The commands described in Table 4–7 control the Status Queue and the responses to your event query.

Table 4–7: Commands associated with the Status Queue

Command	Description
SYSTem:ERRor?	Returns the next event including the error code and text.
SYSTem:ERRor:ALL?	Returns all events including the error code and text.

Table 4–7: Commands associated with the Status Queue (cont.)

Command	Description
SYSTem:ERRor:CODE?	Returns only the error code for the next event.
SYSTem:ERRor:CODE:ALL?	Returns only the error codes for all events.
SYSTem:ERRor:COUNt?	Returns the number of events stored in the Status Queue.

You can control which register events send a message to the Status Queue. The following registers have commands that control event queueing:

- Standard Event Status Register provides the command `STATus:SESR:QENable` so you can disable queuing for one or more types of events. The default after `*RST` is to report only errors.
- Operation Status Register provides the commands `STATus:OPERation:QENable:PTRansition` and `:NTRansition` so you can disable queueing for positive or negative transitions of event lines. Event lines are monitored at the Operation Condition Register. The default after `*RST` is all queueing disabled.
- Questionable Status Register provides the commands `STATus:QUEStionable:QENable:PTRansition` and `:NTRansition` so you can disable queueing for positive or negative transitions of event lines. Event lines are monitored at the Questionable Condition Register. The default after `*RST` is all queueing disabled.

***NOTE**. The TVS600A driver provides functions that support control of event queueing; see your online reference installed as part of the TVS600A VXI-*plug&play *software for information on driver functions.*

The Output Queue

The waveform analyzer temporarily stores query responses in the Output Queue. When the Output queue has a message, it sets bit 4 (MAV) in the Status Byte Register, but if you read it with the command `*STB`, you will overwrite your query response in the Output queue with the bit status. Instead, use the VISA level command `viReadSTB`. See your VISA documentation for information on this command.

The Output Queue is emptied each time you send a new command or query message after it receives an End Of Message (EOM). Therefore, if the controller does not read a query response from the Output Queue before it sends the next command (or query), it will lose the response to the previous query and set a query error.

Status and Event Reporting Process

Figure 4–2 shows how to use the Status and Event Reporting system. In the explanation that follows, numbers in parentheses refer to the circled numbers in Figure 4–2.

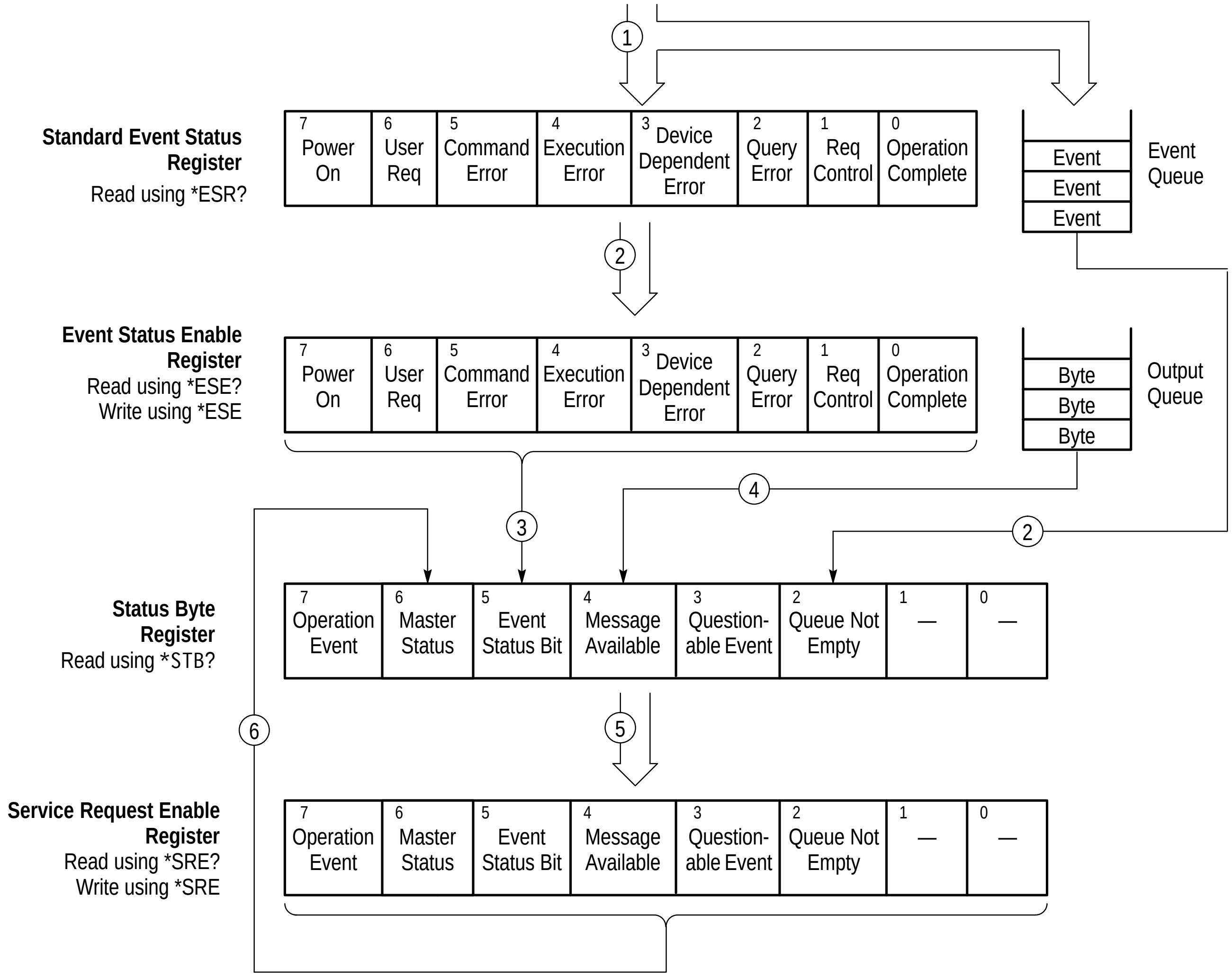


Figure 4–2: Status and event reporting process

When an event occurs the appropriate bit in the Standard Event Status Register is set to one and the event is recorded in the Event Queue (1). If the corresponding bit in the Event Status Enable Register is also enabled (2), then the event status bit in the Status Byte Register is set to one (3). When an event enters the Event Queue, it sets the Queue Not Empty bit in the Status Byte Register (2).

When the Status Reporting System sends output to the Output Queue (for example, a response to a query), it sets the message available bit in the Status Byte Register to one (4).

When a bit in the Status Byte Register is set to one and the corresponding bit in the Service Request Enable Register is enabled (5), the master status summary bit in the Status Byte Register is set to one (6).

Synchronization Methods

Although most commands are completed soon after being received by the waveform analyzer, some commands start processes requiring a longer period. For example, after you send the `INITiate` command, you must wait until acquisition completes before you give another command or query.

Sometimes the result of an operation depends on the result of an earlier operation (the first operation must be completed before the next one is initiated). One example is performing a calculation after a waveform acquisition. The status and event reporting system provides this capability.

Using the *OPC? Query

Use the `*OPC?` query to synchronize commands. The `*OPC?` query places a 1 in the Output Queue once an operation is complete. Set the controller time out so it exceeds the longest time expected to complete an operation. Set up a chained message as follows and wait for the 1 to appear in the Output Queue between sending commands:

```
/* Set up a chained message */
INITiate;*OPC?
```

***NOTE**. The TVS600A driver provides a function that supports *OPC synchronization; see your online reference installed as part of the TVS600A VXI*plug&play *software for information on driver functions.*

Error Messages

The waveform analyzer generates error messages in response to events caused by commands or queries. Each type of event sets a specific bit in the Standard Event Status Register. Thus, each message is associated with a specific Standard Event Status Register bit. In the message tables that follow, the associated Standard Event Status Register bit is specified in the table title. Not shown in the tables are secondary messages giving more detail about the cause of the error or the meaning of the message. These secondary messages are shown for each command and query in your *TVS600 and TVS600A Command Reference.*

Table 4–8 shows the error messages generated by improper command syntax. Check to see that the command is properly formatted and that it follows the rules in your *TVS600 and TVS600A Command Reference.*

Table 4–8: Command error messages (bit 5 in Standard Event Status Register)

Code	Message
100	Command error
101	Invalid character
102	Syntax error
103	Invalid separator
104	Data type error
105	Get not allowed
106	Invalid program data separator
108	Parameter not allowed
109	Missing parameter
110	Command header error
111	Header separator error
112	Mnemonic too long
113	Undefined header
118	Query not allowed
120	Numeric data error
121	Invalid char in number
123	Exponent too large
124	Too many digits
128	Numeric data not allowed
130	Suffix error
131	Invalid suffix
134	Suffix too long

Table 4–8: Command error messages (bit 5 in Standard Event Status Register) (cont.)

Code	Message
138	Suffix not allowed
140	Character data error
141	Invalid character data
144	Character data too long
148	Character data not allowed
150	String data error
151	Invalid string data
158	String data not allowed
160	Block data error
161	Invalid block data
168	Block data not allowed

Table 3–31 lists the execution error messages that can occur during execution of a command.

Table 4–9: Execution error messages (bit 4 in Standard Event Status Register)

Code	Message
200	Execution error
220	Parameter error
221	Settings conflict
222	Data out of range
223	Too much data
224	Illegal parameter value
230	Data corrupt or stale
240	Hardware error
241	Hardware missing
250	Mass storage error
252	Missing mass storage
252	Missing media
253	Corrupt media
254	Media full
255	Directory full

Table 4–9: Execution error messages (bit 4 in Standard Event Status Register) (cont.)

Code	Message
256	File name not found
257	File name error
258	Media protected

Table 4–10 lists the device dependent error messages that can occur during waveform analyzer operation.

Table 4–10: Device dependent error messages (Bit 3 in Standard Event Status Register)

Code	Message
300	Device specific error
310	System error
361	Autoscan failed

Table 4–11 lists the system events.

Table 4–11: System events

Code	Message
401	Power on[1]
402	Operation complete[2]

[1] **Sets bit 7 in the Standard Event Status Register.**

[2] **Sets bit 0 in the Standard Event Status Register.**

Table 4–12 lists the execution warnings that can occur during execution of a command.

Table 4–12: Execution warning messages (Bit 3 in Standard Event Status Register)

Code	Message
500	Execution warnings

Appendix A: Specifications

This chapter contain the complete specifications for the waveform analyzer. For a general description of the waveform analyzer, see *Product Description* (below) page 1–1.

All specifications are guaranteed unless noted “typical.”

The performance limits in this specification are valid with these conditions:

- The waveform analyzer must have been calibrated/adjusted at an ambient temperature between +20° C and +30° C.
- The waveform analyzer must be in an environment with temperature, altitude, humidity, and vibration within the operating limits described in these specifications.
- The waveform analyzer must have had a warm-up period of at least 20 minutes.
- The waveform analyzer must have had its signal-path-compensation routine (self cal) last executed after at least a 20 minute warm-up period at an ambient temperature within ±5° C of the current ambient temperature.

Table A–1: Signal acquisition system

Characteristic	Description	
Accuracy, DC Gain	±1.5% for full scale ranges from 20 mV to 100 V ±2.0% for full scale ranges <20 mV	
Accuracy, DC Voltage Measurement	±(1.5% of input signal + 1% of full scale range) with instrument temperature within 5° C of the temperature when last Self Cal’ed and for input ranges ≥50 mV full scale	
Accuracy, Delta DC Voltage Measurement	±(1.5% of input signal + 0.1% of full scale range) with instrument temperature within 5° C of the temperature when last Self Cal’ed	
Accuracy, Offset[1]	*Full Scale Range Setting*	*Offset Accuracy*
	10 mV – 1 V	±[(0.2% × | offset |) + 1.5 mV + (6% × full scale range)]
	1.01 V – 10 V	±[(0.25% × | offset |) + 15 mV + (6% × full scale range)]
	10.1 V – 100 V	±[(0.25% × | offset |) + 150 mV + (6% × full scale range)]

[1] **Net offset is the nominal voltage level at the waveform analyzer input that corresponds to the center of the A/D-Converter dynamic range. Offset accuracy describes the precision of the net offset voltage.**

Table A–1: Signal acquisition system (cont.)

Characteristic	Description		
Analog Bandwidth, DC–50 Ω Coupled or DC–1 MΩ Coupled	*Full Scale Range Setting*	*Bandwidth*[2]	
	10.1 V – 100 V	DC – 500 MHz (TVS625, TVS625A, TVS645 and TVS645A) DC – 250 MHz (TVS621, TVS621A, TVS641 and TVS641A)	
	100 mV – 10 V	DC – 1 GHz (TVS625, TVS625A, TVS645 and TVS645A) DC – 250 MHz (TVS621, TVS621A, TVS641 and TVS641A)	
	50 mV – 99.8 mV	DC – 750 MHz (TVS625, TVS625A, TVS645 and TVS645A) DC – 250 MHz (TVS621, TVS621A, TVS641 and TVS641A)	
	20 mV – 49.8 mV	DC – 600 MHz (TVS625, TVS625A, TVS645 and TVS645A) DC – 250 MHz (TVS621, TVS621A, TVS641 and TVS641A)	
	10 mV – 19.8 mV	DC – 500 MHz (TVS625, TVS625A, TVS645 and TVS645A) DC – 250 MHz (TVS621, TVS621A, TVS641 and TVS641A)	
Bandwidth, Analog, Selections	20 MHz, 250 MHz, and FULL		
Calculated Rise Time, typical[3] Typical full-bandwidth rise times are shown in the chart to the right	*Full Scale Range Setting*	*TVS621, TVS621A, TVS641 and TVS641A*	*TVS625, TVS625A, TVS645 and TVS645A*
	10.1 V – 100 V	900 ps	1.8 ns
	100 mV – 10 V	450 ps	1.8 ns
	50 mV – 99 mV	600 ps	1.8 ns
	20 mV – 49.9 mV	750 ps	1.8 ns
	10 mV – 19.9 mV	900 ps	1.8 ns
Crosstalk (Channel Isolation)	≥300:1 at 100 MHz and ≥100:1 at the rated bandwidth for the channel's sensitivity (Full Scale Range) setting, for any two channels having equal sensitivity settings		
Delay Between Channels, Full Bandwidth	≤100 ps with equal Full Scale Range and Coupling settings		

[2] **The limits given are for the ambient temperature range of 0° C to +30° C. Reduce the upper bandwidth frequencies by 5 MHz for each °C above +30° C. The bandwidth must be set to FULL.**

[3] **Rise time (rounded to the nearest 50 ps) is calculated from the bandwidth when Full Bandwidth is selected. It is defined by the following formula:**

Rise Time (ns) = 450 ÷ BW (MHz)

Table A–1: Signal acquisition system (cont.)

Characteristic	Description	
Frequency Limit, Upper, 20 MHz Bandwidth Limited, typical	20 MHz	
Frequency Limit, Upper, 250 MHz Bandwidth Limited, typical	180 MHz	
Input Channels, Number of	*Product*	*Channels*
	TVS641, TVS641A, TVS645 and TVS645A	Four
	TVS621, TVS621A, TVS625 and TVS625A	Two
Input Coupling	DC, AC, or GND[4]	
Input Impedance, DC–1 MΩ Coupled	1 MΩ ±0.5% in parallel with 10 pF ±3 pF	
Input Impedance Selections	1 MΩ or 50 Ω	
Input Resistance, DC–50 Ω Coupled	50 Ω ±1%	
Input VSWR, DC–50 Ω Coupled	≤1.3:1 from DC – 500 MHz, ≤1.5:1 from 500 MHz – 1 GHz	
Input Voltage, Maximum, DC–1 MΩ, AC–1 MΩ, or GND Coupled	The greater of ±300 Vrms or 420 Vpeak DC, derated at 20 dB/decade above 1 MHz CAT II (see *Installation Category Descriptions* on page A–14 for more information)	
Input Voltage, Maximum, DC–50 Ω or AC–50 Ω Coupled	5 V_{RMS}, with peaks ≤ ±25 V	
Lower Frequency Limit, AC Coupled, typical	≤10 Hz when AC–1 MΩ Coupled; ≤200 kHz when AC–50 Ω Coupled[5]	
Random Noise	*Bandwidth Selection*	*RMS Noise*
	Full	≤(350 µV + 0.5% of the Full Scale Range setting)
	250 MHz	≤(165 µV + 0.5% of the Full Scale Range setting)
	20 MHz	≤(75 µV + 0.5% of the Full Scale Range setting)
Range, Offset	*Full Scale Range Setting*	*Offset Range*
	10 mV – 1 V	±1 V
	1.01 V – 10 V	±10 V
	10.1 V – 100 V	±100 V

[4] **GND input coupling disconnects the input connector from the attenuator and connects a ground reference to the input of the attenuator.**

[5] **The AC Coupled Lower Frequency Limits are reduced by a factor of 10 when 10X passive probes are used.**

Table A–1: Signal acquisition system (cont.)

Characteristic	Description				
Range, Sensitivity (Full Scale Range), All Channels	10 mV to 100 V[6]				
Step Response Settling Errors, typical[7] The maximum absolute difference between the value at the end of a specified time interval after the mid-level crossing of the step, and the value one second after the mid-level crossing of the step, expressed as a percentage of the step amplitude. See IEEE std. 1057, Section 4.8.1, *Settling Time Parameters*.	*Full Scale Range Setting*	*± Step Response*	*Maximum Settling Error (%) at* *20 ns*	*100 ns*	*20 ms*
	10 mV – 1 V	≤2 V	0.5%	0.2%	0.1%
	1.01 V – 10 V	≤20 V	1.0%	0.5%	0.2%
	10.1 V – 100 V	≤200 V	1.0%	0.5%	0.2%

[6] **The sensitivity ranges are 10 mV to 100 V full scale, switching in a 1–2–5 sequence of coarse settings. Between these coarse settings, you can adjust the sensitivity with a resolution equal to 1% of the more sensitive coarse setting. For example, between the 500 mV and 1 V ranges, the sensitivity can be set with 5 mV resolution.**

[7] **The Full Bandwidth settling errors are typically less than the percentages from the table.**

Table A–2: Timebase system

Characteristic	Description
Accuracy, Long Term Sample Rate and Delay Time	±100 ppm over any interval ≥1 ms
Accuracy, Trigger-to-Trigger Measurement (TVS600A models only)	±100 ps + (timebase accuracy x reading))
Range, Extended Realtime Sampling Rate	5 S/s to 10 MS/s in a 1–2.5–5 sequence
Range, Realtime Sampling Rate	
TVS625, TVS625A, TVS645 and TVS645A	20 MS/s to 5 GS/s on all channels simultaneously in a 1–2.5–5 sequence
TVS621, TVS621A, TVS641 and TVS641A	20 MS/s to 1 GS/s on all channels simultaneously in a 1–2.5–5 sequence
Record Length	256, 512, 1024, 2048, 4096, 8192, 15,000 30,000 (extended realtime sampling mode only)
Time Stamping	125 ns resolution 0.1% variance
Digitized Bits, Number of	8 bits

Table A–3: Trigger system

Characteristic	Description		
Accuracy (Time) for Pulse Glitch or Pulse Width Triggering	*Time Range*	*Accuracy*	
TVS600 models only	1 ns to 1 µs	±(20% of Setting + 0.5 ns)	
	1.02 µs to 1 s	±(204.5 ns + 0.01% of Setting)	
TVS600A models only: limits are valid when using a time-base reference frequency of 10 MHz ±0.01%	2 ns to 500 ns	±(20% of Setting + 0.5 ns)	
	520 ns to 1 s	±(104.5 ns + 0.01% of Setting)	
Accuracy (DC) for External Trigger Level	±(5% + 150 mV) for signals having rise and fall times ≥20 ns		
Accuracy (DC) for Internal Trigger Level, DC Coupled	±[(2% × | Setting) | + 0.03 of Full Scale Range + Offset Accuracy)] for signals having rise and fall times ≥20 ns		
Holdoff, Variable Main Trigger, typical[1]	For all sampling rates, the minimum holdoff is 250 ns and the maximum holdoff is 12 s; the minimum resolution is 8 ns for settings ≤1.2 µs		
Input, External Trigger, typical	50 Ω input resistance; ±5 V (DC + peak AC) maximum safe input voltage; DC coupled only		
Range, Delayed Trigger Time[2]	16 ns to 250 s		
Range, Delta-Time, Slew-Rate Triggering	1 ns to 1 second		
Range, Events Delay	1 to 10,000,000		
Ranges, (Setup/Hold Times) for Setup/Hold Violation Trigger	*Feature*	*Minimum*	*Maximum*
	Setup time[3]	–100 ns	100 ns
	Hold time[4]	1 ns	102 ns
	Setup + Hold time[5]	2 ns	NA
Range (Time) for Pulse Glitch and Pulse Width Triggering	2 ns to 1 s		
Range (Time) for Time-Qualified Runt Triggering (TVS600A models only)	2 ns to 1 s		

[1] **Main Trigger is controlled with the TRIGger:A commands.**

[2] **Delayed Trigger is controlled with the TRIGger:B commands.**

[3] **Positive numbers define times before the clock edge and negative numbers define times after the clock edge.**

[4] **Positive numbers define times after the clock edge and negative numbers define times before the clock edge.**

[5] **The algebraic sum of the setup and hold time set by the user.**

Table A–3: Trigger system (cont.)

Characteristic	Description		
Range, Trigger Level	*Source*	*Range*	
	Any Channel External Input	±100% of full scale range ±1 V	
Range, Trigger Point Position	Minimum: 0 Maximum: 30,000		
Resolution, Trigger Level	0.02% of full scale for any Channel source and 2 mV for the External Input source		
Resolution, Trigger Position	One sample interval at all sample rates		
Sensitivities, Events Delay, DC Coupled, typical	10% of full scale, from DC to 500 MHz, for Full Scale Range settings >100 mV and ≤10 V at the BNC input		
Sensitivities, Logic-Type Trigger (TVS600A models only)	10% of full scale, from DC to 500 MHz, for Full Scale Range settings >100 mV and ≤10 V at the BNC input		
Sensitivities, Pulse-Type Runt Trigger, typical	10% of full scale, from DC to 500 MHz, for Full Scale Range settings >100 mV and ≤10 V at the BNC input		
Sensitivities, Pulse-Type Trigger Width and Glitch, typical	10% of full scale, for Full Scale Range settings >100 mV and ≤10 V at the BNC input		
Sensitivity, Edge-Type Trigger, DC Coupled[6]	The minimum signal levels required for stable edge triggering of an acquisition when the source is DC-coupled.		
	Products	*Trigger Source*	*Sensitivity*
	TVS625, TVS625A, TVS645 and TVS645A	Any Channel	3.5% of Full Scale Range from DC to 50 MHz, increasing to 10% of Full Scale Range at 1 GHz
	TVS621, TVS621A, TVS641 and TVS641A	Any Channel	3.5% of Full Scale Range from DC to 50 MHz, increasing to 10% of Full Scale Range at 250 MHz
	All models	External	25 mV from DC to 50 MHz, increasing to 50 mV at 100 MHz

[6] **Delayed Trigger has the same specifications as Main Trigger.**

Table A–3: Trigger system (cont.)

<table>
<tr><th>Characteristic</th><th colspan="12">Description</th></tr>
<tr><td>Sensitivity, Edge-Type Trigger, Not DC Coupled, typical</td><td colspan="6"><i>Trigger Coupling</i></td><td colspan="6"><i>Typical Signal Level for Stable Triggering</i></td></tr>
<tr><td></td><td colspan="6">AC</td><td colspan="6">Same as the DC-coupled limits for frequencies above 60 Hz; attenuates signals below 60 Hz</td></tr>
<tr><td></td><td colspan="6">High Frequency Reject</td><td colspan="6">One and one-half times the DC-coupled limits from DC to 30 kHz; attenuates signals above 30 kHz</td></tr>
<tr><td></td><td colspan="6">Low Frequency Reject</td><td colspan="6">One and one-half times the DC-coupled limits for frequencies above 80 kHz; attenuates signals below 80 kHz</td></tr>
<tr><td></td><td colspan="6">Noise Reject</td><td colspan="6">Three times the DC-coupled limits</td></tr>
<tr><td>Time, Minimum Pulse or Rearm, and Minimum Transition Time, for Pulse-Type Triggering, typical</td><td colspan="12">For Full Scale Range settings >100 mV and ≤10 V at the BNC input</td></tr>
<tr><td></td><td colspan="4"><i>Pulse Class</i></td><td colspan="4"><i>Minimum Pulse Width</i></td><td colspan="4"><i>Minimum Rearm Width</i></td></tr>
<tr><td></td><td colspan="4">Glitch</td><td colspan="4">2 ns</td><td colspan="4">2 ns + 5% of Glitch Width Setting</td></tr>
<tr><td></td><td colspan="4">Width</td><td colspan="4">1 ns</td><td colspan="4">2 ns + 5% of Width Upper Limit Setting</td></tr>
<tr><td>TVS600A models only</td><td colspan="4">Runt
Time-qualified</td><td colspan="4">2 ns
2 ns</td><td colspan="4">2 ns
8.5 ns + 5% of Width Setting</td></tr>
<tr><td>TVS600A models only</td><td colspan="4">Slew Rate</td><td colspan="4">600 ps[7]</td><td colspan="4">8.5 ns + 5% of Delta Time Setting</td></tr>
<tr><td>Time, Minimum Pulse or Rearm, for Events Delay Triggering, typical</td><td colspan="12">The following chart shows the minimum values for input range settings >100 mV and ≤10 V at the BNC input</td></tr>
<tr><td></td><td colspan="3"><i>Triggering Type</i></td><td colspan="3"><i>Minimum Pulse Width</i></td><td colspan="3"><i>Minimum Rearm Time</i></td><td colspan="3"><i>Minimum Time Between Channels[8]</i></td></tr>
<tr><td></td><td colspan="3">Events Delay</td><td colspan="3">1 ns (for either + or – pulsewidths)</td><td colspan="3">N/A</td><td colspan="3">2 ns</td></tr>
<tr><td></td><td colspan="3">Logic</td><td colspan="3">N/A</td><td colspan="3">1 ns</td><td colspan="3">1 ns</td></tr>
</table>

[7] For slew rate triggering, this is actually the minimum transition time defined t be the time the test signal spends between the two trigger threshold settings.

[8] For Events Delay, the time is the minimum time between a main and delayed event that will be recognized if more than one channel is used. For Logic, time between channels refers to the length of time a logic state derived from more than one channel must exist to be recognized.

Table A–3: Trigger system (cont.)

Characteristic	Description	
Trigger Position Error, Edge Triggering, typical	*Acquisition Mode*	*Trigger Position Error*[9]
	Sample, Average	±(1 Sample Interval + 1 ns)
	Envelope	±(2 Sample Intervals + 2 ns)

[9] **The trigger position errors are typically less than the values given here. These values are for triggering signals having a slew rate at the trigger point of ≥5% of full scale/ns.**

Table A–4: Front-panel connectors

Characteristic	Description		
Arm Input	This input provides external arming capability with a BNC connector		
	Characteristic	*Limits*	
	Arming Threshold Voltage	≤0.8 V	
	Input Voltage Range	0 to 5 V_{pk}, TTL-compatible (arms on a switch closure to ground; internal pull-up resistor to +5 volts is provided)	
	Latency	10 µs	
	Minimum Pulsewidth	10 µs	
Fiducial Input, typical[1]	This input provides fiducial input capability with a BNC connector; the polarity of the signal acquired is inverted with respect to the input		
	Characteristic	*Limits*	
	Fiducial Amplitude	75 mV to 207 mV	
	Input Impedance	0.01 µF in series with 50 Ω	
	Input LF Attenuation	Attenuates signals below 100 MHz (high-pass time constant of 5 ns)	
	Input Sensitivity	*CH 1 Full Scale Range*	*Fiducial Full Scale Range*
		10 mV to 1 V	6 times the CH 1 Full Scale Range setting
		1.01 V to 10 V	0.6 times the CH 1 Full Scale Range setting
		10.1 V to 100 V	0.06 times the CH 1 Full Scale Range setting
	Input Voltage Range	±1 V	
	Maximum Input	2 V_{RMS}	
	Rise Time	*Products*	*Rise Time*
		TVS625, TVS625A, TVS645 and TVS645A	≤3.5 ns (10% to 90%)
		TVS621, TVS621A, TVS641 and TVS641A	≤4.0 ns (10% to 90%)

[1] **The FIDUCIAL Input signal should be a short-duration (≤3 ns), fast rise time (≤2 ns) pulse.**

Table A–4: Front-panel connectors (cont.)

Characteristic	Description	
Output, Reference	*Characteristic*	*Limits*
	Output Voltage	8 V ±1%
	Internal Frequency Reference	Frequency is 10 MHz ±0.015% Amplitude is ≥1 $V_{p\text{-}p}$ into 50 Ω
Probe Compensation, Output Frequency, typical	1 kHz ±25%	
Probe Compensation, Output Voltage	0.5 V (base-top) ±1% into a ≥50 Ω load	
Serial Interface	This front panel-mounted 9-pin D connector provides a serial interface with the following pin assignments: 1 DCD 2 RXD 3 TXD 4 DTR 5 GND 6 DSR 7 RTS 8 CTS 9 No Connection	

Table A–5: VXI interface

<table>
<tr><th>Characteristic</th><th colspan="2">Description</th></tr>
<tr><td>Addressing</td><td colspan="2">Dynamic autoconfigure</td></tr>
<tr><td>Inputs, ECLTRG</td><td colspan="2">Either of the two ECLTRG lines may be individually selected to arm or trigger an acquisition. Arming can occur on either sense of the ECL signal. Trigger can be specified to occur on either the high-to-low transition or the low-to-high transition.</td></tr>
<tr><td>Inputs, TTLTRG</td><td colspan="2">Any of the eight TTLTRG lines may be individually selected to arm or trigger an acquisition. Arming can occur on either sense of the TTL signal. Trigger can be specified to occur on either the high-to-low transition or the low-to-high transition.</td></tr>
<tr><td>Interface Type</td><td colspan="2">Message based (I4)</td></tr>
<tr><td>Interrupts</td><td colspan="2">Programmable interrupter level 1–7</td></tr>
<tr><td>Outputs, ECLTRG</td><td colspan="2">Either of the two ECLTRG lines can be driven by the following signals:
ARM — The waveform analyzer is armed and waiting for a trigger
ATR — Main trigger event has occurred
BTR — Delayed trigger event has occurred
CALC — DSP Calc function “TRG ()” evaluated TRue
OPC — Operation pending complete</td></tr>
<tr><td>Outputs, TTL</td><td colspan="2">Each of the TTLTRG lines (TTLTRG0*–TTLTRG7*) can be driven by the following signals:
ARM — The waveform analyzer is armed and waiting for a trigger
ATR — Main trigger event has occurred
BTR — Delayed trigger event has occurred
CALC — DSP Calc function “TRG ()” evaluated TRue
OPC — Operation pending complete</td></tr>
<tr><td rowspan="4">Outputs, TTLTRG, Logic Levels</td><td colspan="2">Based on the VXIbus Specification RULE B.6.17</td></tr>
<tr><td>*Characteristic*</td><td>*Limits*</td></tr>
<tr><td>Vout(HI)</td><td>Determined by the mainframe termination; the TTL outputs are open collector</td></tr>
<tr><td>Vout(LO)</td><td>≤0.6 V when sinking 48 mA</td></tr>
<tr><td>Outputs, TTLTRG, Logic Polarity</td><td colspan="2">Normal polarity: Negative TRUE; high-to-low transition indicates the event occurred
Inverted polarity: Positive TRUE; low-to-high transition indicates the event occurred</td></tr>
<tr><td>Protocols</td><td colspan="2">Word Serial (WSP)
Fast Data Channel FDC TEK V2.1</td></tr>
<tr><td>VXI</td><td colspan="2">Complies with revision 1.4</td></tr>
</table>

Table A–6: Power distribution and data handling

Characteristic	Description		
Current Requirements, TVS641 and TVS641A and TVS645 and TVS645A typical	*Voltage*	*DC Current*	*Dynamic Current*
	+12 V	1.3 A	0.45 A
	+5 V	11.0 A	0.8 A
	–5.2 V	4.6 A	0.09 A
	–12 V	1.0 A	0.4 A
Current Requirements, TVS621 and TVS621A and TVS625 and TVS625A typical	*Voltage*	*DC Current*	*Dynamic Current*
	+12 V	1.0 A	0.45 A
	+5 V	8.0 A	0.7 A
	–5.2 V	2.7 A	0.05 A
	–12 V	0.8 A	0.4 A
Nonvolatile Memory Retention Time, typical[1]	Battery life is ≥5 years		
Power Requirements, typical	*Products*	*Power Requirements*	
	TVS641, TVS641A, TVS645 and TVS645A	106.5 Watts	
	TVS621, TVS621A, TVS625 and TVS625A	75.6 Watts	

Table A–7: Environmental

Characteristic	Description
Airflow Resistance	≤0.204 mm H_2O air pressure with 6.6 l/s airflow
Altitude, Operating and Nonoperating	Operating: to 15,000 feet (4570 m) Nonoperating: to 40,000 feet (12,190 m)
Classification	This instrument is suitable for operation in Tektronix and MIL-T-28800E Class 5 environments, provide that it is operated in a mainframe which has been qualified for Class 5 environments and which imparts stresses to this module within the limits specified in this table. Nonoperating: to 40,000 feet (12,190 m)
Humidity, Operating and Nonoperating	To 95% relative humidity at or below +30° C; to 45% relative humidity up to +50° C
Temperature, Operating and Nonoperating	Operating: 0° C to +50° C for exterior air when operated in a mainframe with 15° C internal temperature rise Nonoperating: –40° C to +71° C

Table A–8: Certifications and compliances

Characteristic	Description
EC Declaration of Conformity – EMC	Meets intent of Directive 89/336/EEC for Electromagnetic Compatibility. Compliance was demonstrated to the following specifications as listed in the Official Journal of the European Communities[1]: EN 50081-1 Emissions: EN 55011 Class A Radiated and Conducted Emissions EN 60555-2 AC Power Line Harmonic Emissions EN 50082-1 Immunity: IEC 801-2 Electrostatic Discharge Immunity IEC 801-3 RF Electromagnetic Field Immunity IEC 801-4 Electrical Fast Transient/Burst Immunity IEC 801-5 Power Line Surge Immunity[2] AS/NZS 2064, Australian emissions standard for Industrial, Scientific, and Medical Equipment[2] [1] To assure the product operates in conformance with the specifications listed above, the product must be used in a mainframe that is equipped with backplane shields that comply with Rule B.7.US of the VXI specification. [2] TVS600A models only.
EMC Compliance	Meets the intent of Directive 89/336/EEC for Electromagnetic Compatibility when it is used with the product(s) stated in the specifications table. Refer to the EMC specification published for the stated products. May not meet the intent of the Directive if used with other products.

Table A–8: Certifications and compliances (cont.)

Characteristic	Description
EC Declaration of Conformity – Low Voltage	Compliance was demonstrated to the following specification as listed in the Official Journal of the European Communities: Low Voltage Directive 73/23/EEC EN 61010-1:1993 Safety requirements for electrical equipment for measurement, control, and laboratory use
Approvals	UL3111-1 – Standard for electrical measuring and test equipment CAN/CSA C22.2 No. 1010.1 – Safety requirements for electrical equipment for measurement, control and laboratory use
Pollution Degree 2 (TVS600 models)	Do not operate in environments where conductive pollutants may be present.
Safety Certification of Plug-in or VXI Modules	For modules (plug-in or VXI) that are safety certified by Underwriters Laboratories, UL Listing applies only when the module is installed in a UL Listed product. For modules (plug-in or VXI) that have cUL or CSA approval, the approval applies only when the module is installed in a cUL or CSA approved product.
Installation Category Descriptions	Terminals on this product may have different installation category designations. The installation categories are: CAT III Distribution-level mains (usually permanently connected). Equipment at this level is typically in a fixed industrial location CAT II[3] Local-level mains (wall sockets). Equipment at this level includes appliances, portable tools, and similar products. Equipment is usually cord-connected CAT I Secondary (signal level) or battery operated circuits of electronic equipment [3] See *Input Voltage, Maximum, DC–1 MΩ, AC–1 MΩ, or GND Coupled* on page A–3)

Table A–9: Mechanical

Characteristic	Description	
Construction Material	Chassis parts constructed of aluminum alloy; front panel constructed of plastic laminate; circuit boards constructed of glass laminate; cabinet is aluminum	
Weight	*Products*	*Weight*
	TVS641, TVS641A, TVS645 and TVS645A	2.6 kg (5 lbs 12 oz)
	TVS621, TVS621A, TVS625 and TVS625A	2.5 kg (5 lbs 8 oz)
Overall Dimensions	Height: 262 mm (10.3 in) Width: 61 mm (2.4 in) Depth: 373 mm (14.5 in)	

Appendix B: Algorithms

The waveform analyzer can take many automatic measurements and perform a variety of other calculations. By knowing how they make these calculations, you may better understand how to use your waveform analyzer and how to interpret the results.

Measurement Variables

The waveform analyzer uses a variety of variables in its calculations. This section discusses each variable and how to set it.

High and Low

High is the value used as the 100% level in measurements such as fall time and rise time. For example, if you request the 10% to 90% rise time, then the waveform analyzer calculates 10% and 90% as percentages with High representing 100%.

Low is the value used as the 0% level in measurements such as fall time and rise time.

The exact meaning of High and Low depends on which calculation method you choose. To set the method used to determine High you use the CALCulate:WMParameter:HMEThod command. To set the method used to determine Low you use the CALCulate:WMParameter:LMEThod command. The methods are PEAK, MODE, AUTO, and ABSolute.

PEAK defines the 0% and the 100% waveform levels as the lowest amplitude (most negative) and the highest amplitude (most positive) samples. The PEAK method is useful for measuring frequency, width, and period for many types of signals. PEAK is sensitive to waveform ringing and spikes, however, and does not always accurately measure rise time, fall time, overshoot, and undershoot.

MODE attempts to find the highest density of points above and below the waveform midpoint. It attempts to ignore ringing and spikes when determining the 0% and 100% levels. This method works well when measuring square waves and pulse waveforms.

The waveform analyzer calculates the histogram-based High and Low values as follows:

1. It makes a histogram of the record with 256 bins.

2. It splits the histogram into two sections at the halfway point between Min and Max (also called Mid).

3. The level with the most points in the upper histogram is the High value, and the level with the most points in the lower histogram is the Low value. (Choose the levels where the histograms peak for High and Low.)

   If Mid gives the largest peak value within the upper or lower histogram, then return the Mid value for both High and Low (this is probably a very low amplitude waveform).

   If more than one histogram level (bin) has the maximum value, choose the bin farthest from Mid.

This algorithm does not work well for two-level waveforms with greater than about 100% overshoot.

AUTO attempts to use the MODE method but will switch to the PEAK method if the histogram does not show an obvious consistent high level. For example, the MODE histogram operating on a triangle wave would not find significant High and Low levels so AUTO would select PEAK. On a square wave AUTO would pick the MODE method.

ABSolute uses the absolute value set with the commands CALCulate:WMParameter:HIGH or CALCulate:WMParameter:LOW.

Measurement Reference Levels

You can set the various reference levels used to take the automated measurements. You can choose to set reference levels in absolute vertical units or in relative units of percent or ratio. Use the command CALCulate:WMParameter:RMEThod to choose the reference method. The reference levels are as follows:

HREFerence the waveform high reference or distal level. Used in fall time and rise time calculations. In the RELative mode, you use the command CALCulate:WMParameter:HREFerence:RELative to set it from 0% to 100%, with a reset value of 90%. You can set it to a voltage level with the command CALCulate:WMParameter:HREFerence.

MREFerence the waveform middle reference or mesial level. In the RELative mode, you use the command CALCulate:WMParameter:MREFerence:RELative to set MREFerence from 0% to 100%, with a reset value of 50%. You can set it to a voltage level with the command CALCulate:WMParameter:MREFerence. You can also specify a hysteresis value for the MREFerence that reduces the effects of noise on measurements. The HYSTeresis value is a percent or ratio of the AMPLitude value.

LREFerence the waveform low reference or proximal level. Used in fall time and rise time calculations. In the RELative mode, you use the command CALCulate:WMParameter:LREFerence:RELative to set it from 0% to 100%, with a reset value of 10%. You can set it to a voltage level with the command CALCulate:WMParameter:LREFerence.

Other Variables

The waveform analyzer also measures several values itself that it uses to help calculate measurements.

RecordLength is the number of data points in the waveform record.

Hysteresis reduces the effects of noise on measurements by providing a guard band of 5% (the default) of the waveform amplitude, above and below the midpoint value. Hysteresis can be set in the range 0% to 50%. It is used in MCross1, MCross2, and MCross3 calculations.

For example, once a crossing has been measured in a negative direction, the waveform data must fall below 5% of the amplitude from the MidRef point before the measurement system is armed and ready for a positive crossing. Similarly, after a positive MidRef crossing, waveform data must exceed 5% of the amplitude before a negative crossing can be measured. Hysteresis is useful when you are measuring noisy signals, because it allows the waveform analyzer to ignore minor fluctuations in the signal.

MCross Calculations

MCross1, MCross2, and MCross3 refer to the first, second, and third MidRef cross times, respectively. (See Figure B–1.)

The polarity of the crossings does not matter for these variables, but the crossings alternate in polarity; that is, MCross1 could be a positive or negative crossing, but if MCross1 is a positive crossing, MCross2 will be a negative crossing.

The waveform analyzer calculates these values as follows:

1. Find the first MidRefCrossing in the waveform record (or the gated region). This is MCross1.
2. Continuing from MCross1, find the next MidRefCrossing in the waveform record (or the gated region) of the opposite polarity of MCross1. This is MCross2.
3. Continuing from MCross2, find the next MidRefCrossing in the waveform record (or the gated region) of the same polarity as MCross1. This is MCross3.

MCross1Polarity is the polarity of first crossing (no default). It can be rising or falling.

StartCycle is the starting time for cycle measurements. It is a floating-point number with values between 0.0 and (RecordLength – 1.0), inclusive.

StartCycle = MCross1

EndCycle is the ending time for cycle measurements. It is a floating-point number with values between 0.0 and (RecordLength – 1.0), inclusive.

EndCycle = MCross3

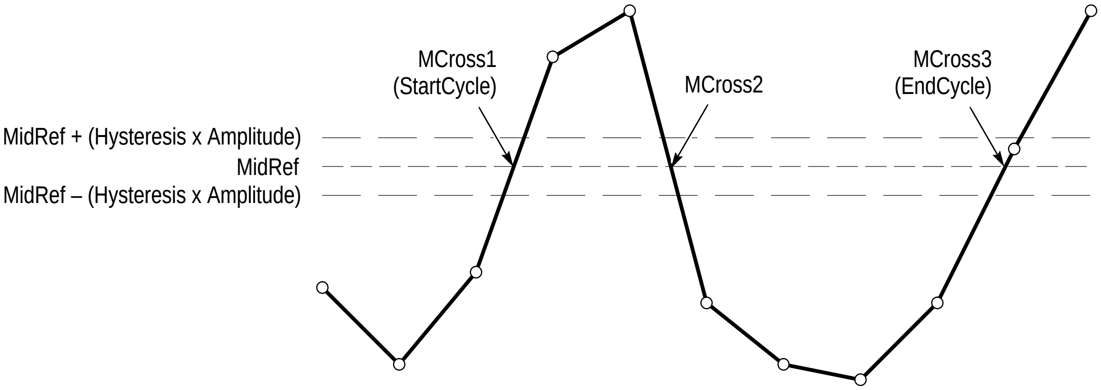

MCross1
(StartCycle)
MCross2
MCross3
(EndCycle)
MidRef + (Hysteresis x Amplitude)
MidRef
MidRef – (Hysteresis x Amplitude)

Figure B–1: MCross calculations

Waveform[<0.0 ... RecordLength–1.0>] — holds the acquired data.

Measurement Zone and Edge Selection

The waveform analyzer can take measurements over the entire waveform or over a user-specified measurement zone. Some measurements also require that the user specify the waveform edge to be measured. Usage of the commands needed to specify zoned (also called gated) measurements is discussed in the section *Measurements* in chapter 3. This section briefly describes each zoned measurements and waveform edge selection.

Zoned Measurements

The `CALCulate<n>:WMParameter:GATE` commands control whether zoned measurements are on or off:

- When `CALC<n>:WMP:GATE` is set to `ON`, measurements are taken within a measurement zone defined using other parameters to the `:GATE` commands.

- When CALC<n>:WMP:GATE is set to OFF, the measurement zone becomes the entire waveform record. The measurement zone defined by the :GATE command parameters is ignored.

All measurements included in the CALC block are affected by these commands, but gating for each CALC block is settable independently.

Edge Selection

The CALCulate<n>:WMParameter:EDGE command selects the waveform edge that the measurement is taken on for these edge-based measurements:

- CROSs, NCROSs, and PCROSs
- DELay
- FTIMe and RTIMe
- COPulse

The CALCulate<n>:WMParameter:SLOPE command selects the polarity of the edges used in DELay measurements; the polarity of the edge used for the other edge measurements is determined implicitly (for example, RTIME use positive slope, etc.).

When CALC:WFM:GATE is set to OFF, the waveform analyzer finds the edge within the entire waveform record; when set to ON, the waveform analyzer finds the edge with in the measurement zone.

Measurement Algorithms

The automated measurements are defined and calculated as follows.

Amplitude

Amplitude = High – Low

Area

The arithmetic area of one waveform or of its measurement zone. Remember that one waveform is not necessarily equal to one cycle. For cyclical data you may prefer to use the cycle area rather than the arithmetic area.

if Start = End then return the (interpolated) value at Start.

Otherwise,

$$Area = \int_{Start}^{End} Waveform(t)\mathrm{dt}$$

For details of the integration algorithm, see page B–17.

COPulse

Center-Of-Pulse timing-measurement. Returns the average of the times, relative to the trigger point, of the three measurement reference levels on the pulse leading edge that contains MCRoss1 and the three levels on the pulse trailing edge containing MCRoss2. See *Mcross Calculations* on page B–3 for definitions of crossings. See *Measurement Reference Levels* on page B–2 for descriptions of the reference levels.

COPulse used the edge index set by `CALCulate<n>:WMParameter:EDGE` as the leading edge and next edge of opposite polarity as the trailing edge.

Cross

Timing measurement. The time relative to the trigger point at which the crossing that you specify occurs. The CROSs measurement searches for the N^{th} occurrences of an edge; during the search it counts edges of either polarity.

1. Searching from Start to End of waveform record, find the first transition — either negative-going or positive-going — through MREF (middle ref).
2. Continue the search process until the N^{th} crossing is found (user specifies N using the `CALCulate:WMParameter:EDGE` command).
3. Cross = Time@crossing, where Time@trigger = 0.

   Pretrigger crossings return negative times; posttrigger crossings return positive times.

Positive values for N force the search at the start of the waveform record; Negative values and zero at the end (zero designates the last crossing, –1 the next to the last crossing, and so on in the waveform record).

Cycle Area

Amplitude (voltage) measurement. The area over one waveform cycle. For non-cyclical data, you might prefer to use the Area measurement.

If StartCycle = EndCycle then return the (interpolated) value at StartCycle.

$$CycleArea = \int_{StartCycle}^{EndCycle} Waveform(t)\mathrm{dt}$$

For details of the integration algorithm, see page B–17.

Cycle Mean

Amplitude (voltage) measurement. The mean over one waveform cycle. For non-cyclical data, you might prefer to use the Mean measurement.

If StartCycle = EndCycle then return the (interpolated) value at StartCycle.

$$CycleMean = \frac{\int_{StartCycle}^{EndCycle} Waveform(t)\mathrm{dt}}{(EndCycle - StartCycle) \times SampleInterval}$$

For details of the integration algorithm, see page B–17.

Cycle RMS

The true Root Mean Square voltage over one cycle.

If StartCycle = EndCycle then CycleRMS = Waveform[Start].

Otherwise,

$$CycleRMS = \sqrt{\frac{\int_{StartCycle}^{EndCycle} (Waveform(t))^2\mathrm{dt}}{(EndCycle - StartCycle) \times SampleInterval}}$$

For details of the integration algorithm, see page B–17.

Delay

Timing measurement. The amount of time between the *MidRef* crossings of two different traces or two different places on the same trace.

Delay measurements are actually a group of measurements. To get a specific delay measurement, you must specify the target and reference crossing polarities and edges.

Gain

Ratio of two amplitudes measurement. An amplitude measurement (see *Amplitude* on page B–5) is taken of the reference and target waveforms.

$$Gain = \frac{Amplitude_{target}}{Amplitude_{refernece}}$$

Fall Time

Timing measurement. The time taken for the falling edge of a pulse to drop from a HighRef value (default = 90%) to a LowRef value (default = 10%).

Figure B–2 shows a falling edge with the two crossings necessary to calculate a fall time measurement.

1. Based on the index n specified by the `CALCulate:WMParameter:EDGE` command, count edges according to the following rules:

   - Positive arguments search forward from the start-to-end of the waveform record; zero and negative arguments search backward from the end-to-start.
   - Zero specifies the last edge in the waveform record.
   - Count only edges that go through both HREF and LREF (see Figure B–2).

2. Continue the search process until the nth negative-going edge is found (user specifies n using the `CALCulate:WMParameter:EDGE` command).
3. Find the time of the HREF crossing for the nth edge. This time is THF. (Use linear interpolation if necessary.)
4. Find the time of the LREF crossing for the nth edge. This time is TLF. (Use linear interpolation if necessary.)
5. FallTime = TLF – THF

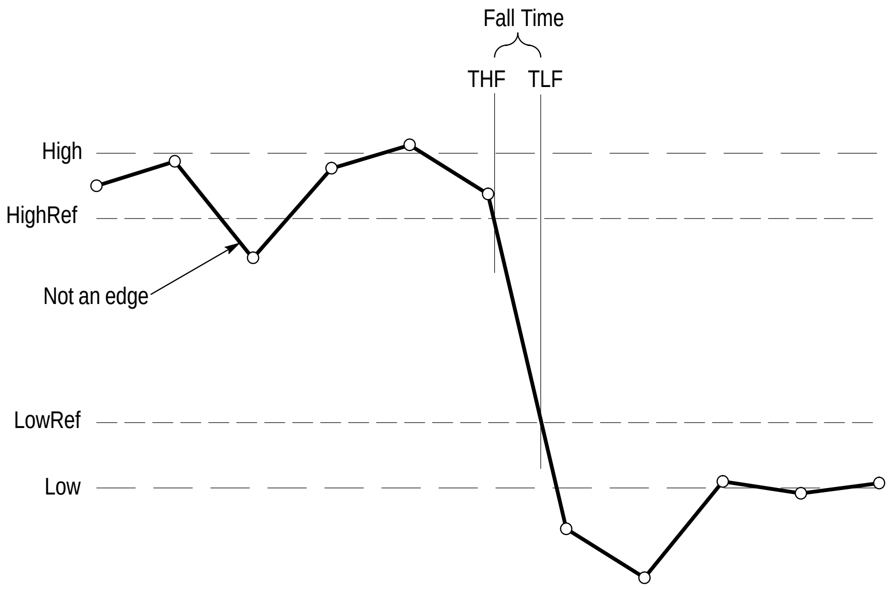


Figure B–2: Fall time

Frequency

Timing measurement. The reciprocal of the period. Measured in Hertz (Hz) where 1 Hz = 1 cycle per second.

If Period = 0 or is otherwise bad, return an error.

Frequency = 1/Period

High

100% (highest) voltage reference value. (See *High and Low* on page B–1.)

Using the min-max measurement technique:

High = Max

Low

0% (lowest) voltage reference value calculated. (See *High and Low* on page B–1.)

Using the min-max measurement technique:

Low = Min

Maximum

Amplitude (voltage) measurement. The maximum voltage. Typically the most positive peak voltage.

Examine all Waveform[ ] samples from Start to End inclusive, and set Max equal to the greatest magnitude Waveform[ ] value found.

Mcross1 Mcross2 Mcross3

Timing-measurement functions. These three measurements are of the times relative to the trigger point at which the first, second, and third Mid Ref crossings occur in the waveform. See *Mcross Calculations* on page B–3 for definitions of crossings.

Mcross1, Mcross2, and Mcross3 cannot be included as measurements in the measurement lists AAMList or WMList. Rather they can be used as functions in calculation expressions.

Pretrigger crossings return negative times; posttrigger crossings return positive times.

Mean

The arithmetic mean for one waveform. Remember that one waveform is not necessarily equal to one cycle. For cyclical data you may prefer to use the cycle mean rather than the arithmetic mean.

$$Mean = \sum_{i=0}^{i=RecordLength-1} \frac{Waveform\ [i]}{RecordLength}$$

Minimum

Amplitude (voltage) measurement. The minimum amplitude. Typically the most negative peak voltage.

Examine all Waveform[ ] samples from Start to End inclusive, and set Min equal to the smallest magnitude Waveform[ ] value found.

Negative Duty Cycle

Timing measurement. The ratio of the negative pulse width to the signal period expressed as a percentage.

NegativeWidth is defined in **Negative Width**, below.

If Period = 0 or undefined then return an error.

$$NegativeDutyCycle = \frac{NegativeWidth}{Period} \times 100\%$$

Negative Width

Timing measurement. The distance (time) between MidRef (default = 50%) amplitude points of a negative pulse.

If MCross1Polarity = ‘–’

then

NegativeWidth = (MCross2 – MCross1)

else

NegativeWidth = (MCross3 – MCross2)

Ncross

Timing measurement. The time relative to the trigger point at which the negative-going crossing that you specify occurs. The NCROSs measurement searches for the Nth occurrence of an edge; during the search it counts only negative edges.

1. Searching from Start to End of waveform record, find the first negative-going transition through MREF (middle reference).

2. Continue the search process until the Nth negative-going crossing is found (user specifies N using the `CALCulate:WMParameter:EDGE` command).

3. NCross = Time@ncrossing, whereTime@trigger = 0.

   Pretrigger crossings return negative times; posttrigger crossings return positive times.

Positive values for N force the search at the start of the waveform record; Negative values and zero, at the end (zero designates the last crossing, –1 the next to the last crossing, and so on in the waveform record).

Overshoot

Amplitude (voltage) measurement. Overshoot finds the first positive-going edge in the waveform record (or gated area) and calculates value as follows:

$$Overshoot = \frac{Max - High}{Amplitude} \times 100\%$$

This measurement ignores settings of the `CALCulate:WMParameter:EDGE` and :SLOPE commands when locating the edge to measure.

Pcross

Timing measurement. The time relative to the trigger point at which the positive-going crossing that you specify occurs. The PCROSs measurement searches for the Nth occurrence of an edge; during the search it counts only positive edges.

1. Searching from Start to End of waveform record, find the first positive-going transition through MREF (middle reference).

2. Continue the search process until the Nth positive-going crossing is found (user specifies N).

3. PCross = Time@pcrossing, whereTime@trigger = 0.

   Pretrigger crossings return negative times; posttrigger crossings return positive times.

Positive values for N force the search at the start of the waveform record; Negative values and zero, at the end (zero designates the last crossing, –1 the next to the last crossing, and so on in the waveform record).

Peak to Peak

Amplitude measurement. The absolute difference between the maximum and minimum amplitude.

PeaktoPeak = Max – Min

Period

Timing measurement. Time taken for one complete signal cycle. The reciprocal of frequency. Measured in seconds.

Period = MCross3 – MCross1

Phase

Timing measurement. The amount of phase shift, expressed in degrees of the target waveform cycle, between the *MidRef* crossings of two different waveforms. Waveforms measured should be of the same frequency or one waveform should be a harmonic of the other.

Phase is a dual waveform measurement; that is, it is measured from a target waveform to a reference waveform. To get a specific phase measurement, you must specify the target and reference sources.

Phase is determined in the following manner:

1. The first *MidRefCrossing (MCross1Target)* and third *(MCross3*) in the source (target) waveform are found.
2. The period of the target waveform is calculated (see *Period* above).
3. The first *MidRefCrossing (MCross1Ref)* in the reference waveform crossing in the same direction (polarity) as that found *MCross1Target* for the target waveform is found.
4. The phase is determined by the following:

$$Phase = \frac{MCross1Ref - MCross1Target}{Period} \times 360$$

If the target waveform leads the reference waveform, phase is positive; if it lags, negative.

Positive Area

Amplitude (voltage) measurement. The arithmetic area over the absolute value of one waveform or if measurement gating is on, over its gated area. Remember that one waveform is not necessarily equal to one cycle. For cyclical data you may prefer to use the cycle area rather than the arithmetic area.

if Start = End then return the (interpolated) value at Start.

Otherwise,

$$Area = \int_{Start}^{End} \mathrm{ABS}(Waveform(t))\mathrm{dt}$$

For details of the integration algorithm, see page B–17.

Positive Cycle Area

Amplitude (voltage) measurement. The area over the absolute value of one waveform cycle. For non-cyclical data, you might prefer to use the Area measurement.

If StartCycle = EndCycle then return the (interpolated) value at StartCycle.

$$CycleArea = \int_{StartCycle}^{EndCycle} \mathrm{ABS}(Waveform(t))\mathrm{dt}$$

For details of the integration algorithm, see page B–17.

Positive Duty Cycle

Timing measurement. The ratio of the positive pulse width to the signal period, expressed as a percentage.

PositiveWidth is defined in **Positive Width**, following.

If Period = 0 or undefined then return an error.

$$PositiveDutyCycle = \frac{PositiveWidth}{Period} \times 100\%$$

Positive Width

Timing measurement. The distance (time) between MidRef (default = 50%) amplitude points of a positive pulse.

```
    If MCross1Polarity = '+'
then
    PositiveWidth = (MCross2 – MCross1)
else
    PositiveWidth = (MCross3 – MCross2)
```

Preshoot

Amplitude (voltage) measurement. Preshoot finds the first positive-going edge in the waveform record (or gated area) and calculates value as follows:

$$Preshoot = \frac{Low - Min}{Amplitude} \times 100\%$$

This measurement ignores settings of the `CALCulate:WMParameter:EDGE` and :SLOPE commands when locating the edge to measure.

Rise Time

Timing measurement. Time taken for the leading edge of a pulse to rise from a LowRef value (default = 10%) to a HighRef value (default = 90%).

Figure B–3 shows a rising edge with the two crossings necessary to calculate a rise time measurement.

1. Based on the index n specified by the `CALCulate:WMParameter:EDGE` command, count edges according to the following rules:

   - Positive arguments search forward from the start-to-end of the waveform record; zero and negative arguments search backward from the end-to-start.
   - Zero specifies the last edge in the waveform record.
   - Count only edges that go through both HREF and LREF (see Figure B–2).

2. Continue the search process until the n^{th} positive-going edge is found_(user specifies n using the `CALCulate:WMParameter:EDGE` command).
3. Find the time of the HREF crossing for the n^{th} edge. This time is THF. (Use linear interpolation if necessary.)
4. Find the time of the LREF crossing for the n^{th} edge. This time is TLF. (Use linear interpolation if necessary.)
5. RiseTime = THR – TLR

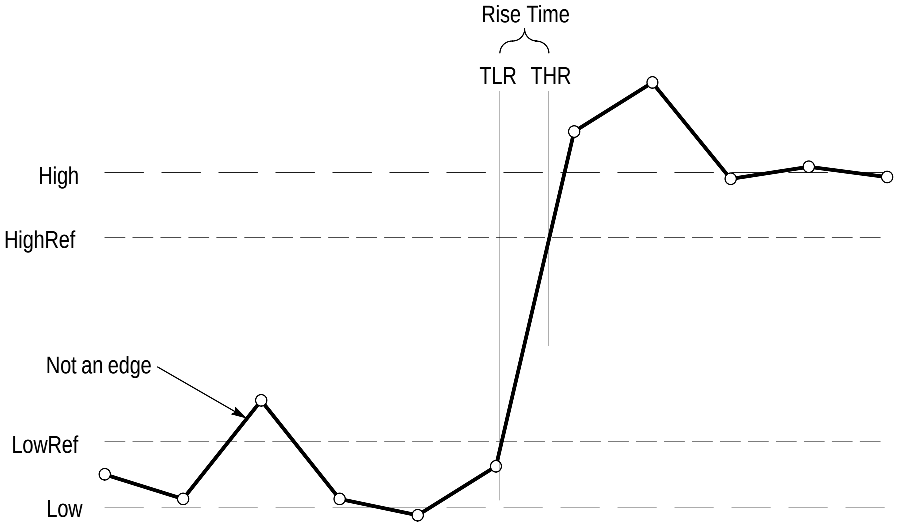


Figure B–3: Rise time

RMS:

Amplitude (voltage) measurement. The true Root Mean Square voltage.

If Start = End then RMS = the (interpolated) value at Waveform[Start].

Otherwise,

$$RMS = \sqrt{\frac{\int_{Start}^{End} (Waveform(t))^{2}\mathrm{dt}}{(End - Start) \times SampleInterval}}$$

For details of the integration algorithm, see *Integration Algorithm* in this section.

TTrig Timing measurement. The time difference between the main and the delay triggers.

TTrig = Time@delaytrig – Time@maintrigger

Value returned is independent of channel number. Value returned is valid only when the delay trigger source is not set to immediate.

Differentiation Algorithm

The differentiation algorithm used by the waveform analyzer is as follows:

$$\begin{aligned}
&Diff(w(n)) = 0\\
&\quad for\ n = 0\\
&Diff(w(n)) = [w(n + 1) - w(n - 1)]/(2T)\\
&\quad for\ 1 \le n \le (R-1)\\
&Diff(w(n)) = [w(R-1) - w(R-2)]/T\\
&\quad for\ n = (R-1)
\end{aligned}$$

where:

n = index into the record of data points
$w(n)$ = input sampled data point
T = time interval between successive samples
R = record length

Integration Algorithm

The integration algorithm used by the waveform analyzer is as follows:

$$\begin{aligned}
&Intg(w(n)) = 0\\
&\quad for\ n = 0\\
&Intg(w(n)) = \left[1/2\ w(0) + \sum_{m=1}^{n-1} w(m) + 1/2\ w(n) \right] \times T\\
&\quad for\ 1 \le n \le R
\end{aligned}$$

where:

n = index into record of data points
$w(n)$ = input sampled data point
T = time interval between successive samples
R = record length in points

Smooth Algorithm

The smoothing algorithm used by the waveform analyzer is as follows:

$$Smooth(w(n)) = (1/s)\left[\sum_{m=0}^{n+h} w(m) + (h - n) \times w(0)\right]$$

$$for\ n < h$$

$$Smooth(w(n)) = (1/s)\left[\sum_{m=n-h}^{n+h} w(m)\right]$$

$$for\, h \le n \le R-1-h$$

$$Smooth(w(n)) = (1/s)\left[\sum_{m=n-h}^{R-1} w(m) + (R-1-n) \times w(R-1)\right]$$

$$for\ n > R-1-h$$

where:

n = index into record of data points
$w(n)$ = input sampled data point
s = smoothing interval in samples; the second argument
h = half interval: (s – 1)/2 rounded down
R = record length in points

The smoothed waveform is derived by computing the average value of the corresponding point of the original waveform and a certain number of points of the original waveform on either side of the corresponding point. The number of points on either side is derived from the smoothing interval, which you set with the command CALC:SMO:POIN.

Near the ends of the waveform, nonexistent points beyond the ends of the waveform are required for averaging. The nonexistent points are assumed to be the value of the corresponding end points. This method of extending the waveform is arbitrary, so the results within a smoothing interval of the ends of the waveform must be interpreted accordingly.

Digital Filter Algorithms

This section describes how the digital filter of the waveform analyzer operates. The commands in the CALCulate:FILTer subsystem control the digital filter.

An Ideal Filter

The filter functions in the waveform analyzer instrument allow lowpass, highpass, bandpass and notch filters to be applied to any acquired set of data. A perfect filter would have unity transmission (with linear phase response) in the pass band, infinite attenuation in the stop band and abruptly change from pass to stop band. The transfer function for an ideal bandpass filter is depicted in Figure B–4.

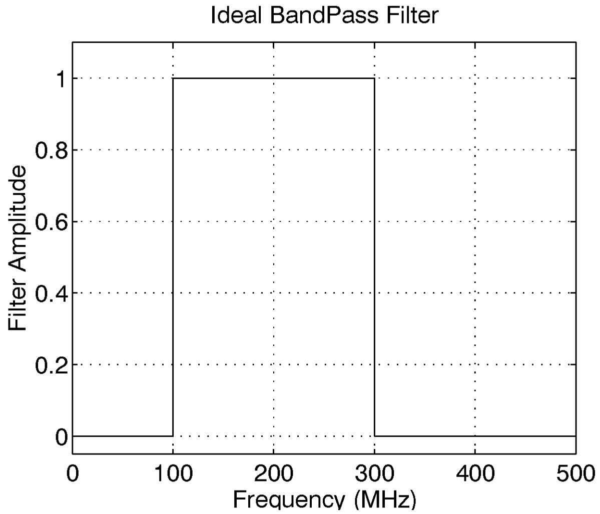


Figure B–4: Transfer function H(f) for an ideal bandpass filter

When you use a filter in the waveform analyzer, the frequency response of the desired filter is inverse Fourier transformed to calculate the response of the filter for a time domain impulse and this impulse response is convolved with the waveform data as shown in the following equation:

$$h(t) = \mathcal{F}^{-1}\{H(f)\}$$

$$output\ wfm = (input\ wfm) * h(t)$$

These equations are mathematically correct, however, it is impossible to implement them. For any ideal filter, which has abrupt changes in the transfer function, the impulse response extends for all time. Clearly an infinitely long impulse response cannot be convolved with the waveform data.

Rectangular Window

One possible solution for dealing with the infinitely long impulse function, h(t) is to reduce it to a manageable length. The simplest technique is to use the central points of the filter and throw away the remaining points. What this does is apply a time domain rectangular window filter to the impulse response h(t). Figure B–5 shows the transfer function for an ideal lowpass filter. Figure B–6 shows the time domain impulse for the lowpass filter and depicts one possible rectangular window which selects the central filter points.

Truncating the infinitely long impulse response of the filter to a finite length results in a filter frequency response that is no longer ideal. In the time domain the filter impulse response has been multiplied by a window w(t):

$$new\ \ h(t) = h(t) \cdot w(t)$$

To see how the frequency domain transfer function for the filter, H(f), has changed it is necessary to transfer the above equation to the frequency domain. Multiplication in the time domain corresponds to convolution in the frequency domain.

$$W(f) = \mathcal{T}\{w(t)\}$$

$$New\ \ H(f) = H(f) * W(f)$$

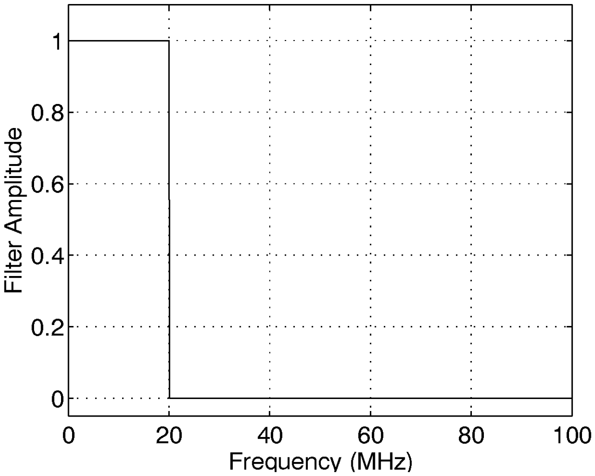


Figure B–5: Transfer function for an ideal lowpass filter

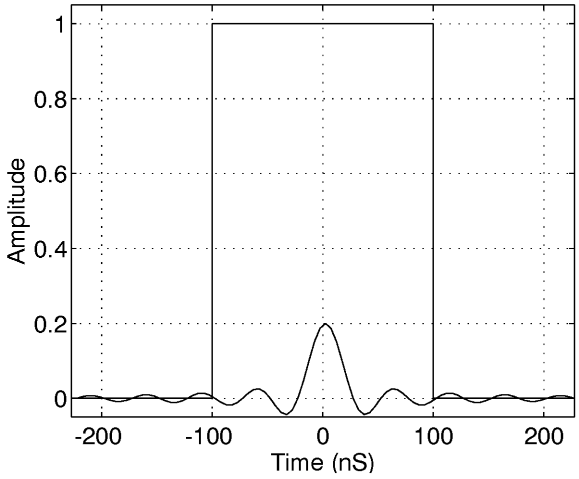


Figure B–6: Using a rectangular window to truncate the data from Figure B–5 to a finite number of points

The frequency domain convolution of H(f) with W(f) has three effects: the filter edges are no longer abrupt, the pass band transmission is no longer exactly unity, and the stop band attenuation is no longer infinite. Figures B–7 through B–9 show the original, ideal filter and the resultant filter with three different lengths of rectangular windows. Note that these plots use a dB scale for the filter.

As more points are used in the filter (corresponding to a longer window) the transition becomes sharper and sharper. However, in this example, the worst case attenuation of the filter in the stop band stays fixed at about –21 dB. Examining Figures B–7 through B–9 carefully, you can see that the peak amplitude of the ripple in the stop band (i.e., minimum attenuation) remains fixed at about –21 dB. As more points are used, the filter becomes sharper but this side lobe remains 21 dB down. For other filter shapes, such as bandpass and notch filters, the worst case stop band attenuation can be as low as –15 dB. The limitation on the stop band attenuation is the major drawback of using a rectangular window.

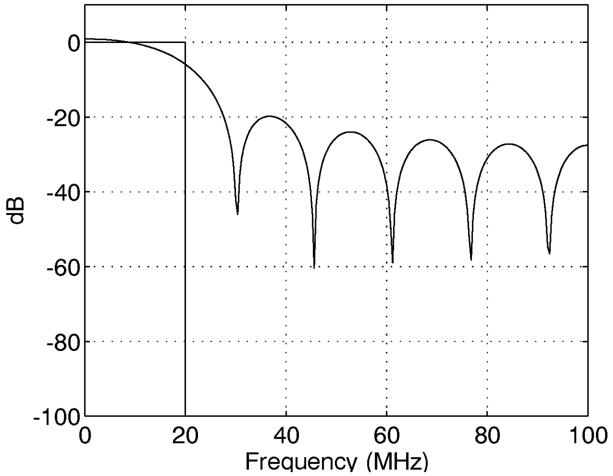


Figure B–7: Lowpass filter transfer function obtained by truncating the impulse response to just a few points

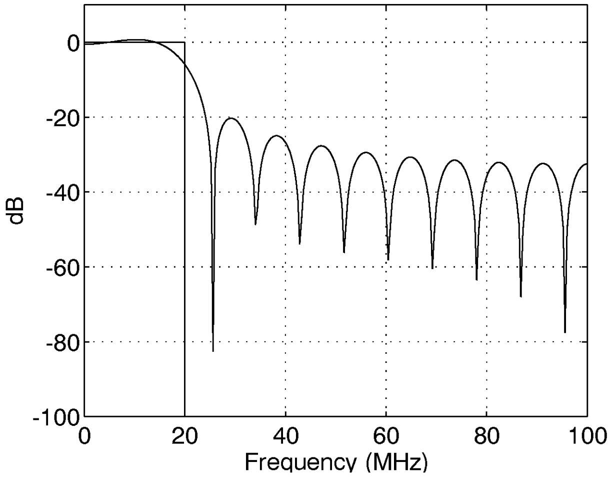


Figure B–8: Using more points in the lowpass filter results in a steeper transition at the cutoff frequency

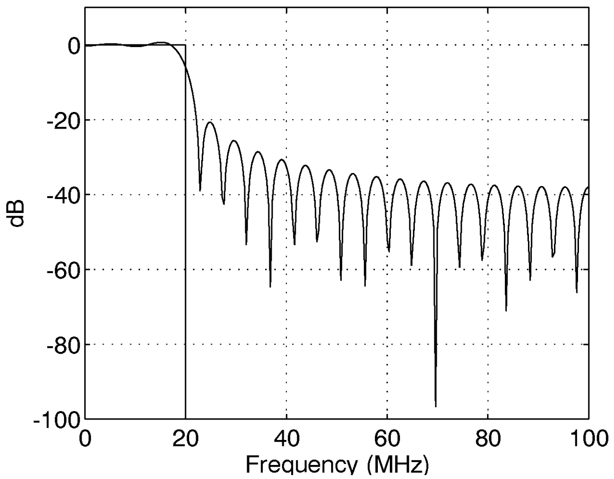


Figure B–9: Using many more points in the lowpass filter results in a quicker transition but a minimum attenuation of 21 dB

Kaiser Window

When the filter response is truncated with a rectangular window the minimum attenuation in the stop band is at best 21 dB. In order to achieve greater attenuation in the stop band a non–rectangular window must be applied to the filter data.

There are many choices for non–rectangular windows. Common windows include Bartlett, Hamming, Hanning, and Blackman. The filter in the waveform analyzer employs a Kaiser window. This window was chosen because it offers a range of possible window shapes, and thus different stop band attenuations. For a window that is M+1 points long, the Kaiser window is defined as follows:

$$w[n] = \left\{ \begin{array}{ll} \dfrac{I_0\left[\beta\left(1-\left[\frac{(n-M/2)}{M/2}\right]^2\right)^{1/2}\right]}{I_0\left[\beta\right]} & 0 \le n \le M \\ 0 & otherwise \end{array} \right\}$$

I_0 represents the zero order modified Bessel function of the first kind. β is a parameter that ranges from 0 to infinity. The larger the value of β, the more the window tapers at the edges. When β=0 the Kaiser window reduces to a rectangular window. Figure B–10 shows three Kaiser windows with 200 points in the window and a β of 1, 5 and 20.

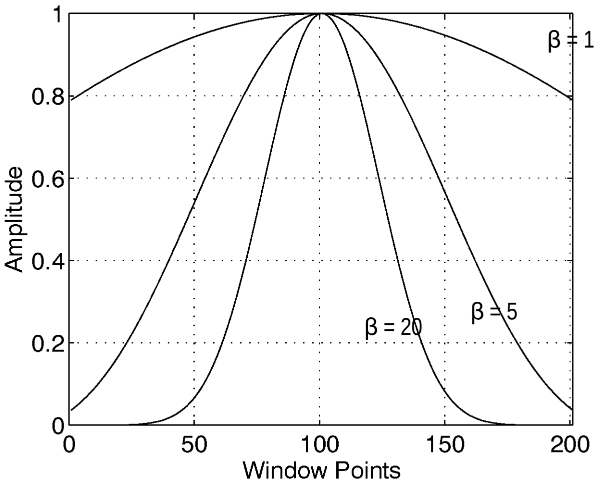


Figure B–10: Kaiser window with 200 points and β = 1, 5 and 20

For larger values of β, the Kaiser window tapers off slowly towards the edges of the window. Using the same number of data points and taking a Fourier transform of a Kaiser window and a rectangular window, the transform of the Kaiser window is broader than the rectangular window but the side lobes are much farther down.

As stated in the last section, the frequency domain transfer function of the filter is given by convolving the transfer function of the ideal filter with that of the window.

$$W(f) = \mathcal{T}\{w(t)\}$$

$$New\ H(f) = H(f) * W(f)$$

If a Kaiser window is used with the same number of points as a rectangular window, then the transition width will not be as narrow but the minimum stop band attenuation will be much greater than the 21 dB achieved with a rectangular window. To graphically see this effect, refer to Figure B–11 which shows a lowpass filter obtained with a Kaiser window with β=2.65. Compare the transfer function of this filter with that in Figure B–9 where the same number of points were used but with a rectangular window.

By setting β high enough, the stop band attenuation can be increased. The cost for this increase is a wider filter transition region which can be countered by using more points in the filter. As pointed out previously, the greater the number of points in the filter, the narrower the filter transition region. However, there is a limit to the number of points in the filter. As described more fully in the section on edge effects, the number of points in the filter is limited to a maximum of 10% of the record length.

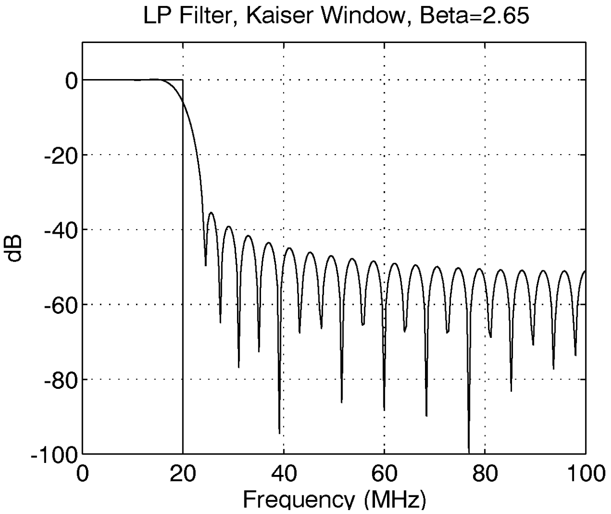


Figure B–11: Compare this result with Figure B–9 with the same number of points but a rectangular window

Defining Filter Specifications

The waveform analyzer filter is specified in a manner which may be unfamiliar to those used to working with analog filters. One difference is that cutoff and start/stop frequencies are specified as the –6 dB point, not the –3 dB point. Figure B–12 shows waveform analyzer specifications for a lowpass filter.

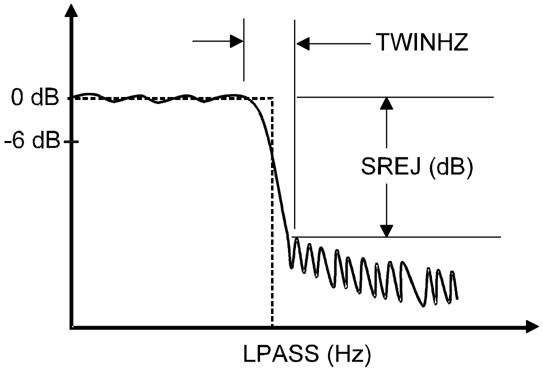


Figure B–12: Filter specifications for a lowpass filter

The cutoff frequency of the filter is specified in Hertz with the LPAS command. As an example, to set a lowpass cutoff frequency of 20 MHz, use the command:

```
CALC1:FILT:FREQ:LPAS 20E6
```

LPAS specifies a lowpass filter and 20E6 sets the cutoff frequency (at –6 dB).

The stop band attenuation or stop band rejection is set with the SREJ command. The SREJ is given in dB. The minimum attenuation is 15 dB and the maximum is 100 dB (the default value is 60 dB). As an example, to set the stop band attenuation to 40 dB use the command:

```
CALC1:FILT:FREQ:SREJ 40
```

The final specification of the filter is the relative filter transition width, TWID. The TWID is directly related to the TWIDHZ depicted in Figure B–12. TWIDHZ is a measure of how quickly the filter response changes from pass band to stop band. The TWID is specified relative to the Nyquist frequency for the acquisition record. The Nyquist frequency is defined as:

$$F_{NYQ} = \frac{1}{2 \cdot TINT}$$

Where TINT is the sample interval. Equivalently, the Nyquist frequency is also equal to 1/2 of the sampling rate.

The TWID is then defined as:

$$TWID = \frac{TWIDHZ}{F_{NYQ}} = TWIDHZ \cdot 2 \cdot TINT$$

Valid ranges for TWID are between 0 and 1. The smaller the value, the narrower the transition region. The default value is 0.1. As an example, to set the TWID to 0.05 use the command:

```
CALC1:FILT:FREQ:TWID 0.05
```

Define highpass filter in the same manner as the lowpass filter. The only difference is that it is necessary to specify a HPAS frequency. As an example, to set the highpass frequency to 100 MHz, use the command:

```
CALC1:FILT:FREQ:HPAS 100E6
```

Figure B–13 depicts the specifications for a band pass filter. The same specifications are used for the notch filter, but the two filters have swapped pass band and stop band regions. Like the lowpass filter, a bandpass filter has a specification for SREJ and TWID. SREJ and TWID are equal on both sides of the bandpass region. For the bandpass and notch filters it is necessary to specify a start and stop frequency. This is done with the STAR and STOP commands. As an example to set up a bandpass filter with a start frequency of 50 MHz and a stop frequency of 75 MHz use the command:

```
CALC1:FILT:FREQ:BPAS; STAR 50E6; STOP 75E6
```

Instead of specifying a STAR and STOP frequency, it is possible to specify CENT and SPAN frequencies.

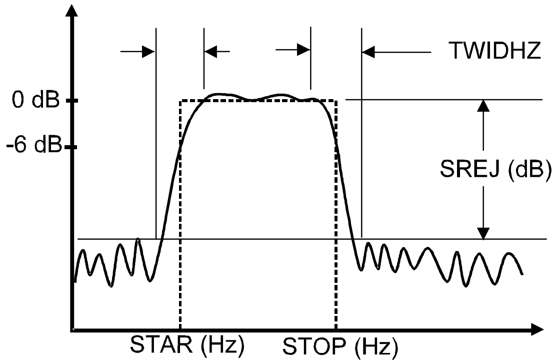


Figure B–13: Filter specifications for a bandpass filter

Filter Length Limitations and Edge Effects

After the TWID and the SREJ are specified, the β for the window is calculated from the following expressions:

$$SATT = SREJ + 6.0206$$

$$\beta = \begin{cases} 0.1102(SATT-8.7) & SATT > 50 \\ 0.58422(SATT-21)^{0.4} + 0.07886(SATT-21) & 21 \le SATT \le 50 \\ 0 & SATT < 21 \end{cases}$$

The value 6.0206 is added to SREJ to ensure that the additive ripple from the notch and bandpass filters stay within specifications. Adding 6.0206 to SREJ is equivalent to dividing the ripple specification by two.

The number of points in the filter are calculated from the expression:

$$NFILT = \frac{SATT-8}{2.285 \cdot \pi \cdot TWID} + 1$$

Actually the waveform analyzer algorithm uses the smallest odd integer greater than the NFILT calculated in the above expression (if NFILT=75.1 then 77 would be used). If there is a high level of stop band attenuation, SREJ, or if the relative filter transition width, TWID, is set very small then there may be a large number of points in the filter. For example, if SREJ =80dB and the TWID=.01 then the number of points in the filter is 1087. The input waveform itself may be less than 1087 points long and there are edge effects, which often preclude the use of such a long filter.

The output waveform is calculated by convolving the input waveform with the impulse response of the filter:

$$output\ wfm = (input\ wfm) * h(t)$$

If the length of the original data is N and the length of the filter is NFILT, then the result of the convolution is a record N+NFILT–1 points long. See Figure B–14. From this record, (NFILT–1)/2 points are cut off each end to return a record which is N points long, the same size as the original record. However, the filter specifications are not guaranteed throughout the length of the new data record. For (NFILT–1)/2 points on either end of the new record, the data is questionable. This is due to edge effects in the convolution when the filter record is not fully within the data record.

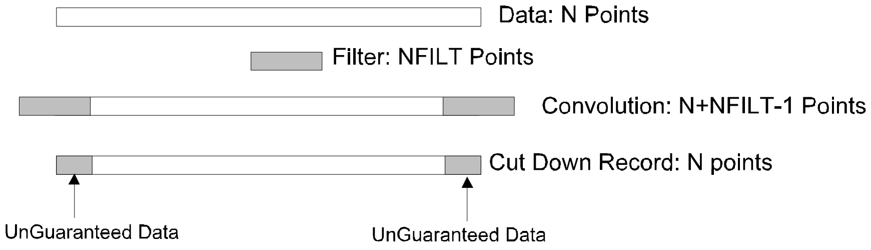


Figure B–14: Record resulting from convolving the filter impulse response with the waveform record

An example will help illustrate the edge effects. Figure B–15 shows a test signal created from a 1 V amplitude, 10 MHz sine wave and a 0.5 V, 125 MHz sine wave. This record is 500 points long, TINT=800 ps and the Nyquist frequency F_{nyq}=625 MHz. To filter out the high frequency signal, a lowpass filter was applied with LPASS = 62.5 MHz, TWID=0.1 (TWIDHZ=62.5 MHZ) and SREJ=26 dB. This resulted in a filter with β=1.51 and 53 points in the filter. Figure B–16 shows the result of applying the lowpass filter to the data. The filter did a good job of cutting out the high frequency components. To illustrate the edge effects, Figure B–17 shows a close up view of the left end of the filtered data and the 10 MHz sine signal. It is clear in that the filtered data does not initially track the source 10 MHz sine signal.

On each end of the record, (NFILT–1)/2 data points of the filtered data are not guaranteed to be within the specification of the filter. This is an undesirable situation. To limit this effect, the waveform analyzer digital filter algorithm limits the number of filter points to be a maximum of 10% of the acquisition record length. With this constraint, in the worst case condition only 5% of the data on either end of the filtered record is not guaranteed to be within the filter specification. Hence, all measurements should be on the central 90% of the data record.

To insure that your measurement does not include bad data, the waveform analyzer sets the (NFILT–1)/2 data points on either end of the filtered record to NULLs. The output waveforms of CALC blocks (which include the filter function) are always floating point numbers. NULL points are defined as

9.910000E+37 for ASCII format, and IEEE NAN (Not A Number) for REAL,32 format.

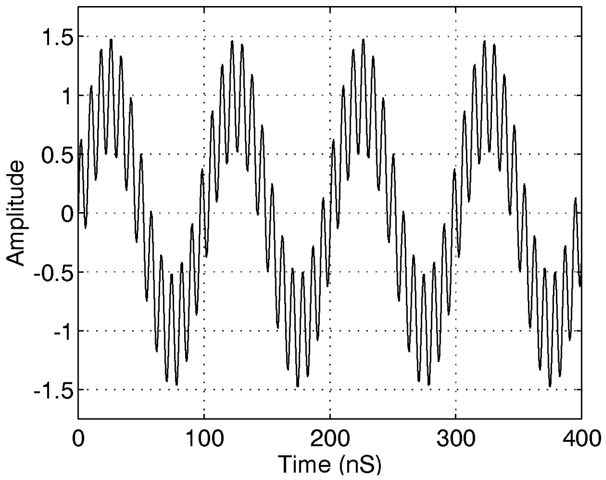


Figure B–15: Filter test signal with a 125 MHz signal modulating a 10 MHz signal

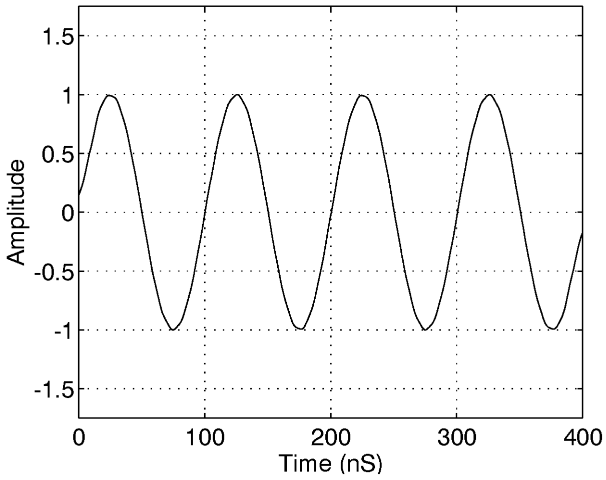


Figure B–16: Test signal after being filtered with a lowpass filter

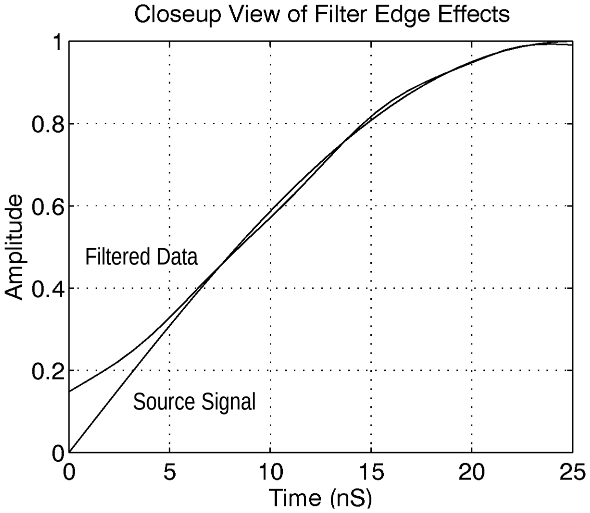


Figure B–17: View of the filtered record showing the first 5% of the filtered data

Filter Performance

Stop Band Attenuation. The stop band attenuation, or stop band rejection, is set by the SREJ parameter. The attenuation in the stop band is at least the value given by SREJ. Typically the minimum attenuation occurs at the start of the stop band. Further into the stop band the attenuation is typically several to tens of dB greater than SREJ.

Pass Band Ripple. The ripple in the pass band is not explicitly set through the filter commands. For the Kaiser window algorithm used in the waveform analyzer the pass band ripple is directly related to the stop band attenuation:

$$PassBand\ Ripple\ (dB)\ =\ 20log_{10}\left[1\ +\ 10^{\frac{-SREJ}{20}}\right]$$

For example, if the SREJ is set to 40dB then the pass band ripple will be less than 0.0864 dB; if SREJ is set to 60dB, pass band ripple will be less than 0.00868 dB.

Filter Cutoff and Roll Off. The digital filters implemented in the waveform analyzer are different than traditional analog filters. One important distinction is that the STAR, STOP, LPAS, and HPAS frequencies are not the –3 dB cutoff frequencies for the filter but are instead –6 dB cutoff frequencies.

Analog filter designers often characterize filters by the roll off rate beyond a cutoff frequency. For example, analog filters are characterized with a roll off of 20 dB/decade, 40 dB/decade, etc. Unfortunately, these familiar analog terms do not apply to digital filters. Unlike analog filters, digital filters do not continue to drop off above a cutoff frequency. Instead, the filter response drops rapidly in the transition region and then flattens out somewhat in the stop band. In the transition region, the roll off cannot be well approximated as a constant roll off per decade of frequency (such as 40 dB/decade).

The Kaiser window filter technique does not provide a constant dB/decade roll off in the transition region. In fact, in the transition region, the Kaiser window technique only specifies that the transfer function will decrease from the pass band level to the stop band attenuation. What the Kaiser window does guarantee is the specifications in the pass band and stop band:

$$Passband\ Ripple \;\le\; 20log_{10}\left[1 + 10^{\frac{-SREJ}{20}}\right] \qquad f \;\le\; LPAS - \left(\frac{TWID}{4 \cdot TINT}\right)$$

$$Stopband\ Rejection \;\ge\; SREJ \qquad\qquad f \;\le\; LPAS + \left(\frac{TWID}{4 \cdot TINT}\right)$$

Similar specifications are achieved for highpass, bandpass and notch filters. For notch filters, be sure that (STOP– START) is greater than TWIDHZ or else no guarantee is made about the attenuation in any portion of the notch region.

Group Delay. The digital filters have linear phase in the pass band. The group delay, which is the derivative of the phase, is therefore constant in the pass band.

Practically speaking, this means that if you have a signal which is made up of many frequency components, the relative phase of these frequency components are preserved in the filter.

Error Conditions

There are two main causes of errors from the digital filter code. One of the sources of error is a filter specification that generates too many filter coefficients. The other class of errors is from cutoff frequencies that violate certain constraints.

Too Many Filter Coefficients. If the stop band attenuation SREJ and/or the relative filter transition width TWID is set to too high, then the number of points required by the filter may exceed 10% of the acquired record length. Since the digital filter implementation limits the number of coefficients to 10% of the record length, waveform analyzer reports an error, and performs no filtering.

Suppose, for example, you acquire a record with 1024 points at 1 GSample/second acquisition rate. You set the lowpass filter to a cutoff frequency of 200 MHz, a stop band attenuation, SREJ, of 80 dB and relative filter transition width, TWID, of 0.05. Such a filter requires 219 points, which is more than 102 points (10% of the data record), and the following error is reported:

```
2100,"CalculateN questionable; digital filter error – filter
specs require too many coefficients"
```

When this error occurs, there are a number of things that can be done to allow the filter generation to succeed. One change you could do is to change the value for TWID (and equivalently TWIDHZ) and keep the remaining filter parameters fixed. This value of TWID is calculated from:

$$TWID = \frac{SREJ - 1.9794}{7.1785 \cdot (MaxNPTS)}$$

Where MaxNPTS in this case is set to 102. The maximum possible value for TWID is 1 (for realistic filters it is desirable to have TWID less than 0.2). If the specified SREJ is too large then the value of TWID calculated in the above expression may exceed 1. In this case, it is not possible to design a filter that meets the SREJ specifications and uses less than the maximum possible number of points.

Another change you could do is to change the value of SREJ and keep the remaining filter parameters fixed. This value of SREJ is calculated from:

$$SREJ = 7.1785 \cdot TWID \cdot (MaxNPTS) + 1.9794$$

The minimum value for SREJ is 15 dB. If the TWID was initially chosen too small then the above formula may predict a value for SREJ which is less than 15 dB. In this case, it is impossible to create a filter which meets the specification for TWID and which uses less than the maximum possible number of points.

Another change would be to acquire the data using a longer record length. Since the filter can have more points if the input data record length is larger, you can use a tighter specified filter with a longer record length. In our example, going to a record length of 4096 would have allowed the filter to operate without error.

Incorrect Cutoff Frequencies. The basic problem is that you can't have the cutoff frequency for low or highpass filters close to 0 or Nyquist. For bandpass and notch filters, there is the additional constraint that start and stop frequencies can't be too close to each other. The rules are as follows:

- Insure that the cutoff frequency (LPAS or HPAS) minus half the transition width is greater than 0.
- Insure that the cutoff frequency (LPAS or HPAS) plus half the transition width is less than Nyquist.

For bandpass/notch filters, insure that START minus half the transition width is greater than 0, STOP plus half the transition width is less than Nyquist, and START plus half the transition width is less than STOP minus half the transition width.

If these constraints are violated, the following error messages appear:

```
2100,"CalculateN questionable; digital filter error – lowpass
filter cutoff invalid", or

2100,"CalculateN questionable; digital filter error –
highpass filter cutoff invalid", or

2100,"CalculateN questionable; digital filter error –
bandpass/notch filter start/stop invalid"
```

Note that for both major sources of error, errors are detected and reported when the calculate block is executed, not when filter parameters are defined or when the filter expression is defined. The errors are delayed because the sample rate (which defines Nyquist) and record length (which sets the maximum number of filter coefficients) may change after you define the filter parameters.

When these errors occur, the waveform analyzer CALC block copies its input waveform to its output and performs *no* filtering.

General Guidelines

Edge effects from the filter can set up to 5% of the data on either end of the filtered record to NULL values: ASCII, 9.910000E+37 or binary, IEEE NAN. If you are going to filter the data, make sure all acquisitions have a sufficient number of points on either side of the data of interest.

If you apply a filter to a set of data and get an error, you can either specify a filter with less stringent specifications or you acquire the data a second time using a longer record length. The maximum length of the filter is 10% of the data record, so if you use a longer record you have more points available in the filter and you can use a higher specified filter.

To increase the number of points in your data, use the same sample rate but a longer record length. See the commands SWEep:POINts and SWEep:TIME to set the record length. Make sure you acquire additional points before and after the data of interest. Do not try to obtain more points in your data record by simply increasing the sampling rate. Increasing the sampling rate provides more points but it also increases the Nyquist sampling rate which can make the desired filter more difficult to achieve.

Appendix C: SCPI Conformance Information

All commands in the TVS600 and TVS600A Series of waveform analyzers conform to SCPI Version 1995.0. Table C–1 lists all commands supported by the waveform analyzers. The columns at right show whether a command is defined in the SCPI 1995.0 Standard or not. Some commands are used by TVS600A models only; see *Command Groups*, which starts on page 3–65.

Table C–1: SCPI conformance information

Command				Defined in SCPI 1995.0	Not Defined In SCPI 1995.0
AADVance	[:STATe][?]				✓
	:COUNt[?]				✓
	:RECord	:COUNt[?]			✓
		:STARt[?]			✓
ABORt				✓	
ARM	[:A][:LAYer[1]]	:DEFine?		✓	
		:SOURce[?]		✓	
AVERage	[:STATe][?]			✓	
	:COUNt[?]			✓	
	:TYPE[?]			✓	
CALCulate[n]	:AAMList[?]				✓
		:STATe[?]			✓
	:DATA?			✓	
		:PREamble?		✓	
	:DERivative	:STATe[?]		✓	
	:FEED[1][?]			✓	
	:FEED2[?]	:CONTEXT[?]		✓	
	:FILTer	:FREQuency	[:TYPE][?]	✓	
			:CENTer[?]	✓	
			:HPASs[?]		✓
			:LPASs[?]		✓
			:SPAN[?]	✓	
			:SREJection[?]		✓
			:STARt[?]	✓	

Table C–1: SCPI conformance information (cont.)

Command				Defined in SCPI 1995.0	Not Defined In SCPI 1995.0
			:STATe[?]	✓	
			:STOP[?]	✓	
			:TWIDth[?]		✓
	:FORMat[?]			✓	
	:IMMediate[?]			✓	
	:INTegral	:STATe[?]		✓	
	:PATH[?]			✓	
		:EXPRession[?]		✓	
	:SMOothing	[:STATe][?]			✓
		:POINts[?]		✓	
	:TRANsform	:FREQuency	:STATe[?]	✓	
			:WINDow[?]	✓	
	:WMList[?]	:STATe[?]			✓
	:WMParameter	:EDGE[?]			✓
		:GATE[?]	[:METHod][?]		✓
			:STARt[?]		✓
			:STOP[?		✓
		:HIGH[?]			✓
		:HMEThod[?]			✓
		:LOW[?]			✓
		:LMEThod[?]			✓
		:HREFerence	[:ABSolute][?]		✓
			:RELative[?]		✓
		:LREFerence	[:ABSolute][?]		✓
			:RELative[?]		✓
		:MREFerence	[:ABSolute][?]		✓
			:HYSTeresis[?]		✓
			:RELative[?]		✓
		:RMEThod[?]			✓
		:SLOPe[?]			✓
CALibration	:ALL[?]			✓	
	:RESults	[:CODE]?			✓

Table C–1: SCPI conformance information (cont.)

Command				Defined in SCPI 1995.0	Not Defined In SCPI 1995.0
		:VERBose?			✓
	:PROBe[n][?]	:RESults?		✓	
DATA?				✓	
	:PREamble?			✓	
FORMat	[:DATA][?]			✓	
		:CALCulate[n][?]			✓
		:TRACe	AATS[?]		✓
		:DINTerchange[?]			✓
	:BORDer[?]			✓	
FUNCtion	[:ON][?]			✓	
		:ALL		✓	
		:COUNt?		✓	
	:OFF[?]			✓	
		:ALL		✓	
		:COUNt?		✓	
	:CONCurrent[?]			✓	
	:STATe?			✓	
	:STATe				✓
INITiate	[:IMMediate]			✓	
	:CONTinuous[?]			✓	
	:COUNt[?]				✓
INPut[n]	:COUPling[?]			✓	
	:FILTer	[:LPASs]	[:STATe][?]	✓	
			:FREQuency[?]	✓	
	:IMPedance[?]			✓	
	:PROBe	:ATTenuation[?]		✓	
		:IDENtification[?]		✓	
		:OFFSet[?]		✓	
	:PROTection	:STATe[?]			✓
MEMory	:DATA[?]			✓	
	:NSTates?			✓	
	:STATe	:CATalog?		✓	

Table C–1: SCPI conformance information (cont.)

Command				Defined in SCPI 1995.0	Not Defined In SCPI 1995.0
		:DEFine?		✓	
OUTPut	:ECLTrg[n]	[:STATE][?]		✓	
		:SOURce?		✓	
	:PCOMpensate	[:STATE][?]			✓
	:REFerence	[:STATE][?]			✓
		:FUNCtion[?]			✓
	:TTLTrg[n]	[:STATE][?]		✓	
		:POLarity[?]		✓	
		:SOURce?		✓	
ROSCillator	:SOURce[?]			✓	
STATus	:OPERation?			✓	
		:CONDition?		✓	
		:ENABle[?]		✓	
		:NTRansition[?]		✓	
		:PTRansition[?]		✓	
		:QENable	:NTRansition[?]		✓
			:PTRansition[?]		✓
	:PRESet			✓	
	:QUEStionable	[EVENt]?		✓	
		:CONDition?		✓	
		:ENABle[?]		✓	
		:NTRansition[?]		✓	
		:PTRansition[?]		✓	
		:QENable	:NTRansition[?]		✓
			:PTRansition[?]		✓
	:SESR	:QENable			✓
SWEep	:OFFSet	:POINts[?]		✓	
		:TIME[?]		✓	
	:OREFerence	:LOCation[?]		✓	
	:POINts[?]			✓	
	:TIME[?]			✓	
	:TINTerval[?]			✓	

Table C–1: SCPI conformance information (cont.)

Command					Defined in SCPI 1995.0	Not Defined In SCPI 1995.0
SYSTem	:AUToset	:VOLTage			✓	
		:SWEep			✓	
		:TRIGger			✓	
	:BDATE[?]					✓
	:CDATE[?]					✓
	:COMMunicate	:SERial	:BAUD?		✓	
			:CONTrol	:DCD[?]		✓
				:RTS[?]	✓	
			:ECHO[?]			✓
			:ERESponse[?]			✓
			:LBUFfer[?]			✓
			:PACE[?]		✓	
			:PARity[?]		✓	
			:PRESet	[:ALL]		✓
				:RAW		✓
				:TERMinal		✓
			:SBITs[?]		✓	
	:ERRor?				✓	
		:ALL?				✓
		:CODE[?]				✓
			:ALL?			✓
		:COUNt?				✓
	:PROTect[?]					✓
	:SECurity	:IMMediate			✓	
	:SET[?]				✓	
	:VERSion?				✓	
TEST	[:ALL][?]					✓
	:RESults	[:CODE]?				✓
		:VERBose?				✓
	:STOP				✓	
TRACe	[:DATA]?				✓	
		:PREamble?			✓	

Table C–1: SCPI conformance information (cont.)

Command					Defined in SCPI 1995.0	Not Defined In SCPI 1995.0
	:CATalog?				✓	
	:COPY				✓	
	:DELete	:NAME			✓	
		:ALL			✓	
	:FEED?				✓	
	:LIST[?]					✓
	:POINts?				✓	
TRIGger	[:A]	:ATRigger	[:STATe][?]		✓	
		:COUPling[?]			✓	
		:COUPling	:<preset>			✓
		:DEFine?			✓	
		:DELay[?]			✓	
		:FILTer	[:LPASs]	[:STATe][?]	✓	
			:HPASs	[:STATe][?]	✓	
		:HOLDoff	:TIME[?]			✓
		:HYSTeresis	:SELect[?]			✓
		:LEVel[?]			✓	
		:LOGic	:CLASs[?]			✓
			:CONDition[?]			✓
			:FUNCtion[?]			✓
			:PATTern	:QUALify[?]		✓
				:WIDTh[?]		✓
			:STATe	:SLOPe[?]		✓
			:THReshold[?]			✓
		:METastable	:STATe[?]			✓
		:PULSe				✓
			:CLASs[?]			✓
			:GLITch	:POLarity[?]		✓
				:QUALify[?]		✓
				:WIDTh[?]		✓
			:SOURce[?]			✓
			:THReshold[?]			✓

Table C–1: SCPI conformance information (cont.)

Command				Defined in SCPI 1995.0	Not Defined In SCPI 1995.0
		:TIMEout	:POLarity[?]		✓
			:WIDth[?]		✓
		:WIDTh	:LLIMit[?]		✓
			:POLarity[?]		✓
			:QUALify[?]		✓
			:ULIMit[?]		✓
	:SHOLDtime	:CLOCk	:POLarity[?]		✓
			:SOURce[?]		✓
			:THReshold[?]		✓
		:DATA	SOURce[?]		✓
			:THReshold[?]		✓
		:HTIMe			✓
		:STIMe			✓
	:TRANsition	:CLASS	:RUNT[?]		✓
			:SLEW[?]		✓
		:RUNT	:QUALify[?]		✓
			:SLOPe[?]		✓
		:SLEW	:QUALify[?]		✓
			:SLOPe[?]		✓
		:SOURce[?]			✓
		:THReshold	:HIGH[?]		✓
			:LOW[?]		✓
		:TIME[?]			✓
	:SLOPe[?]			✓	
	:SOURce[?]			✓	
	:TYPE[?]			✓	
	:DEFine?			✓	
:B	:COUPling[?]			✓	
	:COUPling	:<preset>			✓
	:DEFine?			✓	
	:DELay[?]			✓	
	:ECOunt[?]			✓	

Table C–1: SCPI conformance information (cont.)

Command					Defined in SCPI 1995.0	Not Defined In SCPI 1995.0
		:FILTer	[:LPASs]	[:STATe][?]	✓	
			:HPASs	[:STATe][?]	✓	
		:HYSTeresis	:SELect[?]			✓
		:LEVel[?]			✓	
		:SLOPe[?]			✓	
		:SOURce[?]			✓	
VOLTage[n]	[:DC]	:RANGe	[:UPPer][?]		✓	
			:LOWer[?]		✓	
			:OFFSet[?]		✓	
			:PTPeak[?]		✓	
IEEE 488.2 Common Commands[1]						
*CAL?					✓	
*CLS					✓	
*ESE[?]					✓	
*ESR?					✓	
*IDN?					✓	
*LRN?					✓	
*OPC[?]					✓	
*OPT?					✓	
*PUD[?]					✓	
*RCL					✓	
*RST					✓	
*SAV					✓	
*SRE[?]					✓	
*STB?					✓	
*TST?					✓	
*WAI?					✓	

[1] **Defined in IEE Std. 488.2–1992**

Appendix D: Supported Preambles

Each waveform or other DATA CURVe that you transfer has an associated preamble. The preamble is usually uploaded and may be downloaded as a separate DIF expression that contains information about the data, such as the horizontal scale, the vertical scale, and other settings in place when the data was created.

See *I/O of Waveforms* on page 3–117 for more information on waveform preambles and uploading and downloading waveforms and other data.

TVS600A Preambles

Table D–1 lists the different types of preambles that the waveform analyzer supplies based on the type of data (Y-T curve, envelope, calculation) it transfers the controller and the format you specify for that data (see *Data Uploads* on page 3–117). These preambles are a subset of those defined by *Vol. 3: Data Interchange Format* of the *Standard Commands for Programmable Instruments,* published by the SCPI Consortium; see *Data Interchange Format* on page 3–131.

When downloading waveforms, remember the waveform analyzer will also accept the preambles listed in the table. Keywords in the preambles are defined in Table D–2 on page D–5.

Table D–1: TVS600A Preambles and their formats

Preamble Type	Preamble Format

CALCulate Data (binary)	DIF(VERS 1995.0 SCOP PRE)
	IDEN(NAME "CALC1" INST(NAME "TVS641A" ID "B010100"))
	ENC(FORM IFP32)
	DIM=X(TYPE IMPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "S")
	DIM=Y(TYPE EXPL SIZE <nr1> UNIT "V")
	DATA(CURV(CTYP NONE))

CALCulate Data (ASCII)	DIF(VERS 1995.0 SCOP PRE)
	IDEN(NAME "CALC1" INST(NAME "TVS641A" ID "B010100"))
	ENC(FORM ASC NVAL 9.91E+37 ORAN 9.9E+37 URAN –9.9E+37)
	DIM=X(TYPE IMPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "S")
	DIM=Y(TYPE EXPL SIZE <nr1> UNIT "V")
	DATA(CURV(CTYP NONE))

SENSe Data (binary)	DIF(VERS 1995.0 SCOP PRE)
	IDEN(NAME "CHAN1" INST(NAME "TVS641A" ID "B010100"))
	ENC(FORM INT16 NVAL –32768 ORAN 32767 URAN –32767)
	DIM=X(TYPE IMPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "S")
	DIM=Y(TYPE EXPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "V")
	DATA(CURV(CTYP NONE))

Table D–1: TVS600A Preambles and their formats (cont.)

Preamble Type	Preamble Format

SENSe Data (ASCII)	DIF(VERS 1995.0 SCOP PRE)
	IDEN(NAME "CHAN1" INST(NAME "TVS641A" ID "B010100"))
	ENC(FORM ASC NVAL –32768 ORAN 32767 URAN –32767)
	DIM=X(TYPE IMPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "S")
	DIM=Y(TYPE EXPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "V")
	DATA(CURV(CTYP NONE))

SENSe Data (binary auto-advance)	DIF(VERS 1995.0 SCOP PRE)
	IDEN(NAME "CHAN1" INST(NAME "TVS641A" ID "B010100"))
	ENC(FORM INT16 NVAL –32768 ORAN 32767 URAN –32767)
	DIM=REC(TYPE IMPL SIZE <nr1> UNIT "")
	DIM=X(TYPE IMPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "S")
	DIM=Y(TYPE EXPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "V")
	DATA(CURV(CTYP NONE))

SENSe Data (ASCII auto-advance)	DIF(VERS 1995.0 SCOP PRE)
	IDEN(NAME "CHAN1" INST(NAME "TVS641A" ID "B010100"))
	ENC(FORM ASC NVAL –32768 ORAN 32767 URAN –32767)
	DIM=REC(TYPE IMPL SIZE <nr1> UNIT "")
	DIM=X(TYPE IMPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "S")
	DIM=Y(TYPE EXPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "V")
	DATA(CURV(CTYP NONE))

SENSe Data (binary envelope)	DIF(VERS 1995.0 SCOP PRE)
	IDEN(NAME "CHAN1" INST(NAME "TVS641A" ID "B010100"))
	ENC(FORM INT16 NVAL –32768 ORAN 32767 URAN –32767)
	DIM=X(TYPE IMPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "S")
	DIM=UPP(TYPE EXPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "V")
	DIM=LOW(TYPE EXPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "V")
	TRAC=TU(IND(LAB X) DEP(LAB UPP))
	TRAC=TL(IND(LAB X) DEP(LAB LOW))
	VIEW=ENVELOPE(ENV(UPP TU LOW TL))
	DATA(CURV(CTYP NONE))

SENSe Data (ASCII envelope)	DIF(VERS 1995.0 SCOP PRE)
	IDEN(NAME "CHAN1" INST(NAME "TVS641A" ID "B010100"))
	ENC(FORM ASC NVAL –32768 ORAN 32767 URAN –32767)
	DIM=X(TYPE IMPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "S")
	DIM=UPP(TYPE EXPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "V")
	DIM=LOW(TYPE EXPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "V")
	TRAC=TU(IND(LAB X) DEP(LAB UPP))
	TRAC=TL(IND(LAB X) DEP(LAB LOW))
	VIEW=ENVELOPE(ENV(UPP TU LOW TL))
	DATA(CURV(CTYP NONE))

Table D–1: TVS600A Preambles and their formats (cont.)

Preamble Type	Preamble Format

CALCulate Data (binary envelope)	DIF(VERS 1995.0 SCOP PRE)
	IDEN(NAME "CALC1" INST(NAME "TVS641A" ID "B010100"))
	ENC(FORM IFP32)
	DIM=X(TYPE IMPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "S")
	DIM=UPP(TYPE EXPL SIZE <nr1> UNIT "V")
	DIM=LOW(TYPE EXPL SIZE <nr1> UNIT "V")
	TRAC=TU(IND(LAB X) DEP(LAB UPP))
	TRAC=TL(IND(LAB X) DEP(LAB LOW))
	VIEW=ENVELOPE(ENV(UPP TU LOW TL))
	DATA(CURV(CTYP NONE))

CALCulate Data (ASCII envelope)	DIF(VERS 1995.0 SCOP PRE)
	IDEN(NAME "CALC1" INST(NAME "TVS641A" ID "B010100"))
	ENC(FORM ASC NVAL 9.91E+37 ORAN 9.9E+37 URAN –9.9E+37)
	DIM=X(TYPE IMPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "S")
	DIM=UPP(TYPE EXPL SIZE <nr1> UNIT "V")
	DIM=LOW(TYPE EXPL SIZE <nr1> UNIT "V")
	TRAC=TU(IND(LAB X) DEP(LAB UPP))
	TRAC=TL(IND(LAB X) DEP(LAB LOW))
	VIEW=ENVELOPE(ENV(UPP TU LOW TL))
	DATA(CURV(CTYP NONE))

CALCulate Data (binary complex)	DIF(VERS 1995.0 SCOP PRE)
	IDEN(NAME "CALC1" INST(NAME "TVS641A" ID "B010100"))
	ENC(FORM IFP32)
	DIM=X(TYPE IMPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "S")
	DIM=REAL(TYPE EXPL SIZE <nr1> UNIT "V")
	DIM=IMAG(TYPE EXPL SIZE <nr1> UNIT "V")
	TRAC=TR(IND(LAB X) DEP(LAB REAL))
	TRAC=TI(IND(LAB X) DEP(LAB IMAG))
	VIEW=COMPLEX(RCOM(REAL TR IMAG TI))
	DATA(CURV(CTYP NONE))

CALCulate Data (ASCII complex)	DIF(VERS 1995.0 SCOP PRE)
	IDEN(NAME "CALC1" INST(NAME "TVS641A" ID "B010100"))
	ENC(FORM ASC NVAL 9.91E+37 ORAN 9.9E+37 URAN –9.9E+37)
	DIM=X(TYPE IMPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "S")
	DIM=REAL(TYPE EXPL SIZE <nr1> UNIT "V")
	DIM=IMAG(TYPE EXPL SIZE <nr1> UNIT "V")
	TRAC=TR(IND(LAB X) DEP(LAB REAL))
	TRAC=TI(IND(LAB X) DEP(LAB IMAG))
	VIEW=COMPLEX(RCOM(REAL TR IMAG TI))
	DATA(CURV(CTYP NONE))

Table D–1: TVS600A Preambles and their formats (cont.)

Preamble Type	Preamble Format

CALCulate Data (binary polar)	DIF(VERS 1995.0 SCOP PRE)
	IDEN(NAME "CALC1" INST(NAME "TVS641A" ID "B010100"))
	ENC(FORM IFP32)
	DIM=X(TYPE IMPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "S")
	DIM=MAGN(TYPE EXPL SIZE <nr1> UNIT "V")
	DIM=PHAS(TYPE EXPL SIZE <nr1> UNIT "RAD")
	TRAC=TM(IND(LAB X) DEP(LAB MAGN))
	TRAC=TP(IND(LAB X) DEP(LAB PHAS))
	VIEW=POLAR(PCOM(MAGN TM PHAS TP))
	DATA(CURV(CTYP NONE))

CALCulate Data (ASCII polar)	DIF(VERS 1995.0 SCOP PRE)
	IDEN(NAME "CALC1" INST(NAME "TVS641A" ID "B010100"))
	ENC(FORM ASC NVAL 9.91E+37 ORAN 9.9E+37 URAN –9.9E+37)
	DIM=X(TYPE IMPL SCAL <nr3> OFFS <nr3> SIZE <nr1> UNIT "S")
	DIM=MAGN(TYPE EXPL SIZE <nr1> UNIT "V")
	DIM=PHAS(TYPE EXPL SIZE <nr1> UNIT "RAD")
	TRAC=TM(IND(LAB X) DEP(LAB MAGN))
	TRAC=TP(IND(LAB X) DEP(LAB PHAS))
	VIEW=POLAR(PCOM(MAGN TM PHAS TP))
	DATA(CURV(CTYP NONE))

Preamble Keywords

Table D–2 defines many of the keywords used in the TVS600A preambles listed in Table D–1. (For information on DIF expressions, see *Data Interchange Format* on page 3–131.)

Table D–2: Definitions for preamble keywords

Preamble Element	Description
CTYPe	Indicates the checksum type.
CURVe	Indicates the data that follows is dimensioned.
DATA	Begins the block containing the actual waveform data. When the preamble is requested, the DATA block is empty.
DEPendent	Indicates a dependent DIMension that constitutes the TRACe.
DIF	Data Interchange Format. Always followed by the SCPI DIF version and the block type, such as SCOPe PREamble.
DIM=LOW()	Defines the type, scale, offset, size, and units for the lower envelope of the vertical dimension of envelope waveform data.
DIM=UPP()	Defines the type, scale, offset, size, and units for the upper envelope of the vertical dimension of envelope waveform data.
DIM=X()	Defines the type, scale, offset, size and units for the horizontal dimension of the waveform data.
DIM=Y()	Defines the type, scale, offset, size, and units for the vertical dimension of non-envelope waveform data.
ENCode	Describes the FORMat used to encode the associated waveform data. The FORMat may be one of the following: ■ ASC 8 bit ASCII characters ■ IFP32 32 bit binary floating point ■ INT16 16-bit binary integer
EXPLicit	Values for the dimension are present in DATA(CURVe).
IDENtify	Identifies the source of the data and the instrument used to acquire the data.
IMPLicit	Values for this dimension are derived from a linear function, y=mx + b.
LABel	Uniquely identifies the DIMension statement that applies to a specified trace data block. Only labels defined by DIM blocks, such as X, Y, UPP, or LOW, can appear in a LABel statement.
NVALue	Defines the numeric value of NaN, Not a Number in the data block.
OFFSet	Describes an offset value to be added to data in DATA(CURVe) after the SCALe factor is applied.
ORANge	Defines the numeric value of over range values in the data block.

Table D–2: Definitions for preamble keywords (Cont.)

Preamble Element	Description
SCALe	Describes the scaling factor to be applied to the data in DATA(CURVe).
SIZE	Defines the number of data points in the DATA(CURVe) waveform.
TRACe	Describes relationships between DIMensions. TRACe statements define a name for the correlation between vertical and horizontal DIMensions.
TYPE	Indicates if the dimension is implicit or explicit.
UNIT	Describes the correct units for the data, such as Seconds, Volts, and Hz.
URANge	Defines the numeric value of under range values in the data block.
VIEW	Describes how the defined TRACes are to be interpreted. Using an envelope waveform as an example, the VIEW block defines which TRACe is the upper vertical information and which is the lower.

Glossary

AC coupling
A type of signal transmission that blocks the DC component of a signal but uses the dynamic (AC) component. Useful for observing an AC signal that is normally riding on a DC signal.

Accuracy
The closeness of the indicated value to the true value.

Acquisition
The process of sampling signals from input channels, digitizing the samples into data points, and assembling the data points into a waveform record. The waveform record is stored in memory. The trigger marks time zero in that process.

Acquisition interval
The time duration of the waveform record divided by the record length. The TVS600A Waveform Analyzer stores one data point for every acquisition interval.

Acquisition record
An array of definite length containing digital values that represent an analog input signal. The digital values are derived by measuring the voltage level of the input signal at precisely timed intervals (the sample interval).

ADC
Analog to digital converter. A circuit device that converts an analog signal, such as a temperature sensor, into a digital value.

Aliasing
A false representation of a signal due to insufficient sampling of high frequencies or fast transitions. A condition that occurs when a waveform analyzer digitizes at an effective sampling rate that is too slow to reproduce the input signal. The waveform stored by the waveform analyzer may have a lower frequency than the actual input signal.

Amplitude
The High waveform value less the Low waveform value.

Area
Measurement of the waveform area taken over the entire waveform or the gated region. Expressed in volt-seconds. Area above ground is positive; area below ground is negative.

ASCII
Acronym for the American Standard Code for Information Interchange.

Controllers transmit commands to the waveform analyzer using ASCII character encoding.

Attenuation

The degree the amplitude of a signal is reduced when it passes through an attenuating device such as a probe or attenuator. That is, the ratio of the input measure to the output measure. For example, a 10X probe will attenuate, or reduce, the input voltage of a signal by a factor of 10.

Automatic trigger mode

A trigger mode that causes the waveform analyzer to automatically acquire if triggerable events are not detected within a specified time period.

Average acquisition mode

In this mode, the waveform analyzer acquires a waveform that is the averaged result of several acquisitions. Averaging reduces the apparent noise. The waveform analyzer acquires data as in the sample mode and then averages it according to a specified number of averages.

Bandwidth

The highest frequency signal the waveform analyzer can acquire with no more than 3 dB (× .707) attenuation of the original (reference) signal.

Burst width

A timing measurement of the duration of a burst.

Channel

One type of input used for signal acquisition. The waveform analyzer has four channels.

Class 2

A term for ECL-level signals on the VXI Local Bus.

Command

Specifies an action for the instrument to perform.

Common Commands

See IEEE 488.2 Common Commands.

Controller

A computer or other device that sends commands to and accepts responses from the waveform analyzer.

Coupling

The association of two or more circuits or systems in such a way that power or information can be transferred from one to the other. You can couple the input signal to the trigger and vertical systems several different ways.

Cycle area

A measurement of waveform area taken over one cycle. Expressed in volt-seconds. Area above ground is positive; area below ground is negative.

Cycle mean
An amplitude (voltage) measurement of the arithmetic mean over one cycle.

Cycle RMS
The true Root Mean Square voltage over one cycle.

DC coupling
A mode that passes both AC and DC signal components to the circuit. Available for both the trigger system and the vertical system.

Delay time
The time between the trigger event and the acquisition of data.

Digitizing
The process of converting a continuous analog signal such as a waveform to a set of discrete numbers representing the amplitude of the signal at specific points in time. Digitizing is composed of two steps: sampling and quantizing.

Driver, functions
The TVS600A VXI*plug&play* Driver library included as part of the TVS600A VXI*plug&play* software that accompanies this product. The driver provides high-level functions for control of the waveform analyzer. The functions may be compiled and called as library functions in the programs you create.

DSP (Digital Signal Processor)
The DSP coordinates operation of the acquisition system and performs waveform transforms, calculations, and measurements.

Edge Trigger
Triggering occurs when the waveform analyzer detects the source passing through a specified voltage level in a specified direction (the trigger slope).

Envelope acquisition mode
A mode in which the waveform analyzer acquires a waveform that shows the variation extremes of several acquisitions.

Chained Commands and Queries
Commands and queries grouped together into a single message to be sent to the instrument. Each command and query in the chained message must be separated by a semicolon (;).

EOM
A generic acronym referring to the end-of-message terminator. The end-of-message terminator can be either a byte with the END bit set or the ASCII code for line feed (LF).

ERT sampling
Extended real time sampling. See Extended Real-time sampling for the definition.

Event Status Enable Register
Controls which events are summarized in the event status bit (bit 5) of the Status Byte Register.

Extended Real-time sampling (ERT)
A sampling mode where the waveform analyzer samples fast enough to completely fill a waveform record from a single trigger event and continues until Acquisition memory is full. Use ERT sampling to acquire very long acquisition records for a single channel.

Fall time
A measurement of the time it takes for trailing edge of a pulse to fall from a HighRef value (typically 90%) to a LowRef value (typically 10%) of its amplitude.

Fast Data Channel (FDC)
A fast transfer protocol for moving waveform records and other data between the waveform analyzer and a VXIbus controller. FDC is not supported by GPIB systems.

Frequency
A timing measurement that is the reciprocal of the period. Measured in Hertz (Hz) where 1 Hz = 1 cycle per second.

Ground (GND) coupling
Coupling option that disconnects the input signal from the vertical system.

GHZ
Gigahertz, 10^9 cycles per second

GPIB
Acronym for General Purpose Interface Bus, the common name for the communications interface system defined in IEEE Std 488.1.

High
The value used as 100% in automated measurements (whenever high ref, mid ref, and low ref values are needed as in fall time and rise time measurements). May be calculated using either the min/max or the histogram method. With the min/max method (most useful for general waveforms), it is the maximum value found. With the histogram method (most useful for pulses), it refers to the most common value found above the mid point. See *Appendix B: Algorithms* for details.

Holdoff, trigger
A specified amount of time after a trigger signal that elapses before the trigger circuit will accept another trigger signal. Trigger holdoff helps ensure a stable acquisition.

HZ
Hertz, cycles per second

IEEE

Acronym for the Institute for Electrical and Electronic Engineers.

IEEE 488.2 Common Commands

The set of commands and queries defined by the ANSI/IEEE Standard 488.2.

KHZ

Kilohertz, 10^3 cycles per second

Logical address

A specific, unique address setting for modules in a VXIbus system.

Low

The value used as 0% in automated measurements (whenever high ref, mid ref, and low ref values are needed as in fall time and rise time measurements). May be calculated using either the min/max or the histogram method. With the min/max method (most useful for general waveforms), it is the minimum value found. With the histogram method (most useful for pulses), it refers to the most common value found below the mid point. See *Appendix B: Algorithms* for details.

Maximum

Amplitude (voltage) measurement of the maximum amplitude. Typically the most positive peak voltage.

Mean

Amplitude (voltage) measurement of the arithmetic mean over the entire waveform.

Mb/s

Megabits per second.

MS/s

Megasamples per second.

MHZ

Megahertz, 10^6 cycles per second

Minimum

Amplitude (voltage) measurement of the minimum amplitude. Typically the most negative peak voltage.

Negative duty cycle

A timing measurement representing the ratio of the negative pulse width to the signal period, expressed as a percentage.

Negative width

A timing measurement of the distance (time) between two amplitude points — falling-edge *MidRef* (default 50%) and rising-edge *MidRef* (default 50%) — on a negative pulse.

Normal trigger mode

A mode on which the waveform analyzer does not acquire a waveform record unless a valid trigger event occurs. It waits for a valid trigger event before acquiring waveform data.

Output Queue

Stores query responses from the instrument.

Overshoot

Amplitude (voltage) measurement.

$$PositiveOvershoot = \frac{Max - High}{Amplitude} \times 100\%$$

Peak-to-Peak

Amplitude (voltage) measurement of the absolute difference between the maximum and minimum amplitude.

Period

A timing measurement of the time covered by one complete signal cycle. It is the reciprocal of frequency and is measured in seconds.

Positive duty cycle

A timing measurement of the ratio of the positive pulse width to the signal period, expressed as a percentage.

Positive width

A timing measurement of the distance (time) between two amplitude points — rising-edge *MidRef* (default 50%) and falling-edge *MidRef* (default 50%) — on a positive pulse.

Posttrigger

The specified portion of the waveform record that contains data acquired after the trigger event.

Preshoot measurement

Amplitude (voltage) measurement.

$$NegativeOvershoot = \frac{Low - Min}{Amplitude} \times 100\%$$

Pretrigger

The specified portion of the waveform record that contains data acquired before the trigger event.

Probe

A waveform analyzer input device.

Probe compensation

Adjustment that improves low-frequency response of a probe.

Pulse trigger
A trigger mode in which triggering occurs if the waveform analyzer finds a pulse, of the specified polarity, with a width between, or optionally outside, the user-specified lower and upper time limits.

Quantizing
The process of converting an analog input that has been sampled, such as a voltage, to a digital value.

Query
A message sent to the instrument that returns information about the state of the instrument.

Real-time sampling, RT Acquisition
A sampling mode where the waveform analyzer samples fast enough to completely fill a waveform record from a single trigger event. Use real-time sampling to capture single-shot or transient events.

Record length
The specified number of samples in a waveform.

Reference memories
TVS600A models only. Memory locations in a waveform analyzer used to store waveforms or other data. There are ten, REF1 through REF10. Stored waveforms are lost when the waveform analyzer is turned off or unplugged.

Rise time
The time it takes for a leading edge of a pulse to rise from a *LowRef* value (typically 10%) to a *HighRef* value (typically 90%) of its amplitude.

RMS
Amplitude (voltage) measurement of the true Root Mean Square voltage.

Normal acquisition mode
The waveform analyzer creates a record point by saving the first sample during each acquisition interval. That is the default mode of the acquisition.

Sample interval
The time interval between successive samples in a time base. For real-time digitizers, the sample interval is the reciprocal of the sample rate.

Sampling
The process of capturing an analog input, such as a voltage, at a discrete point in time and holding it constant so that it can be quantized.

SCPI
An acronym for Standard Commands for Programmable Instruments. A standard that provides guidelines for remote programming of instruments.

:Sequence[1]
The first trigger system in the SCPI standard model for triggering.

TRIGger:A provides Sequence1 functions. TRIGger: A is the main trigger system.

:Sequence[2]
The second trigger system in the SCPI standard model for triggering. TRIGger:B provides Sequence2 functions. TRIGger:B is the delayed trigger system.

Settings memory
Memory in a waveform analyzer used to store settings. The waveform analyzer saves the data even when the waveform analyzer is turned off or unplugged.

Slope
The direction at a point on a waveform. You can calculate the direction by computing the sign of the ratio of change in the vertical quantity (Y) to the change in the horizontal quantity. The two values are rising and falling.

Slot 0
The location in a VXIbus mainframe for a controller or resource manager module.

Soft Front Panel (SFP)
The TVS600A VXI*plug&play* Soft Front Panel application included as part of the TVS600A VXI*plug&play* software that accompanies this product. The SFP provides graphical user interface to most of the features of the waveform analyzer.

Stale data
Requesting data with a command such as TRACe when that channel was not previously acquired results in Error 230: Data corrupt or stale.

Standard Event Status Register
Records many types of events that occur in the instrument such as execution error and operation complete.

Status Byte Register
Summarizes information from other registers in the Status and Event Reporting System.

Subsystem Hierarchy Tree
A graphical representation of a subsystem of commands and queries used in the *TVS600 & TVS600A Command Reference* manual.

System Error and Event Queue
Stores error and event messages.

TEKSecure
A Tektronix custom command that initializes settings and waveform

memories. This command (`SYSTem:SECurity:IMMediate`) overwrites any previously stored data.

This feature erases all waveform and setup memory locations (setup memories are replaced with the factory setup). Then it checks each location to verify erasure. This feature finds use where this TVS600A Waveform Analyzer is used to gather security sensitive data, such as is done for research or development projects.

TTLTRG*
The TTL-level trigger bus on the VXIbus backplane.

Time base
The set of parameters that let you define the time and horizontal axis attributes of a waveform record. The time base determines when and how long to acquire record points.

Trigger
An event that marks time zero in the waveform record. It results in the acquisition of a waveform record.

Trigger level
The vertical level the trigger signal must cross to generate a trigger (on edge trigger mode).

VXIbus
A standardized backplane and system specification for modular instrumentation.

VXI Local Bus
Lines in the VXIbus backplane for direct communication between adjacent modules.

Waveform
The shape or form (visible representation) of a signal.

Waveform interval
The time interval between record points.

Index