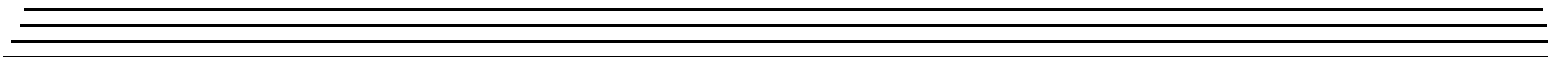
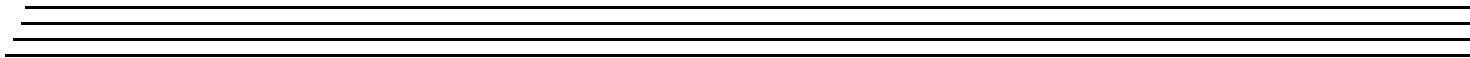
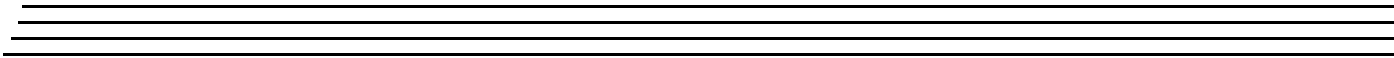




UM-23333-C

SCPI Programmer's Manual for the DT8837



Trademark and Copyright Information

Measurement Computing Corporation, InstaCal, Universal Library, and the Measurement Computing logo are either trademarks or registered trademarks of Measurement Computing Corporation. Refer to the Copyrights & Trademarks section on mccdaq.com/legal for more information about Measurement Computing trademarks. Other product and company names mentioned herein are trademarks or trade names of their respective companies.

© 2013 Measurement Computing Corporation. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form by any means, electronic, mechanical, by photocopying, recording, or otherwise without the prior written permission of Measurement Computing Corporation.

Notice

Measurement Computing Corporation does not authorize any Measurement Computing Corporation product for use in life support systems and/or devices without prior written consent from Measurement Computing Corporation. Life support devices/systems are devices or systems that, a) are intended for surgical implantation into the body, or b) support or sustain life and whose failure to perform can be reasonably expected to result in injury. Measurement Computing Corporation products are not designed with the components required, and are not subject to the testing required to ensure a level of reliability suitable for the treatment and diagnosis of people.

Table of Contents

About this Manual	9
Intended Audience.....	9
What You Should Learn from this Manual.....	9
Conventions Used in this Manual	9
Related Documents	10
Where to Get Help	10
 Chapter 1: Syntax Conventions	 11
Introduction.....	12
Types of SCPI Messages	12
Types of SCPI Commands	12
Common SCPI Commands	12
SCPI Subsystem Commands	14
Syntax of Program Messages	20
Syntax of Common SCPI Commands	21
Syntax of SCPI Subsystem Commands	21
About Short- and Long-Form Mnemonics	22
About Brackets, Braces, and Vertical Bars	22
Syntax of Response Messages.....	23
SCPI Data Types	24
Character Data Types	24
String Data Types	24
<NR1>Value Data Type	25
<NR2> Value Data Type.....	25
<NRr>Value Data Type	25
<NRf>Value Data Type	26
<Boolean> Data Type	27
<Block> Data Type	27
SCPI Expression Types	29
Numeric Expressions	29
Channel Lists	30
Numeric Lists.....	31
Data Interchange Format (DIF) Expressions.....	32
Instrument-Specifier Expressions	33
 Chapter 2: Using SCPI Commands with DT8837 Instrument Modules	 35
Installing SCPI Support Files for the DT8837	36
Configuring the LAN Settings of the DT8837	37
Setting the Static IP and Subnet Mask of the Instrument Module.....	37

Specifying a Description of the Instrument Module	37
Getting Information about the Instrument Module	38
Performing Input Operations	39
Enabling and Configuring Analog Input Channels	39
Enabling and Configuring the Tachometer Input Channel	40
Enabling and Configuring Counter/Timer Channels	42
Enabling the Analog Output Readback Channel	44
Configuring the Input Buffer	44
Configuring the Input Clock Source	45
Configuring the Input Trigger	48
Software Trigger	48
External, Digital (TTL) Trigger	49
Analog Threshold Trigger	49
LAN Trigger Packet	49
Trigger Bus	51
Arming Input Operations	53
Starting Input Operations	53
Determining the Status of the Analog Input Subsystem	54
Retrieving Scan Data from the Input Buffer	54
Stopping Input Operations	57
Performing Analog Output Operations	58
Enabling the Analog Output Channel	58
Configuring the Output Buffer	58
Configuring the Output Clock Source	59
Configuring the Trigger for Analog Output Operations	61
Software Trigger	61
External, Digital (TTL) Trigger	61
Analog Threshold Trigger	62
LAN Trigger Packet	62
Trigger Bus	62
Arming Output Operations	64
Starting Analog Output Operations	64
Determining the Status of the Analog Output Subsystem	64
Stopping Output Operations	65
Performing Digital Output Operations	67
 Chapter 3: Common SCPI Commands	69
Clear Status (*CLS)	70
Standard Event Status Enable Register (*ESE)	70
Standard Event Status Enable Register Query (*ESE?)	71
Standard Event Status Register Query (*ESR?)	71
Identification Query (*IDN?)	73

Operation Complete (*OPC)	74
Operation Complete Query (*OPC?)	74
Reset (*RST)	74
Self-Test Query (*TST?)	75
Service Request Enable (*SRE)	75
Service Request Enable Query (*SRE?)	75
Read Status Byte Query (*STB?)	76
Wait-to-Continue (*WAI)	77
Chapter 4: SCPI Subsystem Commands for the DT8837	79
SCPI Subsystem Command Hierarchy	80
STATus Subsystem Commands	83
Operation Condition Register Query	84
Operation Event Register Enable	85
Operation Enable Register Query	85
Operation Event Register Query	86
Presetting Registers	86
Questionable Condition Register Query	86
Questionable Enable Register	87
Questionable Enable Register Query	87
Questionable Event Register Query	88
SYSTem Subsystem Commands	89
Date Query	90
Description	90
Description Query	91
Error Query All	91
Error Query All Codes	92
ERRor Query Count	93
Error Query Next	94
Error Query Next Code	95
LAN Static IP - IP Address Configuration	96
LAN Static IP - IP Address Query	97
LAN Static IP - Subnet Mask Configuration	97
LAN Static IP - Subnet Mask Query	98
SCPI Version Query	98
Time Query	99
Time Zone Query	100
AD Subsystem Commands	101
Abort Analog Input Operation	102
Arm Analog Input Operation	103
ADC Synchronization Source Query	104
Analog Input Buffer Mode Configuration	104
Analog Input Buffer Mode Query	105

Analog Input Buffer Size Query	106
Analog Input Channel Enable	107
Analog Input Channel Enable Query	108
Analog Input Clock Source Configuration	109
Analog Input Clock Source Query	110
Analog Input Coupling Configuration	111
Analog Input Coupling Query	111
Analog Input Current Source Enable	112
Analog Input Current Source Enable Query	113
Analog Input Gain Configuration	114
Analog Input Gain Query	114
Analog Input Sampling Frequency Configuration	115
Analog Input Sampling Frequency Query	116
Analog Input Scan Status Query	117
Analog Input Status Bits Query	118
Analog Input Trigger Source Configuration	118
Analog Input Trigger Source Query	121
Analog Input Trigger Threshold (AIN1) Configuration	123
Analog Input Trigger Threshold (AIN1) Query	124
Fetch Analog Input Data	125
Initiate Analog Input Operation	128
TACHometer Subsystem Commands	130
Tachometer Channel Enable	131
Tachometer Channel Enable Query	131
Tachometer Period Type Configuration	132
Tachometer Period Type Query	133
Tachometer Self Clear Flag Configuration	133
Tachometer Self Clear Flag Query	134
Tachometer Stale Value Flag Configuration	135
Tachometer Stale Value Flag Query	135
COUNter Subsystem Commands	137
Counter/Timer Channel Enable	138
Counter/Timer Channel Enable Query	138
Counter/Timer Edge Configuration	139
Counter/Timer Edge Query	141
Counter/Timer Self Clear Flag Configuration	142
Counter/Timer Self Clear Flag Query	143
DA Subsystem Commands	145
Abort Analog Output Operation	146
Arm Analog Output Operation	147
Analog Output Buffer Mode Configuration	149

Analog Output Buffer Mode Query	150
Analog Output Buffer Read All Values	150
Analog Output Buffer Set Values	151
Analog Output Buffer Size Configuration	153
Analog Output Buffer Size Query	155
Analog Output Channel Enable	155
Analog Output Channel Enable Query	156
Analog Output Clock Source Configuration	157
Analog Output Clock Source Query	158
Analog Output Readback Enable	159
Analog Output Readback Enable Query	160
Analog Output Status Bits Query	160
Analog Output Trigger Source Configuration	161
Analog Output Trigger Source Query	164
Analog Output Update Frequency Configuration	166
Analog Output Update Frequency Query	167
Initiate Analog Output Operation	168
WTB (Wired Trigger Bus) Subsystem Commands	170
Trigger Bus Lines Configuration	171
Trigger Bus Lines Query	172
EVENT Subsystem Command	174
LAN Event Domain Configuration	175
LAN Event Domain Query	175
LAN Event Name Configuration	176
LAN Event Name Query	177
LAN Event Transmission Enable	178
LAN Event Transmission Enable Query	179
DOUTput Subsystem Commands	181
Digital Output AND Output Values	182
Digital Output OR Output Values	183
Digital Output Set State	184
Digital OUTput Query State	184
Chapter 5: Programming Flowcharts	187
Configuring the Instrument Module on the LAN	188
Performing Input Operations	189
Performing Analog Output Operations	192
Performing Digital Output Operations	193
Chapter 6: Product Support	199

Appendix A: Errors	201
Error Codes	202
Troubleshooting Errors	204
Error –110 Command Header Error	204
Error –410 Query Interrupted	205
Appendix B: Registers	207
Status Byte Register (STB)	208
Standard Event Status Enable Register (ESE)	209
Standard Event Status Register (ESR)	211
Operation Status Register	212
Appendix C: Examples	215
Features of the DT8837 SCPI Example Program	217
Opening and Running the DT8837 SCPI Example Program	218
Appendix D: Using HyperTerminal to Send and Receive SCPI Commands	219
Index	225

About this Manual

This manual describes how to use (Standard Commands for Programmable Instruments (SCPI) to communicate with the DT8837 LXI (LAN eXtensions for Instrumentation) instrument modules.

Intended Audience

This document is intended for instrument programmers who are responsible for writing SCPI-based programs for DT8837 LXI instrument modules.

What You Should Learn from this Manual

This manual provides detailed information about the SCPI commands that are available for communicating with DT8837 LXI instrument modules. This manual is organized as follows:

- [Chapter 1, “Syntax Conventions,”](#) describes the syntax conventions used.
- [Chapter 2, “Using SCPI Commands with DT8837 Instrument Modules,”](#) provides an introduction to SCPI commands.
- [Chapter 3, “Common SCPI Commands,”](#) describes the common SCPI commands that are available for DT8837 LXI instrument modules, including the command syntax, functional description, examples, and so on.
- [Chapter 4, “SCPI Subsystem Commands for the DT8837,”](#) describes the device-specific SCPI commands that available for DT8837 LXI instrument modules, including the command syntax, functional description, examples, and so on.
- [Chapter 5, “Programming Flowcharts,”](#) provides flow diagrams that show how to use the SCPI commands together to write a program that communicates with DT8837 LXI instrument modules.
- [Appendix A, “Errors,”](#) lists the errors that can be returned and describes how to troubleshoot frequently occurring errors.
- [Appendix B, “Registers,”](#) describes the registers that are used by SCPI commands.
- [Appendix C, “Examples,”](#) describes example applications that illustrate how to use SCPI commands to program DT8837 LXI instrument modules.
- [Appendix D, “Using HyperTerminal to Send and Receive SCPI Commands,”](#) describes how to use HyperTerminal to send and receive SCPI commands.
- An index completes this manual.

Conventions Used in this Manual

The following conventions are used in this manual:

- Notes provide useful information or information that requires special emphasis, cautions provide information to help you avoid losing data or damaging your equipment, and

warnings provide information to help you avoid catastrophic damage to yourself or your equipment.

- Items that you select or type are shown in **bold**.

Related Documents

Refer to the following documents for more information:

- *DT8837 User's Manual* (UM-23331). This manual describes the operation of the DT8837 instrument module.
- Standard Commands for Programmable Instruments (SCPI), Volume 1-4, Version 1999.0 May 1999, SCPI Consortium, 2515 Camino del Rio South, Suite 340, San Diego, CA 92108.
- IEEE Std 488.2-1992, The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017-2394, USA (ISBN 1-55937-238-9)

Where to Get Help

Should you run into problems installing or using SCPI commands to communicate with DT8837 LXI instrument modules, the Data Translation Technical Support Department is available to provide technical assistance. Refer to [Chapter 6](#) for more information. If you are outside the United States or Canada, call your local distributor, whose number is listed on Data Translation's web site (www.mccdaq.com).



Syntax Conventions

Introduction.....	12
Syntax of Program Messages	20
Syntax of Response Messages.....	23
SCPI Data Types	24
SCPI Expression Types	29

Introduction

SCPI (Standard Commands for Programmable Instruments) is a universal programming language for electronic test and measurement instruments, based on the IEEE 488.1 and IEEE 488.2 standards. DT8837 LXI instrument modules comply with the SCPI language and implement the IEEE-488.2 STD status structure.

You can issue these commands over VISA or sockets using TCP port 5025. Refer to [Appendix D](#) starting on [page 219](#) for information on using HyperTerminal to send and receive SCPI commands over TCP port 5025.

Types of SCPI Messages

To program a DT8837 LXI instrument module, you create program messages. A program message consists of one or more properly formatted SCPI commands sent from the controller to the DT8837 LXI instrument module. The program message, which may be sent at any time, requests that the instrument perform some action or send back data or status information; these requests are also called queries.

When queried, the DT8837 LXI instrument module sends a response message back to the controller. The response message consists of data in a specific SCPI format.

Refer to [page 20](#) for more information on the syntax of program messages and to [page 23](#) for more information on the syntax of response messages.

The following documents provide more information about SCPI programming:

- Standard Commands for Programmable Instruments (SCPI), Volume 1-4, Version 1999.0 May 1999, SCPI Consortium, 2515 Camino del Rio South, Suite 340, San Diego, CA 92108.
- IEEE Std 488.2-1992, The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017-2394, USA (ISBN 1-55937-238-9)

Types of SCPI Commands

Two types of SCPI commands are available: common commands, described below, and device-specific subsystem commands, described on [page 14](#). DT8837 LXI instrument modules respond to all of the required IEEE-488.2 common commands and support subsystem commands for measuring data and performing other device-specific functions.

Common SCPI Commands

Common SCPI commands, defined by IEEE488.2, control and manage generic system functions such as reset, self-test, configuration storage, and device identification. [Table 1](#) summarizes the common SCPI commands that are available for programming DT8837 LXI instrument modules. Refer to [Chapter 3](#) starting on [page 69](#) for a detailed description of these commands.

Table 1: Common SCPI Commands for Programming DT8837 LXI Instrument Modules

Type	Mnemonic	Description
Clear Status	*CLS	Clears all event registers summarized in the Status Byte (STB) register, described on page 208 .
Event Status Enable Register	*ESE	Enables specified bits in the Standard Event Status Enable register, described on page 209 .
Event Status Enable Register Query	*ESE?	Returns the current value of the Standard Event Status Enable register, described on page 209 .
Event Status Register Query	*ESR?	Returns the current value of the Standard Event Status register, described on page 211 , and then clears the register.
Identification Query	*IDN?	Returns the unique identity of the DT8837 LXI instrument module.
Operation Complete	*OPC	The Operation Complete bit (bit 0) of the Standard Event Status register, described on page 209 , is always enabled. Therefore, this command has no effect when used with the DT8837 LXI instrument module.
Operation Complete Query	*OPC?	The Operation Complete bit (bit 0) of the Standard Event Status register, described on page 209 , is always enabled. Therefore, this command always places the ASCII character 1 into the device's Output Queue.
Reset	*RST	Clears the Standard Event Status register, message queue, error queue, and Status Byte register, and stops any scans that are in progress.
Self-Test Query	*TST?	Always returns 0 for DT8837 LXI instrument modules.
Service Request Enable	*SRE	The Service Request Enable register is not used on these instrument modules. Therefore, this command has no effect when used with DT8837 LXI instrument modules.
Service Request Enable Query	*SRE?	The Service Request Enable register is not used on these instrument modules. Therefore, this command has no effect when used with DT8837 LXI instrument modules.
Status Byte Query	*STB?	Returns the current value of the Status Byte register, described on page 208 .
Wait	*WAI	Has no effect on DT8837 LXI instrument modules.

SCPI Subsystem Commands

SCPI subsystem commands are either measurement-related or other device-specific commands for programming DT8837 LXI instrument modules.

The following subsystems are available on DT8837 LXI instrument modules:

- **STATus** – The STATus subsystem includes commands that are related to the operational status of a DT8837 LXI instrument module. This subsystem is mandatory for SCPI-compliant devices.
- **SYSTem** – The SYSTem subsystem includes commands for querying errors returned by the instrument module, and configuring and querying system settings, including the LAN configuration of the instrument module.
- **AD** – The AD subsystem includes commands and queries for setting up the analog input features of the instrument module, including the clock, trigger, buffer, gain, and IEPE functions, arming the analog input subsystem, starting the analog input operation, stopping the analog input operation, getting status information, and getting data from the input buffer on the DT8837 instrument module.
- **DA** – The DA subsystem includes commands and queries for setting up the analog output features of the instrument module, including the clock, trigger, buffer, arming the analog output subsystem, starting the analog output operation, stopping the analog output operation, and getting status information.
- **TACHometer** – The TACHometer subsystem includes commands and queries for setting up the tachometer features of the instrument module.
- **COUNter** – The COUNter subsystem includes commands and queries for setting up the counter/timer features of the instrument module.
- **WTB** – The WTB (Wired Trigger Bus) subsystem includes commands and queries for configuring the Trigger Bus.
- **EVENT** – The EVENT subsystem includes commands and queries for configuring the LAN events.
- **DOUTput** – The DOUTput subsystem includes commands for setting the state of the digital output lines on the instrument module and a query for returning the state of the digital output lines on the instrument module.

[Table 2](#) summarizes the SCPI commands and queries available in each subsystem for programming DT8837 LXI instrument modules. Refer to [Chapter 4](#) starting on [page 79](#) for a detailed description of these commands.

**Table 2: Subsystem SCPI Commands for Programming
DT8837 LXI Instrument Modules**

Subsystem	Mnemonic	Description
STATus	STATus:OPERation[:EVENT]?	Has no effect on DT8837 LXI instrument modules.
	STATus:OPERation:CONDition?	Returns the current value of the Operation Condition register, described on page 212 .
	STATus:OPERation:ENABLE	Has no effect on DT8837 LXI instrument modules.
	STATus:OPERation: ENABLE?	Always returns 0 for DT8837 LXI instrument modules.
	STATus:PRESet	Has no effect on DT8837 LXI instrument modules.
	STATus:QUEStionable[:EVENT]?	Always returns 0 for DT8837 LXI instrument modules.
	STATus:QUEStionable:CONDition?	Always returns 0 for DT8837 LXI instrument modules.
	STATus:QUEStionable:ENABLE	Has no effect on DT8837 LXI instrument modules.
	STATus:QUEStionable:ENABLE?	Always returns 0 for DT8837 LXI instrument modules.
SYSTem	SYSTem:DATE?	Returns the current date of the DT8837 LXI instrument module. This date is updated automatically by an SNTP (Simple Network Time Protocol) server
	SYSTem:DESCRiption	Specifies a description for the device.
	SYSTem:DESCRiption?	Returns the description of the device.
	SYSTem:ERRor:ALL?	Queries the error/event queue for all the unread errors, returns an integer and a string that describes each error, and removes the error from the queue.
	SYSTem:ERRor:CODE:ALL?	Queries the error/event queue for all the unread errors, returns the error codes that corresponds to the unread errors in a comma-separated list, and removes the errors from the queue.
	SYSTem:ERRor:COUNt?	Queries the error/event queue for the number of unread errors and returns the count.
	SYSTem:ERRor[:NEXT]?	Queries the error/event queue for the next error, returns an integer and a string that describes the error, and removes the error from the queue.
	SYSTem:ERRor:CODE[:NEXT]?	Queries the error/event queue for the next error, returns an integer that corresponds to the error code, and removes the error from the queue.
	SYSTem:COMMunicate:NETwork: IPAddr	Sets the static IP address used by the DT8837 LXI instrument module on the network.
	SYSTem:COMMunicate:NETwork: IPAddr?	Returns the static IP address currently used by the DT8837 LXI instrument module on the network.
	SYSTem:COMMunicate:NETwork: MASK	Sets the subnet mask of the static IP address used by the DT8837 LXI instrument module on the network.
	SYSTem:COMMunicate:NETwork: MASK?	Returns the subnet mask of the static IP address currently used by the DT8837 LXI instrument module on the network.

**Table 2: Subsystem SCPI Commands for Programming
DT8837 LXI Instrument Modules (cont.)**

Subsystem	Mnemonic	Description
SYSTem (cont.)	SYSTem:VERSion?	Returns the SCPI version number to which the DT8837 LXI instrument module complies.
	SYSTem:TIME?	Returns the current time used by the DT8837 LXI instrument module. This time is updated automatically by an SNTP server.
	SYSTem:TZONe?	Returns the time zone that is currently used by the instrument module, as an offset from GMT (Greenwich Mean Time).
AD	AD:BUFFer:MODE	Configures the wrap mode of the input buffer.
	AD:BUFFer:MODE?	Returns the currently configured wrap mode for the input buffer.
	AD:ENABle	Enables or disables sampling of the specified analog input channels.
	AD:ENABle?	Returns whether sampling is enabled or disabled for the specified analog input channels.
	AD:GAIN[:CONFigure]	Configures the gain as 1 or 10 for the specified analog input channels.
	AD:GAIN[:CONFigure]?	Returns the gain for the specified analog input channels.
	AD:COUPling[:CONFigure]	Configures AC or DC coupling for the specified analog input channels.
	AD:COUPling[:CONFigure]?	Returns the coupling type for the specified analog input channels.
	AD:BIAS:CURREnt:ENABle	Enables or disables the use of the 4 mA current source for the specified analog input channels.
	AD:BIAS:CURREnt:ENABle?	Returns whether the 4 mA current source is enabled or disabled for the specified analog input channels.
	[AD:]CLOCK:SOURce	Specifies the clock source (internal or external) for the ADC and DAC master clock generators.
	[AD:]CLOCK:SOURce?	Returns the currently configured clock source for the ADC and DAC master clock generators.
	AD:CLOCK:FREquency[:CONFigure]	Configures the sampling rate for the input operation.
	AD:CLOCK:FREquency[:CONFigure]?	Returns the currently configured sampling rate for the input operation.
	AD:SYNc:SOURce?	Returns the currently configured source of the ADC synchronization pulse from the Trigger bus.
	AD:TRIGger[:SOURce]	Specifies the source of the trigger (immediate, external TTL, analog threshold, Trigger Bus, LAN) that starts the input operation.

**Table 2: Subsystem SCPI Commands for Programming
DT8837 LXI Instrument Modules (cont.)**

Subsystem	Mnemonic	Description
AD (cont.)	AD:TRIGger[:SOURce]?	Returns the currently configured trigger source for the analog input subsystem.
	AD:TRIGger:LEVel[:AIN1]	When the trigger source is analog threshold, specifies the programmable trigger threshold level (between 200 and 9800 mV) of analog input channel 1 that starts the input or output operation.
	AD:TRIGger:LEVel[:AIN1]?	Returns the currently configured threshold level of the analog threshold trigger.
	AD:ARM	Arms the A/D subsystem by writing the input configuration to the instrument module.
	AD:INITiate	Starts the analog input operation when the configured trigger is detected.
	AD:ABORt	Stops the analog input operation on the DT8837 LXI instrument module.
	AD:FETCh?	Returns a specified number of samples from the input buffer.
	AD:STATus:SCAN?	Returns the indices of the chronologically oldest and most recent scan records in the input buffer on the instrument module.
	AD:STATus?	Returns the status bits for the A/D subsystem.
TACHometer	TACHometer:ENABle	Enables or disables sampling of the tachometer input channel.
	TACHometer:ENABle?	Returns whether sampling is enabled or disabled for the tachometer input channel.
	TACHometer:PERiod[:CONFigure]	Specifies the type of period to measure for the tachometer input signal.
	TACHometer:PERiod[:CONFigure]?	Returns the currently configured type of period to measure for the tachometer input signal.
	TACHometer:SCLR[:CONFigure]	Specifies whether the value read between measurements is cleared or retained.
	TACHometer:SCLR[:CONFigure] ?	Returns whether the value read between measurements is cleared or retained.
	TACHometer:SFLAG[:CONFigure]	Specifies whether to use the most significant bit (MSB) of the measurement value to indicate new or old data.
	TACHometer:SFLAG[:CONFigure]?	Returns whether the most significant bit (MSB) of the measurement value is used to indicate new or old data.

**Table 2: Subsystem SCPI Commands for Programming
DT8837 LXI Instrument Modules (cont.)**

Subsystem	Mnemonic	Description
COUNter	COUNter[<n>]:CONFigure	Configures the operation of the counter/timer channels.
	COUNter[<n>]:CONFigure?	Returns the configuration of the counter/timer channels.
	COUNter[<n>]:ENABle	Enables or disables sampling of the counter/timer channels.
	COUNter[<n>]:ENABle?	Returns whether sampling is enabled or disabled for the counter/timer channels.
DA	DA:ENABle	Enables or disables updating of the analog output channel at the configured clock rate.
	DA:ENABle?	Returns the state (enabled or disabled) of updating the analog output channel.
	DA:READ:ENABle	Enables or disables reading the value of the analog output channel in the analog input data stream.
	DA:READ:ENABle?	Returns the state (enabled or disabled) of reading the value of the analog output channel in the analog input data stream.
	[DA:]CLOCK:SOURce	Specifies the clock source (internal or external) for the ADC and DAC master clock generators.
	[DA:]CLOCK:SOURce?	Returns the currently configured clock source for the ADC and DAC master clock generators.
	DA:CLOCK:FREquency[:CONFigure]	Configures the clock frequency for the analog output operation.
	DA:CLOCK:FREquency[:CONFigure]?	Returns the currently configured clock frequency for the output operation.
	DA:TRIGger[:SOURce]	Specifies the source of the trigger (immediate, external TTL, analog threshold, Trigger Bus, LAN) that starts the output operation.
	DA:TRIGger[:SOURce]?	Returns the currently configured trigger source for the analog output subsystem.
	DA:BUFFer:SIZe[:SCANs]	Specifies the number of samples to write to the output buffer; the analog output channel is updated with these values when the output operation is armed and triggered.
	DA:BUFFer:SIZe[:SCANs]?	Returns the number of scans that the output buffer was configured to hold.
	DA:BUFFer[:DATA]	Stores samples in the output buffer; the analog output channel is updated with these samples when the analog output operation is armed and triggered.
	DA:BUFFer[:DATA]?	Returns the analog output samples that were written to the output buffer.

**Table 2: Subsystem SCPI Commands for Programming
DT8837 LXI Instrument Modules (cont.)**

Subsystem	Mnemonic	Description
DA (cont.)	DA:ARM	Arms the D/A subsystem by writing the output configuration to the instrument module.
	DA:INITiate	Starts the analog output operation when the configured trigger is detected.
	DA:ABORt	Stops the analog output operation on the DT8837 LXI instrument module.
	DA:STATus?	Returns the status bits for the D/A subsystem.
WTB	WTB:LXI<n>:ENABle	Configures a specific line of the Trigger Bus as either driven (an output) or disabled.
	WTB:LXI<n>:ENABle?	Returns the configuration of a specified line of the Trigger Bus.
EVENT	EVENT:DOMain	Sets the value of the LXI domain octet that is transmitted and received in all LXI LAN event packets.
	EVENT:DOMain?	Returns the value of the LXI domain octet that is transmitted and received in all LXI LAN event packets.
	EVENT:LAN<n>:TRANsmit[:ENABle]	Enables or disables a specific LAN event for transmission from the instrument module when the analog input subsystem is triggered.
	EVENT:LAN<n>:TRANsmit[:ENABle]?	Returns the enabled/disabled state of a specific LAN event.
	EVENT:LAN<n>:NAME	Configures the LAN event ID field in the LXI LAN packet.
	EVENT:LAN<n>:NAME?	Returns the event ID in the LXI LAN packet.
DOUTput	DOUTput	Sets the value of the four digital output lines on the DT8837 LXI instrument module.
	DOUTput?	Returns the current value of the four digital output lines on the DT8837 LXI instrument module.
	DOUTput:AND	Performs a bitwise AND operation on the specified value and the current value of the digital output port.
	DOUTput:OR	Performs a bitwise OR operation on the specified value and the current value of the digital output port.

Syntax of Program Messages

A program message consists of one or more properly formatted SCPI commands sent from the controller to a DT8837 LXI instrument module to request some action or to query the instrument module for a response.

Figure 1 shows the syntax of a program message:

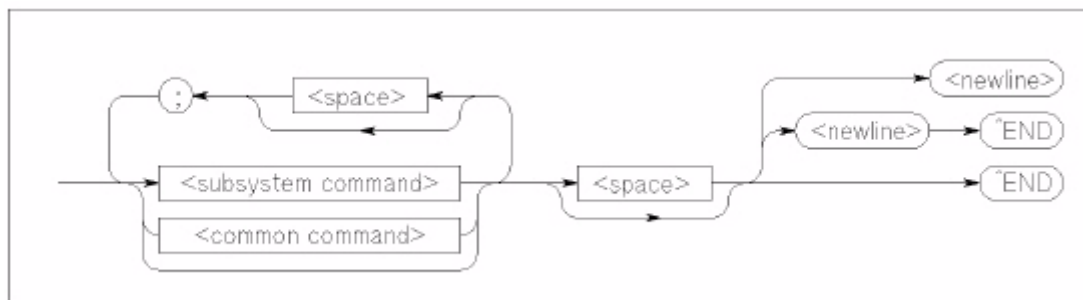


Figure 1: Syntax of Program Messages

A semicolon (;) is used to separate commands within the same command group, and can also minimize typing. For example, you could send the following program message to set the state of the digital output port (all 4 digital output lines) and read the status of the digital output lines:

```
:OUTPut 15; :OUTput?
```

Use a semicolon and a colon to link commands from different groups. For example, this program message returns the contents of the Operation Status register, and then sets the state of the digital output port:

```
:STATUS:OPERation:CONDition? ;:OUTPut 15
```

To terminate a program message, use one of the following program message terminators:

- <newline> or the <NL> character
- <^END>

<^END> means that the IEEE-488 EOI (End-Or-Identify) message was asserted at the same time that the last data byte was sent. <^END> is interpreted as a <NL> character and can be used to terminate a command string in place of a <NL> character.

- <newline><^END>

Terminating the program message always resets the current SCPI command path to the root level.

The following subsections describe the syntax of both the common SCPI commands and the subsystem SCPI commands.

Syntax of Common SCPI Commands

Figure 2 shows the syntax of common SCPI commands.

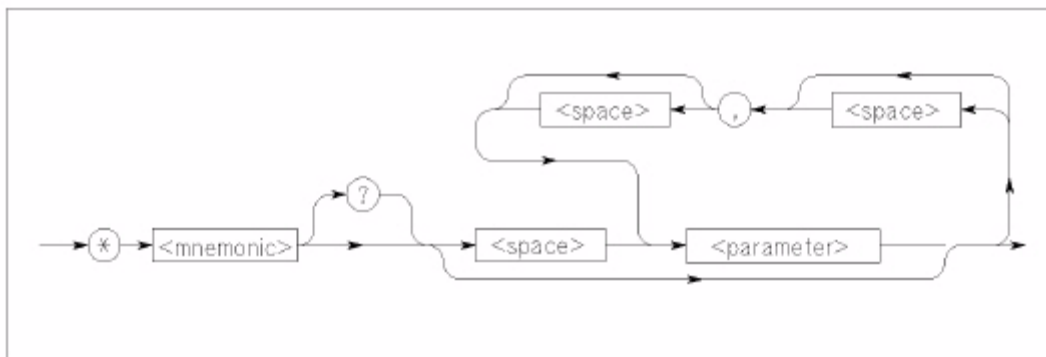


Figure 2: Syntax of Common SCPI Commands

Common SCPI commands begin with an asterisk (*). The command mnemonic is case-insensitive. For example, the following commands have the same effect:

```
*RST
*rst
*Rst
```

Queries requires a question mark (?) at the end of the command header, as follows:

```
*IDN?
```

Syntax of SCPI Subsystem Commands

Figure 3 shows the syntax of SCPI subsystem commands.

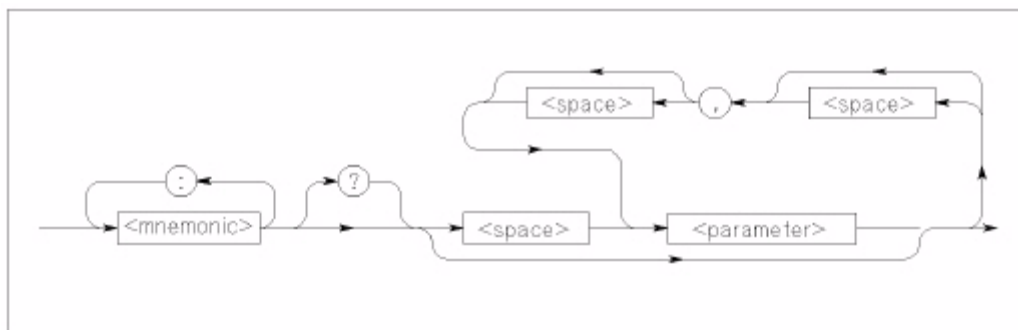


Figure 3: Syntax of SCPI Subsystem Commands

Use a colon (:) to separate command keywords or mnemonics, as shown below:

```
:STATus:OPERation:ENABle
```

Queries require a question mark (?) at the end of the command header, as follows:

```
:STATus:OPERation:CONDition?
```

About Short- and Long-Form Mnemonics

Note that the command syntax shows most command mnemonics (and some parameters) as a mixture of upper- and lower-case letters. The upper-case letters indicate the abbreviated spelling for the command. For shorter program lines, use the abbreviated form; for better program readability, use the long form.

You can use either upper- or lower-case letters to specify the command, as the mnemonic is case-insensitive. For example, here is the long form of a command:

```
SYSTem:COMMunicate:NETwork:MASk?
```

And, here is the short-form of this command:

```
Syst:Comm:NET:Mas?
```

About Brackets, Braces, and Vertical Bars

Some parameters are enclosed in square brackets ([]), indicating that the parameter is optional and can be omitted. The brackets are not sent with the command. If you do not specify a value for an optional parameter, the instrument module uses a default value.

Triangle brackets (< >) indicate that you must specify a value for the enclosed parameter. The brackets are not sent with the command.

Braces ({ }) enclose the parameter choices for a given command. The braces are not sent with the command.

A vertical bar (|) separates multiple parameter choices for a given command.

Syntax of Response Messages

A response message consists of data in a specific SCPI format that was requested from a DT8837 LXI instrument module from the controller.

Figure 4 shows the syntax of a response message from a DT8837 LXI instrument module:

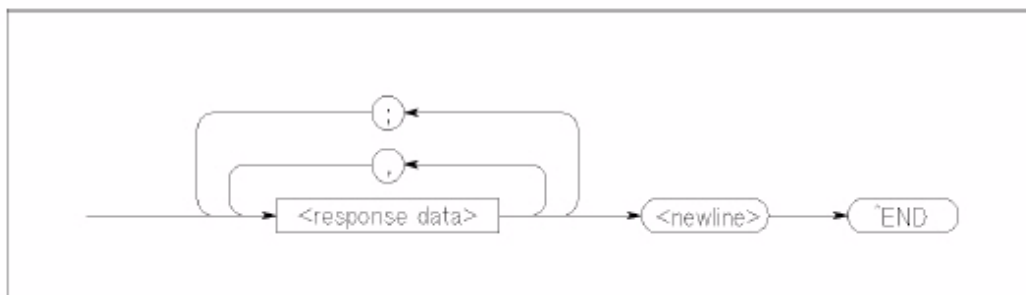


Figure 4: Response Message Syntax

Response messages may contain both commas (,) and semicolons (;) as separators. When a single query command returns multiple values, a comma is used to separate each data item. When multiple queries are sent in the same message, the groups of data items corresponding to each query are separated by semicolons.

The terminator for a response message is always <newline><^END>.

Note: For character data types in response messages, only the short-form of the mnemonic is returned in all uppercase letters.

SCPI Data Types

SCPI defines several different data types for use in program messages to a DT8837 LXI instrument module and for use in response messages from a DT8837 LXI instrument module.

DT8837 LXI instrument modules use the following subset of SCPI data types:

- Character
- String
- <NR1>
- <NR2>
- <NRr>
- <NRf>
- <Boolean>
- <Block>

This section summarizes these data types. Refer to the SCPI standards document for more information about these data types.

Character Data Types

If a command parameter takes a character data type, a specific number of settings are allowed for the parameter. For example, in the following command, you can specify one of the following character data types: *AC* for AC coupling, or *DC* for DC coupling:

```
AD:COUPling[:CONFIgure] {AC|DC} [,<source list>]
```

Character data types have the following characteristics:

- Can have either the short or long form in program messages and are returned in short-form only in response messages
- Are case insensitive in program messages and are in uppercase only in response messages
- Must have a specific length

String Data Types

Strings used in command parameters and responses follow these rules:

- Strings are enclosed in double quotes " "
For example,
"This is an example"
specifies the following string: This is an example

- Use double quotes within double quotes to indicate the part of the string that should appear in quotes; note that double and single quotes used inside the string must be duplicated. For example,

```
"This is the "example"
```

specifies the following string: This is the “example”

- ? Strings are case sensitive

<NR1>Value Data Type

The <NR1> value data type is used to specify zero, and positive and negative integer decimal values, including optional signs. If you need to indicate decimal points, use the <NR2> value data type, instead.

The following values are examples of the <NR1> data type:

```
0          255          -2
```

<NR2> Value Data Type

The <NR2> value data type is used to specify zero, and positive and negative decimal values, including optional signs and decimal points.

The difference between <NR1> and <NR2> is the explicit decimal point. Note that 0 is a special case and redundant decimal points are ignored.

The following values are examples of the <NR2> data type:

```
-1.234      1.0          0.0
```

<NRr>Value Data Type

The <NRr> data type is used to specify a non-decimal numeric value, such a hexadecimal, octal, or binary numeric value.

[Figure 5](#) shows the format of the <NRr> data type:

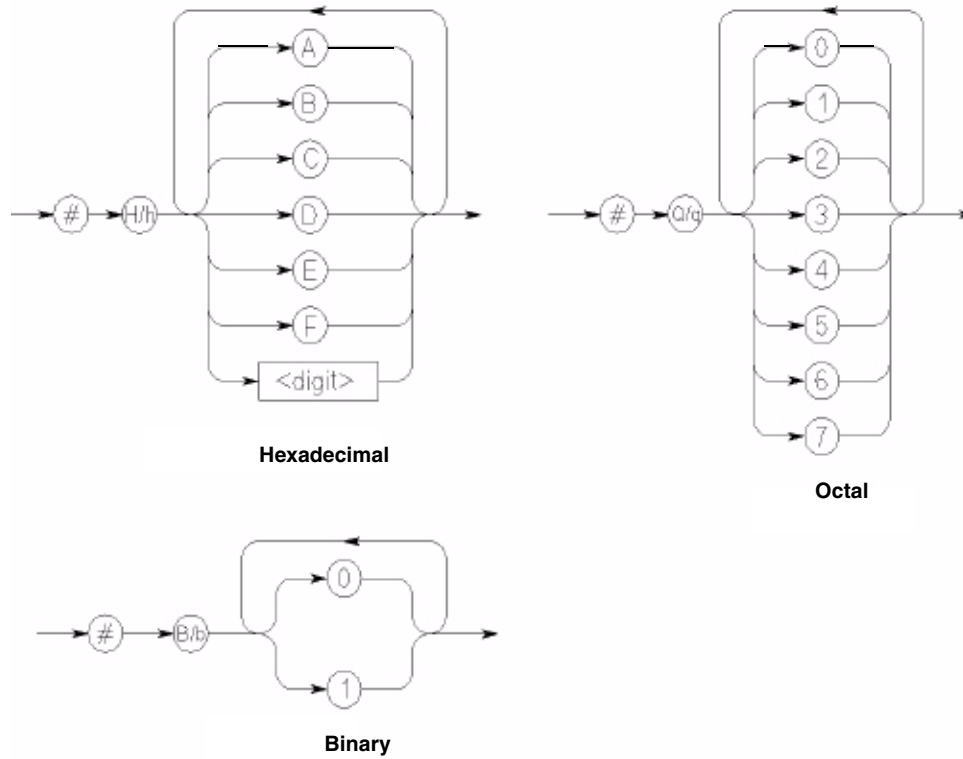


Figure 5: <NRr>Value Data Type

The following examples show how the number 16384 in decimal format is represented as a **<NRr>** data type:

- *Hexadecimal <NRr> value:* #H4000
- *Octal <NRr> value:* #Q40000
- *Binary <NRr> value:* #B100000000000000

<NRf>Value Data Type

The **<NRf>** data type is used to specify a floating-point value. These values include digits with an implied decimal point, an explicit decimal point, or an explicit decimal point and an exponent.

The following values are examples of the **<NRf>** data type:

6553 1.525e-4 0.100000

<Boolean> Data Type

A Boolean data type for a parameter and response represents a single binary condition that is either True or False. Boolean values are defined as follows:

- 0 or OFF – indicates that the condition is False
- 1 or ON – indicates that the condition is True

Note that the characters OFF and ON are not case sensitive. While a DT8837 LXI instrument module accepts the characters OFF and ON instead of 0 and 1, if queried, these instrument modules return Boolean responses as either 0 or 1.

<Block> Data Type

The <Block> data type is used to transfer array and system-defined blocks of data between the controller and the instrument module. This is the most efficient data format for transferring data.

Figure 6 shows the format of the <Block> data type:

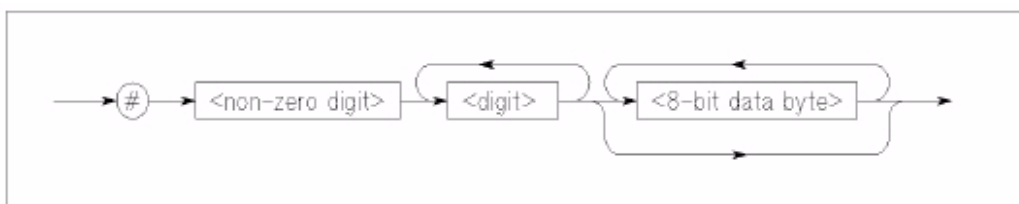


Figure 6: Block Data Type

where:

- # is the starting character of the block
- <non-zero digit> is a single decimal value that specifies how many digits will follow in the next field
- <digit> is a value in <NR1> format that specifies how many <8-bit data byte>s follow. In effect, this field specifies the length, in bytes, of the remainder of the block.
- <8-bit data byte> specifies each 8-bit byte of the block. (The number of 8-bit data bytes is specified in the preceding field.) The contents of this block of bytes is dependent on the specific command.

Note: Prior to interpreting multi-byte values, such as floats, integers, unsigned integers, and so on, you must convert values from network-byte order (where the highest bytes are sent first; i.e., big-endian) to host-byte-order (where the lowest bytes are read first, i.e., little endian).

The following example shows how a block of five data bytes is formatted:

```
#1<5><b0><b1><b2><b3><b4>
```

SCPI Expression Types

SCPI defines five types of expressions:

- Numeric expressions
- Channel lists
- Numeric lists
- Data Interchange Format (DIF) expressions
- Instrument-specifier expressions

The following subsections summarize these expressions. Refer to the SCPI standard document for more information about these expressions.

Numeric Expressions

A numeric expression is a collection of terms which evaluates to a trace, number, array, or other data element. Expressions can contain terms which are numbers, traces, variables, or expressions. The syntax of a numeric expression is shown in [Figure 7](#):

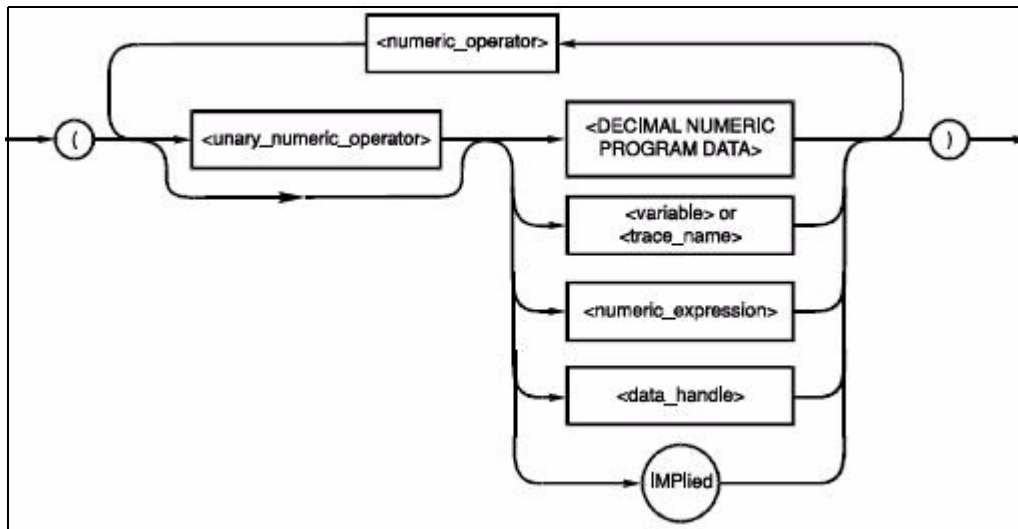


Figure 7: Syntax of a Numeric Expression

where,

- `<numeric_operator>` is defined as one of the following operators: `+`, `-`, `*`, `/`, `^`, `MOD`, `DIV`, `ADD`, `OR`, `EXOR`
- `<unary_numeric_operator>` is defined as one of the following operators: `-`, `+`, `NOT`. Unary operators should not be used with signed numbers.
- `<variable>` or `<trace_name>` is defined as character program data

Expressions are evaluated according to the following precedence rules:

1. Enclosed by parentheses
2. Unary operators (+ and -)
3. ^(exponentiation)
4. * (multiplication), / (division), MOD and DIV
5. + (addition) and - (subtraction)
6. NOT
7. AND
8. OR and EXOR
9. Left to right

Elements in a numeric expression are promoted to the size and type of the most complex element, and the result of the expression is of that type. For example, an expression containing a <trace_name> is evaluated according to the rules for arithmetic on traces, and the result is a trace. If an expression contains both a <trace_name> and scalar data, a trace is created with all elements set to the value of the scalar data before arithmetic is performed. For example, if TREF is a trace_name, the expression (TREF-3) results in a trace that is the same size as TREF, with each data element lower by three.

Channel Lists

DT8837 LXI instrument modules use channel lists as parameters in certain commands and in responses to certain queries.

Channel lists are used to specify the analog input channels on a DT8837 LXI instrument module that are used to measure input channels. A channel list may appear in a measurement, configuration, or command. By design, all analog input channels that are specified in the channel list are measured simultaneously.

The syntax of a channel list expression is shown in [Figure 8](#):

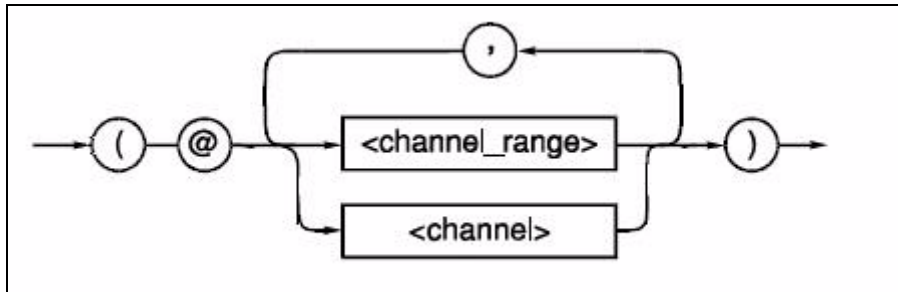


Figure 8: Syntax of a Channel List Expression

where:

- (is the starting character of the channel list
- @ is the next character of the channel list
- <channel range> consists of two channels in <NR1> format separated by a colon.

The first channel in the range must be lower than the second in the range. For example, for analog input channels on the DT8837 instrument module, the first channel is 1 and the last channel is 4.

Separate multiple <channel_range> elements with commas.

- <channel> consists of one channel number in <NR1> format. Separate multiple <channel> elements with commas.
-) terminates the channel list expression

For example, to set the gain on analog input channels 1 through 4, use one of the following commands:

```
:AD:GAIN 10 (@1:4)
```

```
:AD:GAIN 10 (@1,2,3,4)
```

To set the gain for analog input channel 3 only, use this command:

```
:AD:GAIN 10 (@3)
```

Numeric Lists

A numeric list is an expression format for compactly expressing numbers and ranges of numbers in a single parameter. The syntax of a numeric list expression is shown in [Figure 9](#):

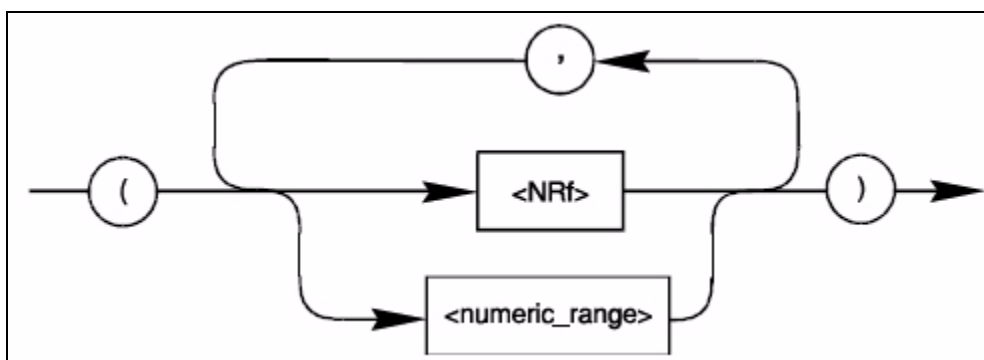


Figure 9: Syntax of Numeric List Expressions

where:

- <NRf> is an extended format, described on [page 26](#).
- <numeric_range> is defined as two <NRf> data types separated by colons. The range is inclusive of the specified numbers.

Data Interchange Format (DIF) Expressions

The data interchange format is block-structured and lets software packages and instruments share waveform and other data.

The following block types are available in DIF expressions:

- DATA block – Contains the data.
- IDENTify block – Describes the manner and environment in which the data was obtained.
- DIMension, ORDer, and ENCode blocks – Describe how the data is physically represented and logically organized.
- TRACe and VIEW blocks – Provide semantic information about the data.
- REMark block – Contains textual comments regarding the data.
- DIF block – Identifies the block as a <dif_expression> and describes the version of SCPI that is used.

In a DIF expression, each hierarchical level of a block is introduced by a block name with its subordinate elements enclosed in parentheses. A block may have a modifier and may contain subordinate blocks and keyword units, or both. Keyword units consist of a keyword followed by one or more values. If a keyword has more than one value, the values are separated by commas. The following example shows a simple data set. All blocks and keywords are indented to show their hierarchical relationship:

```
(DIF (VERsion 1993.0)
IDENTify (
    NAME "Data Format Example"
    TEST (
        NUMBer "7D4", "2.4")
    ENCode (
        HRANge 75
        RANge 25)
DIMension=X (
    TYPE IMPLicit
    SCALe 0.01
    SIZE 7
    UNITs "S")
DIMension=Y (
    TYPE EXPLicit
    SCALe 0.02
    OFFSet 0.1UNITs "V")
DATA (
    CURVe(
        VALues 49.0, 48.0, 50.2, 61.3, 68.5, 38.6, 48.0)))
```

The data interchange format overall is formatted as an IEEE 488.2 <EXPRESSION PROGRAM DATA> element. Within this element, the various blocks, modifiers, keywords, and value types are composed of syntactic elements that, for the most part, are identical to the corresponding types specified in IEEE 488.2 or a subset of them. Refer to the IEEE 488.2 standard for more information.

Instrument-Specifier Expressions

An `<instrument_specifier>` is a combination of one or more base functionality keywords along with optional additional functionality keywords that define an instrument class. The syntax of an instrument-specifier expression is shown in [Figure 10](#).

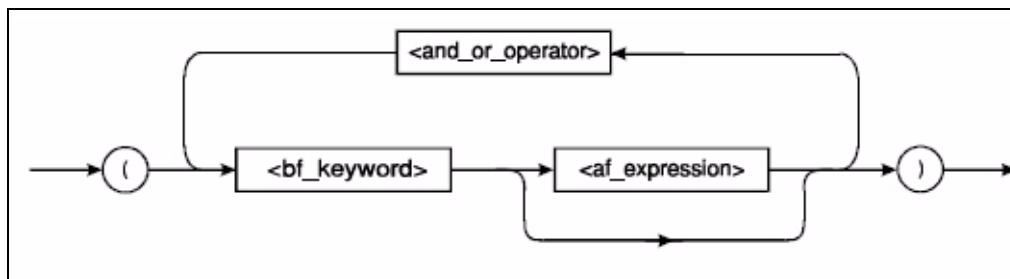


Figure 10: Syntax of an Instrument-Specifier Expression

where:

- (is the starting character of the instrument-specifier expression
- `<and_or_operator>` is either `'&'` (ASCII hexadecimal 26) or `'|'` (ASCII hexadecimal 7C)
- `<bf_keyword>` is a base functionality keyword
- `<af_expression>` includes additional `<and_or_operator>`s and additional functionality keywords
-) terminates the instrument-specifier expression



Using SCPI Commands with DT8837 Instrument Modules

Installing SCPI Support Files for the DT8837	36
Configuring the LAN Settings of the DT8837	37
Performing Input Operations	39
Performing Analog Output Operations	58
Performing Digital Output Operations	67

Installing SCPI Support Files for the DT8837

If you have not already done so, install the SCPI support files, including the documentation for your DT8837 LXI instrument module, this manual, and the SCPI example programs for the DT8837, by performing the following steps:

1. Insert the DT8837 CD into your CD-ROM or DVD drive.
The installation program should automatically start, and the DT8837 installation program should appear.
2. If the installation program does not automatically start, double-click **Setup.exe** from the CD.
The DT8837 installation program appears.
3. Click **Install from Web (recommended)** to get the latest version of the software or **Install from CD** to install the software from the CD.
4. If you are installing from the web, click the link to install the DT8837 SCPI software.
5. If you are installing from the DT8837 CD, perform these steps:
 - a. Click **Install DT8837 Software**.
 - b. Click **Install Selected Features** and follow the prompts to install the example programs and documentation.
 - c. When you are finished with the DT8837 CD, click **Quit Installer**.

To access the SCPI documentation and examples for the DT8837, from the Windows Start menu, click **Programs -> Data Translation, Inc -> Instruments -> DT8837 -> SCPI Support**.

Configuring the LAN Settings of the DT8837

The Ethernet address of a DT8837 instrument module consists of two parts:

- The IP address – The “Internet Protocol” numeric address that identifies where messages are sent or received on the LAN (Local Area Network).
- Subnet mask – A 32-bit value that enables the recipient of IP packets to distinguish the network ID and host ID portions of the IP address.

The instrument module can acquire an Ethernet address in one of the following ways:

- DHCP (Dynamic Host Configuration Protocol server) – The address is set by the network server automatically when the DT8837 instrument module is powered on. The address is different each time the instrument module is powered on.

By default, DHCP is enabled for DT8837 instrument modules.

- Auto-IP – If the DHCP server is not available, the instrument module configures its own IP address in the range of 169.254.0.0 to 169.254.255.255 with a subnet mask of 255.255.0.0. Like with DHCP, the address is different each time the instrument module is powered on.

By default, Auto-IP is enabled for DT8837 instrument modules.

- Static IP – Using SCPI commands, you can specify the static IP address of the instrument module. The static IP address does not change when the instrument module is powered on.

The following subsections provide information about configuring and returning the LAN settings of DT8837 instrument modules.

Setting the Static IP and Subnet Mask of the Instrument Module

You can configure the static IP address and subnet mask of the instrument module on the LAN using the following SCPI commands:

- Set the static IP address of the instrument module using the **SYSTem:COMMunicate:NETwork:IPaddr** command, described on [page 96](#).
- Set the subnet mask of the instrument module using the **SYSTem:COMMunicate:NETwork:MASK** command, described on [page 97](#).

Specifying a Description of the Instrument Module

To make identifying the instrument module easier, you can specify a description of the instrument module using the **SYSTem:DESCription** command, described on [page 90](#).

Getting Information about the Instrument Module

Using SCPI commands, you can return the following information about each DT8837 instrument module on the LAN:

- IP address – Use the **SYSTem:COMMunicate:NETwork:IP?** query, described on [page 97](#), to return the IP address of the instrument module, regardless of the method (DHCP, Auto-IP, static IP) used to acquire the address.
- Subnet mask – Use the **SYSTem:COMMunicate:NETwork:MASK?** command, described on [page 98](#), to return the subnet mask of the instrument module, regardless of the method (DHCP, Auto-IP, static IP) used to acquire the subnet mask.
- Description – Use the **SYSTem:DESCRiption?** command, described on [page 91](#), to return the description of the instrument module.

Performing Input Operations

You can acquire data from up to four analog input channels as well the tachometer input channel, counter/timer input channels, and the analog output readback channel on the DT8837 LXI instrument module. The input operation starts when the specified trigger source is detected. The operation is paced by the specified clock source at the specified input clock frequency.

To set up and acquire input data from the DT8837 using SCPI commands, perform the following steps:

1. Enable and configure the analog input channels that you want to sample, as described on [page 39](#).
2. If desired, enable and configure the tachometer input, as described on [page 40](#).
3. If desired, enable and configure the counter/timer inputs, as described on [page 42](#).
4. If desired, enable and configure the analog output readback channel, as described on [page 44](#).
5. Set up the input buffer, as described on [page 44](#).
6. Set up the clock that paces the input operation, as described on [page 45](#).
7. Set up the trigger that starts the input operation, as described on [page 48](#).
8. Arm the analog input subsystem, as described on [page 53](#).
9. If you are using the software trigger source (IMMediate), initiate the input operation, as described on [page 53](#).
10. (Optional) Determine the status of the input operation, as described on [page 54](#).
11. Retrieve the data from the input buffer on the DT8837, as described on [page 54](#).
12. If desired, stop the operation, as described on [page 57](#).

Enabling and Configuring Analog Input Channels

The DT8837 instrument module supports four, 24-bit, differential analog input channels (numbered 1 to 4).

Enable the analog input channels you want to sample by using the **AD:ENABle** command. To determine whether specified analog input channels are enabled or disabled, use the **AD:ENABle?** query.

The following sections describe how to configure the analog input channels.

Configuring IEPE Functions

Applications that require accelerometer, vibration, noise, or sonar measurements often use IEPE sensors. IEPE conditioning is built-in to the analog input circuitry of the DT8837 instrument module. The instrument module supports the following software-programmable IEPE functions for each of the four analog inputs:

- Excitation current source – You can enable or disable the use of a 4 mA, internal excitation current source for each analog input channel with the **AD:BIAS:CURRent:ENABle** command. To determine whether the current source is enabled or disabled, use the **AD:BIAS:CURRent:ENABle?** query.
- Coupling type – You can specify whether AC coupling or DC coupling is used for each analog input channel with the **AD:COUPling[:CONFigure]** command. To return the current coupling type, use the **AD:COUPling[:CONFigure]?** query.

Note: If you enable the use of the internal 4 mA excitation current source, it is recommended that you choose AC coupling. Refer to the *DT8837 User's Manual* for more information on wiring IEPE inputs.

Input Ranges and Gains

The DT8837 provides an input range of ± 10 V and software-selectable gains of 1 and 10. This provides effective input ranges of ± 10 V (when the gain is 1) and ± 1 V (when the gain is 10).

You can set the gain for each analog input channel programmatically using the **AD:GAIN[:CONFigure]** command. To return the current gain setting for each analog input channel, use the **AD:GAIN[:CONFigure]?** query.

Enabling and Configuring the Tachometer Input Channel

The DT8837 accepts one tachometer input signal with a range of ± 30 V. You can measure the frequency or period of the tachometer input signal by reading the value of the tachometer input channel in the analog input stream. The frequency or period measurement allows you to calculate the rotation speed of tachometer input signals. An internal 12 MHz counter is used for the measurement, yielding a resolution of 83 ns (1/12 MHz).

Note: You can also measure the phase of the tachometer input signal in relation to the analog input samples using one of the counter/timer channels, described on [page 42](#).

To read the number of counts between two edges of the tachometer input signal, enable the tachometer input channel in the analog input data stream using the **TACHometer:ENABle** command. To determine whether the tachometer is enabled or disabled, use the **TACHometer:ENABle?** query.

To measure the frequency or period of a tachometer signal, configure the following parameters for the tachometer:

- Period type – Use the **TACHometer:PERiod[:CONFigure]** command to specify the type of period (rising-to-rising edge or falling-to-falling edge) to measure on the tachometer input signal.

You can determine which period type is configured using the **TACHometer:PERiod[:CONFigure]?** query.

- Self Clear flag – Use the **TACHometer:SCLR[:CONFigure]** command to specify whether the value read between measurements is cleared or retained. To clear the value to zero between measurements, set the Self Clear flag to 1. To retain the previous measurement value until the next measurement is complete, set the Self Clear flag to 0.

You can determine the configuration of the Self Clear flag using the **TACHometer:SCLR[:CONFigure]?** query.

- Stale Value flag – Use the **TACHometer:SFLAG[:CONFigure]** command to specify whether to use the most significant bit (MSB) of the returned 32-bit value to indicate new or old data. To use the MSB to indicate whether the measurement is new or old, set the Stale Value flag to active (1). When this flag is active, the MSB of the value is set to 0 to indicate new data or 1 to indicate old data. Reading the value before the measurement is complete returns an MSB of 1 to indicate old data.

If you are not interested in whether the data is new or old, set the Stale Value flag to 0. In this case, the MSB is always set to 0.

You can determine the configuration of the Stale Value flag using the **TACHometer:SFLAG[:CONFigure]?** query.

When the tachometer input is enabled, the internal 12 MHz counter starts incrementing when it detects the first specified edge (rising or falling) of the tachometer input. When the next specified edge is detected, the count is stored in a holding register and the counter resets and starts counting again. The tachometer holding register is read at the input sample rate and included as part of the input scan data. Tachometer counting is independent of the input sample rate. Therefore, depending on the sample rate and the tachometer input rate, you could either always read a new count value or, depending on the value of the Self Clear flag, read several old count values or 0.

The hardware automatically aligns the value of the tachometer input with the analog input measurements, so that all measurements are correlated in time based on the specified input clock and input trigger.

When you read the value of the tachometer input as part of the analog input data stream, you might see results similar to the following (note that this assumes that the previous measurement value is returned between new measurement values):

Table 3: An Example of Reading the Tachometer Input as Part of the Analog Input Data Stream

Time	A/D Value	Tachometer Input Value	Status of Operation
10	5002	0	Operation started, but is not complete
20	5004	0	Operation not complete
30	5003	0	Operation not complete
40	5002	12373	Operation complete
50	5000	12373	Next operation started, but is not complete
60	5002	12373	Operation not complete
70	5004	12373	Operation not complete
80	5003	14503	Operation complete
90	5002	14503	Next operation started, but is not complete

Using the count that is returned from the tachometer input, you can determine the following:

- Frequency of a signal pulse (the number of periods per second). You can calculate the frequency as follows:
 - $\text{Frequency} = 12 \text{ MHz} / (\text{Number of counts} - 1)$
where 12 MHz is the internal counter/timer clock frequency

For example, if the count is 21, the measured frequency is 600 kHz (12 MHz/20).

- Period of a signal pulse. You can calculate the period as follows:
 - $\text{Period} = 1 / \text{Frequency}$
 - $\text{Period} = (\text{Number of counts} - 1) / 12 \text{ MHz}$
where 12 MHz is the internal counter/timer clock frequency

Enabling and Configuring Counter/Timer Channels

The DT8837 provides two counter/timer channels for measuring the frequency, period, or phase between any of the following signals:

- Completion of the A/D sample to the rising or falling edge of the tachometer input signal
- Completion of the A/D sample to the rising or falling edge of the gate input signal
- Rising or falling edge of the gate input signal to the rising or falling edge of the tachometer input signal
- Rising or falling edge of the gate input signal to the rising or falling edge of the gate input signal, which you can use to determine the pulse width or period of the gate signal
- Rising or falling edge of the tachometer input signal to the rising edge or falling edge of the tachometer input signal, which you can use to determine the pulse width or period of the tachometer input signal

- Rising or falling edge of the tachometer input signal to the completion of the A/D sample
- Rising or falling edge of the gate input signal to the completion of the A/D sample

An internal 48 MHz clock (with 21 ns resolution) is used to calculate the measurement, which allows you to precisely correlate these measurements with the analog input data.

To read the value of the counter in the analog input data stream, enable the counter/timer using the **COUNTer[<n>]:ENABLE** command. To determine whether the counter/timer is enabled or disabled, use the **COUNTer[<n>]:ENABLE?** query.

Configure the following parameters for the counter/timer measurement:

- Start signal edge – Use the **COUNTer[<n>]:EDGE:START[:CONFigure]** command to specify the signal that starts the measurement. You can choose one of the following signals to start the measurement: ADC conversion complete, rising edge of the tachometer input signal, falling edge of the tachometer input signal, rising edge of the gate input signal, or falling edge of the gate input signal.

You can determine which signal is configured to start the measurement using the **COUNTer[<n>]:EDGE:START[:CONFigure]?** query.

- Stop signal edge – Use the **COUNTer[<n>]:EDGE:STOP[:CONFigure]** command to specify the signal that stops the measurement. You can choose one of the following signals to stop the measurement: ADC conversion complete, rising edge of the tachometer input signal, falling edge of the tachometer input signal, rising edge of the gate input signal, or falling edge of the gate input signal.

Note: Note that if you choose to start the measurement using the A/D conversion complete signal, choose a different signal to stop the measurement. Likewise, if you choose to stop the measurement using the A/D conversion complete signal, choose a different signal to start the measurement.

You can determine which signal is configured to stop the measurement using the **COUNTer[<n>]:EDGE:STOP[:CONFigure]?** query.

- Self Clear flag – Use the **COUNTer[<n>]:SCLR[:CONFigure]** command to specify whether the value read between measurements is cleared or retained. To clear the value of the counter/timer to zero between measurements, set this flag to 1. To retain the previous measurement value until the next measurement is complete, set this value to 0.

You can determine the configuration of the Self Clear flag using the **COUNTer[<n>]:SCLR[:CONFigure]?** query.

When the counter is enabled, the internal 48 MHz counter starts incrementing when it detects the first specified start edge. When the specified stop edge is detected, the count is stored in a holding register and the counter resets and waits for next start edge. The counter holding register is read at the input sample rate and included as part of the input scan data. Counter/timer counting is independent of the input sample rate. Therefore, depending on the sample rate and the counter input rate, you could either always read a new count value or, depending on the value of the Self Clear flag, read several old count values or 0.

Since these counters have selectable start and stop edges, if the same edge is selected as the start and stop edge, the counter will count every other input signal period and not every period like the tachometer input counter.

Using the count that is returned from the counter/timer, you can determine the following:

- Frequency of a signal pulse (the number of periods per second). You can calculate the frequency as follows:
 - $\text{Frequency} = 48 \text{ MHz} / (\text{Number of counts} - 1)$
where 48 MHz is the internal counter/timer clock frequency

For example, if the count is 81, the measured frequency is 600 kHz (48 MHz/80).

- Period of a signal pulse. You can calculate the period as follows:
 - $\text{Period} = 1 / \text{Frequency}$
 - $\text{Period} = (\text{Number of counts} - 1) / 48 \text{ MHz}$
where 48 MHz is the internal counter/timer clock frequency

Enabling the Analog Output Readback Channel

You can read back the value of the analog output channel in the analog input data stream by enabling the analog output readback channel using the **DA:READ:ENABle** command. To determine whether the analog output readback feature is enabled or disabled, use the **DA:READ:ENABle?** query.

When the analog input operation is started, the value of the analog output channel is returned in the analog input data stream. The hardware automatically aligns the value of the analog output channel with the analog input measurements, so that all measurements are correlated in time.

Refer to [page 58](#) for information on configuring and starting analog output operations.

Configuring the Input Buffer

DT8837 instrument modules use an 8 MB input buffer for storing data from each of up to 8 enabled input channels (analog input channels 1, 2, 3, 4, tachometer input channel, counter/timer 1, counter/timer 2, and the analog output readback channel). One sample from each of the enabled input channels is called a scan.

You can use the **AD:BUFFer:MODE** command to specify one of the following wrap modes for the input buffer:

- Continuous wrap mode – Specify WRAP mode if you want the input operation to continue indefinitely until you stop it using the **AD:ABORt** command. In this case, when the end of the input buffer is reached, the operation wraps to the beginning of the input buffer overwriting the oldest scan data with the latest scan data.
- No wrap mode – Specify NOWRAP mode if you want the input operation to stop automatically when the input buffer is filled.

To verify the currently configured wrap mode for the input buffer, use the **AD:BUFFer:MODE?** query.

You can use the **AD:BUFFer:SIZE[:SCAns]?** query to determine how many scans can be stored in the input buffer based on the number of enabled input channels and the input sampling rate.

Note: The sampling rate determines DMA block size that is used for the scan data, where each block has a fixed number of header bytes that is not used to store scans. The DMA block size is smaller for the minimum sampling frequency, resulting in more space for the headers, and greater for the maximum sampling frequency.

Since the maximum input buffer size is approximately 8 MB and each sample is 4 bytes, you can store a maximum of approximately 2 Msamples in the input buffer. If you sample each input channel at the maximum input frequency (52.734 kHz), the input buffer will fill in approximately 5 seconds (52 ksamples/s per channel).

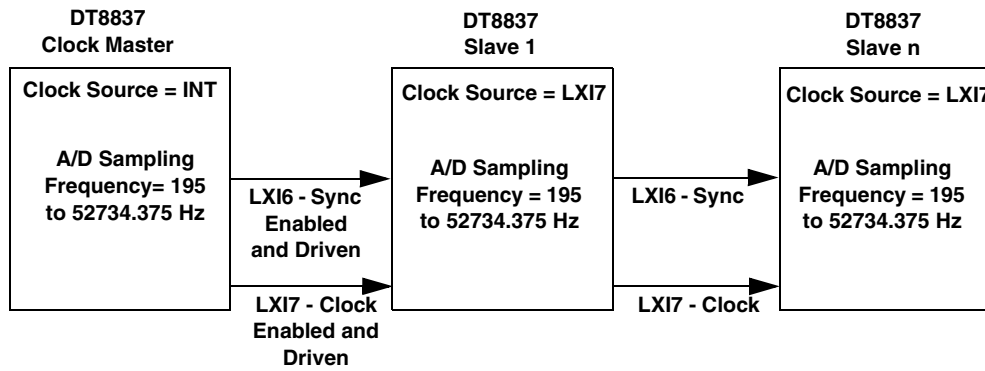
The maximum size of data transfer over Ethernet using the **AD:FETCh?** command is 32 kB. Since each sample is 4 bytes, the maximum number of samples per **AD:FETCh?** is 8 ksamples. Therefore, if you are using continuous wrap mode, ensure that you call **AD:FETCh?** in a tight loop to ensure that you retrieve all the samples in the input buffer before the buffer is overwritten.

Configuring the Input Clock Source

The same 48 MHz reference clock is used for both the analog input and analog output subsystems on the DT8837. Using the **[AD:]CLOCK:SOURce** command, you can specify one of the following clock sources as the source of the reference clock for the ADC and DAC master clock generators:

- **Internal clock** – Selects the internal reference clock on the DT8837 instrument module.
- **LXI7** – When multiple DT8837 instrument modules are connected together using the Trigger Bus, as shown in [Figure 11](#), LXI7 selects the dedicated clock signal on the Trigger Bus that is used to synchronize the clock on the connected DT8837 instrument modules.

In this configuration, one DT8837 instrument module is the clock master and the remaining DT8837 instrument modules are slaves.



Note: LXI bus terminators must be installed for the first and last instrument modules in the Trigger Bus chain.

Figure 11: Synchronizing the Clock on Multiple DT8837 Instrument Modules for Analog Input Operations

If you are using only one DT8837 instrument module or if you are using multiple DT8837 instrument modules but do not want to synchronize their clocks, set up the clocks as follows:

1. Use the **[AD:]CLOCK:SOURce** command to set the clock source to *INTernal*. This command configures the device to use the internal clock reference.
2. Use the **AD:CLOCK:FREQuency[:CONFigure]** command to specify the sampling frequency of the analog input subsystem. The sampling frequency ranges from 195 Hz to 52734.375Hz.

If you want to synchronize the clocks of multiple DT8837 instrument modules that are connected together using the Trigger Bus, set up the DT8837 instrument modules as follows:

Note: Ensure that you perform this procedure before you set up the triggers (as described on [page 48](#)), and before you arm the subsystems (as described on [page 53](#)).

1. On the DT8837 slave, use the **[AD:]CLOCK:SOURce** command to set the reference clock source to *LXI7*. This command configures the slave device to receive the clock signal (LXI7) from the Trigger Bus; in to receive the synchronization pulse (LXI6) from the Trigger Bus.
2. On the DT8837 clock master, use the **[AD:]CLOCK:SOURce** command to set the reference clock source to *INTernal*. This command configures the clock master to use the internal clock reference; in addition, this command automatically configures the clock master to use the internal ADC synchronization pulse.
3. On the DT8837 clock master, use the **WTB:LXI7[:OUTput]:ENABLE** command to drive out the clock signal (LXI7) on the Trigger Bus. This command also drives out the ADC synchronization pulse (LXI6) on the Trigger Bus.

4. On each DT8837 slave, use the **AD:CLOCK:FREQuency[:CONFigure]** command to specify the sampling frequency of the analog input subsystem. The sampling frequency ranges from 195 Hz to 52734.375 Hz.

Note: For proper operation, it is important that you configure the clock frequency of the slave instrument modules before configuring the clock frequency of the clock master.

It is also recommended that you choose the same sampling frequency for all instrument modules to avoid problems. Note, however, that the analog input and analog output frequencies can be different from each other.

5. On the DT8837 clock master, use the **AD:CLOCK:FREQuency[:CONFigure]** command to specify the sampling frequency of the analog input subsystem. The sampling frequency ranges from 195 Hz to 52734.375 Hz.

Once the clocks are configured, the clock and sync signals are sent over the Trigger Bus to all slaves. The analog input operation on each slave starts on the first sample clock pulse after the trigger is received. Refer to [page 48](#) for information on triggering DT8837 instrument modules.

Note: If you change the sampling frequency of one or more slave instrument modules, you must reconfigure the clock frequency of the clock master, even if its parameters have not changed.

To return the currently configured reference clock source, use the **[AD:]CLOCK:SOURce?** query. To return the currently configured ADC synchronization source, use the **AD:SYNc:SOURce?** query.

The actual sampling frequency that the device can achieve may be slightly different than the frequency you specified due to the accuracy of the clock. You can determine the actual sampling frequency that is configured using the **AD:CLOCK:FREQuency[:CONFigure]?** query.

Note: DT8837 instrument modules use 24-bit Delta-Sigma ADCs that provide anti-aliasing filters based on the sampling frequency.

According to sampling theory (Nyquist Theorem), specify a frequency that is at least twice as fast as the input's highest frequency component. For example, to accurately sample a 20 kHz signal, specify a sampling frequency of at least 40 kHz to avoid aliasing.

DT8837 instrument modules support a wide pass band of 0.5 Hz to 25.8 kHz ($0.49 \times$ sampling frequency) to eliminate aliasing, allowing you to measure low frequency signals accurately at the Nyquist sampling rate.

The value that you specify for the sampling frequency is multiplied by 512 internally to set the internal reference clock on the instrument module. For example, if you specify a sampling frequency of 50000, the instrument module sets the internal reference clock to 25.6 MHz. The maximum timebase is 27 MHz.

Once the sample clock is started, the instrument module requires 39 sample clock pulses before the first A/D conversion is completed (39/sample rate) due to the group delay of the converters. The hardware adjusts for the group delay to provide only valid data. The first sample that is returned corresponds to the value of the analog input signal at the time the trigger occurred.

Note: Data from the tachometer, counter/timer, and analog output readback channel (which does not have the 39 sample group delay) is fed through a hardware queue to realign it with the adjusted analog input data stream.

Configuring the Input Trigger

A trigger is an event that occurs based on a specified set of conditions. Once the analog input subsystem is armed with the **AD:ARM** command (and, if the trigger source is **IMMediate**, started with the **AD:INITiate** command), acquisition starts with the first sample after the trigger is received. Refer to [page 53](#) for more information on the **AD:ARM** and **AD:INITiate** commands.

Note: The Input Trigger LED on the front panel of the DT8837 instrument module turns solid amber when the analog input subsystem is armed and solid green when the analog input subsystem is triggered. When the analog input subsystem is idle, the Input Trigger LED is off.

This section describes each of the supported trigger sources in detail. To return the currently configured trigger source, use the **AD:TRIGger[:SOURce]** query.

Software Trigger

A software trigger event occurs when you start the analog input operation with the **AD:INITiate** command (the computer issues a write to the instrument module to begin conversions).

Use the **AD:TRIGger[:SOURce]** command with the *IMMediate* parameter to select the software trigger.

External, Digital (TTL) Trigger

An external digital (TTL) trigger event occurs when the instrument module detects a rising-edge transition on the signal connected to the Trigger In input on the instrument module.

Use the `AD:TRIGger[:SOURce]` command with the *EXternal* parameter to specify the external, digital (TTL) trigger.

Analog Threshold Trigger

An analog threshold trigger event occurs when the signal attached to analog input channel 1 rises above a user-specified threshold value from 0.2 V to 9.8 V with 0.1 V of hysteresis.

Use the `AD:TRIGger[:SOURce]` command with the *AIN1* parameter to specify the analog threshold trigger. Then, use the `AD:TRIGger:LEVel[:AIN1]` command to set the threshold level.

LAN Trigger Packet

When multiple DT8837 instrument modules are connected together over the local area network (LAN), as shown in [Figure 12](#), you can synchronize the start of acquisition by transmitting one of eight LAN trigger packets (LAN0 to LAN7) over the network.

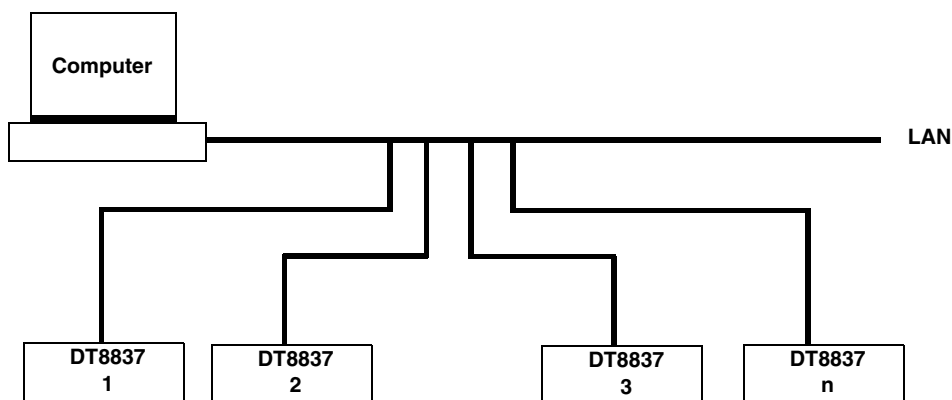


Figure 12: Synchronizing the Start of Acquisition When Triggering Instrument Modules Over the LAN

The timing of the trigger packets from device to device is dependent on the network traffic on the LAN.

The DT8837 supports multicast transmission of LAN trigger packets using UDP (User Datagram Protocol) port 5044 and UDP multicast address 224.0.23.159. The DT8837 does not support unicast LAN events transmitted via TCP connections.

[Figure 13](#) shows the format of a LAN trigger packet:

HW Detect (3 bytes)	Domain (1 byte)	Event ID (16 bytes)	Sequence (UInt32)	Time stamp (10 bytes)	Epoch (UInt16)	Flags (UInt16)	Data Fields	0 (2 Bytes)
------------------------	--------------------	------------------------	----------------------	--------------------------	-------------------	-------------------	-------------	----------------

Figure 13: Format of a LAN Trigger Packet

The fields of the LAN trigger packet are described as follows:

- **HW Detect** – Identifies valid packets and is reserved for future hardware detection of LAN events. This field should be set to *LXI*. Note that the third byte, ASCII "T", is also used as a version identifier; future revisions to the LXI standard may change this value.
- **Domain** – This field represents a group of devices that are managed by a single directory and use shared resources. Values for the LXI domain range from 0 to 255. The default value is 0. This field is treated as an unsigned byte.

To transmit LAN trigger packets to or from a DT8837 instrument module, ensure that you set the domain field of each DT8837 instrument module to the same value using the **EVENT:DOMain** command.

- **Event ID** – This field contains the first 16 bytes of the event name (a string). By default, the LAN event names are defined as LAN0 to LAN7.

To transmit LAN trigger packets to or from a DT8837 instrument module, ensure that you set the event ID of each DT8837 instrument module to the same value using the **EVENT:LAN<n>:NAME** command.

- **Sequence** – Each instrument module maintains its own sequence; the sequence number is incremented every time a unique packet is transmitted. (Note that if packets are retransmitted to enhance reliability, re-transmitted packets contain the same sequence number as the original.)
- **Time stamp** – For DT8837 instrument modules, the time stamp of received LAN event packets is ignored, while generated LAN event packets have a time stamp of 0.
- **Epoch** – For DT8837 instrument modules, this value is 0.
- **Flags** – Contains data about the packet. This value is generated automatically by DT8837 instrument modules.
- **Data Fields** – Arbitrary number of bytes, up to the capacity of the data packet. This field is generated automatically by DT8837 instrument modules.

To set up a LAN trigger, do the following:

1. Set the trigger source to a LAN trigger packet (LAN0 to LAN7) using the **AD:TRIGger[:SOURce]** command.
2. Set the value of the LXI domain octet using the **EVENT:DOMain** command.
3. Enable the LAN event for transmission using the **EVENT:LAN<n>:TRANsmit[:ENABle]** command.

4. Configure the LAN event ID field of the LXI packet using the **EVENT:LAN<n>:NAME** command.
5. Arm the analog input subsystem, as described on [page 53](#).

Acquisition starts with the first sample after the specified LAN trigger packet is received.

Trigger Bus

When multiple DT8837 instrument modules are connected together using the Trigger Bus, as shown in [Figure 14](#), you can synchronize the start of acquisition using one of six LXI trigger signals: LXI0 to LXI5.

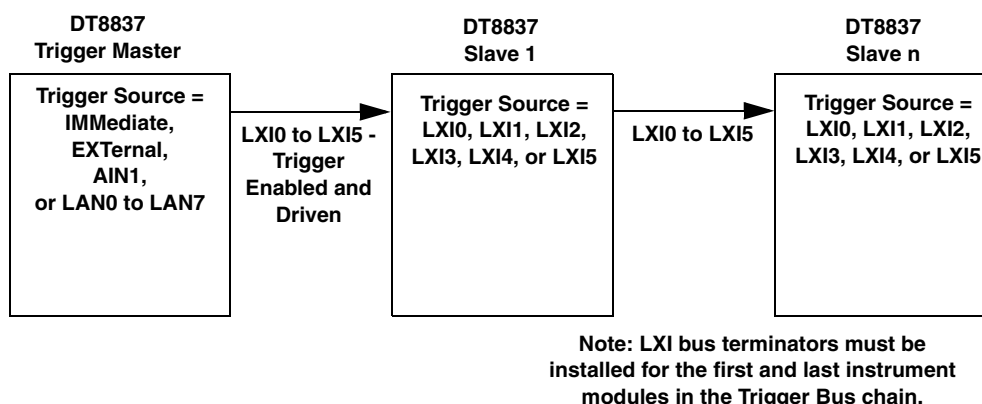
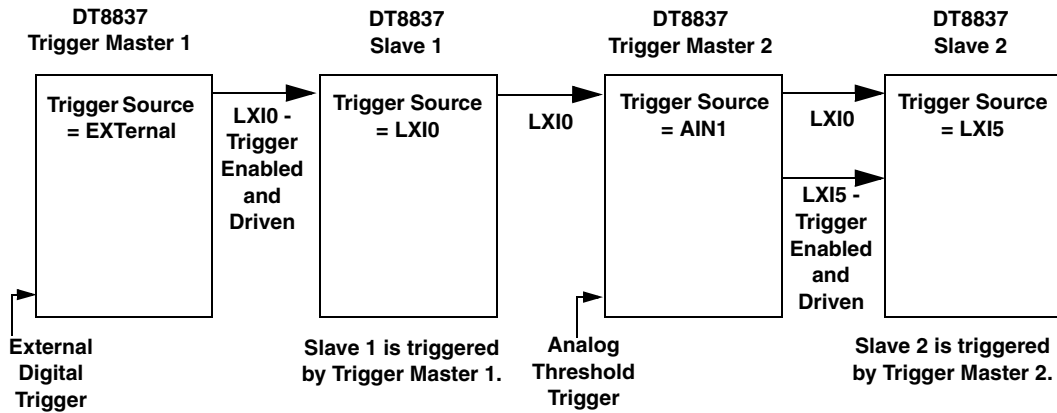


Figure 14: Synchronizing the Start of Acquisition When Connecting Multiple Instrument Modules to the Trigger Bus

Note: When using LXI triggers, the timing from device to device is guaranteed to be within one sample pulse.

Unlike when synchronizing the clock of multiple instrument modules, which requires one DT8837 instrument module dedicated as the clock master, you can have more than one trigger master, if desired, to trigger multiple instrument modules. Additionally, the trigger master need not be the clock master; for example, a device other than the DT8837 can generate the trigger for the DT8837 instrument modules, if desired. [Figure 15](#) shows an example of using two DT8837 trigger masters on the Trigger Bus.



Note: LXI bus terminators must be installed for the first and last instrument modules in the Trigger Bus chain.

Figure 15: An Example of Using Two Trigger Masters on the Trigger Bus

Note: Ensure that you set up the clocks (as described on [page 45](#)) before you set up the triggers. Arm the subsystems (as described on [page 53](#)) after both the clocks and triggers have been set up.

Set up the triggers as follows:

1. On the DT8837 trigger master, use the **AD:TRIGger[:SOURce]** command to set the trigger source to software trigger (IMMediate), external digital trigger (EXternal), analog threshold trigger (AIN1), or LAN trigger packet (LAN0 to LAN7).
2. If the trigger source AIN1, set the trigger level using the **AD:TRIGger:LEVel[:AIN1]** command.
3. If the trigger source is LAN0 to LAN7, configure the LAN event trigger, as follows:
 - a. Set the value of the LXI domain octet using the **EVENT:DOMain** command.
 - b. Enable the LAN event for transmission using the **EVENT:LAN<n>:TRANsmit[:ENABLE]** command.
 - c. Configure the LAN event ID field of the LXI packet using the **EVENT:LAN<n>:NAME** command.
4. On each DT8837 slave, use the **AD:TRIGger[:SOURce]** command to select the trigger signal (LXI0 to LXI5) to receive from the Trigger Bus.
5. On the DT8837 trigger master, use the **WTB:LXI<n>[:OUTput]:ENABLE** command to drive out the LXI trigger signal (LXI0 to LXI5) on the Trigger Bus.

Note: If you change the input trigger source on the master, you must call the **WTB:LXI<n>[:OUTput]:ENABLE** command again to ensure that the LXI trigger signal is driven out on the Trigger Bus.

While a trigger master can drive out more than one trigger signal on the Trigger Bus, the slave can accept only one trigger signal to start the analog input operation.

6. Arm the analog input subsystem on each slave, as described on [page 53](#).
7. Arm the analog input subsystem on the trigger master, as described on [page 53](#).
8. If the trigger source of the DT8837 trigger master is IMMediate (software trigger), initiate the operation, as described on [page 53](#).

Once the master is triggered, the specified LXI trigger signal is driven out on the Trigger Bus to all slaves that were configured to accept it. Acquisition starts on the first sample after the trigger is received.

Arming Input Operations

Once you have configured all the parameters of the analog input subsystem, call the **AD:ARM** command to arm the subsystem. The arming process writes all the input configuration settings to the DT8837 hardware.

Note: If you want to perform analog input and analog output operations simultaneously, call **ARM** rather than **AD:ARM**, or explicitly call **DA:ARM** before calling **AD:ARM**. Once the analog input subsystem is armed, the analog output subsystem cannot be armed.

Then, use the **AD:STATus?** command, described below, or read bit 10 of the Operation Status register using the **STATus:OPERation:CONDition?** command to verify that the arming process is complete.

Starting Input Operations

If you are using the software (IMMediate) trigger source and have armed the analog input subsystem, you must start the analog input operation by issuing the **AD:INITiate** command. For all other trigger sources, **AD:INITiate** is ignored.

When the software trigger is detected, the analog input subsystem simultaneously samples all the enabled input channels, and converts the analog inputs to voltage.

How the operation stops depends on the buffer wrap mode, described on [page 44](#).

Determining the Status of the Analog Input Subsystem

You can use the **AD:STaTus?** command to determine the status of the analog input subsystem. The returned value, in the range of 0 to 255, represents the status bits for the analog input subsystem.

The status bits are defined as follows:

- Bit 0 – The value of this bit is 1 if the analog input subsystem is active or 0 if the analog input subsystem is inactive.
- Bit 1 – The value of this bit is 1 if the analog input subsystem is armed or 0 if the analog input subsystem is not armed.
- Bit 2 – The value of this bit is 1 if the analog input subsystem is triggered or 0 if the analog input subsystem is not triggered.
- Bit 3 – The value of this bit is 1 if AD SYNC was detected or 0 if AD SYNC was not detected.
- Bit 4 – The value of this bit is 1 if the AD FIFO overflowed or 0 if the AD FIFO did not overflow.

Retrieving Scan Data from the Input Buffer

To retrieve scan data from the input buffer, use the **AD:FETCh?** command.

Note: At any instant, up to 12 SCPI clients can retrieve a client-specific number of scans concurrently from the input buffer of the instrument module using the **AD:FETCh?** command.

Of the 12 SCPI clients, 4 can be VXI-11 clients, which use the VISA::INSTR resource to access the instrument module.

Up to 8 additional clients can access the DT8837 instrument modules concurrently using the web interface provided with the device. Refer to the documentation for your instrument module for more information.

At this time, the SCPI and web interfaces cannot be "locked;" therefore, one client can change the configuration of the instrument module that another client is accessing. However, you can optionally lock the VXI-11 interface using the VISA APIs `viLock/viUnlock`; this prevents other VXI-11 clients (including VXI-11 discovery) from accessing the instrument module.

The **AD:FETCh?** command takes two parameters:

- *index* – a required parameter that specifies where in the input buffer to begin reading the scan data
- *number of scans* – an optional parameter that specifies the number of scans to retrieve from the input buffer

By using *index* and *number of scans*, any client can request data at arbitrary intervals. To request all the data from the input buffer, set the *index* to 0 and omit the *number of scans* parameter.

Note: If the *number of scans* parameter is omitted, a fixed number of scan records is returned.

Since the maximum input buffer size is approximately 8 MB and each sample is 4 bytes, you can store a maximum of approximately 2 Msamples in the input buffer. If you sample each input channel at the maximum input frequency (52.734 kHz), the input buffer will fill in approximately 5 seconds (52 ksamples/s per channel).

The maximum size of data transfer over Ethernet using the **AD:FETCh?** command is 32 kB. Since each sample is 4 bytes, the maximum number of samples per **AD:FETCh?** is 8 ksamples. Therefore, if you are using continuous wrap mode, ensure that you call **AD:FETCh?** in a tight loop to ensure that you retrieve all the samples in the input buffer before the buffer is overwritten.

You can use the **AD:BUFFer:SIZE[:SCAns]?** query to determine how many scans can be stored in the input buffer based on the number of enabled input channels and the input sampling rate. Note that the sampling rate determines DMA block size that is used for the scan data, where each block has a fixed number of header bytes that is not used to store scans. The DMA block size is smaller for the minimum sampling frequency, resulting in more space for the headers, and greater for the maximum sampling frequency.

If you want to read only a specified number of scans (instead of all the scans available), specify the number of scans that you want to retrieve in the *number of scans* parameter.

If more than the requested number scans has been acquired, the DT8837 instrument module returns only the scans that you requested between the starting index (*index*) and the ending index (*index + number of scans*). Conversely, if fewer than the requested number of scans has been acquired, the instrument module returns the subset of the requested scans that are available between the starting index (*index*) and the ending index (*index + number of scans*).

If all the input channels are enabled, samples returned in the scan record are ordered as follows:

- Chan 1 – analog channel 1
- Chan 2 – analog channel 2
- Chan 3 – analog channel 3
- Chan 4 – analog channel 4
- Chan 5 – tachometer
- Chan 6 – counter/timer 1
- Chan 7 – counter/timer 2
- Chan 8 – analog output channel readback

Reading data from the input buffer with **AD:FETCh?** does not destroy the data; therefore, you can read the same data multiple times, if desired, providing that the input buffer has not been overwritten.

If the client wants to read a number of scans that chronologically follow a previous response, get the *EndingIndex* returned by the **AD:STATus:SCAn?** command, described on [page 117](#), add one to this value, and set the *Index* parameter of the next **AD:FETCh?** command to this value. For example, if *EndingIndex* returned by **AD:STATus:SCAn?** is 2050 and you want to continue reading data from the location where you left off, set *Index* in the next **AD:FETCh?** command to 2051.

Stopping Input Operations

To stop an input operation that is in progress, issue the **AD:ABORt** command.

Note that if the buffer mode is NOWRAp, the operation stops automatically when the input buffer has been filled. Refer to [page 44](#) for more information on buffer modes.

Performing Analog Output Operations

DT8837 instrument modules feature one 24-bit analog output channel. The analog output operation is started when the designated trigger source is detected and is paced by the specified clock source at the specified output clock frequency.

To set up and update the analog output channel of a DT8837 instrument module using SCPI commands, perform the following steps:

1. Enable the analog output channel, as described on [page 58](#).
2. Set up the output buffer, as described on [page 58](#).
3. Specify the clock source and the frequency at which to update the analog output channel, as described on [page 59](#).
4. Specify the trigger source that starts the analog output operation, as described on [page 61](#).
5. Arm the analog output subsystem, as described on [page 64](#).
6. If you are using the software trigger source (IMMediate), initiate the analog output operation, as described on [page 64](#).
7. (Optional) Determine the status of the analog output subsystem, as described on [page 64](#).
8. If desired, stop the output operation, as described on [page 65](#).

Enabling the Analog Output Channel

DT8837 instrument modules support one, 24-bit analog output channel (channel 1).

Enable the analog output channel by using the **DA:ENABle** command. You can determine whether the analog output channel is enabled or disabled by using the **DA:ENABle?** query.

Configuring the Output Buffer

DT8837 LXI instrument modules use an 128K sample output buffer (also known as a hardware FIFO) for storing analog output data to write to the analog output channel.

Specify the number of samples to output to the analog output channel using the **DA:BUFFer:SIZE[:SCAns]** command. This value can range between 1 and 131072.

Specify the actual values that you want to write to the analog output channel by filling the output buffer using the **DA:BUFFer[:DATA]** command. Using this command, you specify a data block that contains a sequence of samples to be written to the analog output buffer. Within the block, you specify the number of samples the block contains and the 4 byte analog output values to write to the analog output buffer.

Note: Due to internal buffer limits, the number of samples in the block must not exceed 1024 for SCPI connections to port 5025 ("raw" SCPI) or 256 for SCPI connections over VXI-11.

To verify the values in the output buffer, use the **DA:BUFFer[:DATA]?** query to return the samples in the buffer.

Use the **DA:BUFFer:MODE** command to specify one of the following wrap modes for the output buffer:

- Continuous wrap mode – Specify WRAP mode if you want the output operation to continue indefinitely until you stop it with the **DA:ABORt** command. In this case, when the end of the output buffer is reached, the operation wraps to the beginning of the output buffer and continues updating the analog output channel with the values stored in the buffer.
- No wrap mode – Specify NOWRAP mode if you want the output operation to stop automatically when the number of samples specified by **DA:BUFFer:SIZE** has been output.

To verify the currently configured wrap mode for the output buffer, use the **DA:BUFFer:MODE?** query.

Configuring the Output Clock Source

The same 48 MHz reference clock source is used for both the analog input and analog output subsystems.

If you have not already done so, you can use the **[DA:]CLOCK:SOURce** command to specify one of the following clock sources as the source of the reference clock for the ADC and DAC master clock generators:

- **Internal clock** – Selects the internal reference clock on the DT8837 instrument module.
- **LXI7** – When multiple DT8837 instrument modules are connected together using the Trigger Bus, as shown in [Figure 16](#), LXI7 selects the dedicated clock signal (LXI7) on the Trigger Bus to synchronize the clock on the connected DT8837 instrument modules.

In this configuration, one DT8837 instrument module is the clock master and the remaining DT8837 instrument modules are slaves.

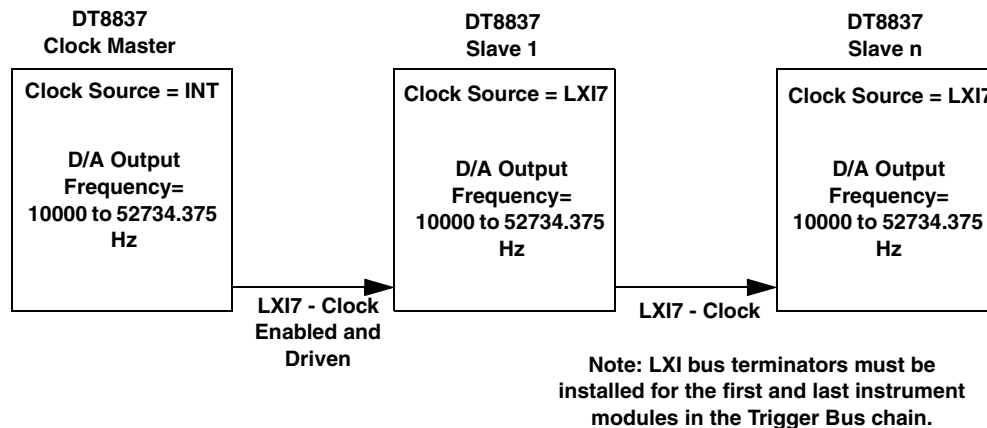


Figure 16: Synchronizing the Clock on Multiple DT8837 Instrument Modules For Analog Output Operations

If you are using only one DT8837 instrument module or if you do not want to synchronize the clocks on multiple DT8837 instrument modules, set up the clocks as follows:

1. Use the **[DA:]CLOCK:SOURce** command to set the clock source to *INTernal*. This command configures the device to use the internal clock reference.
2. Use the **DA:CLOCK:FREQuency[:CONFigure]** command to specify the frequency at which to pace the output operation. The output frequency ranges from 10000 Hz to 52734.375 Hz.

If you want to synchronize the clocks of multiple DT8837 instrument modules that are connected together using the Trigger Bus, set up the clocks on the DT8837 instrument modules, as follows:

1. On the DT8837 slave, use the **[DA:]CLOCK:SOURce** command to set the clock source to *LXI7*. This command configures the slave device to receive the clock signal (*LXI7*) from the Trigger Bus.
2. On the DT8837 clock master, use the **[DA:]CLOCK:SOURce** command to set the clock source to *INTernal*. This command configures the clock master device to use the internal clock reference.
3. On the DT8837 clock master, use the **WTB:LXI7[:OUTput]:ENABLE** command to drive out the clock signal (*LXI7*) on the Trigger Bus.
4. On each DT8837 slave, use the **DA:CLOCK:FREQuency[:CONFigure]** command to specify the frequency at which to pace the output operation. The output frequency ranges from 10000 Hz to 52734.375 Hz.

Note: For proper operation, it is important that you configure the clock frequency of the slave instrument modules before configuring the clock frequency of the clock master.

It is also recommended that you choose the same sampling frequency for all instrument modules to avoid problems. Note, however, that the analog input and analog output frequencies can be different from each other.

5. On the DT8837 clock master, use the **DA:CLOCK:FREQuency[:CONFigure]** command to specify the frequency at which to pace the output operation. The output frequency ranges from 10000 Hz to 52734.375 Hz.

Once the clocks are configured, a clock signal is sent over the Trigger Bus to all slaves. The analog output operation on each slave starts on the first sample clock pulse after the trigger is received. Refer to [page 61](#) for information on triggering DT8837 instrument modules.

Note: If you change the update frequency of one or more slave instrument modules, you must reconfigure the update frequency of the clock master, even if its parameters have not changed.

To return the currently configured clock source, use the **[DA:]CLOCK:SOURce?** query.

To verify the output clock frequency, use the **DA:CLOCK:FREquency[:CONFigure]?** query.

Note: Due to the group delay of the Delta-Sigma D/A converter, the DT8837 requires 29 sample clocks once the analog output subsystem has been triggered before the value of the analog output channel reflects the first value in the output buffer.

Configuring the Trigger for Analog Output Operations

A trigger is an event that occurs based on a specified set of conditions. Once the analog output subsystem is armed with the **DA:ARM** command (and, if the trigger source is *IMMediate*, started with the **DA:INITiate** command), the analog output operation starts after the specified trigger is received and stops when either the specified number of samples have been output or you stop the operation. Refer to [page 64](#) for more information on the **DA:ARM** and **DA:INITiate** commands.

Note: The Output Trigger LED on the front panel of the DT8837 instrument module turns solid amber when the analog output subsystem is armed and solid green when the analog output subsystem is triggered. When the analog output subsystem is idle, the Output Trigger LED is off.

This section describes each of the supported trigger sources in detail. To return the currently configured trigger source, use the **DA:TRIGger[:SOURce]?** query.

Software Trigger

A software trigger event occurs immediately once you start the operation with the **DA:INITiate** command (the computer issues a write to the instrument module to begin conversions).

Use the **DA:TRIGger[:SOURce]** command with the *IMMediate* parameter to select the software trigger.

External, Digital (TTL) Trigger

An external digital (TTL) trigger event occurs when the instrument module detects a rising-edge transition on the signal connected to the Trigger In input on the instrument module.

Use the **DA:TRIGger[:SOURce]** command with the *EXternal* parameter to specify the external, digital (TTL) trigger.

Analog Threshold Trigger

An analog threshold trigger event occurs when the signal attached to analog input channel 1 rises above a user-specified threshold value from 0.2 V to 9.8 V with 0.1 V of hysteresis.

Use the **DA:TRIGger[:SOURce]** command with the *AIN1* parameter to specify the analog threshold trigger. Then, use the **AD:TRIGger:LEVel[:AIN1]** command to set the threshold level.

LAN Trigger Packet

When multiple DT8837 instrument modules are connected together over the local area network (LAN), you can synchronize the start of analog output operations by transmitting one of eight LAN trigger packets (LAN0 to LAN7) over the network.

To trigger analog output operations on a DT8837 instrument module using a LAN trigger packet, set up the LAN trigger as follows:

1. Set the trigger source to a LAN trigger packet (LAN0 to LAN7) using the **DA:TRIGger[:SOURce]** command.
2. Set the value of the LXI domain octet using the **EVENT:DOMain** command.
3. Enable the LAN event for transmission using the **EVENT:LAN<n>:TRANsmit[:ENABLE]** command.
4. Configure the LAN event ID field of the LXI packet using the **EVENT:LAN<n>:NAME** command.
5. Arm the analog output subsystem, as described on [page 64](#).

The analog output operation starts after the specified LAN trigger packet is received.

Trigger Bus

When multiple DT8837 instrument modules are connected together using the Trigger Bus, as shown in [Figure 17](#), you can synchronize the start of analog output operations using one of six LXI trigger signals: LXI0 to LXI5.

Note: When using LXI triggers, the timing from device to device is guaranteed to be within one sample pulse.

Internally, the trigger signals from the Trigger Bus are wired to the analog input subsystem. Therefore, choosing an LXI trigger for analog output operations is appropriate only if you want to trigger both the analog input and analog output subsystem of multiple instrument modules at the same time.

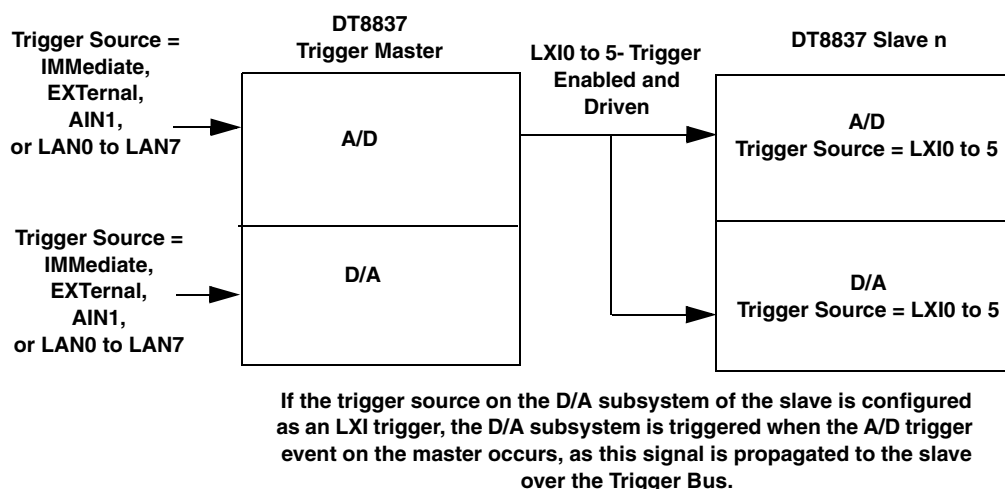


Figure 17: Synchronizing the Start of Analog Output Operations When Using the Trigger Bus

Set up the triggers as follows:

1. On the DT8837 trigger master, use the **DA:TRIGger[:SOURce]** command to set the trigger source to software trigger (IMMediate), external digital trigger (EXTernal), analog threshold trigger (AIN1), or LAN trigger packet (LAN0 to LAN7).
2. If the trigger source AIN1, set the trigger level using the **AD:TRIGger:LEVel[:AIN1]** command.
3. If the trigger source is LAN0 to LAN7, configure the LAN event trigger, as follows:
 - a. Set the value of the LXI domain octet using the **EVENT:DOMain** command.
 - b. Enable the LAN event for transmission using the **EVENT:LAN<n>:TRANsmit[:ENABLE]** command.
 - c. Configure the LAN event ID field of the LXI packet using the **EVENT:LAN<n>:NAME** command.
4. On each DT8837 slave, use the **DA:TRIGger[:SOURce]** command to select the trigger signal (LXI0 to LXI5) to receive from the Trigger Bus.
5. On the DT8837 trigger master, use the **WTB:LXI<n>[:OUTput]:ENABLE** command to drive out the LXI trigger signal (LXI0 to LXI5) on the Trigger Bus.

Note: If you change the trigger source on the master, you must call the **WTB:LXI<n>[:OUTput]:ENABLE** command again to ensure that the LXI trigger signal is driven out on the Trigger Bus.

6. Arm the analog output subsystem on each DT8837 slave, as described on [page 64](#).

7. Arm the analog output subsystem on the DT8837 trigger master, as described on [page 64](#).
8. If the trigger source of the DT8837 trigger master is IMMediate (software trigger), initiate the operation, as described on [page 64](#).

Once the analog input subsystem of the master is triggered, the specified LXI trigger signal is driven out on the Trigger Bus to all slaves that were configured to accept it. The output operation starts after the specified LXI trigger is received.

Arming Output Operations

Once you have configured all the parameters of the analog output subsystem, call the **DA:ARM** command to arm the subsystem. The arming process writes all the analog output configuration settings to the DT8837 hardware.

Note: If you want to perform analog output and analog input operations simultaneously, call **ARM** rather than **DA:ARM**, or explicitly call **DA:ARM** before calling **AD:ARM**. Once the analog input subsystem is armed, the analog output subsystem cannot be armed.

Then, use the **DA:STATus?** command, described below, or read bit 10 of the Operation Status register using the **STATus:OPERation:CONDition?** command to verify that the arming process is complete, before starting the analog output subsystem (described in the next section).

Starting Analog Output Operations

If you are using the software (IMMediate) trigger source and have armed the analog output subsystem, you must start the analog output operation by issuing the **DA:INITiate** command. For all other trigger sources, **DA:INITiate** is ignored.

When the software trigger is detected, the analog output subsystem updates the analog output channel with the values in the output buffer.

How the operation stops depends on the buffer wrap mode, described on [page 58](#).

Determining the Status of the Analog Output Subsystem

You can use the **DA:STATus?** command to determine the status of the analog output subsystem. The returned value, in the range of 0 to 255, represents the status bits for the analog output subsystem.

The status bits are defined as follows:

- Bit 0 – The value of this bit is 1 if the analog output subsystem is active or 0 if the analog output subsystem is inactive.
- Bit 2 – The value of this bit is 1 if the analog output subsystem is triggered or 0 if the analog output subsystem is not triggered.

- Bit 3 – The value of this bit is 1 if the DA FIFO is done (the end of the buffer is reached) when no wrap mode is selected or 0 if the DA FIFO is not done (the end of the buffer has not been reached) when no wrap mode is selected.
- Bit 4 – The value of this bit is 1 if the DA FIFO is empty or 0 if the DA FIFO is not empty.

Stopping Output Operations

To stop an analog output operation that is in progress, issue the **DA:ABORT** command. You can specify one of the following behaviors when the analog output operation is stopped:

- **Abrupt stop, hold last output value** – If you want the analog output operation to stop immediately and hold the last value that was output until the next **DA:ARM** command, as shown in [Figure 18](#), specify the optional *DEFault* parameter for the **DA:ABORT** command or do not specify any of the optional parameters for **DA:ABORT**.

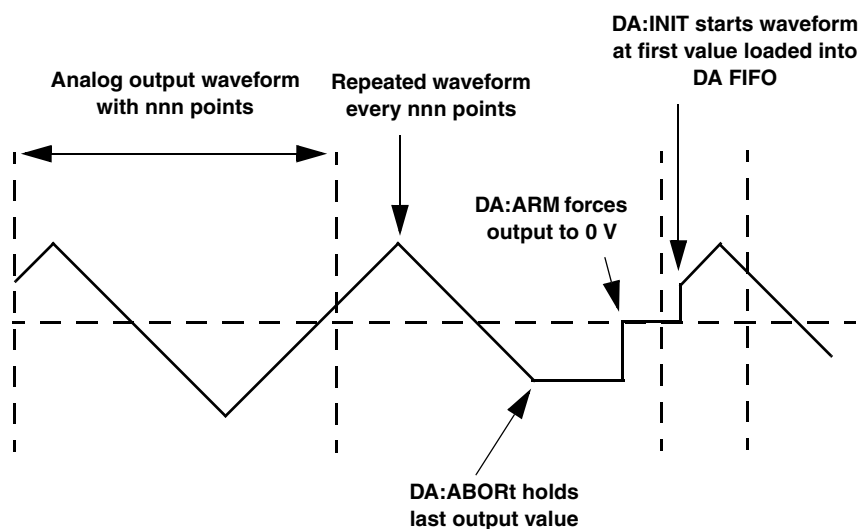


Figure 18: Abrupt Stop, Hold Last Output Value

- **Abrupt stop, set output value to zero** – If you want the analog output operation to stop immediately and return the analog output value to 0 V, as shown in [Figure 19](#), specify the optional *RETZero* parameter for the **DA:ABORT** command.

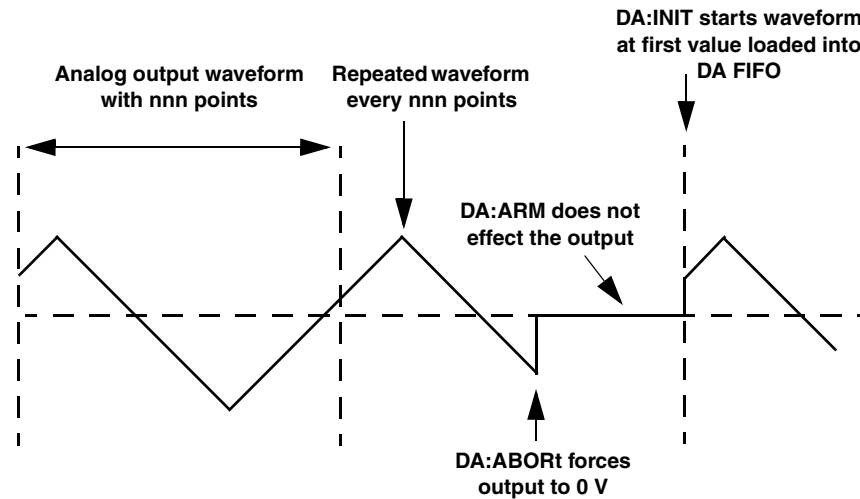


Figure 19: Abrupt Stop, Set Output Value to Zero

- **Stop when last value in buffer is output; hold last output value** – If you want the analog output operation to stop after the last value has been output from the buffer and then hold the last value that was output until the next **DA:ARM** command, as shown in [Figure 20](#), specify the optional *ENDBuffer* parameter for the **DA:ABORT** command.

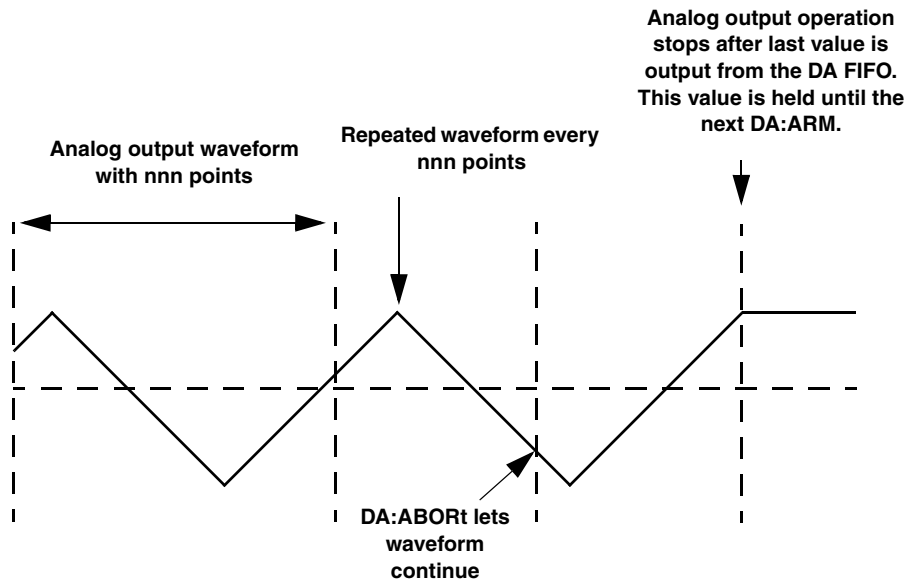


Figure 20: Stop When Last Value in Buffer is Output and Hold Last Output Value

Note: If the buffer mode is NOWRAP, you do not need to call **DA:ABORT**, since the operation stops automatically when the number of samples specified by **DA:BUFFER:SIZE** has been output. Refer to [page 58](#) for more information on buffer modes.

Performing Digital Output Operations

DT8837 instrument modules feature four, isolated digital output lines. The digital outputs are solid-state relays that operate at ± 30 V and 400 mA peak (AC or DC). Switching time is 2 ms maximum. Digital outputs resemble a switch; the switch is closed if the state of the digital output line is 1, and the switch is open if the state of the digital output line is 0. On power up or reset, the digital outputs are disabled.

Use the **DOUTput** command to close or open a digital relay.

You can also change the value of specific digital output lines without affecting all of the digital output lines by performing the following bitwise operations:

- Use the **DOUTput:AND** command to compare the binary representations of a specified value between 0 and 15 and the current value of the digital output port, and perform a logical AND operation on each pair of corresponding bits. If both bits are 1, the result is 1. If both bits are 0, the result is 0. If one bit is 1 and the other bit 0, the result is 0.

The following example shows the result of a logical AND operation:

Table 4:

	0101
AND	0011
=	0001

- OR – Use the **DOUTput:OR** command to compare the binary representations of a specified value between 0 and 15 and the current value of the digital output port, and perform a logical OR operation on each pair of corresponding bits. If both bits are 1, the result is 1. If both bits are 0, the result is 0. If one bit is 1 and the other bit 0, the result is 1.

The following example shows the result of a logical OR operation:

Table 5:

	0101
OR	0011
=	0111

You can query the current configuration of the digital output port by using the **DOUTput?** command.

Common SCPI Commands

Clear Status (*CLS).....	70
Standard Event Status Enable Register (*ESE).....	70
Standard Event Status Enable Register Query (*ESE?).....	71
Standard Event Status Register Query (*ESR?)	71
Identification Query (*IDN?)	73
Operation Complete (*OPC).....	74
Operation Complete Query (*OPC?).....	74
Reset (*RST).....	74
Self-Test Query (*TST?)	75
Service Request Enable (*SRE)	75
Service Request Enable Query (*SRE?)	75
Read Status Byte Query (*STB?).....	76
Wait-to-Continue (*WAI).....	77

Clear Status (*CLS)

Description	Clears all event registers summarized in the Status Byte (STB) register, described on page 208 .
Syntax	*CLS
Parameters	None
Response Data	None
Notes	All queues that are summarized in the Status Byte (STB) register, except the output queue, are emptied. The device is forced into the operation complete idle state and the operation complete query idle state.
Example	> *CLS

Standard Event Status Enable Register (*ESE)

Description	Enables specified bits in the Standard Event Status Enable register, described on page 209 .
Syntax	*ESE <DECIMAL NUMERIC PROGRAM DATA>
Required Parameters	
Name:	DECIMAL NUMERIC PROGRAM DATA
Data Format:	<0+NR1>
Description:	An integer value expressed in base 2 (binary) that represents the weighted bit value of the Standard Event Status Enable register. Values range from 0 to 255.
Response Data	None
Notes	The following table shows the bits of the Standard Event Status Enable Register and the binary-weighted decimal value for each bit; see page 209 for more information on each of these bits:

Binary		
Bit	Weight	Description
0	1	OPC (Operation Complete)
1	2	RQC (Request Control)
2	4	QYE (Query ERROR)
3	8	DDE (Device-Dependent ERROR)
4	16	E (Execution ERROR)
5	32	CME (Command ERROR)
6	64	NU (Not Used)
7	128	PON (Power on)

Refer to *IEEE Std 488.2-1992*, section 11.5.1.3, for more information.

Example The following command enables bits 0, 2, 3, 4, 5, and 7 of the Standard Event Status Enable register:

```
> *ESE 189
```

See Also *ESE?, described next
*ESR?, described on [page 71](#)

Standard Event Status Enable Register Query (*ESE?)

Description Returns the current value of the Standard Event Status Enable register, described on [page 209](#).

Syntax *ESE?

Parameters None

Response Data <NUMERIC RESPONSE DATA>

Name: NUMERIC RESPONSE DATA

Data Format: <0+NR1>

Description: An integer value expressed in base 2 (binary) that represents the weighted bit value of the Standard Event Status Enable register. Values range from 0 to 255.

Notes Refer to *IEEE Std 488.2-1992*, section 11.4.2.3.2, and *IEEE Std 488.2*, section 8.7.1, for more information.

Example The following command queries the Standard Event Status Enable register:

```
> *ESE?  
< 189
```

This value indicates that bits 0, 2, 3, 4, 5, and 7 of the Standard Event Status Enable register are enabled.

See Also *ESE, described on [page 70](#)
*ESR?, described on [page 71](#)

Standard Event Status Register Query (*ESR?)

Description Returns the current value of the Standard Event Status register, described on [page 211](#), and then clears the register.

Syntax *ESR?

Parameters None

Response Data <NUMERIC RESPONSE DATA>

Name: NUMERIC RESPONSE DATA

Data Format: <0+NR1>

Description: An integer value expressed in base 2 (binary) that represents the weighted bit value of the Standard Event Status register.

The bits of the Standard Event Status Register are listed below along with the binary-weighted decimal value for each bit; refer to [page 211](#) for more information on each of these bits:

Binary		
Bit	Weight	Description
0	1	OPC (Operation Complete)
1	2	RQC (Request Control)
2	4	QYE (Query ERROR)
3	8	DDE (Device-Dependent ERROR)
4	16	E (Execution ERROR)
5	32	CME (Command ERROR)
6	64	NU (Not Used)
7	128	PON (Power on)

Notes Bits in the Standard Event Status register should be unmasked by setting the corresponding bit in the Standard Event Status Enable register. On power up, the Standard Event Status Enable register is '0'; therefore, all bits in the Standard Event Status register are masked. The summary of the Standard Event Status register is reflected in the Standard Event Status Bit Summary (ESB) of the Status Byte register, described on [page 208](#).

Example The following example unmask all error bits in the Standard Event Status register:

```
> *ESE?;*ESE 255;*ESE?  
< 0;;255
```

Then, an illegal command is sent and the Standard Event Status register is queried; a value of 32 is returned, indicating that bit 5 (Command Error) of the Standard Event Status register was set:

```
> *bad  
> *ESR?  
< 32
```

In the following example, ESE is set to an invalid value, causing bit 4 (E) to be set:

```
> *ESE 65535  
> *ESR?  
< 16
```


See Also *ESE, described on [page 70](#)
 *ESE?, described on [page 71](#)
 *STB?, described on [page 76](#)

Identification Query (*IDN?)

Description	This command returns the unique identity of a DT8837 LXI instrument module.
Syntax	*IDN?
Parameters	None
Response Data	Manufacturer, Model, Serial number, Firmware revision
Data Format	<ARBITRARY ASCII RESPONSE DATA>
Name:	Manufacturer
Description:	Defines the manufacturer of the instrument module. For DT8837 LXI instrument modules, this response is Data Translation.
Name:	Model
Description:	Identifies the model of the DT8837 LXI instrument module.
Name:	Serial number
Description:	Identifies the serial number of the instrument module.
Name:	Firmware revision
Description:	Identifies the version of firmware that is loaded on the instrument module.
Notes	<p>Since the data format of the response is <ARBITRARY ASCII RESPONSE DATA>, the *IDN? query should be the last <QUERY MESSAGE UNIT> in a <TERMINATED PROGRAM MESSAGE>.</p> <p>Refer to <i>IEEE 488.2-1992</i>, sections 6.5.7.5, 8.7.11 and 10.14, for more information.</p>
Example	<pre>> *IDN? < Data Translation,DT8837,8029023,1.1</pre> <p>This response indicates that Data Translation is the manufacturer of the device, DT8837 is the model of the instrument module, 8029023 is the serial number of the instrument module, and 1.1 is the version of the firmware.</p>

Operation Complete (*OPC)

Description	The Operation Complete bit (bit 0) of the Standard Event Status register, described on page 211 , is always enabled. Therefore, this command has no effect when used with a DT8837 LXI instrument module.
Syntax	*OPC
Parameters	None
Response Data	None
Notes	This command is implemented for SCPI compliance to avoid error messages when it is sent by the controller to the instrument module.
Example	> *OPC
See Also	*OPC?, described below

Operation Complete Query (*OPC?)

Description	The Operation Complete bit (bit 0) of the Standard Event Status register, described on page 211 , is always enabled. Therefore, this command always places the ASCII character 1 into the device's Output Queue.
Syntax	*OPC?
Parameters	None
Response Data	<NUMERIC RESPONSE DATA>
Name:	NUMERIC RESPONSE DATA
Data Format:	<NR1>
Description:	A single ASCII-encoded byte for 1 (31, 49 decimal).
Example	> *OPC? < 31

Reset (*RST)

Description	Clears the Standard Event Status register, message queue, error queue, and Status Byte register, and stop any scans that are in progress.
Syntax	*RST
Parameters	None
Response Data	None
Notes	Refer to <i>IEEE 488.2-1992</i> , section 10.32, for more information.
Example	> *RST

Self-Test Query (*TST?)

Description	Always returns 0 for DT8837 LXI instrument modules.
Syntax	*TST?
Parameters	None
Response Data	<NUMERIC RESPONSE DATA>
Name:	NUMERIC RESPONSE DATA
Data Format:	<NR1>
Description:	This value is always 0 for DT8837 LXI instrument modules.
Notes	This query has no effect. This command is implemented for SCPI compliance to avoid error messages when it is sent by the controller to the instrument module.
Example	> *TST? < 0

Service Request Enable (*SRE)

Description	The Service Request Enable register is not used on these instrument modules. Therefore, this command has no effect when used with a DT8837 LXI instrument module.
Syntax	*SRE <value>
Required Parameters	
Name:	value
Data Format:	<0+NR1>
Description:	A positive decimal value; this value is ignored.
Response Data	None
Notes	This command is implemented for SCPI compliance to avoid error messages when it is sent by the controller to the instrument module.
Example	> *SRE 255
See Also	*SRE?, described below

Service Request Enable Query (*SRE?)

Description	The Service Request Enable register is not used on these instrument modules. Therefore, this command places the ASCII character 0 into the device's Output Queue.
Syntax	*SRE?
Parameters	None

Response Data	<NUMERIC RESPONSE DATA>
Name:	NUMERIC RESPONSE DATA
Data Format:	<NR1>
Description:	A single ASCII-encoded byte for 0.
Notes	This command is implemented for SCPI compliance to avoid error messages when it is sent by the controller to the instrument module.
Example	> *SRE? < 0

Read Status Byte Query (*STB?)

Description	Returns the current value of the Status Byte register, described on page 208 .
Syntax	*STB?
Parameters	None
Response Data	<NUMERIC RESPONSE DATA>
Name:	NUMERIC RESPONSE DATA
Data Format:	<NR1>
Description:	The weighted sum of the bit values of the Status Byte register, ranging from 0 to 255. The following bits, described in <i>1999 SCPI Syntax & Style</i> , section 9, are supported by DT8837 LXI instrument modules: <ul style="list-style-type: none">• Bit 7 (weighted bit value = 128) – Summary of device dependent Operation Status Register• Bit 5 (weighted bit value = 32) – Event Status Bit Summary (ESB), '1' = ESR is non-zero, '0' otherwise• Bit 4 (weighted bit value = 16) – Message Available Queue Summary (MAV), '1' = message queue not empty• Bit 2 (weighted bit value = 4) – Error/Event Queue Summary, '1' = Error queue not empty
Notes	Refer to <i>IEEE 488.2-1992</i> , section 10.36, for more information.
Example	The following example shows a query that is correct and causes no errors: > *IDN?;*ESR?;*STB? < Data Translation,DT8837,-1,1.2;0;16

Example (cont.) This example shows an illegal command being sent and the status of the Status Byte register and the error queue:

```
> bad
> *STB?
< 36
```

A value of 36 indicates that bits 5 (Standard Event Status Bit Summary) and 2 (Error / Event Queue Summary) of the Status Byte register are set.

The following example shows the status of the Event Status register:

```
> *ESR?
< 32
```

A value of 32 indicates that bit 5 (Command Error) of the Event Status register is set. The following updates the status of the Status Byte register:

```
> *STB?
< 4
```

A value of 4 indicates that bit 2 (Error / Event Queue Summary) of the Status Byte register is set. The following shows the error codes that are returned, and updates the status of the Status Byte register:

```
> :SYST:ERR?
< -110, "Command header error;bad"
> :SYST:ERR?
< 0, "No error"
> *STB?
< 0
```

Wait-to-Continue (*WAI)

Description	This command has no effect when used with DT8837 LXI instrument modules.
Syntax	*WAI
Parameters	None
Response Data	None
Notes	This command is implemented for SCPI compliance to avoid any error message when it is sent by the controller to the instrument module.
Example	> *WAI



SCPI Subsystem Commands for the DT8837

SCPI Subsystem Command Hierarchy	80
STATus Subsystem Commands	83
SYSTem Subsystem Commands	89
AD Subsystem Commands	101
TACHometer Subsystem Commands	130
COUNter Subsystem Commands	137
DA Subsystem Commands	145
WTB (Wired Trigger Bus) Subsystem Commands	170
EVENT Subsystem Command	174
DOUTput Subsystem Commands	181

Note: In the examples in this section, the symbols > and < represent communication to and from the instrument module, respectively. Users do not explicitly send or receive these symbols.

SCPI Subsystem Command Hierarchy

SCPI subsystem commands are hierarchical and are organized as an inverted tree structure with the "root" at the top. Table 6 shows the hierarchy of SCPI subsystem commands for DT8837 LXI instrument modules.

Table 6: Hierarchy of SCPI Subsystem Commands for DT8837 LXI Instrument Modules

STATus	:OPERation	[:EVENT]?	
		:CONDition?	
		:ENABle[?]	
	:PRESet		
	:QUEStionable	[:EVENT]?	
		:CONDition?	
		:ENABle[?]	
SYSTem	:DATE?		
	:DESCRiption[?]		
	:ERRor	:ALL?	
		[:NEXT]	
		:COUNT?	
		:CODE	:ALL?
			[:NEXT]?
	:COMMunicate	:NET	:IP[?]
			:MASK[?]
	:TIME?		
	:TZONE?		
	:VERSion?		
AD	:ABORt		
	:ARM		
	:BIAS	:CURRent	[:ENABle][?]
	:BUFFer	:MODE[?]	
		:SIZe	[:SCAns]?
	:CLOCK	:FREQuency	[:CONFIgure][?]
		:SOURce[?]	
	:COUPling	[:CONFIgure][?]	
	:ENABle[?]		

Table 6: Hierarchy of SCPI Subsystem Commands for DT8837 LXI Instrument Modules (cont.)

AD (cont.)	:FETCh?		
	:GAIN	:CONFigure[?]	
	:INITiate		
	:STATus	:SCAN?	
	:STATus?		
	:SYNc	:SOURce[?]	
	:TRIGger	:LEVel	:AIN1[?]
		:SOURce[?]	
TACHometer	:ENABle[?]		
	:PERiod	:CONFigure[?]	
	:SCLR	:CONFigure[?]	
	:SFLAG	:CONFigure[?]	
COUNter[<n>]	:ENABle[?]		
	:EDGE	:{START STOP}	:CONFigure[?]
	:SCLR	:CONFigure[?]	
DA	:ABORt		
	:BUFFer	:MODE[?]	
		:SIZE[:SCANs][?]	
		:DATA[?]	
	:CLOCK	:FREQuency	:CONFigure[?]
		:SOURce[?]	
	:ENABle[?]		
	:FETCh?		
	:INITiate		
	:READ	:ENABle[?]	
	:TRIGger	:SOURce[?]	
	:STATus?		
WTB	:LXI<n>	:OUTput	:ENABle[?]
EVENT	:DOMain[?]		
	:LAN<n>	:NAME[?]	
		:TRANsmit	:ENABle[?]

Table 6: Hierarchy of SCPI Subsystem Commands for DT8837 LXI Instrument Modules (cont.)

DOUTput	[?]
	:AND
	:OR

Looking at the STATus subsystem, for example, STATus is the root keyword of the command, OPERation is a second-level keyword, and CONDition is a third-level keyword. The next section describes the syntax of program messages and SCPI commands from the controller to a DT8837 LXI instrument module.

STATus Subsystem Commands

The STATus subsystem includes the commands listed in [Table 7](#) for determining the operational status of a DT8837 LXI instrument module. This section describes each of these commands in detail.

Table 7: STATus Subsystem Commands

Type	Mnemonic	See
Operation Condition Register Query	STATus:OPERation:CONDition?	page 84
Operation Enable Register	STATus:OPERation:ENABle	page 85
Operation Enable Register Query	STATus:OPERation:ENABle?	page 85
Operation Event Register Query	STATus:OPERation[:EVENT]?	page 86
Presetting Registers	STATus:PRESet	page 86
Questionable Condition Register Query	STATus:QUESTionable:CONDition?	page 86
Questionable Enable Register	STATus:QUESTionable:ENABle	page 87
Questionable Enable Register Query	STATus:QUESTionable:ENABle?	page 87
Questionable Event Register Query	STATus:QUESTionable[:EVENT]?	page 88

Operation Condition Register Query

Description	Returns the current value of the Operation Status register, described on page 212 .
Syntax	:STATus:OPERation:CONDition?
Parameters	None
Response Data	<NUMERIC RESPONSE DATA>
Name:	NUMERIC RESPONSE DATA
Data Format:	<0+NR1>
Description:	The weighted bit value of the Operation Status register. Values for the response range from 0 to 32767.
Notes	You can query the bits of the Operation Status register to determine whether the input and output buffers are full or empty, and whether the input and output triggers have occurred.
Example	<p>The following example shows the status of the Operation Status register and the Status Byte register when the analog input subsystem is waiting to be armed:</p> <pre>> :STAT:OPER:COND? < 64 > *STB? < 128</pre> <p>A value of 64 indicates that bit 6 (decimal value 64) of the Operation Status registers is set to 1. A value of 128 indicates that bit 7 (Operation Status Register Summary) of the Status Byte register is set to 1.</p> <p>The following example shows the status of the Operation Status register and the Status Byte register when the analog input subsystem is waiting for a trigger:</p> <pre>> :STAT:OPER:COND? < 32 > *STB? < 128</pre> <p>A value of 32 indicates that bit 5 (decimal value 32) of the Operation Status registers is set to 1. A value of 128 indicates that bit 7 (Operation Status Register Summary) of the Status Byte register is set to 1.</p>

Operation Event Register Enable

Description The Operation Event register is not used on DT8837 LXI instrument modules; therefore, this command has no effect on a DT8837 LXI instrument module.

Syntax :STATus:OPERation:ENABle <bitmask>

Required Parameters

Name: bitmask
Data Format: <NRr>
Description: A non-decimal numeric value; this value is ignored by DT8837 LXI instrument modules.

Response Data None

Notes This command is implemented for SCPI compliance to avoid error messages when it is sent by the controller to the instrument module.

Example > :STAT:OPER:ENAB #B00000000

See Also STATus:OPERation:CONDition?, described on [page 84](#)

Operation Enable Register Query

Description The Operation Enable register is not used on DT8837 LXI instrument modules; therefore, this query always returns 0.

Syntax :STATus:OPERation:ENABle?

Parameters None

Response Data <bitmask>

Name: bitmask
Data Format: <0+NR1>
Description: The Operation Enable register is not used. Therefore, this value is always 0.

Notes This command is implemented for SCPI compliance to avoid error messages when it is sent by the controller to the instrument module.

Example > :STAT:OPER:ENAB?
 < 0

See Also STATus:OPERation:CONDition?, described on [page 84](#)

Operation Event Register Query

Description	The Operation Event register is not used on DT8837 LXI instrument modules; therefore, this query always returns 0.
Syntax	:STATus:OPERation[:EVENT]?
Parameters	None
Response Data	<NUMERIC RESPONSE DATA>
Name:	NUMERIC RESPONSE DATA
Data Format:	<0+NR1>
Description:	The Operation Event register is not used. Therefore, the value of this response is always 0.
Notes	This command is implemented for SCPI compliance to avoid error messages when it is sent by the controller to the instrument module.
Example	> :STAT:OPER:EVEN? < 0
See Also	STATus:OPERation:CONDition?, described on page 84

Presetting Registers

Description	This command has no effect when used with a DT8837 LXI instrument module.
Syntax	:STATus:PRESet <value>
Required Parameters	
Name:	value
Data Format:	<NR1>
Description:	For DT8837 LXI instrument modules, this value is ignored.
Response Data	None
Notes	This command is implemented for SCPI compliance.
Example	> :STAT:PRES 1

Questionable Condition Register Query

Description	The Questionable Condition register is not used on DT8837 LXI instrument modules; therefore, this query always returns 0.
Syntax	:STATus:QUEStionable:CONDition?
Parameters	None
Response Data	<regvalue>

Name:	regvalue
Data Format:	<0+NR1>
Description:	The Questionable Condition register is not used. Therefore, the value of this response is always 0.
Notes	This command is implemented for SCPI compliance to avoid error messages when it is sent by the controller to the instrument module.
Example	<pre>> :STAT:QUES:COND? < 0</pre>
See Also	STATus:OPERation:CONDition?, described on page 84

Questionable Enable Register

Description The Questionable Enable register is not used on DT8837 LXI instrument modules. Therefore, this command has no effect when used with a DT8837 LXI instrument module.

Syntax :STATus:QUEStionable:ENABle <bitmask>

Required Parameters

Name:	bitmask
Data Format:	<NRr>
Description:	A non-decimal numeric value; this value is ignored by DT8837 LXI instrument modules.

Response Data None

Notes This command is implemented for SCPI compliance to avoid error messages when it is sent by the controller to the instrument module.

Example > :STAT:QUES:ENAB #B00000000

Questionable Enable Register Query

Description The Questionable Enable register is not used on DT8837 LXI instrument modules; therefore, this query always returns 0.

Syntax :STATus:QUEStionable:ENABle?

Parameters None

Response Data <bitmask>

Name:	bitmask
Data Format:	<NR1>
Description:	The Questionable Enable register is not used. Therefore, the value of this response is always 0.

Notes This command is implemented for SCPI compliance to avoid error messages when it is sent by the controller to the instrument module.

Example > :STAT:QUES:ENAB?
< 0

Questionable Event Register Query

Description The Questionable Event register is not used on DT8837 LXI instrument modules; therefore, this query always returns 0.

Syntax :STATus:QUESTionable[:EVENT]?

Parameters None

Response Data <regvalue>

Name: regvalue

Data Format: <0+NR1>

Description: The Questionable Event register is not implemented. Therefore, the value of this response is always 0.

Notes This command is implemented for SCPI compliance to avoid error messages when it is sent by the controller to the instrument module.

Example > :STAT:QUES:EVEN?
< 0

See Also STATus:OPERation:CONDition?, described on [page 84](#)

SYSTem Subsystem Commands

The SYSTem subsystem includes the commands listed in [Table 8](#). Use these commands to calibrate DT8837 LXI instrument modules, query the status of scan records on DT8837 LXI instrument modules, and configure or query global system settings, including the time, date, time zone, and network address of the instrument module. This section describes each of these commands in detail.

Table 8: SYSTem Subsystem Commands

Type	Mnemonic	See
Date Query	SYSTem:DATE?	page 90
Description	SYSTem:DESCRiption	page 90
Description Query	SYSTem:DESCRiption?	page 91
Error Query All	SYSTem:ERRor:ALL?	page 91
Error Query All Codes	SYSTem:ERRor:CODE:ALL?	page 92
Error Query Count	SYSTem:ERRor:COUNt?	page 93
Error Query Next	SYSTem:ERRor[:NEXT]?	page 94
Error Query Next Code	SYSTem:ERRor:CODE[:NEXT]?	page 95
LAN Static IP - Address Configuration	SYSTem:COMMunicate:NETwork:IPAddr	page 96
LAN Static IP - Address Query	SYSTem:COMMunicate:NETwork:IPAddr?	page 97
LAN Static IP - Subnet Mask Configuration	SYSTem:COMMunicate:NETwork:MASk	page 97
LAN Static IP - Subnet Mask Query	SYSTem:COMMunicate:NETwork:MASk?	page 98
SCPI Version Query	SYSTem:VERSion?	page 98
Time Query	SYSTem:TIME?	page 99
Time Zone Query	SYSTem:TZONE?	page 100

Date Query

Description	Returns the current date of the DT8837 LXI instrument module. This date is updated automatically by an SNTP (Simple Network Time Protocol) server.
Syntax	:SYSTem:DATE?
Parameters	None
Response Data	<year> , <month> , <day>
Name:	year
Data Format:	<NR1>
Description:	A number representing the year, such as 2009.
Name:	month
Data Format:	<NR1>
Description:	A number from 1 to 12 representing the month.
Name:	day
Data Format:	<NR1>
Description:	A number from 1 to 31 representing the day.
Example	<pre>> :SYST:DATE? < 2009,1,15</pre> <p>This response indicates that the date of the DT8837 instrument module is January 15th, 2009.</p>

Description

Description	Specifies a description for the instrument module.
Syntax	:SYSTem:DESCRiption <device_description>
Parameters	<device_description>
Response Data	None
Name:	device_description
Data Format:	<CHARACTER PROGRAM DATA>
Description:	A string with a maximum length of 64 ASCII characters, including the terminating NULL. Strings exceeding this limited are truncated.
Example	<pre>> :SYST:DESC < The DT8837 in the lab</pre> <p>This command sets the description of the DT8837 to "The DT8837 in the lab".</p>

Description Query

Description	Returns the description of the instrument module.
Syntax	:SYSTem:DESCRiption?
Parameters	None
Response Data	<device_description>
Data Format	<CHARACTER RESPONSE DATA>
Name:	device_description
Description:	A string with a maximum length of 64 ASCII characters.
Example	<pre>> :SYST:DESC? < The DT8837 in the lab</pre> <p>In this case, the description of the device is as follows: "The DT8837 in the lab".</p>

Error Query All

Description	Queries the error/event queue for all the unread errors and removes them from the Error/Event Queue.
Syntax	:SYSTem:ERRor:ALL?
Parameters	None
Response Data	<Error/event_number>, <Error/event_description>, <Device-dependent info>
Data Format	<CHARACTER RESPONSE DATA>
Name:	Error/event_number
Description:	A unique integer in the range of -32768 to 32767. A value of 0 indicates that no error or event has occurred. Refer to Appendix A starting on page 201 for a list of errors that can be returned.
Name:	Error/event_description
Description:	A quoted string that contains a description of the error that was read from the Error/Event Queue. Refer to Appendix A starting on page 201 for a list of errors that can be returned.
Name:	Device-dependent info
Description:	Optional text that provides device-dependent information about the error that was read from the Error/Event Queue. Refer to Appendix A starting on page 201 for a list of errors that can be returned.

Notes The maximum string length of `<Error/event_description>` plus `<Device-dependent info>` is 255 characters.

The error queue is a first-in, first-out (FIFO) with a capacity of 32 error messages. By querying the error count before and after a SCPI command (in a single command string), you can unambiguously determine whether the command that you issued caused an error. Use the **SYSTem:ERRor:COUNT?** command, described on [page 93](#), to return the number of unread items in the error queue.

When the queue is full, the message "-350,'Queue overflow'" is returned and subsequent errors can not be added to the queue. Use the ***CLS** command, described on [page 93](#), to clear the error queue as well as the Status Byte register.

Refer to *1999 SCPI Command Reference*, section 21.8.4, for more information.

Example The following shows the responses to this query after an invalid command is sent to the instrument module:

```
>:BADc
>:SYST:ERR:ALL?
< -110, "Command header error;:BADc"
> :SYST:ERR:ALL?
< 0, "No error"
```

See Also **SYSTem:ERRor[:NEXT]?**, described on [page 94](#)
SYSTem:ERRor:COUNT?, described on [page 93](#)
SYSTem:ERRor:CODE:ALL?, described on [page 92](#)
SYSTem:ERRor:CODE[:NEXT]?, described on [page 95](#)

Error Query All Codes

Description	Returns a comma-separated list of only the error/event code numbers in FIFO order. If the queue is empty, the response is 0.
Syntax	<code>:SYSTem:ERRor:CODE:ALL?</code>
Parameters	None
Response Data	<code><Error/event_number></code> , <code><Error/event_number></code> , , <code><Error/event_number></code>
Data Format	<code><CHARACTER RESPONSE DATA></code>
Name:	Error/event_number
Description:	A unique integer in the range of -32768 to 32767. A value of 0 indicates that no error or event has occurred. Refer to Appendix A starting on page 201 for a list of errors that can be returned.
Notes	Refer to <i>1999 SCPI Command Reference</i> , section 21.8.5.1 for more information.

Example The following shows the responses to this query after an invalid command is sent to the instrument module:

```
> :BADc
> :SYST:ERR:CODE:ALL?
< -110,0;
```

See Also **SYSTem:ERRor:ALL?**, described on [page 91](#)
SYSTem:ERRor[:NEXT]?, described on [page 94](#)
SYSTem:ERRor:COUnT?, described on [page 93](#)
SYSTem:ERRor:CODE[:NEXT]?, described on [page 95](#)

ERRor Query Count

Description	Queries the Error/Event Queue for the number of unread items and returns the count.
Syntax	:SYSTem:ERRor:COUnT?
Parameters	None
Response Data	<Count>
Name:	Count
Data Format:	<0+NR1>
Description:	<p>A unique integer that indicates the number of unread errors in the error queue. A value of 0 indicates that the queue is empty.</p> <p>Since errors may occur at any time, more items may be present in the error queue at the time that it is actually read.</p>
Notes	<p>The error queue is a first-in, first-out (FIFO) with a capacity of 32 error messages. The error queue accumulates errors from all clients. By querying the error count before and after a SCPI command (in a single program message), you can unambiguously determine whether the command that you issued caused an error.</p> <p>When the queue is full, the message "-350,'Queue overflow'" is returned and subsequent errors can not be added to the queue. Use the SYSTem:ERRor:ALL? command, described on page 91, to read the error and remove it from the queue. Use the *CLS command, described on page 70, to clear the error queue as well as the Status Byte register.</p> <p>Refer to <i>1999 SCPI Command Reference</i>, section 21.8.6 for more information.</p>

Example The following shows how to query the number of errors in the error queue and the error count:

```
> :BADc
> :SYST:ERR:COUNT?
< 1
> :SYST:ERR?
< -110, "Command header error;:BADc"
> :SYST:ERR?
< 0, "No error"
> :SYST:ERR:COUNT?
< 0
```

See Also `SYSTem:ERRor:ALL?`, described on [page 91](#)
`SYSTem:ERRor[:NEXT]?`, described on [page 94](#)
`SYSTem:ERRor:CODE:ALL?`, described on [page 92](#)
`SYSTem:ERRor:CODE[:NEXT]?`, described on [page 95](#)

Error Query Next

Description	Queries the next error and removes it from the Error/Event Queue.
Syntax	<code>:SYSTem:ERRor[:NEXT]?</code>
Parameters	None
Response Data	<code><Error/event_number>, <Error/event_description>, <Device-dependent info></code>
Data Format	<code><CHARACTER RESPONSE DATA></code>
Name:	Error/event_number
Description:	A unique integer in the range of -32768 to 32767. A value of 0 indicates that no error or event has occurred. Refer to Appendix A starting on page 201 for a list of errors that can be returned.
Name:	Error/event_description
Description:	A quoted string that contains a description of the error that was read from the Error/Event Queue. Refer to Appendix A starting on page 201 for a list of errors that can be returned.
Name:	Device-dependent info
Description:	Optional text that provides device-dependent information about the error that was read from the Error/Event Queue. Refer to Appendix A starting on page 201 for a list of errors that can be returned.

Notes The maximum string length of *<Error/event_description>* plus *<Device-dependent info>* is 255 characters.

The error queue is a first-in, first-out (FIFO) with a capacity of 32 error messages. By querying the error count before and after a SCPI command (in a single command string), you can unambiguously determine whether the command that you issued caused an error. Use the **SYSTem:ERRor:COUNt?** command, described on [page 93](#), to return the number of unread items in the error queue.

When the queue is full, the message "-350, Queue overflow" is returned and subsequent errors can not be added to the queue. Use the ***CLS** command, described on [page 70](#), to clear the error queue as well as the Status Byte register.

Refer to *1999 SCPI Command Reference*, section 21.8.5.2, for more information.

Example The following shows the responses to this query after an invalid command is sent to the instrument module:

```
> :BADc
> :SYST:ERR?
< -110, "Command header error;:BADc"
> :SYST:ERR?
< 0, "No error"
```

See Also **SYSTem:ERRor:ALL?**, described on [page 91](#)

SYSTem:ERRor:COUNt?, described on [page 93](#)

SYSTem:ERRor:CODE:ALL?, described on [page 92](#)

SYSTem:ERRor:CODE[:NEXT]?, described on [page 95](#)

Error Query Next Code

Description	Queries the next error code and removes it from the Error/Event Queue.
Syntax	:SYSTem:ERRor:CODE[:NEXT]?
Parameters	None
Response Data	<Error/event_number>
Data Format	<CHARACTER RESPONSE DATA>
Name:	Error/event_number
Description:	A unique integer in the range of -32768 to 32767. A value of 0 indicates that no error or event has occurred. Refer to Appendix A starting on page 201 for a list of errors that can be returned.
Notes	Refer to <i>1999 SCPI Command Reference</i> , section 21.8.5.2 for more information.

Example The following shows the responses to this query after an invalid command is sent to the instrument module:

```
> :BADc
> :SYST:ERR:CODE:NEXT?
< -110
> :SYST:ERR:CODE:NEXT?
< 0
```

See Also **SYSTem:ERRor:ALL?**, described on [page 91](#)
SYSTem:ERRor[:NEXT]?, described on [page 94](#)
SYSTem:ERRor:COUNT?, described on [page 93](#)
SYSTem:ERRor:CODE:ALL?, described on [page 92](#)

LAN Static IP - IP Address Configuration

Description Configures the static IP address that is used by the DT8837 instrument module on the network.

Syntax :SYSTem:COMMunicate:NETwork:IPAddr <ipaddr>

Required Parameters

Name: ipaddr

Data Format: <0+NR1>, <CHARACTER PROGRAM DATA>

Description: Specifies the IP address of the DT8837 instrument module. The address must be four decimal numbers in the range 0-255, separated by periods, such as "192.43.218.59".

Response Data None

Notes In addition to the static IP address, you must also set the subnet mask using the **SYSTem:COMMunicate:NETwork:MASk** command, described on [page 97](#).

The DT8837 instrument module is configured to use DHCP with Auto-IP, by default.

Example This command sets the IP address of the DT8837 instrument module to 192.43.218.59.

```
> :SYST:COMM:NET:IP 192.43.218.59
```

See Also **SYSTem:COMMunicate:NETwork:IPAddr?**, described on [page 97](#)
SYSTem:COMMunicate:NETwork:MASk, described on [page 97](#)
SYSTem:COMMunicate:NETwork:MASk?, described on [page 98](#)

LAN Static IP - IP Address Query

Description	Returns the IP address that is currently used by the DT8837 instrument module on the network, regardless of the method (DHCP, Auto-IP, static IP) used to acquire the address.
Syntax	:SYSTem:COMMunicate:NETwork:IPAddr?
Response Data	<ipaddr>
Name:	ipaddr
Data Format:	<0+NR1>, <CHARACTER PROGRAM DATA>
Description:	The IP address of the DT8837 instrument module. The address must be four decimal numbers in the range 0-255, separated by periods, such as "192.43.218.59".
Notes	The DT8837 instrument module is configured to use the DHCP server, by default.
Example	This command returns the IP address of the DT8837 instrument module, which in this example is 192.43.218.15. <pre>> :SYST:COMM:NET:IP? < 192.43.218.59</pre>
See Also	SYSTem:COMMunicate:NETwork:IPAddr , described on page 96 SYSTem:COMMunicate:NETwork:MASk , described on page 97 SYSTem:COMMunicate:NETwork:MASk? , described on page 98

LAN Static IP - Subnet Mask Configuration

Description	Configures the subnet mask that is used by the DT8837 instrument module on the network.
Syntax	:SYSTem:COMMunicate:NETwork:MASk <mask>
Required Parameters	
Name:	mask
Data Format:	<0+NR1>, <CHARACTER PROGRAM DATA>
Description:	Specifies the subnet mask of the DT8837 instrument module. The subnet mask must be four decimal numbers in the range 0 to 255, separated by periods, such as "255.255.255.0".
Response Data	None
Notes	In addition to the subnet mask, you must also set the static IP address using the SYSTem:COMMunicate:NETwork:IP command, described on page 96 .
Example	This command sets the subnet mask of the DT8837 instrument module to 255.255.255.0: <pre>> :SYST:COMM:NET:MASk 255.255.255.0</pre>

See Also **SYSTem:COMMunicate:NETwork:IP**, described on [page 96](#)
 SYSTem:COMMunicate:NETwork:IP?, described on [page 97](#)
 SYSTem:COMMunicate:NETwork:MASk?, described on [page 98](#)

LAN Static IP - Subnet Mask Query

Description Returns the subnet mask that is currently used by the DT8837 instrument module on the network, regardless of the method (DHCP, Auto-IP, static IP) used to acquire the address.

Syntax :SYSTem:COMMunicate:NETwork:MASk?

Response Data <mask>

 Name: mask

 Data Format: <0+NR1>, <CHARACTER PROGRAM DATA>

 Description: The subnet mask of the DT8837 instrument module. The subnet mask must be four decimal numbers in the range 0-255, separated by periods, such as "255.255.255.0".

Example This command returns the subnet mask of the DT8837 instrument module, which in this example is 255.255.255.0:

```
> :SYST:COMM:NET:MAS?
< 255.255.255.0
```

See Also **SYSTem:COMMunicate:NETwork:IP**, described on [page 96](#)
 SYSTem:COMMunicate:NETwork:IP?, described on [page 97](#)
 SYSTem:COMMunicate:NETwork:MASk, described on [page 97](#)
 SYSTem:COMMunicate:NETwork:MASk?, described on [page 98](#)

SCPI Version Query

Description This command, which is mandatory for SCPI-compliant devices, returns the SCPI version number to which the DT8837 instrument module complies.

Syntax :SYSTem:VERSion?

Parameters None

Response Data <YYYY> . <V>

Data Format <CHARACTER RESPONSE DATA>

 Name: YYYY

 Data Format: <NR2>

 Description: The year of the version, such as 2009.

Name:	V
Data Format:	<NR2>
Description:	The approved SCPI revision number for that year, such as 0.
Notes	Refer to <i>1999 SCPI Command Reference</i> , section 21.21 for more information.
Example	<pre>> :SYST:VERS? < 1999.0</pre> <p>In this example, the version is year 1999 and the SCPI revision is 0.</p>

Time Query

Description	Returns the current time used by the DT8837 LXI instrument module. This time is updated automatically by an SNTP server.
Syntax	:SYSTem:TIME?
Parameters	None
Response Data	<hour>, <minute>, <second>
Name:	hour
Data Format:	<NR1>
Description:	A number from 0 to 23 representing the current hour of the instrument module.
Name:	minute
Data Format:	<NR1>
Description:	A number from 0 to 59 representing the current minute of the instrument module.
Name:	second
Data Format:	<NR1>
Description:	A number from 0 to 59 representing the current second of the instrument module.
Example	<pre>> :SYST:TIME? < 15, 31, 45</pre> <p>This response indicates that the current time of the DT8837 instrument module is 3:31:45 pm (15 is the hour, 31 is the number of minutes, and 45 is the number of seconds).</p>

Time Zone Query

Description Returns the time zone that is currently used by the instrument module, as an offset from GMT (Greenwich Mean Time).

Syntax :SYSTem:TZONe?

Required Parameters None

Response Data

Name: hour
Data Format: <NR1>
Description: A number from -12 to +12 representing the current hour relative to GMT.

Name: minute
Data Format: <NR1>
Description: A number from -59 to +59 representing the current minute relative to GMT. Minutes are rounded up to 30.

Response Data None

Example
> :SYST:TZON
< +5, -45

This response indicates that the current time zone of the instrument module is four hours and 30 minutes ahead of GMT. Minutes are rounded up to 30.

AD Subsystem Commands

The AD subsystem is used to configure perform analog input operations on a DT8837 LXI instrument module. [Table 9](#) lists the AD subsystem commands. This section describes each of these commands in detail.

Table 9: AD Subsystem Commands

Type	Mnemonic	See
Abort Analog Input Operation	AD:ABORt	page 102
Arm Analog Input Operation	AD:ARM	page 103
ADC Synchronization Source Query	AD:SYNc:SOURce?	page 104
Analog Input Buffer Mode Configuration	AD:BUFFer:MODE	page 104
Analog Input Buffer Mode Query	AD:BUFFer:MODE?	page 105
Analog Input Buffer Size Query	AD:BUFFer:SIZe[:SCAns]?	page 106
Analog Input Channel Enable	AD:ENABle	page 107
Analog Input Channel Enable Query	AD:ENABle?	page 108
Analog Input Clock Source Configuration	[AD:]CLOCK:SOURce	page 109
Analog Input Clock Source Query	[AD:]CLOCK:SOURce?	page 110
Analog Input Coupling Configuration	AD:COUPling[:CONFIgure]	page 111
Analog Input Coupling Query	AD:COUPling[:CONFIgure]?	page 111
Analog Input Current Source Configuration	AD:BIAS:CURREnt:ENABle	page 112
Analog Input Current Source Query	AD:BIAS:CURREnt:ENABle?	page 113
Analog Input Gain Configuration	AD:GAIN[:CONFIgure]	page 114
Analog Input Gain Query	AD:GAIN[:CONFIgure]?	page 114
Analog Input Sampling Frequency Configuration	AD:CLOCK:FREQuency [:CONFIgure]	page 115
Analog Input Sampling Frequency Query	AD:CLOCK:FREQuency[:CONFIgure]?	page 116
Analog Input Scan Status Query	AD:STATus:SCAN?	page 117
Analog Input Status Bits Query	AD:STATus?	page 118
Analog Input Trigger Source Configuration	AD:TRIGGer[:SOURce]	page 118
Analog Input Trigger Source Query	AD:TRIGGer[:SOURce]?	page 121
Analog Input Trigger Threshold Configuration	AD:TRIGGer:LEVel[:AIN1]	page 123
Analog Input Trigger Threshold Query	AD:TRIGGer:LEVel[:AIN1]?	page 124
Fetch Analog Input Data	AD:FETCh?	page 125
Initiate Analog Input Operation	AD:INITiate	page 128

Abort Analog Input Operation

Description Stops the specified analog input operation on the DT8837 LXI instrument module, if it is in progress.

Syntax :AD:ABORt

Optional Keywords

Name: AD

Data Format: <CHARACTER PROGRAM DATA>

Description: Enumerated type that specifies whether to stop the operation on the analog input (AD) subsystem.

Note that if this keyword is omitted, both the analog input (AD) and analog output (DA) subsystems are stopped simultaneously when the **ABORt** command is executed.

Required Parameters None

Optional Parameters None

Response Data None

Notes Use the **AD:STATus?** command, described on [page 118](#), to determine the state of the analog input subsystem.

Example The following example configures a DT8837 LXI instrument module to update the analog input channel at 52.734 kHz when a software trigger is detected, and queries the A/D status bits:

```
> :AD:ENAB ON
> :AD:CLOC INT
> :AD:CLOC:FREQ MAX
> :AD:TRIG IMM
> :AD:STAT?
< 0
```

The analog input operation is then armed and triggered, and the A/D status bits are queried to determine the status of the operation:

```
> :AD:ARM
> :AD:INIT
> :AD:STAT?
< 7
```

The analog input operation is then stopped and the A/D status bits are queried to determine the status of the operation:

```
> :AD:ABOR
> :AD:STAT?
< 4
```

See Also **AD:ARM**, described on [page 103](#)

AD:INITiate, described on [page 128](#)

AD:STATus?, described on [page 118](#)

Arm Analog Input Operation

Description Arms the analog input subsystem, which writes all the configuration settings to the instrument module.

Syntax :AD:ARM

Optional Keywords

Name: AD

Data Format: <CHARACTER PROGRAM DATA>

Description: Enumerated type that specifies whether to arm the analog input (AD) subsystem.

If this keyword is omitted, both the analog input (AD) and analog output (DA) subsystems are armed simultaneously when the **ARM** command is executed.

Required Parameters None

Optional Parameters None

Response Data None

Notes If you want to perform analog input and analog output operations simultaneously, call **ARM** rather than **AD:ARM**, or explicitly call **DA:ARM** before calling **AD:ARM**. Once the analog input subsystem is armed, the analog output subsystem cannot be armed.

If you specified the trigger source for the analog input subsystem as **IMMediate**, you must call the **AD:INITiate** command after arming the operation to start the operation.

Use the **AD:STATus?** command, described on [page 118](#), to determine the state of the analog input subsystem.

Example The following example configures a DT8837 LXI instrument module to sample analog input channels 1 to 3 at 52.734 kHz when a software trigger is detected, and queries the A/D status bits to determine the status of the operation:

```
> :AD:ENAB ON (@1:3)
> :AD:CLOC INT
> :AD:CLOC:FREQ MAX
> :AD:TRIG IMM
> :AD:STAT?
< 0
```

The following example arms the analog input subsystem, and queries the A/D status bits to determine the status of the operation:

```
> :AD:ARM
> :AD:STAT?
< 2
```

A value of 2 indicates that the A/D subsystem was armed.

Example (cont.) The analog input operation is then started, and the the A/D status bits are queried to determine the status of the operation:

```
> :AD:INIT
> :AD:STAT?
< 7
```

A value of 7 indicates that the A/D subsystem is armed, triggered, and active.

See Also **AD:INITiate**, described on [page 128](#)

STATus:OPERation:CONDition?, described on [page 84](#)

ADC Synchronization Source Query

Description Returns the source of the ADC synchronization pulse.

Syntax :AD:SYNc:SOURce?

Required Parameters None

Optional Parameters None

Response Data {INTernal|LXI6}

Name: INTernal|LXI6

Data Format: <CHARACTER RESPONSE DATA>

Description: The source that is currently configured as the ADC synchronization pulse, where INTernal is the internal sync on the DT8837 instrument module and LXI6 is the synchronization signal from the Trigger Bus.

Example The following example returns the currently configured sync source; in this example, the internal sync source is used:

```
> :AD:SYN:SOUR?
< INTernal
```

See Also **AD:CLOCK:SOURce**, described on [page 109](#)

Analog Input Buffer Mode Configuration

Description Configures the wrap mode of the input buffer.

Syntax :AD:BUFFer:MODE {WRAP|NOWRAP|DEFAULT}

Required Parameters

Name: WRAP|NOWRAP|DEFAULT

Data Format: <CHARACTER PROGRAM DATA>

Description: Specifies the wrap mode for the specified buffer. The available choices are as follows:

- **WRAP mode** – After the scan data has been stored in the buffer, the latest scan overwrites the oldest one in the input buffer.

Description (cont.):	Acquisition continues until it is explicitly stopped with AD:ABORT , described on page 102 . <ul style="list-style-type: none"> • NOWRAP mode – After the scan data has been stored in the input buffer, acquisition stops automatically. • DEFault – The default setting is WRAP mode.
Optional Parameters	None
Response Data	None
Notes	Refer to <i>1999 SCPI Command Reference</i> , section 23.6.1, for more information.
Possible Errors	-284 "Program currently running" If you receive this error, call the AD:ABORT command to stop the analog input operation before reconfiguring the analog input subsystem.
Example	The following example configures the input buffer for continuous wrap mode (WRAP): <pre>> :AD:BUFF:MODE WRA</pre>
See Also	AD:BUFFer:MODE? , described on page 104

Analog Input Buffer Mode Query

Description	Returns the currently configured wrap mode for the input buffer.
Syntax	:AD:BUFFer:MODE?
Required Parameters	None
Optional Parameters	None
Response Data	{WRAP NOWRAP}
Name:	WRAP NOWRAP
Data Format:	<CHARACTER RESPONSE DATA>
Description:	The currently configured wrap mode. Possible values are as follows: <ul style="list-style-type: none"> • WRAP – After the scan data has been stored in the buffer, the latest scan overwrites the oldest one in the input buffer and acquisition continues until it is explicitly stopped with AD:ABORT, described on page 102. If DEFault was configured, WRAP is returned. • NOWRAP – After the scan data has been stored in the input buffer, acquisition stops automatically.
Notes	Refer to <i>1999 SCPI Command Reference</i> , section 23.6.1, for more information.

Example The following example returns the currently configured wrap mode of the input buffer; in this case, continuous wrap mode (WRAP) is used:

```
> :AD:BUFF:MODE?  
< WRAP
```

See Also AD:BUFFer:MODE, described on [page 104](#)

Analog Input Buffer Size Query

Description Returns the maximum number of scans that can be stored in the input buffer based on the number of enabled input channels and the input sampling rate.

Syntax :AD:BUFFer:SIZE[:SCANs]?

Required Parameters None

Optional Parameters None

Response Data <Number of scans>

Name: Number of scans

Data Format: <0+NR1>

Description: Indicates the maximum number of scans that can be stored in the input buffer based on the number of input channels and the input sampling rate.

Notes If the buffer wrap mode is NOWRAP, acquisition stops automatically after this number of scans is acquired.

If the buffer wrap mode is WRAP, after this number of scans is acquired, the oldest scan data is overwritten with the latest scan data as the buffer wraps.

The sampling rate determines DMA block size that is used for the scan data, where each block has a fixed number of header bytes that is not used to store scans. The DMA block size is smaller for the minimum sampling frequency, resulting in more space for the headers, and greater for the maximum sampling frequency.

Since the maximum input buffer size is approximately 8 MB and each sample is 4 bytes, you can store a maximum of approximately 2 Msamples in the input buffer. If you sample each input channel at the maximum input frequency (52.734 kHz), the input buffer will fill in approximately 5 seconds (52 ksamples/s per channel).

The maximum size of data transfer over Ethernet using the **AD:FETCh?** command is 32 kB. Since each sample is 4 bytes, the maximum number of samples per **AD:FETCh?** is 8 ksamples. Therefore, if you are using continuous wrap mode, ensure that you call **AD:FETCh?** in a tight loop to ensure that you retrieve all the samples in the input buffer before the buffer is overwritten.

Example The following example returns the number of scans that can be stored in the input buffer under various conditions:

```
> :AD:ENAB ON, (@1);:AD:BUFF:SIZE?
< 2096128
> :AD:ENAB ON, (@2);:AD:BUFF:SIZE?
< 1048064
> :AD:ENAB OFF, (@2);:AD:BUFF:SIZE?
< 2096128
> :AD:ENAB OFF;:AD:BUFF:SIZE
< 0
> :AD:ENAB ON;:AD:BUFF:SIZE?
< 524032
> :AD:CLCOK:FREQ MIN;:AD:BUFF:SIZE?
< 520192
> :SYST:ERR?
< 0,"No error"
> :AD:CLOCK:FREQ 10000;:AD:BUFF:SIZE?
< 524032
> :AD:CLOCK:FREQ 20000;:AD:BUFF:SIZE?
< 524032
> :AD:CLOCK:FREQ MAX;:AD:BUFF:SIZE?
< 524032
> :AD:CLOCK:FREQ MIN;:AD:BUFF:SIZE?
< 520192
> :AD:ENAB 0
> AD:BUFF:SIZE?
< 0
```

See Also AD:BUFFer:MODE, described on [page 104](#)

Analog Input Channel Enable

Description Enables or disables sampling on the specified analog input channels.

Syntax :AD:ENABLE <state> [,<source list>]

Required Parameters

Name: state

Data Format: <Boolean>

Description: Specifies whether to enable or disable sampling on the specified analog input channels.

To enable sampling, set this value to ON or 1. To disable sampling, set this value to OFF or 0.

Optional Parameters

Name:	source list
Data Format:	<channel_list>
Description:	Specifies the list of analog input channels to enable or disable. Analog input channel numbers range from 1 to 4. If <i>source list</i> is omitted, all analog input channels are configured, by default.

Response Data None

Possible Errors -284 "Program currently running"

If you receive this error, call the **AD:ABORT** command to stop the analog input operation before reconfiguring the analog input subsystem.

Example The following example enables sampling on analog input channels 1 and 2 and disables sampling on analog input channels 3 and 4:

```
>:AD:ENAB ON, (@1, 2)
>:AD:ENAB 0, (@3, 4)
```

See Also **AD:ENABLE?**, described on [page 108](#)

Analog Input Channel Enable Query

Description Returns whether sampling is enabled or disabled for the specified analog input channels.

Syntax :AD:ENABLE? [<source list>]

Required Parameters None

Optional Parameters

Name:	source list
Data Format:	<channel_list>
Description:	Specifies the list of analog input channels for which to return the enabled/disabled state. Analog input channel numbers range from 1 to 4. If <i>source list</i> is omitted, the configuration of all analog input channels is returned, by default.

Response Data <state>

Name:	state
Data Format:	<0+NR1>
Description:	Contains the current state for each specified channel, separated by commas, where 1 indicates that sampling is enabled and 0 indicates that sampling is disabled.

Example The following example returns whether sampling is enabled or disabled for all analog input channels on the DT8837 instrument module; in this example, analog input channels 1 and 2 are enabled and analog input channels 3 and 4 are disabled.

```
> :AD:ENAB?
< 1,1,0,0;
```

See Also AD:ENABle, described on [page 107](#)

Analog Input Clock Source Configuration

Description Specifies the source of the reference clock used for the ADC and DAC master clock generators on the DT8837 instrument module.

Syntax :AD:CLOCK:SOURce {INTernal|LXI7}

Optional Keywords

Name: AD

Data Format: <CHARACTER PROGRAM DATA>

Description: Optional enumerated type that specifies the analog input (AD) subsystem for which to configure the clock source.

On the DT8837 instrument module, the analog input and analog output subsystems share a single reference clock. Therefore, you can omit this keyword to affect the clock source for both subsystems.

Required Parameters

Name: INTernal|LXI7

Data Format: <CHARACTER PROGRAM DATA>

Description: Enumerated type that specifies the reference clock source to use, where INTernal is the internal reference clock on the DT8837 instrument module and LXI7 is line 7 of the Trigger Bus.

Optional Parameters None

Response Data None

Notes The specified reference clock source is used for pacing both analog input and analog output operations.

If the instrument module is the clock master, specify the reference clock source as INTernal.

If the instrument module is a slave and you want to synchronize the clock with the master using the Trigger Bus, specify the reference clock source as LXI7.

Possible Errors -284 "Program currently running"

If you receive this error, call the **AD:ABORt** command to stop the analog input operation before reconfiguring the analog input subsystem.

Example The following example specifies the clock source on the DT8837 instrument module as the internal reference clock, which automatically configures the sync source as internal:

```
> :CLOC:SOUR INT
```

See Also [AD:CLOCK:SOURce?](#), described on [page 110](#)
[AD:SYNc:SOURce?](#), described on [page 104](#)
[AD:CLOCK:FREQuency\[:CONFigure\]](#), described on [page 115](#)
[AD:CLOCK:FREQuency\[:CONFigure\]?](#), described on [page 116](#)
[WTB:LXI<n>\[OUTput\]:ENABle](#), described on [page 171](#)

Analog Input Clock Source Query

Description Returns the currently configured reference clock source used for the ADC and DAC master clock generators on the DT8837 instrument module.

Syntax : [AD:] CLOCk: SOURce?

Optional Keywords

Name: AD

Data Format: <CHARACTER PROGRAM DATA>

Optional enumerated type that specifies the analog input (AD) subsystem for which to return the clock source.

On the DT8837 instrument module, the analog input and analog output subsystems share a single reference clock. Therefore, you can omit this keyword to return the currently configured clock source for both subsystems.

Required Parameters None

Optional Parameters None

Response Data <INTernal|LXI7>

Name: INTernal|LXI7

Data Format: <CHARACTER RESPONSE DATA>

Description: Enumerated type that represents the reference clock source that is currently configured, where INTernal is the internal reference clock on the DT8837 and LXI7 is the external reference clock from the Trigger Bus.

Example The following example returns the currently configured reference clock source on the DT8837 instrument module; in this case, the internal reference clock is configured:

```
> :CLOC:SOUR?  
< INTernal
```

See Also AD:CLOCK:SOURce?, described on [page 109](#)
 AD:SYNc:SOURce?, described on [page 104](#)
 AD:CLOCK:FREQuency[:CONFigure], described on [page 115](#)
 WTB:LXI<n>[:OUTput]:ENABle, described on [page 171](#)

Analog Input Coupling Configuration

Description Configures the coupling type for the specified analog input channels.

Syntax :AD:COUPling[:CONFigure] {AC|DC} [, <source list>]

Required Parameters

Name: AC|DC

Data Format: <CHARACTER PROGRAM DATA>

Description: Enumerated type that specifies either AC or DC coupling.

Optional Parameters

Name: source list

Data Format: <channel_list>

Description: Specifies the list of analog input channels to which to apply the coupling type. Analog input channel numbers range from 1 to 4.
 If *source list* is omitted, the coupling type is applied to all analog input channels, by default.

Response Data None

Possible Errors -284 "Program currently running"

If you receive this error, call the **AD:ABORt** command to stop the analog input operation before reconfiguring the analog input subsystem.

Example The following example configures analog input channels 1 and 2 for AC coupling and analog input channels 3 and 4 for DC coupling:

```
>:AD:COUP AC, (@1, 2)
```

```
>:AD:COUP DC, (@3, 4)
```

See Also AD:COUPling[:CONFigure]?, described on [page 111](#)

Analog Input Coupling Query

Description Returns the currently configured coupling type for the specified analog input channels.

Syntax :AD:COUPling[:CONFigure]? [<source list>]

Required Parameters None

Optional Parameters

Name:	source list
Data Format:	<channel_list>
Description:	Specifies the list of analog input channels for which to return the coupling type configuration. Analog input channel numbers range from 1 to 4. If <i>source list</i> is omitted, the coupling type configuration is returned for all analog input channels, by default.

Response Data <coupling type>

Name:	coupling type
Data Format:	<CHARACTER RESPONSE DATA>
Description:	Contains the coupling type (AC or DC) for each specified channel, separated by commas.

Example The following example returns the coupling type for all analog input channels on the DT8837 instrument module; in this case channels 1 and 2 use AC coupling and channels 3 and 4 use DC coupling:

```
> :AD:COUP?  
< AC,AC,DC,DC
```

See Also AD:COUPling[:CONFigure], described on [page 111](#)

Analog Input Current Source Enable

Description Enables or disables use of the 4 mA current source for the specified analog input channels.

Syntax :AD:BIAS:CURRent:ENABle <state> [,<source list>]

Required Parameters

Name:	state
Data Format:	<Boolean>
Description:	Specifies the state (enabled or disabled) of the 4 mA current source for the specified analog input channels. To enable the current source, set this value to ON or 1. To disable the current source, set this value to OFF or 0.

Optional Parameters

Name:	source list
Data Format:	<channel_list>
Description:	Specifies the list of analog input channels to which to apply the current source configuration. Analog input channel numbers range from 1 to 4. If <i>source list</i> is omitted, the current source configuration is applied to all analog input channels, by default.

Response Data	None
Possible Errors	<p>-284 "Program currently running"</p> <p>If you receive this error, call the AD:ABORT command to stop the analog input operation before reconfiguring the analog input subsystem.</p>
Example	<p>The following example enables the 4 mA current source for analog input channels 1 and 2 and disables the 4 mA current source for analog input channels 3 and 4:</p> <pre>:AD:BIAS:CURREN:ENAB 1, (@1,2) :AD:BIAS:CURREN:ENAB OFF, (@3,4)</pre>
See Also	AD:BIAS:CURREN:ENABLE? , described on page 113

Analog Input Current Source Enable Query

Description	Returns whether the current source is enabled or disabled for the specified analog input channels.
Syntax	<code>:AD:BIAS:CURREN:ENABLE? [<source list>]</code>
Required Parameters	None
Optional Parameters	
Name:	source list
Data Format:	<channel_list>
Description:	Specifies the list of analog input channels for which to return the current source configuration. Analog input channel numbers range from 1 to 4. If <i>source list</i> is omitted, the current source configuration is returned for all analog input channels, by default.
Response Data	<current source>
Name:	current source
Data Format:	<0+NR1>
Description:	Contains the current source state for each specified channel, separated by commas, where 1 indicates that the current source is enabled and 0 indicates that the current source is disabled.
Example	<p>The following example returns the current source configuration for all analog input channels on the DT8837 instrument module; in this case, the current source is enabled for channels 1 and 2 and disabled for channels 3 and 4:</p> <pre>>:AD:BIAS:CURREN:ENAB? < 1,1,0,0</pre>
See Also	AD:BIAS:CURREN:ENABLE , described on page 112

Analog Input Gain Configuration

Description Configures the gain setting for the specified analog input channels.

Syntax `:AD:GAIN[:CONFigure] <numeric value> [,<source list>]`

Required Parameters

Name: numeric value

Data Format: <+NR1>

Description: Specifies a gain value of 1 or 10.

Optional Parameters

Name: source list

Data Format: <channel_list>

Description: Specifies the list of analog input channels to which to apply the gain value. Analog input channel numbers range from 1 to 4. If *source list* is omitted, the gain value is applied to all analog input channels, by default.

Response Data None

Possible Errors -284 "Program currently running"

If you receive this error, call the **AD:ABORT** command to stop the analog input operation before reconfiguring the analog input subsystem.

Example The following example configures analog input channels 1 and 2 for a gain of 1 and analog input channels 3 and 4 for a gain of 10:

```
:AD:GAIN 1, (@1, 2)
:AD:GAIN 10, (@3, 4)
```

See Also `AD:GAIN[:CONFigure]?`, described on [page 114](#)

Analog Input Gain Query

Description Returns the currently configured gain value for the specified analog input channels.

Syntax `:AD:GAIN[:CONFigure]? [<source list>]`

Required Parameters None

Optional Parameters

Name: source list

Data Format: <channel_list>

Description: Specifies the list of analog input channels for which to return the gain configuration. Analog input channel numbers range from 1 to 4. If *source list* is omitted, the gain configuration is returned for all analog input channels, by default.

Response Data	<gain>
Name:	gain
Data Format:	<+NR1>
Description:	Contains the gain value (1 or 10) for each specified channel, separated by commas.
Example	<p>The following example returns the gain configuration for all analog input channels on the DT8837 instrument module; in this case, channel 1 and 2 are configured for a gain of 1, and channels 3 and 4 are configured for a gain of 10:</p> <pre>>:AD:GAIN? < 1,1,10,10</pre>
See Also	AD:GAIN[:CONFigure], described on page 114

Analog Input Sampling Frequency Configuration

Description	Specifies the sampling frequency for pacing analog input operations on the DT8837 instrument module.
Syntax	:AD:CLOCK:FREQuency[:CONFigure] <freq MINimum MAXimum>

Required Parameters

Name:	freq MINimum MAXimum
Data Format:	<+NRf>
Description:	<p>Specifies the sampling frequency, in Hz, where <i>freq</i> represents a particular sampling frequency, <i>MINimum</i> represents the minimum sampling frequency, and <i>MAXimum</i> represents the maximum sampling frequency for the specified analog input subsystem.</p> <p>For analog input operations, the minimum sampling frequency is 195 Hz and the maximum sampling frequency is 52734.375 Hz (the default value).</p>

Optional Parameters	None
----------------------------	------

Response Data	None
----------------------	------

Note If you are synchronizing the clocks of multiple instrument modules, it is important that you enable LXI7 as an output on the clock master and then configure the clock frequency of the slaves before configuring the clock frequency of the clock master.

It is also recommended that you choose the same sampling frequency for all instrument modules to avoid problems. Note, however, that the analog input and analog output frequencies can be different from each other.

Note (cont.) If you later change the sampling frequency of one or more slave instrument modules, you must reconfigure the clock frequency of the clock master, even if its parameters have not changed.

Refer to [page 45](#) for more information on setting up the clocks.

Possible Errors -131 "Invalid suffix"

If you receive this error, ensure that the sampling frequency is entered in <+NRf> format.

-222 "Data out of range"

If you receive this error, the sampling frequency that was entered was outside the valid range. Enter a valid sampling frequency.

-284 "Program currently running"

If you receive this error, call the **AD:ABORT** command to stop the analog input operation before reconfiguring the analog input subsystem.

Example The following example configures the maximum sampling frequency (52734.375 Hz) for the analog input subsystem on a DT8837:

```
>:AD:CLOC:FREQ MAX
>:AD:CLOC:FREQ?
< 52734.375
>:SYST:ERR?
< 0, "No error"
```

See Also AD:CLOCK:SOURce, described on [page 109](#)

AD:CLOCK:SOURce?, described on [page 110](#)

AD:SYNc:SOURce?, described on [page 104](#)

AD:CLOCK:FREQuency[:CONFigure]?, described on [page 116](#)

WTB:LXI<n>[:OUTput]:ENABle, described on [page 171](#)

Analog Input Sampling Frequency Query

Description Returns the sampling frequency that is currently configured for pacing analog input operations on the DT8837 instrument module.

Syntax :AD:CLOCK:FREQuency[:CONFigure]?

Required Parameters None

Optional Parameters None

Response Data <sampling frequency>

Name:	sampling frequency
Data Format:	<+NRf>
Description:	The currently configured sampling frequency, in Hz, for the specified analog input subsystem.
Example	The following example returns the currently configured sampling frequency, in Hz, for the analog input subsystem; in this case, the sampling frequency is 52734.375 Hz: <pre>> :AD:CLOC:FREQ? < 52734.375</pre>
See Also	AD:CLOCK:SOURce, described on page 109 AD:CLOCK:SOURce?, described on page 110 AD:CLOCK:FREQuency[:CONFigure], described on page 115

Analog Input Scan Status Query

Description	Returns the indices of the chronologically oldest and most recent scan records in the input buffer on the instrument module.
Syntax	:AD:STATus:SCAN?
Parameters	None
Response Data	<StartingIndex>,<EndingIndex>
Name:	StartingIndex
Data Format:	<0+NR1>
Description:	A decimal number that represents the index of the chronologically oldest scan record available in the circular buffer on the instrument module.
Name:	EndingIndex
Data Format:	<0+NR1>
Description:	A decimal number that represents the index of the most recent scan record available in the circular buffer on the instrument module.
Notes	If the input buffer is empty (because a scan has not been started or started and stopped), both <i>StartingIndex</i> and <i>EndingIndex</i> will be 0. Otherwise, these values will be non-zero.
Example	The follow example shows the <i>StartingIndex</i> (1001) and <i>EndingIndex</i> (1050) when the input buffer consists of scan records 1001 to 1050: <pre>> :AD:STAT:SCA? < 1001,1050</pre>
See Also	AD:FETCh?, described on page 125

Analog Input Status Bits Query

Description	Returns the status bits for the analog input subsystem.
Syntax	:AD:STATus?
Required Parameters	None
Optional Parameters	None
Response Data	<status bits>
Name:	status bits
Data Format:	<0+NR1>
Description:	<p>The value, in the range of 0 to 255, of the status bits for the analog input subsystem. The status bits are defined as follows:</p> <ul style="list-style-type: none">• Bit 0 – The value of this bit is 1 if the analog input subsystem is active or 0 if the analog input subsystem is inactive.• Bit 1 – The value of this bit is 1 if the analog input subsystem is armed or 0 if the analog input subsystem is not armed.• Bit 2 – The value of this bit is 1 if the analog input subsystem is triggered or 0 if the analog input subsystem is not triggered.• Bit 3 – The value of this bit is 1 if AD SYNC was detected or 0 if AD SYNC was not detected.• Bit 4 – The value of this bit is 1 if the AD FIFO overflowed or 0 if the AD FIFO did not overflow.
Notes	Refer to <i>IEEE 488.2-1992</i> , section 9.1, for more information.
Example	<p>The following example returns the status bits for the analog input subsystem:</p> <pre>> :AD:STAT? < 4;</pre> <p>In this example, the status bit value is 4, indicating that the analog input subsystem was triggered.</p>

Analog Input Trigger Source Configuration

Description	Specifies the trigger source that starts the analog input operation on the DT8837 instrument module.
Syntax	:AD:TRIGger[:SOURce] {IMMediate EXTernal AIN1 LXI0 LXI1 LXI2 LXI3 LXI4 LXI5 LAN0 LAN1 LAN2 LAN3 LAN4 LAN5 LAN6 LAN7 DEFault}

Required Parameters

Name:	IMMediate EXTernal AIN1 LXI0 LXI1 LXI2 LXI3 LXI4 LXI5 LAN0 LAN1 LAN2 LAN3 LAN4 LAN5 LAN6 LAN7 DEFault <CHARACTER PROGRAM DATA>
Data Format:	Enumerated type that specifies the trigger source to use. The
Description:	<p>following trigger sources are supported:</p> <ul style="list-style-type: none"> • IMMediate – A software trigger. Once you arm the subsystem with the AD:ARM command, the software trigger event occurs when the AD:INITiate command is executed (the computer issues a write to the instrument module to begin conversions). • EXTernal– An external digital (TTL) trigger event. Once you arm the subsystem with the AD:ARM command, this trigger event occurs when the instrument module detects a rising-edge transition on the Trigger In signal connected to the instrument module. • AIN1 – An analog threshold trigger event. Once you arm the subsystem with the AD:ARM command, this trigger event occurs when the signal attached to analog input channel 1 rises above a user-specified threshold value from 0.2 V to 9.8 V with 0.1 V of hysteresis. You specify the trigger threshold using the AD:TRIGger:LEVel[:AIN1] command, described on page 123. • LXI0 – Once you arm the subsystem with the AD:ARM command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 0 of the Trigger Bus. • LXI1 – Once you arm the subsystem with the AD:ARM command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 1 of the Trigger Bus. • LXI2 – Once you arm the subsystem with the AD:ARM command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 2 of the Trigger Bus. • LXI3 – Once you arm the subsystem with the AD:ARM command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 3 of the Trigger Bus. • LXI4 – Once you arm the subsystem with the AD:ARM command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 4 of the Trigger Bus. • LXI5 – Once you arm the subsystem with the AD:ARM command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 5 of the Trigger Bus.

Description (cont.):	<ul style="list-style-type: none">• LAN0 – Once you arm the subsystem with the AD:ARM command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 0.• LAN1 – Once you arm the subsystem with the AD:ARM command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 1.• LAN2 – Once you arm the subsystem with the AD:ARM command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 2.• LAN3 – Once you arm the subsystem with the AD:ARM command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 3.• LAN4 – Once you arm the subsystem with the AD:ARM command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 4.• LAN5 – Once you arm the subsystem with the AD:ARM command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 5.• LAN6 – Once you arm the subsystem with the AD:ARM command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 6.• LAN7 – Once you arm the subsystem with the AD:ARM command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 7.• DEFault – The default setting, which is IMMEDIATE.
Optional Parameters	None
Response Data	None
Notes	<p>The trigger input signal from the Trigger Bus is propagated by the trigger source of the analog input subsystem on the trigger master. Both the analog input and analog output subsystems can be triggered by a Trigger Bus signal, if and only if, the same line (LXI0 to LXI5) is used as the trigger. The order in which to set up the Trigger Bus is important; refer to page 51 for more information about using the Trigger Bus to connect and trigger multiple instrument modules.</p> <p>The analog input and analog output subsystems can be triggered by either the same LAN event or by two different LAN events. Configure the LXI-defined LAN event using the commands and queries under the EVENT subsystem, described on page 174. Refer to page 49 for more information about using LXI LAN events to trigger multiple instrument modules over the network.</p>

Possible Errors	<p>-284 "Program currently running"</p> <p>If you receive this error, call the AD:ABORT command to stop the analog input operation before reconfiguring the analog input subsystem.</p>
Example	<p>The following example configures an external digital trigger source to start an analog input operation on the DT8837 instrument module:</p> <pre>>:AD:TRIG EXT >:AD:ARM</pre>
See Also	<p>AD:ARM, described on page 103</p> <p>AD:INITiate, described on page 128</p> <p>AD:TRIGger[:SOURce]?, described on page 121</p> <p>AD:TRIGger:LEVel[:AIN1], described on page 123</p> <p>AD:TRIGger:LEVel[:AIN1]?, described on page 124</p> <p>AD:CLOCK:FREQuency[:CONFigure], described on page 115</p> <p>AD:CLOCK:FREQuency[:CONFigure]?, described on page 116</p>

Analog Input Trigger Source Query

Description	Returns the trigger source that is currently configured for starting analog input operations on the DT8837 instrument module.
Syntax	:AD:TRIGger[:SOURce]?
Required Parameters	None
Optional Parameters	None
Response Data	<p>{IMMediate EXTernal AIN1 LXI0 LXI1 LXI2 LXI3 LXI4 LXI5 LAN0 LAN1 LAN2 LAN3 LAN4 LAN5 LAN6 LAN7}</p> <p>Name: IMMediate EXTernal AIN1 LXI0 LXI1 LXI2 LXI3 LXI4 LXI5 LAN0 LAN1 LAN2 LAN3 LAN4 LAN5 LAN6 LAN7</p>
Data Format:	<CHARACTER RESPONSE DATA>
Description:	<p>Enumerated type that specifies the trigger source to use. The following trigger sources are supported:</p> <ul style="list-style-type: none"> • IMMediate – A software trigger. Once you arm the subsystem with the AD:ARM command, the software trigger event occurs when the AD:INITiate command is executed (the computer issues a write to the instrument module to begin conversions). • EXTernal – An external digital (TTL) trigger event. Once you arm the subsystem with the AD:ARM command, this trigger event occurs when the instrument module detects a rising-edge transition on the Trigger In signal connected to the instrument module.

Description (cont.):

- **AIN1** – An analog threshold trigger event. Once you arm the subsystem with the **AD:ARM** command, this trigger event occurs when the signal attached to analog input channel 1 rises above a user-specified threshold value from 0.2 V to 9.8 V with 0.1 V of hysteresis. You specify the trigger threshold using the **AD:TRIGger:LEVel[:AIN1]** command, described on [page 123](#).
- **LXI0** – Once you arm the subsystem with the **AD:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 0 of the Trigger Bus.
- **LXI1** – Once you arm the subsystem with the **AD:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 1 of the Trigger Bus.
- **LXI2** – Once you arm the subsystem with the **AD:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 2 of the Trigger Bus.
- **LXI3** – Once you arm the subsystem with the **AD:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 3 of the Trigger Bus.
- **LXI4** – Once you arm the subsystem with the **AD:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 4 of the Trigger Bus.
- **LXI5** – Once you arm the subsystem with the **AD:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 5 of the Trigger Bus.
- **LAN0** – Once you arm the subsystem with the **AD:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 0.
- **LAN1** – Once you arm the subsystem with the **AD:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 1.
- **LAN2** – Once you arm the subsystem with the **AD:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 2.
- **LAN3** – Once you arm the subsystem with the **AD:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 3.
- **LAN4** – Once you arm the subsystem with the **AD:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 4.

- Description (cont.):
- LAN5 – Once you arm the subsystem with the **AD:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 5.
 - LAN6 – Once you arm the subsystem with the **AD:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 6.
 - LAN7 – Once you arm the subsystem with the **AD:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 7.

Example The following example returns the currently configured trigger source for analog input operations; in this example, the external digital trigger is used:

```
>:AD:TRIG?
< EXternal
```

See Also **AD:TRIGger[:SOURce]**, described on [page 118](#)
AD:TRIGger:LEVel[:AIN1], described on [page 123](#)
AD:TRIGger:LEVel[:AIN1]?, described on [page 124](#)
AD:ARM, described on [page 103](#)
AD:INITiate, described on [page 128](#)
AD:CLOCK:FREQuency[:CONFigure], described on [page 115](#)
AD:CLOCK:FREQuency[:CONFigure]?, described on [page 116](#)

Analog Input Trigger Threshold (AIN1) Configuration

Description For the analog threshold trigger source, specifies the threshold level on analog input channel 1. An analog threshold trigger event occurs when the signal attached to analog input channel 1 rises above this threshold level.

Syntax **:AD:TRIGger:LEVel[:AIN1]** <numeric value|MINimum|MAXimum>

Required Parameters

Name: numeric value | MINimum | MAXimum

Data Format: <+NR1> | CHARACTER PROGRAM DATA | CHARACTER PROGRAM DATA

Description: The threshold level, where *numeric value* represents a particular threshold value ranging from 200 to 9800 mV, *MINimum* represents the minimum threshold value (200 mV), and *MAXimum* represents the maximum threshold value (9800 mV).

Optional Parameters None

Response Data None

Notes This command applies only to the analog threshold trigger source (AIN1), specified with the **AD:TRIGger[:SOURce]** command, described on [page 118](#), or the **DA:TRIGger[:SOURce]** command, described on [page 161](#).

The threshold value has 0.1 V of hysteresis.

Possible Errors -120 "Numeric data error"

If you receive this error, enter a valid threshold value between 200 and 9800 mV.

-284 "Program currently running"

If you receive this error, call the **AD:ABORT** command to stop the analog input operation before reconfiguring the analog input subsystem.

Example The following example configures a threshold level of 750 mV on analog input channel 1:

```
>:AD:TRIG AIN1
>:AD:TRIG:LEV 750
>:AD:ARM
```

See Also **AD:ARM** described on [page 103](#)

AD:TRIGger[:SOURce], described on [page 118](#)

DA:TRIGger[:SOURce], described on [page 161](#)

AD:TRIGger:LEVel[:AIN1]?, described on [page 124](#)

Analog Input Trigger Threshold (AIN1) Query

Description For the analog threshold trigger source, returns the threshold level that is currently configured for analog input channel 1.

Syntax **:AD:TRIGger:LEVel[:AIN1]?**

Required Parameters None

Optional Parameters None

Response Data <threshold level>

Name: threshold level

Data Format: <+NR1>

Description: The threshold level, ranging from 200 to 9800 mV.

Notes This command applies only to the analog threshold trigger source (AIN1), specified with the **AD:TRIGger[:SOURce]** command, described on [page 118](#), and the **DA:TRIGger[:SOURce]** command, described on [page 161](#).

The threshold value has 0.1 V of hysteresis.

Example The following example returns the currently configured threshold level for the analog threshold trigger; in this example, the threshold level is 750 mV:

```
> :AD:TRIG:LEV?
< 750
```

See Also AD:ARM described on [page 103](#)
 AD:TRIGger[:SOURce], described on [page 118](#)
 DA:TRIGger[:SOURce], described on [page 161](#)
 AD:TRIGger:LEVel, described on [page 123](#)

Fetch Analog Input Data

Description For scans that were started using the AD:INITiate command, described on [page 128](#), returns a number of time-stamped, sequenced measurements from the input buffer on a DT8837 LXI instrument module, in the form of scan records.

Syntax :AD:FETCh? <index> [, <number of scans>]

Required Parameters

Name:	index
Data Format:	<0+NR1>
Description:	<p>Specifies the index (the offset in the input buffer) from which to retrieve scan data.</p> <p>To read from the first location in the buffer, specify 0 for <i>index</i>.</p> <p>To read a sequence of measurements that chronologically follow the sequence of measurements from an earlier AD:FETCh?, specify an <i>index</i> that is one greater than the number of the last scan from the previous AD:FETCh?.</p> <p>If only a subset of the scans that you requested are available (either because the scan is not yet complete or the scan location in the input buffer was overwritten with a later scan, the DT8837 LXI instrument module returns the scans that exist between the starting index (<i>index</i>) and the ending index (<i>index+number of scans</i>).</p> <p>For example, assume that you want to read 10 scans starting at index 5, but the scans at indices 5 through 8 have been overwritten. In this case, AD:FETCh? 5, 10 returns only scans 9 through 15.</p> <p>If no scans exist between the starting index and the ending index, an empty scan record is returned.</p>

Optional Parameters

Name:	number of scans
Data Format:	<+NR1>
Description:	<p>Specifies the number of scans to retrieve from the input buffer, starting with the scan number specified by <i>index</i>.</p> <p>If this parameter is either omitted, a fixed number of scans is returned. Since the size of the response in <DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA> is limited to 2 KB, the maximum number of scan records that AD:FETCh? can return is limited to the integral number of scan records that can fit in the input buffer.</p>

Response Data <Header><Scan record><Scan record>...<Scan record>

Response Data Format <DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA>

Name:	Header
Description:	<p>The header contains the following 4-byte fields:</p> <ul style="list-style-type: none">• Index – Index of the first scan record in this block.• Number of scans – Number of scan records in this block.• Samples per scan – Number of samples per enabled channel or scan record.• Time stamp seconds – The time stamp of the first scan record in the block, in seconds.• Time stamp nanoseconds – The time stamp of the first scan record in the block, in nanoseconds.
Name:	Scan record
Description:	<p>Consists of 4-byte "raw values" that represent the samples from the enabled input channels. If all the input channels are enabled, samples are returned in the following order:</p> <ul style="list-style-type: none">• Chan 1 – analog channel 1• Chan 2 – analog channel 2• Chan 3 – analog channel 3• Chan 4 – analog channel 4• Chan 5 – tachometer• Chan 6 – counter/timer 1• Chan 7 – counter/timer 2• Chan 8 – analog output channel readback

Notes Since the maximum input buffer size is approximately 8 MB and each sample is 4 bytes, you can store a maximum of approximately 2 Msamples in the input buffer. If you sample each input channel at the maximum input frequency (52.734 kHz), the input buffer will fill in approximately 5 seconds (52 ksamples/s per channel).

The maximum size of data transfer over Ethernet using the **AD:FETCh?** command is 32 kB. Since each sample is 4 bytes, the maximum number of samples per **AD:FETCh?** is 8 ksamples. Therefore, if you are using continuous wrap mode, ensure that you call **AD:FETCh?** in a tight loop to ensure that you retrieve all the samples in the input buffer before the buffer is overwritten.

While a client can repeatedly issue a **AD:FETCh?** request, be aware that the network traffic is high and that the client has to determine if there is any chronological overlap between responses.

When an error is encountered, an error string is appended to the error queue, and bit 2 (EAV) of the Status Byte register is set.

If bit 5 (CME) of the Standard Event Status Enable register is enabled, a command error sets bit 5 (CME) of Event Status Register.

If bit 4 (E) of the Standard Event Status Enable register is enabled, an Execution Error sets bit 4 (E) of the Standard Event Status register.

The summary of the Standard Event Status register is available in bit 5 (ESB) of the Status Byte register.

For more information on measurements, see *1999 SCPI Command Reference*, section 3.1 and 3.4. For more information on <DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA>, see *IEEE Std 488.2-1992*, section 8.7.9.

The IEEE488.2 standard defines that binary formatted data shall have a prefix that begins with #xyyyyyyyy format. The *x* is always a 1-digit ASCII numeric that specifies the "number of digits" for the *y* part; *y* has multiple digits (specified by *x*) of decimal expression that specify the number of bytes that follow.

Example The following example returns two scan records, starting with index 0; in this example, analog input channel 1 was enabled:

```
> :AD:FETC? 0,2
< #6000028<<x00><<x00><<x00><<x02><<x00>
<<x00><<x00><<x04><<x00><<x00><<x00>
<<x01>J???<<x00><<x00><<x00><<x00>
<<x00>?32<<x00>?2q
```

Looking at the response, you can see the following:

- # is the header
- 6 (after #) specifies the number of characters that will follow
- 000028 are the six characters (after 6) that specify how many bytes (28, in this case) that will follow
- The rest of the response includes the actual data in binary format

See Also **AD:INITiate**, described on [page 128](#)
AD:ABORt, described on [page 102](#)

Initiate Analog Input Operation

Description If the configured trigger source for the analog input subsystem is IMMEDIATE, starts the analog input operation immediately once the subsystem has been armed with the **AD:ARM** command. For all other trigger sources, this command has no effect.

Syntax :AD:INITiate

Optional Keywords

Name: AD

Data Format: <CHARACTER PROGRAM DATA>

Description: Enumerated type that specifies whether to start the analog input (AD) subsystem.

If this keyword is omitted, both the analog input (AD) and analog output (DA) subsystems are started simultaneously when the **INITiate** command is executed.

Required Parameters None

Optional Parameters None

Response Data None

Notes An Execution Error is generated by this command if one of the following conditions occurs:

- The analog input operation is already running
- No input channels are enabled
- The analog input sample frequency is invalid

If bit 4 (E) of the Standard Event Status Enable register is enabled, an Execution Error sets bit 4 (E) of the Standard Event Status register.

To determine if an Execution Error occurred, query bit 4 of the Standard Event Status register using the ***ESR?** command, described on [page 71](#).

Use the **AD:STATus?** command, described on [page 118](#), to determine the state of the analog input subsystem.

To read measurements, use the **AD:FETCh?** command, described on [page 125](#).

Example The following example configures a DT8837 LXI instrument module to sample analog input channels 1 to 3 at 52.734 kHz when a software trigger is detected, and queries the A/D status bits to determine the status of the operation:

```
> :AD:ENAB ON (@1:3)
> :AD:CLOC INT
> :AD:CLOC:FREQ MAX
> :AD:TRIG IMM
> :AD:STAT?
< 0
```

The analog input operation is then armed and started, and the A/D status bits are queried to determine the status of the operation:

```
> :AD:ARM
> :AD:INIT
> :AD:STAT?
< 7
```

The analog input operation is then stopped and the the A/D status bits are queried to determine the status of the operation:

```
> :AD:ABOR
> :AD:STAT?
< 4
```

See Also **AD:TRIGger[:SOURce]**, described on [page 118](#)

AD:ARM, described on [page 103](#)

AD:ABORt, described on [page 102](#)

AD:STATus?, described on [page 118](#)

AD:FETCh?, described on [page 125](#)

TACHometer Subsystem Commands

The TACHometer subsystem is used to configure period measurement operations on the tachometer input of a DT8837 LXI instrument module. Once you enable a tachometer channel, you can read the channel as part of the analog input data stream using the AD subsystem commands, described on [page 101](#), and the **AD: FETCh?** command, described on [page 125](#).

[Table 10](#) lists the commands that apply to the TACHometer subsystem. This section describes each of these commands in detail.

Table 10: TACHometer Subsystem Commands

Type	Mnemonic	See
Tachometer Channel Enable	TACHometer:ENABLE	page 131
Tachometer Channel Enable Query	TACHometer:ENABLE?	page 131
Tachometer Period Type Configuration	TACHometer:PERiod[:CONFigure]	page 132
Tachometer Period Type Query	TACHometer:PERiod[:CONFigure]?	page 133
Tachometer Self Clear Flag Configuration	TACHometer:SCLR[:CONFigure]	page 133
Tachometer Self Clear Flag Query	TACHometer:SCLR[:CONFigure]?	page 134
Tachometer Stale Data Flag Configuration	TACHometer:SFLAG[:CONFigure]	page 135
Tachometer Stale Data Flag Query	TACHometer:SFLAG[:CONFigure]?	page 135

Tachometer Channel Enable

Description Enables or disables the ability to return the value of the tachometer input channel in the analog input data stream.

Syntax :TACHometer:ENABle <state>

Required Parameters

Name: state

Data Format: <Boolean>

Description: Specifies whether to enable or disable the ability to return the value of the tachometer input channel in the analog input data stream.

To enable this feature, set this value to ON or 1. To disable this feature, set this value to OFF or 0.

Optional Parameters None

Response Data None

Notes Once you enable a tachometer channel, you can read the channel as part of the analog input data stream using the AD subsystem commands, described on [page 101](#), and the **AD:FETCh?** command, described on [page 125](#).

Possible Errors -284 "Program currently running"

If you receive this error, call the **AD:ABORt** command to stop the analog input operation before reconfiguring the TACHometer subsystem.

Example The following example enables the ability to return the value of the tachometer input channel in the analog input data stream:

```
> :TACH:ENAB ON
```

See Also TACHometer:ENABle?, described on [page 131](#)

Tachometer Channel Enable Query

Description Returns whether reading the value of the tachometer input channel in the analog input data stream is enabled or disabled.

Syntax :TACHometer:ENABle?

Required Parameters None

Optional Parameters None

Response Data <state>

Name:	state
Data Format:	<0+NR1>
Description:	Contains the current state of the tachometer input channel, where 1 indicates that the tachometer input channel is enabled and 0 indicates that the tachometer input channel is disabled.
Notes	Once you enable a tachometer channel, you can read the channel as part of the analog input data stream using the AD subsystem commands, described on page 101 , and the AD:FETCh? command, described on page 125 .
Example	<p>The following example returns whether reading the value of the tachometer input channel in the analog input data stream is enabled or disabled; in this example, this feature is enabled:</p> <pre>> :TACH:ENAB? < 1</pre>
See Also	TACHometer:ENABLE , described on page 131

Tachometer Period Type Configuration

Description	Configures the type of period to measure on the tachometer input signal.
Syntax	<code>:TACHometer:PERiod[:CONFigure]</code> <code>{NWIDth PWIDth DEFault}</code>
Required Parameters	
Name:	NWIDth PWIDth DEFault
Data Format:	<CHARACTER PROGRAM DATA>
Description:	<p>An enumerated type that specifies which period to measure on the tachometer input signal. The following period types are supported:</p> <ul style="list-style-type: none">• NWIDth – Measures from falling edge to falling edge of the tachometer input signal.• PWIDth – Measures from rising edge to rising edge of the tachometer input signal.• DEFault – The default setting, which is PWIDth.
Optional Parameters	None
Response Data	None
Possible Errors	<p>-284 "Program currently running"</p> <p>If you receive this error, call the AD:ABORt command to stop the analog input operation before reconfiguring the TACHometer subsystem.</p>

Example The following example configures the TACHometer subsystem to measure the period between rising edges of the tachometer input signal:

```
> :TACH:PER PWID
```

See Also TACHometer:PERiod[:CONFigure]?, described on [page 133](#)

Tachometer Period Type Query

Description Returns the currently configured period type for the tachometer input signal.

Syntax :TACHometer:PERiod[:CONFigure]?

Parameters None

Response Data {NWIDth|PWIDth}

Name: PWIDth|NWIDth

Data Format: <CHARACTER PROGRAM DATA>

Description: An enumerated type that specifies which period to measure on the tachometer input signal. The following period types are supported:

- NWIDth – Measures from falling edge to falling edge of the tachometer input signal.
- PWIDth – Measures from rising edge to rising edge of the tachometer input signal.

Example The following example returns the currently configured period type for the tachometer input signal. In this example, rising-to-rising edges are used for the period measurement:

```
> :TACH:PER?
< PWID
```

See Also TACHometer:PERiod[:CONFigure], described on [page 132](#)

Tachometer Self Clear Flag Configuration

Description Configures the Self Clear flag for the tachometer, which determines the value of the tachometer between period measurements.

Syntax :TACHometer:SCLR[:CONFigure] <self clear>

Required Parameters

Name:	self clear
Data Format:	<Boolean>
Description:	<p>Specifies whether the value read between measurements is cleared or retained.</p> <p>To clear the value to zero between measurements, set <i>self clear</i> to 1.</p> <p>To retain the previous measurement value until the next measurement is complete, set <i>self clear</i> to 0.</p>
Optional Parameters	None
Response Data	None
Possible Errors	<p>-284 "Program currently running"</p> <p>If you receive this error, call the AD:ABORT command to stop the analog input operation before reconfiguring the TACHometer subsystem.</p>
Example	<p>The following example configures the tachometer to clear the value to 0 between measurements:</p> <pre>>:TACH:SCLR 1</pre>
See Also	TACHometer:SCLR[:CONFigure]? , described on page 134

Tachometer Self Clear Flag Query

Description	Returns the value of the Self Clear flag for the tachometer, which determines the value of the tachometer between period measurements.
Syntax	:TACHometer:SCLR[:CONFigure]?
Parameters	None
Response Data	<self clear>
Name:	self clear
Data Format:	<Boolean>
Description:	Indicates whether the value read between measurements is cleared (1) or retained (0).
Example	<p>The following example returns the value of the Self Clear flag for the TACHometer subsystem. In this example, the value of the tachometer is cleared to 0 between measurements:</p> <pre>>:TACH:SCLR? < 1</pre>
See Also	TACHometer:SCLR[:CONFigure] , described on page 133

Tachometer Stale Value Flag Configuration

Description	Configures the value of the Stale Value flag, which determines whether the most-significant bit (MSB) of the period measurement value is used to indicate new or old data.
Syntax	<code>:TACHometer:SFLAG[:CONFigure] <stale value flag></code>
Required Parameters	
Name:	stale value flag
Data Format:	<Boolean>
Description:	<p>Determines whether to use the most significant bit (MSB) of the value to indicate new or old data.</p> <p>To use the MSB of the value to indicate whether the measurement is new or old, set <i>stale value flag</i> to active (1). When this flag is active, the MSB of the value is set to 0 to indicate new data or 1 to indicate old data. Reading the value before the measurement is complete returns an MSB of 1 to indicate old data.</p> <p>If you are not interested in whether the data is new or old, set <i>stale value flag</i> to 0. In this case, the MSB is always set to 0.</p>
Optional Parameters	None
Response Data	None
Possible Errors	<p>-284 "Program currently running"</p> <p>If you receive this error, call the AD:ABORT command to stop the analog input operation before reconfiguring the TACHometer subsystem.</p>
Example	<p>The following example configures the tachometer input signal to use the MSB of the value to indicate whether data is new or old:</p> <pre>>:TACH:SFLAG 1</pre>
See Also	TACHometer:SFLAG[:CONFigure]? , described on page 135

Tachometer Stale Value Flag Query

Description	Returns the currently configured value of the Stale Value flag, which determines whether the most-significant bit (MSB) of the period measurement value is used to indicate new or old data.
Syntax	<code>:TACHometer:SFLAG[:CONFigure]?</code>
Parameters	None
Response Data	<stale value flag>

Name:	stale value flag
Data Format:	<Boolean>
Description:	<p>If the MSB of the value is used to indicate whether the measurement is new or old, this value is 1.</p> <p>If the MSB of the value is not used to indicate whether the measurement is new or old, this value is 0.</p>
Example	<p>The following example returns the configuration of the Stale Value flag used to measure the period of the tachometer input signal. In this example, the MSB of the value is used to indicate whether data is new or old:</p> <pre>> :TACH:SFLAG? < 1</pre>
See Also	TACHometer:SFLAG[:CONFigure], described on page 135

COUNTER Subsystem Commands

The COUNTER subsystem is used to configure and perform frequency, period, or phase measurements using the counter/timer inputs of a DT8837 LXI instrument module. Once you enable a counter/timer channel, you can read the channel as part of the analog input data stream using the AD subsystem commands, described on [page 101](#), and the **AD:FETCh?** command, described on [page 125](#).

[Table 11](#) lists the commands that apply to the COUNTER subsystem. This section describes each of these commands in detail.

Table 11: COUNTER Subsystem Commands

Type	Mnemonic	See
Counter/Timer Channel Enable	COUNTER[<n>]:ENABLE	page 138
Counter/Timer Channel Enable Query	COUNTER[<n>]:ENABLE?	page 138
Counter/Timer Edge Configuration	COUNTER[<n>]:EDGE:{START STOP}[:CONFigure]	page 139
Counter/Timer Edge Query	COUNTER[<n>]:EDGE:{START STOP}[:CONFigure]?	page 141
Counter/Timer Self Clear Flag	COUNTER[<n>]:SCLR[:CONFigure]	page 142
Counter/Timer Self Clear Query	COUNTER[<n>]:SCLR[:CONFigure]?	page 143

Counter/Timer Channel Enable

Description	Enables or disables the ability to return the value of the specified counter/timer channel in the analog input data stream.
Syntax	:COUNter [<n>]:ENABle <state>
Optional Numeric Suffix	
Name:	n
Description:	Specifies which of the counter/timer channels (1 or 2) to configure. If this suffix is omitted, counter/timer 1 is configured.
Required Parameters	
Name:	state
Data Format:	<Boolean>
Description:	Specifies whether to enable or disable the ability to return the value of the specified counter/timer channel in the analog input data stream. To enable this feature, set this value to ON or 1. To disable this feature, set this value to OFF or 0.
Optional Parameters	None
Response Data	None
Notes	Once you enable a counter/timer channel, you can read the channel as part of the analog input data stream using the AD subsystem commands, described on page 101 , and the AD:FETCh? command, described on page 125 .
Possible Errors	-284 "Program currently running" If you receive this error, call the AD:ABORt command to stop the analog input operation before reconfiguring the COUNter subsystem.
Example	The following example enables the ability to return the value of counter/timer 1 in the analog input data stream: >:COUN1:ENAB ON
See Also	COUNter [<n>]:ENABle?, described on page 138

Counter/Timer Channel Enable Query

Description	Returns whether reading the value of the counter/timer channel in the analog input data stream is enabled or disabled.
Syntax	:COUNter [<n>]:ENABle?

Optional Numeric Suffix

Name: n

Description: Specifies which of the counter/timer channels (1 or 2) to query. If this suffix is omitted, counter/timer 1 is queried.

Required Parameters None

Optional Parameters None

Response Data <state>

Name: state

Data Format: <0+NR1>

Description: Contains the current state of the counter/timer channel, where 1 indicates that the counter/timer is enabled and 0 indicates that the counter/timer channel is disabled.

Notes Once you enable a counter/timer channel, you can read the channel as part of the analog input data stream using the AD subsystem commands, described on [page 101](#), and the **AD:FETCh?** command, described on [page 125](#).

Example The following example returns whether reading the value of counter/timer 1 in the analog input data stream is enabled or disabled; in this example, counter/timer 1 is enabled.

```
> :COUN1:ENAB?
< 1
```

See Also **COUNter[<n>]:ENABle**, described on [page 138](#)

Counter/Timer Edge Configuration

Description Configures the edges that start and stop the frequency, period, or phase measurement on a counter/timer input channel.

Syntax :COUNter[<n>]:EDGE:{START|STOP}
[:CONFigure]
{ADCdone|TACHNEGative|TACHPOSitive|GATENEGative
|GATEPOSitive|DEFault}

**Optional Numeric
Suffix**

Name:	n
Description:	Specifies which of the counter/timer channels (1 or 2) to configure. If this suffix is omitted, counter/timer 1 is configured.

Required Parameters

Name:	start edge
Data Format:	<CHARACTER PROGRAM DATA>
Description:	<p>An enumerated type that specifies the edge that starts the period measurement. The following start edge types are supported:</p> <ul style="list-style-type: none">• ADCdone – Measures the completion of the A/D sample to the specified <i>stop edge</i>.• TACHNEGative – Measures from the falling edge of the tachometer input signal to the specified <i>stop edge</i>.• TACHPOSitive – Measures from the rising edge of the tachometer input signal to the specified <i>stop edge</i>.• GATENEGative – Measures from the falling edge of the gate input signal to the specified <i>stop edge</i>.• GATEPOSitive – Measures from the rising edge of the gate input signal to the specified <i>stop edge</i>.• DEFault – The default setting, which is ADCdone

Name:	stop edge
Data Format:	<CHARACTER PROGRAM DATA>
Description:	<p>An enumerated type that specifies the edge that stops the period measurement. The following stop edge types are supported:</p> <ul style="list-style-type: none">• ADCdone – Measures from the specified <i>start edge</i> to the completion of the A/D sample.• TACHNEGative – Measures from the specified <i>start edge</i> to the falling edge of the tachometer input signal.• TACHPOSitive – Measures from the specified <i>start edge</i> to the rising edge of the tachometer input signal.• GATENEGative – Measures from the specified <i>start edge</i> to the falling edge of the gate input signal.• GATEPOSitive – Measures from the specified <i>start edge</i> to the rising edge of the gate input signal.• DEFault – The default setting, which is ADCdone.

Optional Parameters None

Response Data None

Notes	If you choose to start the measurement using the ADCdone signal, choose a different signal to stop the measurement. Likewise, if you choose to stop the measurement using the ADCdone signal, choose a different signal to start the measurement.
Possible Errors	-284 "Program currently running" If you receive this error, call the AD:ABORT command to stop the analog input operation before reconfiguring the COUNTER subsystem.
Example	The following example configures counter/timer 2 to measure the period between the rising edge of the gate input signal to the falling edge of the tachometer input signal: <pre>>:COUN2:EDGE:START GATEPOS >:COUN2:EDGE:STOP TACHNEG</pre>
See Also	COUNTER[<n>]:EDGE:{START STOP}[:CONFIGURE]? , described on page 141

Counter/Timer Edge Query

Description	Returns the currently configured edges that start and stop a frequency, period, or phase measurement on a counter/timer input channel.
Syntax	COUNTER[<n>]:EDGE:{START STOP}[:CONFIGURE]?
Optional Numeric Suffix	
Name:	n
Description:	Specifies which of the counter/timer channels (1 or 2) to query. If this suffix is omitted, counter/timer 1 is queried.
Parameters	None
Response Data	<start edge stop edge>
Name:	start edge
Data Format:	<CHARACTER RESPONSE DATA>
Description:	An enumerated type that identifies the edge that starts the period measurement. The following start edge types are supported: <ul style="list-style-type: none"> • ADCdone – Measures the completion of the A/D sample to the specified <i>stop edge</i>. This value is returned if the <i>start edge</i> was configured as ADCdone or DEFAULT. • TACHNEGative – Measures from the falling edge of the tachometer input signal to the specified <i>stop edge</i>. • TACHPOSitive – Measures from the rising edge of the tachometer input signal to the specified <i>stop edge</i>.

- Description (cont.):
- GATENEGative – Measures from the falling edge of the gate input signal to the specified *stop edge*.
 - GATEPOSitive – Measures from the rising edge of the gate input signal to the specified *stop edge*.

Name: stop edge

Data Format: <CHARACTER RESPONSE DATA>

- Description: An enumerated type that identifies the edge that stops the period measurement. The following stop edge types are supported:
- ADCdone – Measures from the specified *start edge* to the completion of the A/D sample. This value is returned if the *stop edge* was configured as ADCdone or DEFault.
 - TACHNEGative – Measures from the specified *start edge* to the falling edge of the tachometer input signal.
 - TACHPOSitive – Measures from the specified *start edge* to the rising edge of the tachometer input signal.
 - GATENEGative – Measures from the specified *start edge* to the falling edge of the gate input signal.
 - GATEPOSitive – Measures from the specified *start edge* to the rising edge of the gate input signal.

Example The following example returns the starting and stopping measurement edges that were configured for counter/timer 2. In this example, the counter/timer was configured to measure the period between the rising edge of the gate input signal to the falling edge of the tachometer input signal:

```
> :COUN2 :EDGE :START?
< GATEPOS
> :COUN2 :EDGE :STOP?
< TACHNEG
```

See Also COUNter[<n>]:EDGE:{START|STOP}
[:CONFigure], described on [page 139](#)

Counter/Timer Self Clear Flag Configuration

Description Configures the Self Clear flag for the counter/timer, which determines the value of the counter between measurements.

Syntax :COUNter[<n>]:SCLR[:CONFigure] <self clear>

Optional Numeric Suffix

Name: n

Description: Specifies which of the counter/timer channels (1 or 2) to configure. If this suffix is omitted, counter/timer 1 is configured.

Required Parameters

Name: self clear

Data Format: <0+NR1>

Description: Indicates whether the value read between measurements is cleared or retained.

To clear the value to zero between measurements, set *self clear* to 1.

To retain the previous measurement value until the next measurement is complete, set *self clear* to 0.

If this parameter is omitted, the existing configuration is retained in the specified counter/timer.

Optional Parameters None

Response Data None

Possible Errors -284 "Program currently running"

If you receive this error, call the **AD:ABORT** command to stop the analog input operation before reconfiguring the COUNTER subsystem.

Example The following example configures counter/timer 2 to clear the value of the counter to 0 between measurements:

```
>:COUN2:SCLR 1
```

See Also COUNTER[<n>]:SCLR[:CONFigure]?, described on [page 143](#)

Counter/Timer Self Clear Flag Query

Description Returns the currently configured value of the Self Clear flag for the counter/timer, which determines the value of the counter between measurements.

Syntax :COUNTER[<n>]:SCLR[:CONFigure]?

Optional Numeric Suffix

Name: n

Description: Specifies which of the counter/timer channels (1 or 2) to query. If this suffix is omitted, counter/timer 1 is queried.

Parameters None

Response Data	<self clear>
Name:	self clear
Data Format:	<0+NR1>
Description:	Indicates whether the value read between measurements is cleared (1) or retained (0).
Example	<p>The following example returns the value of the Self Clear flag for counter/timer 2. In this example, the counter is cleared to 0 between measurements:</p> <pre>> :COUN2 :SCLR? < 1</pre>
See Also	COUNter[<n>]:SCLR[:CONFigure], described on page 142

DA Subsystem Commands

The DA subsystem is used to configure and perform analog output operations on a DT8837 LXI instrument module. [Table 12](#) lists the DA subsystem commands. This section describes each of these commands in detail.

Table 12: DA Subsystem Commands

Type	Mnemonic	See
Abort Analog Output Operation	DA:ABORt	page 146
Arm Analog Output Operation	DA:ARM	page 147
Analog Output Buffer Mode Configuration	DA:BUFFer:MODE	page 149
Analog Output Buffer Mode Query	DA:BUFFer:MODE?	page 150
Analog Output Buffer Read All Values	DA:BUFFer[:DATA]?	page 150
Analog Output Buffer Set Values	DA:BUFFer[:DATA]	page 151
Analog Output Buffer Size Configuration	DA:BUFFer:SIZE[:SCAns]	page 153
Analog Output Buffer Size Query	DA:BUFFer:SIZE[:SCAns]?	page 155
Analog Output Channel Enable	DA:ENABle	page 155
Analog Output Channel Enable Query	DA:ENABle?	page 156
Analog Output Clock Source Configuration	[DA:]CLOCK:SOURce	page 157
Analog Output Clock Source Query	[DA:]CLOCK:SOURce?	page 158
Analog Output Readback Enable	DA:READ:ENABle:	page 159
Analog Output Readback Enable Query	DA:READ:ENABle?	page 160
Analog Output Status Bits Query	DA:STATus?	page 160
Analog Output Trigger Source Configuration	DA:TRIGger[:SOURce]	page 161
Analog Output Trigger Source Query	DA:TRIGger[:SOURce]?	page 164
Analog Output Update Frequency Configuration	DA:CLOCK:FREQuency[:CONFigure]	page 166
Analog Output Update Frequency Query	DA:CLOCK:FREQuency[:CONFigure]?	page 167
Initiate Analog Output Operation	DA:INITiate	page 168

Abort Analog Output Operation

Description Stops the analog output operation on the DT8837 LXI instrument module, if it is in progress.

Syntax :DA:ABORt [DEfault|RETZero|ENDBuffer]

Optional Keywords

Name: DA

Data Format: <CHARACTER PROGRAM DATA>

Description: Enumerated type that specifies whether to stop the operation on the analog output (DA) subsystem.

Note that if this keyword is omitted, both the analog input (AD) and analog output (DA) subsystems are stopped simultaneously when the **ABORt** command is executed.

Required Parameters None

Optional Parameters

Name: DEfault|RETZero|ENDBuffer

Data Format: <CHARACTER PROGRAM DATA>

Description: Enumerated type that specifies how to abort the analog output operation.

The following options are supported:

- DEfault – The analog output operation stops abruptly and the output value is held to the last value that was output when the operation was stopped.
- RETZero – The analog output operation stops abruptly and the output value is set to zero.
- ENDBuffer – The analog output operation stops after the last value in the output buffer has been output. The output value is held to the last value that was output.

Response Data None

Notes Query the D/A status bits using the **DA:STATus?** command, described on [page 160](#), to determine the status of the analog output subsystem.

Example The following example configures a DT8837 LXI instrument module to update the analog output channel at 52.734 kHz when a software trigger is detected, continuously outputting from the buffer, and queries the D/A status bits to determine the status of the operation:

```
> :DA:ENAB ON
> :DA:BUFFer:MODE WRAP
> :DA:CLOC INT
> :DA:CLOC:FREQ MAX
> :DA:TRIG IMM
> :DA:STAT?
< 0
```

The analog output operation is then armed and started, and the D/A status bits are queried to determine the status of the operation:

```
> :DA:ARM
> :DA:INIT
> :DA:STAT?
< 7
```

The analog output operation is then stopped abruptly and the output value is set to 0. The D/A status bits are queried to determine the status of the operation:

```
> :DA:ABOR RETZero
> :DA:STAT?
< 0
```

See Also [DA:ARM](#), described on [page 147](#)
[DA:INITiate](#), described on [page 168](#)
[DA:BUFFer:MODE](#), described on [page 149](#)
[DA:STATus?](#), described on [page 160](#)

Arm Analog Output Operation

Description Arms the analog output subsystem, which writes all the configuration settings to the instrument module, and forces the analog output to 0 V.

Syntax :DA:ARM

Optional Keywords

Name:	DA
Data Format:	<CHARACTER PROGRAM DATA>
Description:	<p>Enumerated type that specifies whether to arm the analog output (DA) subsystem.</p> <p>If this keyword is omitted, both the analog input (AD) and analog output (DA) subsystems are armed simultaneously when the ARM command is executed.</p>
Required Parameters	None
Optional Parameters	None
Response Data	None
Notes	<p>If you want to perform analog output and analog input operations simultaneously, call ARM rather than DA:ARM, or explicitly call DA:ARM before calling AD:ARM. Once the analog input subsystem is armed, the analog output subsystem cannot be armed.</p> <p>If you specified the trigger source for the analog output operation as IMMediate, you must call the DA:INITiate command after arming the operation to start the operation.</p> <p>Query the D/A status bits using the DA:STATus? command, described on page 160, to determine the status of the analog output subsystem.</p>
Example	<p>The following example configures a DT8837 LXI instrument module to update the analog output channel at 52.734 kHz when a software trigger is detected, continuously outputting from the buffer, and queries the D/A status bits to determine the status of the analog output operation:</p> <pre>> :DA:ENAB ON > :DA:BUFFer:MODE WRAP > :DA:CLOC INT > :DA:CLOC:FREQ MAX > :DA:TRIG IMM > :DA:STAT? < 0</pre> <p>The following example arms the analog output subsystem, and queries the D/A status bits to determine the status of the operation:</p> <pre>> :DA:ARM > :DA:STAT? < 2</pre> <p>The analog output operation is then started, and the D/A status bits are queried to determine the status of the operation:</p> <pre>> :DA:INIT > :DA:STAT? < 7</pre>

See Also **DA:ARM**, described on [page 147](#)
DA:INITiate, described on [page 168](#)
DA:BUFFer:MODE, described on [page 149](#)
DA:STATus?, described on [page 160](#)

Analog Output Buffer Mode Configuration

Description Configures the wrap mode of the output buffer.
Syntax `:DA:BUFFer:MODE {WRAP|NOWRAP|DEFAULT}`

Required Parameters

Name: WRAP|NOWRAP|DEFAULT
Data Format: <CHARACTER PROGRAM DATA>
Description: Specifies the wrap mode for the output buffer. The available choices are as follows:

- **WRAP** – Continuously updates the analog output channel with the number of samples (configured by **DA:BUFFer:SIZE**) stored the output buffer, starting from the first location in the buffer. When the configured number of samples have been output, the operation continues starting at the first location of the buffer, and the process continues indefinitely until you stop it with **DA:ABORT**.
- **NOWRAP** – The analog output channel is updated with the data stored in the output buffer, starting with the first location in the buffer. When the number of samples (configured by **DA:BUFFer:SIZE**) has been reached, the operation stops automatically. The waveform is output only once.
- **DEFAULT** – The default setting is WRAP mode.

Optional Parameters None

Response Data None

Notes Refer to *1999 SCPI Command Reference*, section 23.6.1, for more information.

Possible Errors -200 "Execution error"

If you receive this error, the analog output subsystem is armed. Call the **DA:ABORT** command to stop the analog output operation before reconfiguring the analog output subsystem.

Example The following example configures the output buffer for continuous wrap mode (WRAP):

```
>:DA:BUFF:MODE WRA
```

See Also **DA:BUFFer:MODE?**, described on [page 150](#)
DA:BUFFer:SIZE, described on [page 153](#)

Analog Output Buffer Mode Query

Description	Returns the currently configured wrap mode for the output buffer.
Syntax	:DA:BUFFer:MODE?
Required Parameters	None
Optional Parameters	None
Response Data	{WRAP NOWRAP}
Name:	WRAP NOWRAP
Data Format:	<CHARACTER RESPONSE DATA>
Description:	<p>The currently configured wrap mode. Possible values are as follows:</p> <ul style="list-style-type: none">• WRAP – Continuously updates the analog output channel with the number of samples (configured by DA:BUFFer:SIZE) stored the output buffer, starting from the first location in the buffer. When the configured number of samples have been output, the operation continues starting at the first location of the buffer, and the process continues indefinitely until you stop it with DA:ABORT.• NOWRAP – The analog output channel is updated with the data stored in the output buffer, starting with the first location in the buffer. When the number of samples (configured by DA:BUFFer:SIZE) has been reached, the operation stops automatically. The waveform is output only once.
Notes	Refer to <i>1999 SCPI Command Reference</i> , section 23.6.1, for more information.
Example	<p>The following example returns the currently configured wrap mode of the output buffer; in this case, continuous wrap mode (WRA) is used:</p> <pre>> :DA:BUFF:MODE? < WRAP</pre>
See Also	DA:BUFFer:MODE , described on page 149 DA:BUFFer:SIZE , described on page 153

Analog Output Buffer Read All Values

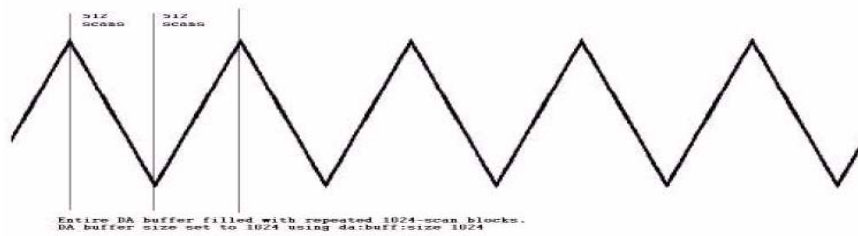
Description	Reads the stored values from the output buffer.
Syntax	:DA:BUFFer[:DATA]?
Required Parameters	None
Response Data	<block>

Name:	block
Data Format:	<ARBITRARY BLOCK PROGRAM DATA>
Description:	Contains an integral number of samples starting from the beginning of the output buffer. You specify the size of the output buffer using the DA:BUFFer:SIZE command, described on page 153 . Each sample is a 4-byte analog output value, in network byte order, with 0 as the most significant byte in each value.
Notes	You can fill the output buffer by using the DA:BUFFer[:DATA] command, described on page 151 . This query can then be used to verify the values that were written to the output buffer. Refer to <i>1999 SCPI Command Reference</i> , section 23.3, and IEEE Std 488.2-1992, section 8.7.9, for more information.
Possible Errors	-221 "Settings conflict" If you receive this error, check the wrap mode of the output buffer.
Example	This example shows how to verify the values that were loaded into the output buffer: <pre>>:DA:BUFF (32768, 0, 65535) >:DA:BUFF? < 32768, 0, 65535</pre>
See Also	DA:BUFFer:SIZE , described on page 153 DA:BUFFer[:DATA] , described on page 151

Analog Output Buffer Set Values

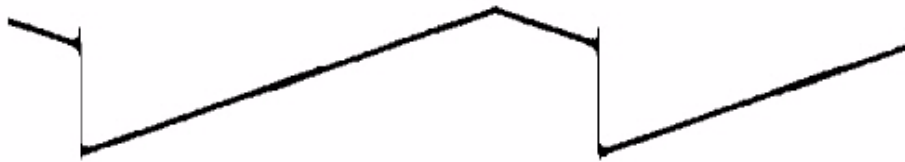
Description	Stores samples in the output buffer that are written to the analog output channel when the analog output subsystem is triggered.
Syntax	:DA:BUFFER[:DATA] <index>, <block>
Required Parameters	
Name:	index
Data Format:	<+NR1>
Description:	The index of the analog output buffer at which to start writing analog output samples.

Name:	block
Data Format:	<DEFINITE LENGTH ARBITRARY BLOCK PROGRAM DATA>
Description:	<p>A data block with a sequence of consecutive samples to be written to the analog output buffer, formatted as follows:</p> <p style="text-align: center;"><sample count><sample><sample>...<sample></p> <p>where,</p> <ul style="list-style-type: none">• <i>sample count</i> is the number of samples in the block. The specified number of samples is written to the analog output buffer starting at the specified <i>index</i>. No samples are written beyond the last index, 131072, in the buffer. <p>Due to internal buffer limits, <i>sample count</i> must not exceed 1024 for SCPI connections to port 5025 ("raw" SCPI) or 256 for SCPI connections over VXI-11.</p> <ul style="list-style-type: none">• <i>sample</i> is the 4-byte analog output value to write to the analog output buffer. The least significant 24-bits of this value is used to update the analog output channel. Each <i>sample</i> is in offset binary format.
Response Data	None
Notes	Refer to page 27 for more information about the block data type. For more information, refer to 1999 SCPI Command Reference, section 23.3, and IEEE Std 488.2-1992, section 7.7.6.
Possible Errors	<p>-120 "Numeric data error"</p> <p>If you receive this error, check the values in <i>block</i>.</p> <p>-284 "Program currently running"</p> <p>If you receive this error, call the DA:ABORt command to stop the analog output operation before reconfiguring the output buffer.</p>
Example	<p>This example sets the number of analog output samples to 1024 and generates a 1024-sample triangular waveform shown below:</p> <pre>> *IDN? < Data Translation,DT8837,N999999999, 0.0.0.10 > DA:STAT? < 0 > DA:CLOCK:FREQ 10240 < DA:BUFF:SIZE 1024 > SYST:ERR? < 0,"No error" > DA:ENAB 1 > SYST:ERR? < 0,"No error" > DA:INIT > DA:ABOR</pre>



Using the same 1024-sample triangular waveform, this example sets the number of analog output samples to 640. The generated waveform is shown below:

```
> DA:BUFF:SIZE 640
> DA:INIT
```



See Also `DA:BUFFer:SIZE`, described on [page 153](#)
`DA:BUFFer:SIZE?`, described on [page 155](#)
`DA:BUFFer[:DATA]?`, described on [page 151](#)

Analog Output Buffer Size Configuration

Description Sets the number of samples in the output buffer. The analog output channel is updated with this number of samples (either continuously or once depending on the buffering mode selected) when the analog output subsystem is triggered.

Syntax `:DA:BUFFer:SIZE[:SCAns]`
 {<value>|MAXimum}

Required Parameters

Name:	< value> MAXimum
Data Format:	<+NR1> CHARACTER PROGRAM DATA>
Description:	<p>Specifies the number of samples in the output buffer. The available choices are as follows:</p> <ul style="list-style-type: none">• <i>value</i> – The number of samples to store in the output buffer. This value must be between 1 and 131072.• <i>MAXimum</i> – The number of samples in the output buffer is set to 131072.
Optional Parameters	None
Response Data	None
Notes	<p>For output buffers, use this command after loading the output buffer using the DA:BUFFer[:DATA] command, described on page 151. Note the following:</p> <ul style="list-style-type: none">• If the size of the buffer equals the number of samples written to the buffer using the DA:BUFFer[:DATA] command, then the analog output channel is updated with all the values in the output buffer.• If the size of the buffer is less than the values written to the buffer using the DA:BUFFer[:DATA] command, then the analog output channel is updated with only the number of samples that fit in the buffer, starting from the the first location of the buffer.• If the size of the buffer is greater than the number of samples that were written to the buffer using the DA:BUFFer[:DATA] command, then the samples at the end of the buffer may have undefined values. <p>Refer to <i>1999 SCPI Command Reference</i>, section 23.8, for more information.</p>
Possible Errors	<p>-222 "Data out of range"</p> <p>If you receive this error, <i>numeric value</i> is out of range. The size of the input buffer defaults to the value of MAXimum number of samples (131072).</p> <p>-284 "Program currently running"</p> <p>If you receive this error, the analog output subsystem is currently running. Call the DA:ABORt command to stop the analog output operation before reconfiguring the output buffer.</p>
Example	<p>The following example sets the number of samples to store in the output buffer to 1024:</p> <pre>> : DA:BUFF:SCA 1024</pre>
See Also	<p>DA:BUFFer:SIZE?, described on page 155</p> <p>DA:BUFFer[:DATA], described on page 151</p> <p>DA:BUFFer:MODE, described on page 149</p>

Analog Output Buffer Size Query

Description	Returns the number of samples currently configured to be stored in the output buffer.
Syntax	:DA:BUFFer:SCAn?
Required Parameters	None
Optional Parameters	None
Response Data	<numeric value>
Name:	numeric value
Data Format:	<+NR1>
Description:	The number of samples currently configured to be stored in the output buffer. Values range from 1 to 131072.
Notes	Refer to <i>1999 SCPI Command Reference</i> , section 23.8, for more information.
Example	<p>The following example returns the number of samples currently configured to be stored in the output buffer; in this case, the output buffer is configured to store 1024 samples:</p> <pre>> :DA:BUFF:SCA? < 1024</pre>
See Also	<p>DA:BUFFer:SIZE, described on page 153</p> <p>DA:BUFFer[:DATA], described on page 151</p>

Analog Output Channel Enable

Description	Enables or disables the ability to update the analog output channel.
Syntax	:DA:ENABle <state>
Required Parameters	
Name:	state
Data Format:	<Boolean>
Description:	<p>Specifies whether to update the value of the analog output channel.</p> <p>To enable this capability, set this value to ON or 1. To disable this capability, set this value to OFF or 0.</p>
Optional Parameters	None
Response Data	None

Notes When disabled, the voltage on the analog output channel is set to 0 V.

When enabled, the analog output channel is updated with the values specified using the **DA:BUFFer[:DATA]** command at the configured update frequency as soon as the configured output trigger is detected. Refer to [page 151](#) for more information on specifying the output values, [page 157](#) for more information on configuring the update frequency, and to [page 161](#) for more information on configuring the output trigger.

If you want to readback the value of the analog output channel in the analog input data stream, use the **DA:READ:ENABLE** command, described on [page 159](#).

Possible Errors -284 "Program currently running"

If you receive this error, call the **DA:ABORT** command to stop the analog output operation before reconfiguring the analog output subsystem.

Example The following example enables the ability to update the analog output channel:

```
>:DA:ENABle ON
```

See Also **DA:ENABLE?**, described on [page 156](#)
DA:CLOCK:FREQuency[:CONFigure], described on [page 157](#)
DA:TRIGger[:SOURce], described on [page 161](#)
DA:BUFFer[:DATA], described on [page 151](#)
DA:READ:ENABLE, described on [page 159](#)

Analog Output Channel Enable Query

Description	Returns whether the ability to update the analog output channel is enabled or disabled.
Syntax	:DA:ENABle?
Required Parameters	None
Optional Parameters	None
Response Data	<state>
Name:	state
Data Format:	<0+NR1>
Description:	Contains the current state of the analog output channel, where 1 indicates that the analog output channel is enabled and 0 indicates that the analog output channel is disabled.

Example The following example returns whether the analog output channel is enabled or disabled; in this example, this feature is enabled.

```
> :DA:ENAB?
< 1
```

See Also [DA:ENABle](#), described on [page 155](#)
[DA:CLOCK:FREQuency\[:CONFigure\]](#), described on [page 157](#)
[DA:TRIGger\[:SOURce\]](#), described on [page 161](#)
[DA:BUFFer\[:DATA\]](#), described on [page 151](#)
[DA:READ:ENABle](#), described on [page 159](#)

Analog Output Clock Source Configuration

Description Specifies the source of the reference clock used for the ADC and DAC master clock generators on the DT8837 instrument module.

Syntax : [DA:]CLOCK:SOURce {INTernal|LXI7}

Optional Keywords

Name: DA

Data Format: <CHARACTER PROGRAM DATA>

Description: Optional enumerated type that specifies the analog output (DA) subsystem for which to configure the clock source.

On the DT8837 instrument module, the analog input and analog output subsystems share a single reference clock. Therefore, you can omit this keyword to affect the clock source for both subsystems.

Required Parameters

Name: INTernal|LXI7

Data Format: <CHARACTER PROGRAM DATA>

Description: Enumerated type that specifies the reference clock source to use, where INTernal is the internal reference clock on the DT8837 and LXI7 is line 7 of the Trigger Bus.

Optional Parameters None

Response Data None

Notes The specified reference clock source is used for pacing both analog input and analog output operations.

If the instrument module is the clock master, specify reference clock source as INTernal.

If the instrument module is a slave and you want to synchronize the clock with the master, specify the reference clock source as LXI7.

Possible Errors	-284 "Program currently running" If you receive this error, call the DA:ABORT command to stop the analog output operation before reconfiguring the analog output subsystem.
Example	The following example specifies the clock source on the DT8837 instrument module as the internal reference clock: <code>>:CLOCK:SOUR INT</code>
See Also	[DA:]CLOCK:SOURce?, described on page 158 DA:CLOCK:FREQuency[:CONFigure], described on page 166 DA:CLOCK:FREQuency[:CONFigure]?, described on page 167 WTB:LXI<n>[:OUTput]:ENABLE, described on page 171

Analog Output Clock Source Query

Description	Returns the currently configured reference clock source used for the ADC and DAC master clock generators on the DT8837 instrument module.
Syntax	: [DA:] CLOCK: SOURce?
Optional Keywords	
Name:	DA
Data Format:	<CHARACTER PROGRAM DATA>
Description:	Optional enumerated type that specifies the analog output (DA) subsystem for which to return the currently configured clock source.
Description (cont.):	On the DT8837 instrument module, the analog input and analog output subsystems share a single reference clock. Therefore, you can omit this keyword to return the currently configured clock source for both subsystems.
Required Parameters	None
Optional Parameters	None
Response Data	<INTernal LXI7>
Name:	INTernal LXI7
Data Format:	<CHARACTER RESPONSE DATA>
Description:	Enumerated type that specifies the reference clock source that is currently configured, where INTernal is the internal reference clock on the DT8837 and LXI7 is the external reference clock from the Trigger Bus.

Example The following example returns the currently configured reference clock source on the DT8837 instrument module; in this case, the internal reference clock is configured:

```
> :CLOC:SOUR?
< INTernal
```

See Also [DA:]CLOCK:SOURce, described on [page 157](#)
 DA:CLOCK:FREQuency[:CONFigure], described on [page 166](#)
 DA:CLOCK:FREQuency[:CONFigure]?, described on [page 167](#)

Analog Output Readback Enable

Description Enables or disables the ability to read back the value from the analog output channel in the analog input data stream.

Syntax :DA:READ:ENABle <state>

Required Parameters

Name: state
Data Format: <Boolean>
Description: Specifies whether to enable or disable reading back the value from the analog output channel.
 To enable the readback, set this value to ON or 1. To disable the readback, set this value to OFF or 0.

Optional Parameters None

Response Data None

Notes To configure the analog output channel, use the DA:ENABle command, described on [page 155](#).
 You configure the values to write to the analog output channel, using the DA:BUFFer[:DATA] command, described on [page 151](#).
 Once the analog output readback channel is enabled, you can read the value of the analog output channel in the analog input data stream using the AD subsystem commands, described on [page 101](#), and the AD:FETCh? command, described on [page 125](#).

Possible Errors -284 "Program currently running"

If you receive this error, call the DA:ABORt command to stop the analog output operation before reconfiguring the analog output subsystem.

Example The following example enables reading back the value from the analog output channel in the analog input data stream:

```
> :DA:READ:ENAB ON
```

See Also **DA:READ:ENABle?**, described on [page 160](#)
 DA:ENABle, described on [page 155](#)

Analog Output Readback Enable Query

Description	Returns whether reading back the value of the analog output channel in the analog input data stream is enabled or disabled.
Syntax	<code>:DA:READ:ENABle?</code>
Required Parameters	None
Optional Parameters	None
Response Data	<code><state></code>
Name:	state
Data Format:	<code><0+NR1></code>
Description:	Contains the current readback state of the analog output channel, where 1 indicates that reading back the analog output value is enabled and 0 indicates that reading back the analog output value is disabled.
Notes	<p>To configure the analog output channel, use the DA:ENABle command, described on page 155.</p> <p>You configure the values to write to the analog output channel, using the DA:BUFFer[:DATA] command, described on page 151.</p>
Notes (cont.)	Once the analog output readback channel is enabled, you can read the value of the analog output channel in the analog input data stream using the AD subsystem commands, described on page 101 , and the AD:FETCh? command, described on page 125 .
Example	<p>The following example returns whether reading back the value of the analog output channel in the analog input data stream is enabled or disabled; in this example, this function is enabled.</p> <pre>> :DA:READ:ENAB? < 1</pre>
See Also	DA:READ:ENABle? , described on page 160 DA:ENABle , described on page 155

Analog Output Status Bits Query

Description	Returns the status bits for the analog output subsystem.
Syntax	<code>:DA:STATus?</code>
Required Parameters	None
Optional Parameters	None

Response Data	<status bits>
Name:	status bits
Data Format:	<0+NR1>
Description:	<p>The value, in the range of 0 to 255, of the status bits for the analog output subsystem. The status bits are defined as follows:</p> <ul style="list-style-type: none"> • Bit 0 – The value of this bit is 1 if the analog output subsystem is active or 0 if the analog output subsystem is inactive. • Bit 1 – The value of this bit is 1 if the analog output subsystem is armed or 0 if the analog output subsystem is not armed. • Bit 2 – The value of this bit is 1 if the analog output subsystem is triggered or 0 if the analog output subsystem is not triggered. • Bit 3 – The value of this bit is 1 if the DA FIFO is done (the end of the buffer is reached) when no wrap mode is selected or 0 if the DA FIFO is not done (the end of the buffer has not been reached) when no wrap mode is selected. • Bit 4 – The value of this bit is 1 if the DA FIFO is empty or 0 if the DA FIFO is not empty.
Notes	Refer to <i>IEEE 488.2-1992</i> , section 9.1, for more information.
Example	<p>The following example returns the status bits for the analog output subsystem:</p> <pre>> :DA:STAT? < 2</pre> <p>In this example, the status bit value is 2, indicating that the analog output subsystem is armed.</p>

Analog Output Trigger Source Configuration

Description Specifies the trigger source that starts the analog output operation on the DT8837 instrument module.

Syntax :DA:TRIGger [:SOURce]
 {IMMediate|EXTernal|AIN1|LXI0|LXI1|LXI2|LXI3|
 LXI4|LXI5|LAN0|LAN1|LAN2|LAN3|LAN4|LAN5|LAN6|
 LAN7|DEFault}

Required Parameters

Name:	IMMediate EXTernal AIN1 LXI0 LXI1 LXI2 LXI3 LXI4 LXI5 LAN0 LAN1 LAN2 LAN3 LAN4 LAN5 LAN6 LAN7 DEFault
Data Format:	<CHARACTER PROGRAM DATA>
Description:	Enumerated type that specifies the trigger source to use.

- Description (cont.): The following trigger sources are supported:
- **IMMediate** – A software trigger. Once you arm the subsystem with the **DA:ARM** command, the software trigger event occurs when you start the operation with the **DA:INITiate** command, described on [page 168](#); the computer issues a write to the instrument module to begin conversions.
 - **EXTernal** – An external digital (TTL) trigger event. Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when the instrument module detects a rising-edge transition on the Trigger In signal connected to the instrument module.
 - **AIN1** – An analog threshold trigger event. Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when the signal attached to analog input channel 1 rises above a user-specified threshold value from 0.2 V to 9.8 V with 0.1 V of hysteresis. You specify the trigger threshold using the **AD:TRIGger:LEVel[:AIN1]** command, described on [page 123](#).
 - **LXI0** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 0 of the Trigger Bus.
 - **LXI1** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 1 of the Trigger Bus.
 - **LXI2** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 2 of the Trigger Bus.
 - **LXI3** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 3 of the Trigger Bus.
 - **LXI4** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 4 of the Trigger Bus.
 - **LXI5** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 5 of the Trigger Bus.
 - **LAN0** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 0.

Description (cont.):	<ul style="list-style-type: none"> • LAN1 – Once you arm the subsystem with the DA:ARM command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 1. • LAN2 – Once you arm the subsystem with the DA:ARM command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 2. • LAN3 – Once you arm the subsystem with the DA:ARM command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 3. • LAN4 – Once you arm the subsystem with the DA:ARM command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 4. • LAN5 – Once you arm the subsystem with the DA:ARM command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 5. • LAN6 – Once you arm the subsystem with the DA:ARM command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 6. • LAN7 – Once you arm the subsystem with the DA:ARM command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 7. • DEFault – The default setting, which is IMMEDIATE.
Optional Parameters	None
Response Data	None
Notes	<p>The trigger input signal from the Trigger Bus is propagated by the trigger source of the analog input subsystem on the trigger master. Both the analog input and analog output subsystems can be triggered by a Trigger Bus signal, if and only if, the same line (LXI0 to LXI5) is used as the trigger. The order in which to set up the Trigger Bus is important; refer to page 62 for more information about using the Trigger Bus to connect and trigger multiple instrument modules.</p> <p>The analog input and analog output subsystems can be triggered by either the same LAN event or by two different LAN events. Configure the LXI-defined LAN event using the commands and queries under the EVENT subsystem, described on page 174. Refer to page 62 for more information about using LXI LAN events to trigger multiple instrument modules over the network.</p>
Possible Errors	<p>-284 "Program currently running"</p> <p>If you receive this error, call the DA:ABORT command to stop the analog output operation before reconfiguring the analog output subsystem.</p>

Example The following example configures an external digital trigger source to start analog input operations and a threshold trigger to start analog output operations on the DT8837 instrument module:

```
>:AD:TRIG EXT
>:AD:ARM
>:DA:TRIG AIN1
>:DA:ARM
```

See Also **DA:ARM**, described on [page 147](#)
DA:INITiate, described on [page 168](#)
DA:TRIGger[:SOURce]?, described on [page 161](#)
AD:TRIGger:LEVel[:AIN1], described on [page 123](#)
AD:TRIGger:LEVel[:AIN1]?, described on [page 124](#)
DA:CLOCK:FREQuency[:CONFigure], described on [page 166](#)
DA:CLOCK:FREQuency[:CONFigure]?, described on [page 167](#)

Analog Output Trigger Source Query

Description Returns the trigger source that is currently configured for starting an analog output operation on the DT8837 instrument module.

Syntax :DA:TRIGger[:SOURce]?

Required Parameters None

Optional Parameters None

Response Data {IMMediate|EXTernal|AIN1|LXI0|LXI1|LXI2|LXI3|LXI4|LXI5|LAN0|LAN1|LAN2|LAN3|LAN4|LAN5|LAN6|LAN7}

Name: IMMediate|EXTernal|AIN1|LXI0|LXI1|LXI2|LXI3|LXI4|LXI5|LAN0|LAN1|LAN2|LAN3|LAN4|LAN5|LAN6|LAN7

Data Format: <CHARACTER RESPONSE DATA>

Description: Enumerated type that specifies the trigger source to use. The following trigger sources are supported:

- **IMMediate** – A software trigger. Once you arm the subsystem with the **DA:ARM** command, the software trigger event occurs when you start the operation with the **DA:INITiate** command, described on [page 168](#); the computer issues a write to the instrument module to begin conversions. If the trigger source was configured as **Default**, **IMM** is returned.
- **EXTernal** – An external digital (TTL) trigger event. Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when the instrument module detects a rising-edge transition on the signal connected to the Trigger In signal connected to instrument module.

Description (cont.):

- **AIN1** – An analog threshold trigger event. Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when the signal attached to analog input channel 1 rises above a user-specified threshold value from 0.2 V to 9.8 V with 0.1 V of hysteresis. You specify the trigger threshold using the **AD:TRIGger:LEVel[:AIN1]** command, described on [page 123](#).
- **LXI0** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 0 of the Trigger Bus.
- **LXI1** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 1 of the Trigger Bus.
- **LXI2** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 2 of the Trigger Bus.
- **LXI3** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 3 of the Trigger Bus.
- **LXI4** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 4 of the Trigger Bus.
- **LXI5** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when a slave DT8837 instrument module detects a trigger input signal on line 5 of the Trigger Bus.
- **LAN1** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 1.
- **LAN2** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 2.
- **LAN3** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 3.
- **LAN4** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 4.
- **LAN5** – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 5.

- Description (cont.):**
- LAN6 – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 6.
 - LAN7 – Once you arm the subsystem with the **DA:ARM** command, this trigger event occurs when the DT8837 instrument module detects LXI-defined LAN event 7.

Example The following example returns the currently configured trigger source for analog output operations; in this example, the analog threshold trigger is used:

```
> :DA:TRIG?  
< AIN1
```

See Also **DA:TRIGger[:SOURce]**, described on [page 161](#)
AD:TRIGger:LEVel[:AIN1], described on [page 123](#)
AD:TRIGger:LEVel[:AIN1]?, described on [page 124](#)
DA:CLOCK:FREQuency[:CONFigure], described on [page 166](#)
DA:CLOCK:FREQuency[:CONFigure]?, described on [page 167](#)

Analog Output Update Frequency Configuration

Description Specifies the output frequency for pacing analog output operations on the DT8837 instrument module.

Syntax **:DA:CLOCK:FREQuency[:CONFigure]**
<freq|MINimum|MAXimum>

Required Parameters

Name: **freq|MINimum|MAXimum**

Data Format: **<+NRf>**

Description: Specifies the output frequency, in Hz, where *freq* represents a particular output frequency, *MINimum* represents the minimum output frequency, and *MAXimum* represents the maximum output frequency for the analog output subsystem.

For analog output operations, the minimum output frequency is 10000 Hz and the maximum output frequency is 52734.375 Hz (the default).

Optional Parameters None

Response Data None

Notes If you are synchronizing the clocks of multiple instrument modules, it is important that you enable LXI7 as an output on the clock master and then configure the clock frequency of the slaves before configuring the clock frequency of the clock master.

Notes (cont.) It is also recommended that you choose the same sampling frequency for all instrument modules to avoid problems. Note, however, that the analog input and analog output frequencies can be different from each other.

If you later change the sampling frequency of one or more slave instrument modules, you must reconfigure the clock frequency of the clock master, even if its parameters have not changed.

Refer to [page 59](#) for more information on setting up the clocks.

Possible Errors

-131 "Invalid suffix"

If you receive this error, ensure that the output frequency is entered in <+NRf> format.

-222 "Data out of range"

If you receive this error, the output frequency that was entered was outside the valid range. Enter a valid output frequency.

-284 "Program currently running"

If you receive this error, call the **DA:ABORT** command to stop the analog output operation before reconfiguring the analog output subsystem.

Example

The following example configures a frequency of 52734.375 Hz for the analog output subsystem:

```
>:DA:CLOC:FREQ MAX
>:DA:CLOC:FREQ?
< 52734.375
>:SYST:ERR?
< 0, "No error"
```

See Also

[DA:]CLOCK:SOURce, described on [page 157](#)

[DA:]CLOCK:SOURce?, described on [page 158](#)

DA:CLOCK:FREQuency[:CONFigure]?, described on [page 167](#)

AD:CLOCK:FREQuency[:CONFigure]?, described on [page 116](#)

WTB:LXI<n>[:OUTput]:ENABle, described on [page 171](#)

Analog Output Update Frequency Query

Description Returns the output frequency that is currently configured for pacing analog output operations on the DT8837 instrument module.

Syntax :DA:CLOCK:FREQuency[:CONFigure]?

Required Parameters None

Optional Parameters None

Response Data <output frequency>

Name:	output frequency
Data Format:	<+NRf>
Description:	The currently configured output frequency, in Hz, for the analog output subsystem.
Example	<p>The following example returns the currently configured output frequency, in Hz, for the analog output subsystem on the DT8837 instrument module; in this case the frequency is 10000 Hz:</p> <pre>> :DA:CLOC:FREQ? < 10000.00</pre>
See Also	<p>[DA:]CLOCK:SOURce, described on page 157</p> <p>[DA:]CLOCK:SOURce?, described on page 158</p> <p>DA:CLOCK:FREQuency[:CONFigure], described on page 166</p>

Initiate Analog Output Operation

Description	If the configured trigger source for the analog output subsystem is IMMEDIATE, starts the analog output operation immediately once the subsystem has been armed with the DA:ARM command. For all other trigger sources, this command has no effect.
Syntax	:DA:INITiate
Optional Keywords	
Name:	DA
Data Format:	<CHARACTER PROGRAM DATA>
Description:	<p>Enumerated type that specifies whether to start the analog output (DA) subsystem.</p> <p>If this keyword is omitted, both the analog input (AD) and analog output (DA) subsystems are started simultaneously when the INITiate command is executed.</p>
Required Parameters	None
Optional Parameters	None
Response Data	None
Notes	<p>An Execution Error is generated by this command if one of the following conditions occurs:</p> <ul style="list-style-type: none">• The analog output operation is already running• The output channel was not enabled• The analog output frequency is invalid <p>If bit 4 (E) of the Standard Event Status Enable register is enabled, an Execution Error sets bit 4 (E) of the Standard Event Status register.</p>

- Notes (cont.)** To determine if an Execution Error occurred, query bit 4 of the Standard Event Status register using the ***ESR?** command, described on [page 71](#).
- Query the D/A status bits using the **DA:STATus?** command, described on [page 160](#), to determine the state of the analog output subsystem.
- To fill the output buffer, use the **DA:BUFFer[:DATA]?** command, described on [page 151](#).

Example The following example configures a DT8837 LXI instrument module to update the analog output channel at 52.734 kHz when a software trigger is detected, and queries the D/A status bits to determine the status of the analog output operation:

```
> :DA:ENAB ON
> :DA:CLOC INT
> :DA:CLOC:FREQ MAX
> :DA:TRIG IMM
> :DA:STAT?
< 0
```

The analog output operation is then armed and started, and the D/A status bits are queried to determine the status of the analog output operation:

```
> :DA:ARM
> :DA:INIT
> :DA:STAT?
< 7
```

The analog output operation is then stopped and the D/A status bits are queried to determine the status of the analog output operation:

```
> :DA:ABOR
> :DA:STAT?
< 4
```

See Also **DA:ARM**, described on [page 147](#)
DA:TRIGger[:SOURce], described on [page 161](#)
DA:ABORt, described on [page 146](#)
DA:STATus?, described on [page 160](#)
DA:BUFFer[:DATA], described on [page 151](#)

WTB (Wired Trigger Bus) Subsystem Commands

The Wired Trigger Bus subsystem includes the commands listed in [Table 13](#). Use these commands when configuring the lines of the Trigger Bus on a DT8837 LXI instrument modules. This section describes each of these commands in detail.

Table 13: Trigger Bus Subsystem Commands

Name	Mnemonic	See
Trigger Bus Lines Configuration	WTB:LXI<n>[:OUTput]:ENABLE	page 171
Trigger Bus Lines Query	WTB:LXI<n>[:OUTput]:ENABLE?	page 172

Trigger Bus Lines Configuration

Description	Enables or disables a specific Trigger Bus line from being driven on the bus.
Syntax	<code>:WTB:LXI<n>[:OUTput]:ENABLE <state></code>
Required Numeric Suffix	
Name:	n
Description:	Specifies which of the Trigger Bus lines to enable for output. Values range from 0 to 7.
Required Parameters	
Name:	state
Data Format:	<Boolean>
Description:	Specifies whether the LXI line is driven out (enabled) on the Trigger Bus or whether the line is ignored (disabled). To enable the Trigger Bus LXI line, set this value to ON or 1. To disable the Trigger Bus LXI line, set this value to OFF or 0. By default, all Trigger Bus LXI lines are disabled.
Optional Parameters	None
Response Data	None
Notes	<p>The LXI clock source is dedicated to line 7 of the Trigger Bus. You can specify the LXI clock source using the AD:CLOCK:SOURce command, described on page 109, or the DA:CLOCK:SOURce command, described on page 157.</p> <p>The LXI sync source is dedicated to line 6 of the Trigger Bus and is automatically enabled as an output when LXI7 is enabled as an output.</p> <p>The LXI trigger source can be output on lines 0 to 5 of the Trigger Bus. You can specify the source of the trigger using the AD:TRIGger[:SOURce] command, described on page 118, or the DA:TRIGger[:SOURce] command, described on page 161.</p> <p>If you change the trigger source on the master, you must call the WTB:LXI<n>[:OUTput]:ENABLE command again to ensure that the LXI trigger signal is driven out on the Trigger Bus.</p> <p>Refer to the flow diagrams in Chapter 5 starting on page 187 for more information about setting up the Trigger Bus for input and output operations.</p>

Possible Errors	-114 "Header suffix out of range"
	If you receive this error, ensure that you specify a value of 0 to 7 for the header suffix <i>n</i> .
	-284 "Program currently running"
	If you receive this error, call the [AD DA:]ABORt command to stop the operation before configuring the Trigger Bus.
Example	This example configures LXI line 0 of the Trigger Bus as driven (enabled): <pre>>:WTB:LXI0:ENAB 1</pre>
See Also	WTB:LXI<n>[:OUTput]:ENABle? , described on page 172

Trigger Bus Lines Query

Description	Returns the current configuration of a specific LXI line of the Trigger Bus.
Syntax	:WTB:LXI<n>[:OUTput]:ENABle?
Required Numeric Suffix	
Name:	n
Description:	Specifies which of the Trigger Bus lines to configure. Values range from 0 to 7.
Required Parameters	None
Optional Parameters	None
Response Data	<state>
Name:	state
Data Format:	<Boolean>
Description:	Returns a value of 1 if the LXI line is enabled to be driven out on the Trigger Bus, or 0 if the LXI line is disabled.
Notes	<p>The LXI clock signal is dedicated to line 7 of the Trigger Bus. You can specify the LXI clock source using the AD:CLOCK:SOURce command, described on page 109, or the DA:CLOCK:SOURce command, described on page 157.</p> <p>The LXI sync source is dedicated to line 6 of the Trigger Bus and is automatically enabled as an output when LXI7 is enabled as an output.</p> <p>The LXI trigger signal from the analog input subsystem can be output on lines 0 to 5 of the Trigger Bus. You can specify the source of the trigger using the AD:TRIGger[:SOURce] command, described on page 118, or the DA:TRIGger[:SOURce] command, described on page 161.</p>

Possible Errors -114 "Header suffix out of range"

If you receive this error, ensure that you specify a value of 0 to 7 for the header suffix *n*.

Example This example returns the configuration of LXI line 0 of the Trigger Bus; in this case LXI line 0 is driven (enabled):

```
>:WTB:LXI0:ENABle?  
< 1
```

See Also WTB:LXI<n>[:OUTput]:ENABle, described on [page 171](#)

AD:CLOCK:SOURce, described on [page 109](#)

DA:CLOCK:SOURce, described on [page 157](#)

AD:TRIGger[:SOURce], described on [page 118](#)

DA:TRIGger[:SOURce], described on [page 161](#)

EVENT Subsystem Command

The EVENT subsystem includes the commands listed in [Table 14](#) used to manage LXI-defined LAN events.

You can configure a DT8837 instrument module to generate LAN events. Specific LAN events can be received by the DT8837 instrument module and used to trigger either the analog input and/or the analog output subsystems. There are 8 LAN event configurations under this subsystem: LAN0, LAN1, LAN2, LAN3, LAN4, LAN5, LAN6, and LAN7. In addition, there are a set of configuration parameters that apply to the entire collection of configurations. This section describes the commands and queries of the EVENT subsystem in detail.

Table 14: EVENT Subsystem Commands

Name	Mnemonic	See
LAN Event Domain Configuration	EVENT:DOMain	page 175
LAN Event Domain Query	EVENT:DOMain?	page 175
LAN Event Name Configuration	EVENT:LAN<n>:NAME	page 176
LAN Event Name Query	EVENT:LAN<n>:NAME?	page 177
LAN Event Transmission Enable	EVENT:LAN<n>:TRANsmit[:ENABle]	page 178
LAN Event Transmission Enable Query	EVENT:LAN<n>:TRANsmit[:ENABle]?	page 179

LAN Event Domain Configuration

Description	Sets the value of the LXI domain octet that is transmitted and received in all LXI LAN trigger event packets. The domain setting is used in all the event configurations (LAN0 to LAN7).
Syntax	:EVENT:DOMain <value>
Required Parameters	
Name:	value
Data Format:	<0+NR1>
Description:	The value of the LXI domain octet that is transmitted and received in all LXI LAN event packets. Valid values range from 0 to 255; the default value is 0.
Optional Parameters	None
Response Data	None
Notes	<p>The LAN even domain setting is saved in nonvolatile memory.</p> <p>A transmitted LAN event carries this domain setting. Any received LAN event that does not have this domain setting is ignored.</p>
Possible Errors	<p>-284 "Program currently running"</p> <p>If you receive this error, call the [AD DA:]ABORt command to stop the operation before configuring the LAN trigger event packet.</p>
Example	<p>This example configures the value of the LXI octet that is transmitted in all the LXI LAN event packets to 1:</p> <pre>> :EVENT:DOM 1</pre>
See Also	EVENT:DOMain?, described on page 175

LAN Event Domain Query

Description	Returns the value of the LXI domain octet that is transmitted and received in all LXI LAN event packets. The domain setting is used in all the event configurations (LAN0 to LAN7).
Syntax	:EVENT:DOMain?
Required Parameters	None
Optional Parameters	None
Response Data	<value>
Name:	value
Data Format:	<0+NR1>
Description:	The value of the LXI domain octet that is transmitted and received in all LXI LAN event packets. Valid values range from 0 to 255; the default value is 0.

Notes A transmitted LAN event carries this domain setting. Any received LAN event that does not have this domain setting is ignored.

Example This example returns the currently configured value of the LXI octet that transmitted in all the LXI LAN event packets; in this case, the value is 1:

```
> :EVENT:DOM?
< 1
> :EVENT:DOM 3
> :EVENT:DOM?
< 3
```

See Also EVENT:DOMain, described on [page 175](#)

LAN Event Name Configuration

Description Configures the LAN event ID field in the LXI LAN packet.

Syntax :EVENT:LAN<n>:NAME <event name>

Required Numeric Suffix

Name: n

Description: Specifies which of the LAN events to configure. Values range from 0 to 7.

Required Parameters None

Optional Parameters

Name: event name

Data Format: <CHARACTER PROGRAM DATA>

Description: A string with a maximum length of 16 ASCII characters. Event names longer than 16 ASCII characters are truncated to the first 16 characters.

If this parameter is omitted, the event name is configured to be the default string that corresponds to the specified LAN event, as follows:

LAN Events	<i>event name</i>
0	LAN0
1	LAN1
2	LAN2
3	LAN3
4	LAN4
5	LAN5

Description (cont.):	6	LAN6
	7	LAN7
Response Data	None	
Notes	<p>A LAN event that is transmitted from the instrument module has this value in the <i>event ID</i> field.</p> <p>LAN events that are received by the instrument module must have this value in the <i>event ID</i> field, otherwise the event is ignored.</p>	
Example	<p>This example specifies the LAN event name for LAN event configuration 3 as MYLAN3:</p> <pre>>:EVENT:LAN3:NAM MYLAN3</pre>	
Possible Errors	<p>-284 "Program currently running"</p> <p>If you receive this error, call the [AD DA:]ABORT command to stop the operation before configuring the LAN trigger event packet.</p> <p>-114 "Header suffix out of range"</p> <p>If you receive this error, ensure that you specify a value of 0 to 7 for the header suffix <i>n</i>.</p>	
See Also	EVENT:LAN<n>:NAME?, described on page 177	

LAN Event Name Query

Description	Returns the LAN event ID field in the LXI LAN packet.
Syntax	:EVENT:LAN<n>:NAME?
Required Numeric Suffix	
Name:	n
Description:	Specifies the LAN event for which to return the event name. Values range from 0 to 7.
Required Parameters	None
Optional Parameters	None
Response Data	<event name>

Name: event name

Data Format: <CHARACTER RESPONSE DATA>

Description: A string with a maximum length of 16 ASCII characters. Event names longer than 16 ASCII characters are truncated to the first 16 characters.

If this parameter is omitted, the event name is configured to be the default string that corresponds to the specified LAN event, as follows:

LAN Event	<i>event name</i>
0	LAN0
1	LAN1
2	LAN2
3	LAN3
4	LAN4
5	LAN5
6	LAN6
7	LAN7

Notes A LAN event that is transmitted from the instrument module has this value in the *event ID* field.

LAN events that are received by the instrument module must have this value in the *event ID* field, otherwise they the event is ignored.

Example This example returns the currently configured event name for LAN event configuration 3; in this case, "MYLAN3" is the event name:

```
>:EVENT:LAN3:NAM?
< MYLAN3
```

Possible Errors -114 "Header suffix out of range"

If you receive this error, ensure that you specify a value of 0 to 7 for the header suffix *n*.

See Also EVENT:LAN<n>:NAME, described on [page 176](#)

LAN Event Transmission Enable

Description Enables or disables a specific LAN event for transmission from the instrument module.

Syntax :EVENT:LAN<n>:TRANsmitt[:ENABle] <state>

Required Numeric Suffix

Name: n

Description: Specifies which of the LAN events to enable or disable. Values range from 0 to 7.

Required Parameters

Name: state

Data Format: <Boolean>

Description: Specifies whether to enable or disable the specific LAN event for transmission from the instrument module.

To enable this capability, set this value to ON or 1. To disable this capability, set this value to OFF or 0.

Optional Parameters None

Response Data None

Notes More than one LAN event configuration can be enabled for transmission. A LAN event configuration is implicitly enabled for reception using the **AD:TRIGger[:SOURce]** command, described on [page 118](#), or the **DA:TRIGger[:SOURce]** command, described on [page 161](#).

Possible Errors -114 "Header suffix out of range"

If you receive this error, ensure that you specify a value of 0 to 7 for the header suffix *n*.

-284 "Program currently running"

If you receive this error, call the **[AD|DA:]ABORt** command to stop the operation before configuring the LAN trigger event packet.

Example This example enables LAN event configuration 3:

```
>:EVEN:LAN3:TRAN 1
```

See Also **EVENT:LAN<n>:TRANsmit[:ENABle]?**, described on [page 179](#)

{AD|DA}:TRIGger[:SOURce], described on [page 118](#)

LAN Event Transmission Enable Query

Description Returns whether a specific LAN event is enabled or disabled for transmission from the instrument module when the analog input subsystem is triggered.

Syntax **:EVENT:LAN<n>:TRANsmit[:ENABle]?**

**Required Numeric
Suffix**

Name: n

Description: Specifies the LAN event for which to return the enable/disable state. Values range from 0 to 7.

Required Parameters None**Optional Parameters** None**Response Data** <state>

Name: state

Data Format: <Boolean>

Description: Returns whether the specific LAN event is enabled or disabled for transmission from the instrument module when the analog input subsystem is triggered.

If this capability is enabled, the returned value is 1. If this capability is disabled, the returned value is 0.

Notes More than one LAN event configuration can be enabled for transmission. A LAN event configuration is implicitly enabled for reception using the {AD|DA}:TRIGger[:SOURce] command, described on [page 118](#), and is used to trigger the analog input and/or analog output subsystems.**Example** This example returns the currently configured state of LAN configuration 3; in this case, this event configuration is enabled.

```
> :EVEN:LAN3:TRAN?  
> 1
```

Possible Errors -114 "Header suffix out of range"If you receive this error, ensure that you specify a value of 0 to 7 for the header suffix *n*.**See Also** EVENT:LAN<n>:TRANsmit[:ENABle], described on [page 178](#)
{AD|DA}:TRIGger[:SOURce], described on [page 118](#)

DOUTput Subsystem Commands

The DOUTput subsystem includes the commands listed in [Table 15](#). Use these commands when updating the digital output port on a DT8837 LXI instrument modules. This section describes each of these commands in detail.

Table 15: Digital OUTPut Subsystem Commands

Name	Mnemonic	See
Digital Output AND Output Values	DOUTput:AND	page 182
Digital Output OR Output Values	DOUTput:OR	page 183
Digital Output Set State	DOUTput	page 184
Digital Output Query State	DOUTput?	page 184

Digital Output AND Output Values

Description Compares the binary representations of a specified value and the current value of the digital output port, and performs a logical AND operation on each pair of corresponding bits.

If both bits are 1, the result is 1. If both bits are 0, the result is 0. If one bit is 1 and the other bit 0, the result is 0.

Syntax :DOUTput:AND <value>

Required Parameters

Name: value

Data Format: <0+NR1>

Description: A value between 0 and 15 that is ANDed with the current value of the digital output port.

This value represents the weighted bit value of the digital output port, where the value of bit 0 (digital output line 0) corresponds to a decimal value of 1 (2^0) if the bit is set, and the value of bit 3 (digital output line 3) corresponds to a decimal value of 8 (2^3) if the bit is set.

Setting a bit closes the relay contact that corresponds to the digital output line.

Response Data None

Notes You can use this command along with the **DOUTput** command, described on [page 184](#), and the **DOUTput:OR** command, described on [page 183](#), to change the state of certain digital output lines without affecting the state of other digital output lines.

Refer to the *1999 SCPI Data Exchange Format*, section 3.4.6.

Example Assume that the current value of the digital output port is decimal 8 (binary 1000). The following example ANDs this value with decimal value 1 (binary 0001):

```
> :DOUT:AND 1
> :DOUT?
< 0
```

The result is decimal value 0 (binary 0000), which opens the relay contacts associated with digital output lines 0, 1, 2, and 3.

See Also **DOUTput**, described on [page 184](#)

DOUTput:OR, described on [page 183](#)

DOUTput?, described on [page 184](#)

Digital Output OR Output Values

Description Compares the binary representations of a specified value and the current value of the digital output port, and performs a logical OR operation on each pair of corresponding bits.

If both bits are 1, the result is 1. If both bits are 0, the result is 0. If one bit is 1 and the other bit 0, the result is 1.

Syntax :DOUTput:OR <value>

Required Parameters

Name: value

Data Format: <0+NR1>

Description: A value between 0 and 15 that is ORed with the current value of the digital output port.

This value represents the weighted bit value of the digital output port, where bit 0 (digital output line 0) corresponds to a decimal value of 1 (2^0) if the bit is set, and the bit 3 (digital output line 3) corresponds to a decimal value of 8 (2^3) if the bit is set.

Setting a bit closes the relay contact that corresponds to the digital output line.

Response Data None

Notes You can use this command along with the **DOUTput** command, described on [page 184](#), and the **DOUTput:AND** command, described on [page 183](#), to change the state of certain digital output lines without affecting the state of other digital output lines.

Refer to the *1999 SCPI Data Exchange Format*, section 3.4.6.

Example Assume that the current value of the digital output port is decimal 8 (binary 1000). The following example ORs this value with decimal value 1 (binary 0001):

```
> :DOUT:OR 1
> :DOUT?
< 9
```

The result is decimal value 9 (binary 1001), which closes the relay contacts associated with digital output lines 0 and 3 and opens the relay contact associated with digital output lines 1 and 2.

See Also **DOUTput**, described on [page 184](#)
DOUTput:AND, described on [page 182](#)
DOUTput?, described on [page 184](#)

Digital Output Set State

Description Closes or opens the relay contacts associated with the digital output port on a DT8837 LXI instrument module.

Syntax :DOUTput <value>

Required Parameters

Name: value

Data Format: <0+NR1>

Description: The weighted bit value of the digital output port, where the value of bit 0 (digital output line 0) corresponds to a decimal value of 1 (2^0) if the bit is set, and the value of bit 3 (digital output line 3) corresponds to a decimal value of 8 (2^3) if the bit is set.

Values range from 0 to 15. Setting a bit closes the relay contact that corresponds to the digital output line.

Response Data None

Notes Refer to the *1999 SCPI Data Exchange Format*, section 3.4.6.

Example The following example closes the relay contacts associated with digital output lines 0, 2, and 3 and opens the relay contact associated with digital output line 1:

```
:DOUT 13
```

See Also **DOUTput?**, described on [page 184](#)
DOUTput:AND, described on [page 182](#)
DOUTput:OR, described on [page 183](#)

Digital OUTput Query State

Description Returns the current state of the relay contacts that are associated with the digital output port on a DT8837 LXI instrument module.

Syntax :DOUTput?

Parameters None

Response Data <value>

Name: value

Data Format: <0+NR1>

Description: The weighted bit value of the digital output port, where the value of bit 0 (digital output line 0) corresponds to a decimal value of 1 (2^0) if the bit is set, and the value of bit 3 (digital output line 3) corresponds to a decimal value of 8 (2^3) if the bit is set.

Values range from 0 to 15. Setting a bit closes the relay contact that corresponds to the digital output line.

Notes Refer to the *1999 SCPI Data Exchange Format*, section 3.4.6.

Example > :DOUT?
 < 15

This response indicates that all the relay contacts of the digital output port are closed.

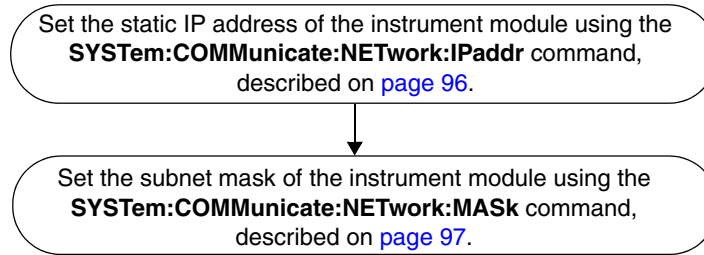
See Also DOUTput, described on [page 184](#)
 DOUTput:AND, described on [page 182](#)
 DOUTput:OR, described on [page 183](#)



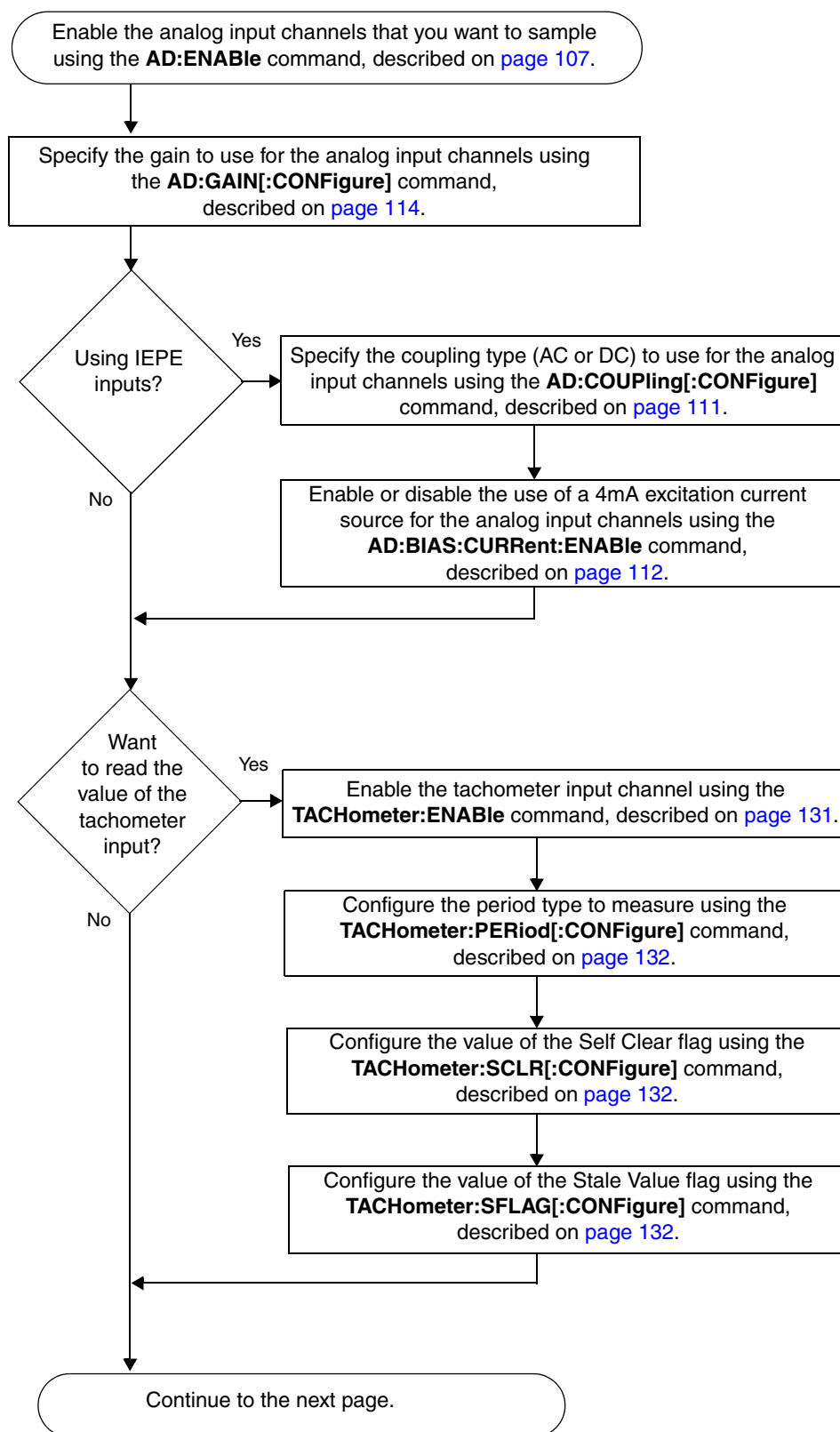
Programming Flowcharts

Configuring the Instrument Module on the LAN.....	188
Performing Input Operations.....	189
Performing Analog Output Operations.....	192
Performing Digital Output Operations.....	193

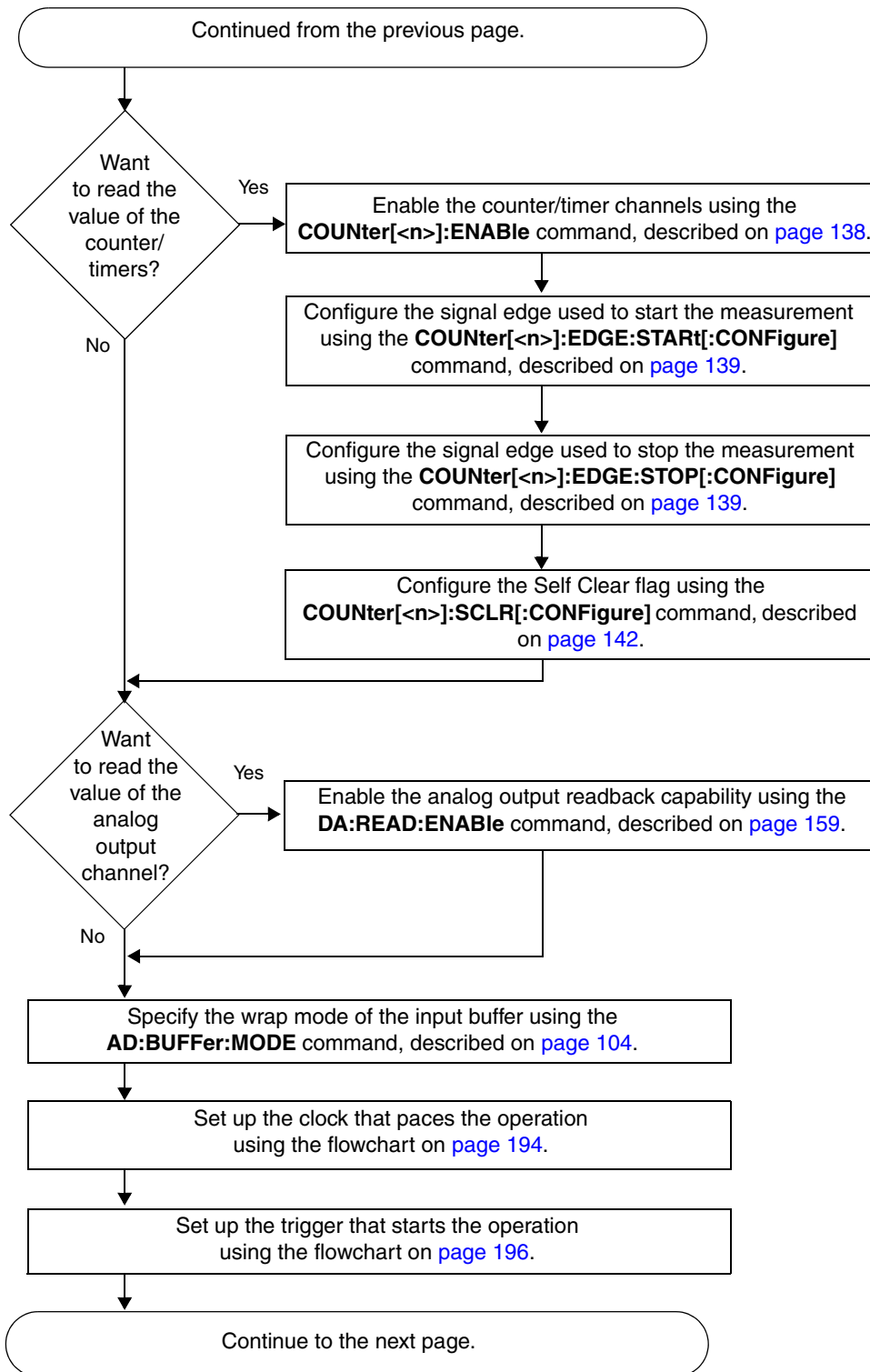
Configuring the Instrument Module on the LAN



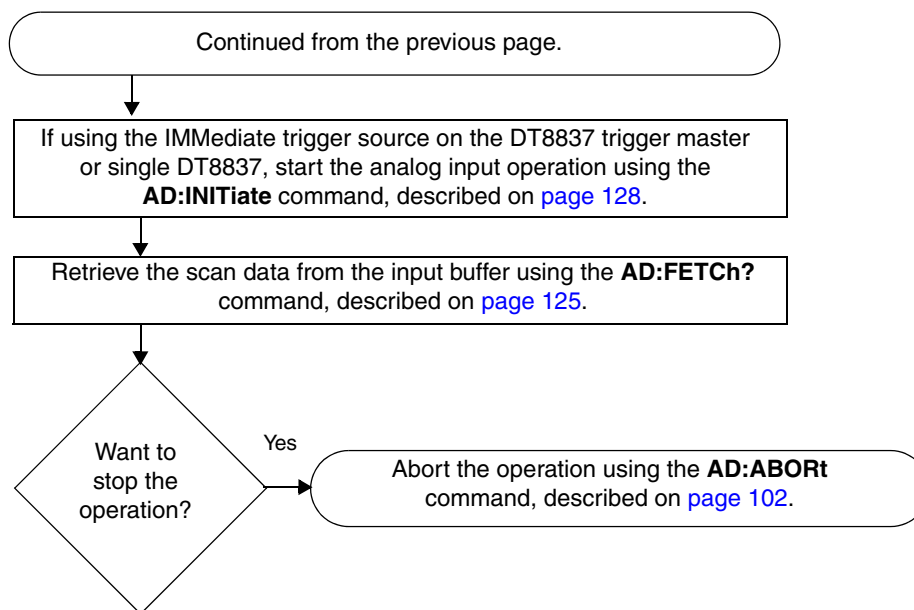
Performing Input Operations



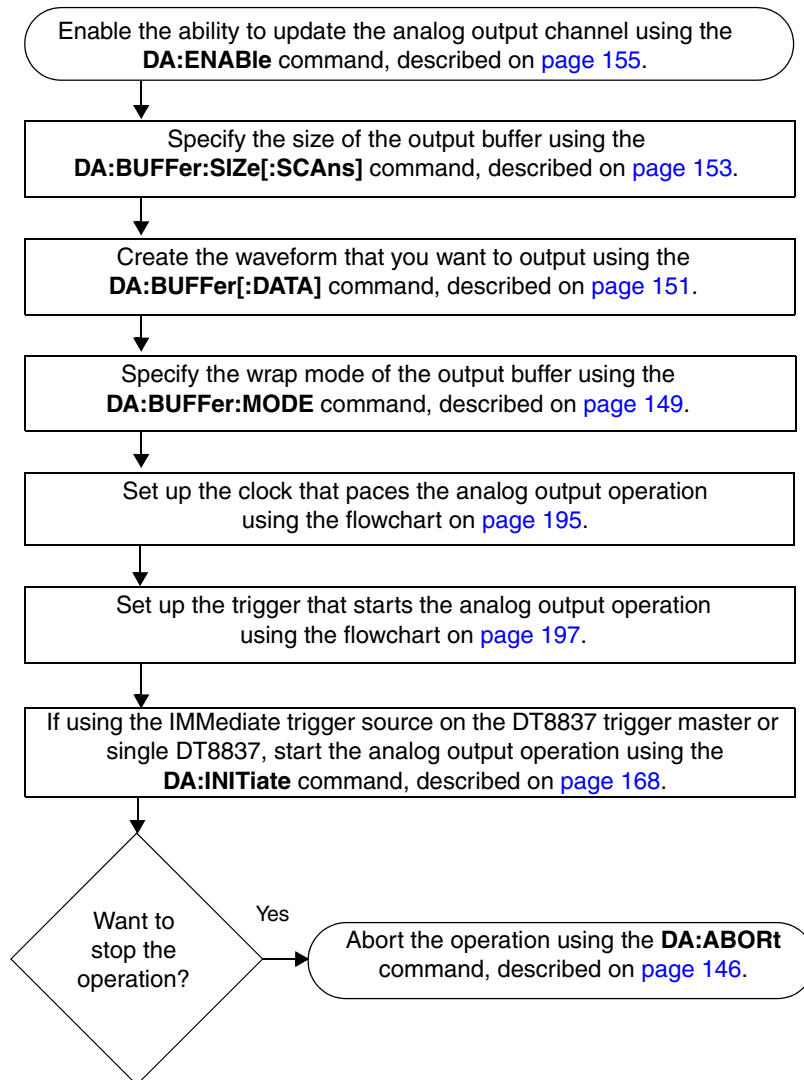
Performing Input Operations (cont.)



Performing Input Operations (cont.)



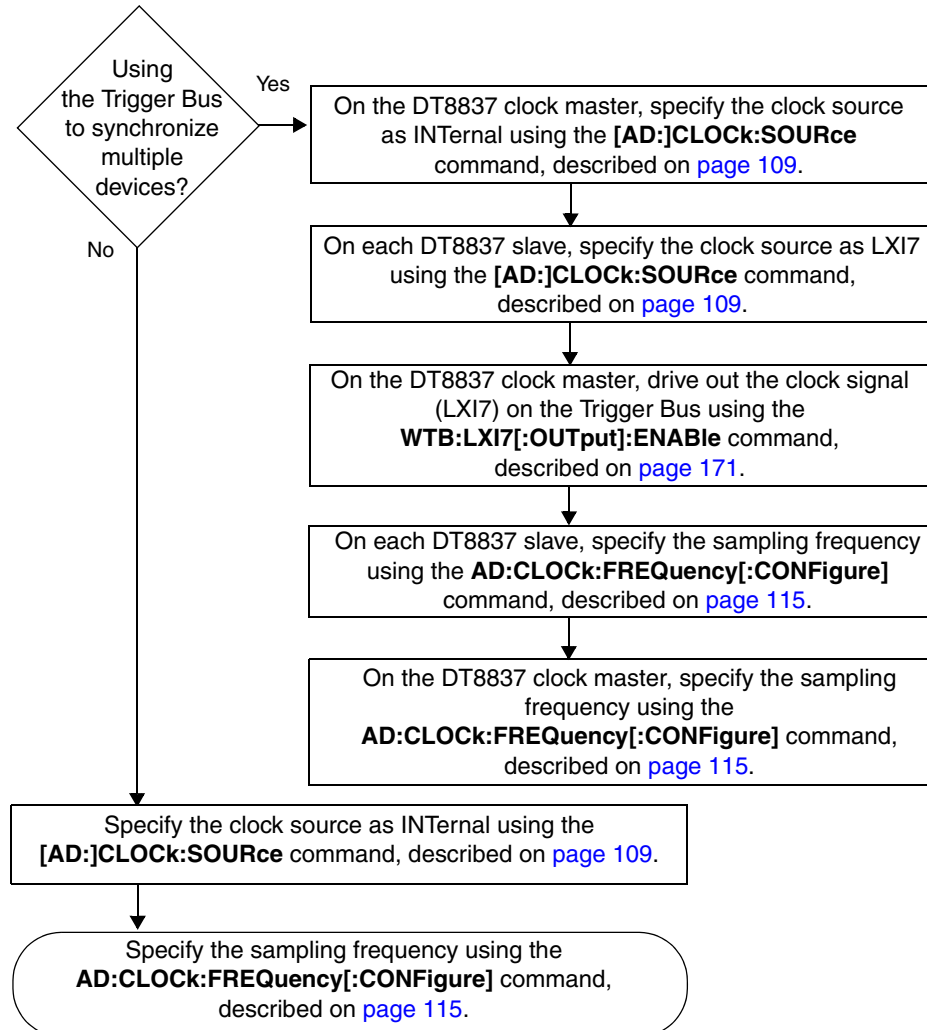
Performing Analog Output Operations



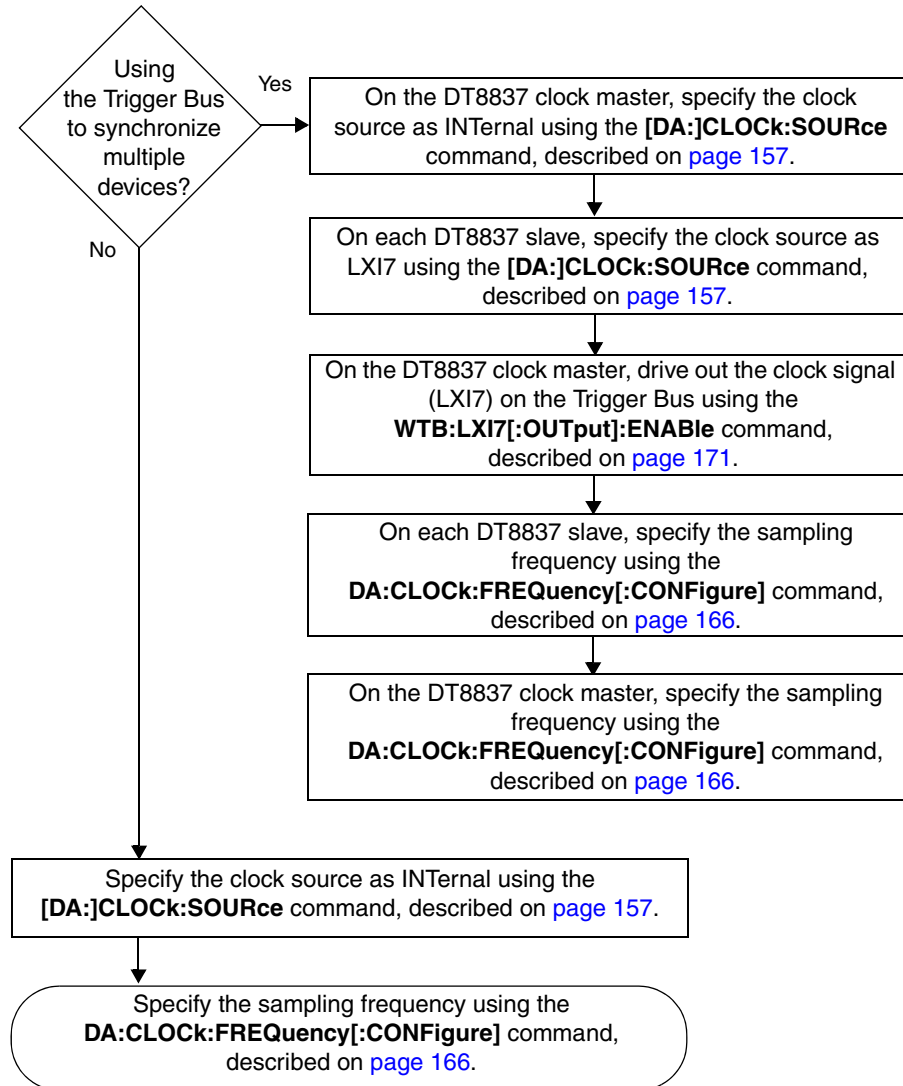
Performing Digital Output Operations

Set the state of the digital output port using the **DOUTput** command, described on [page 184](#).

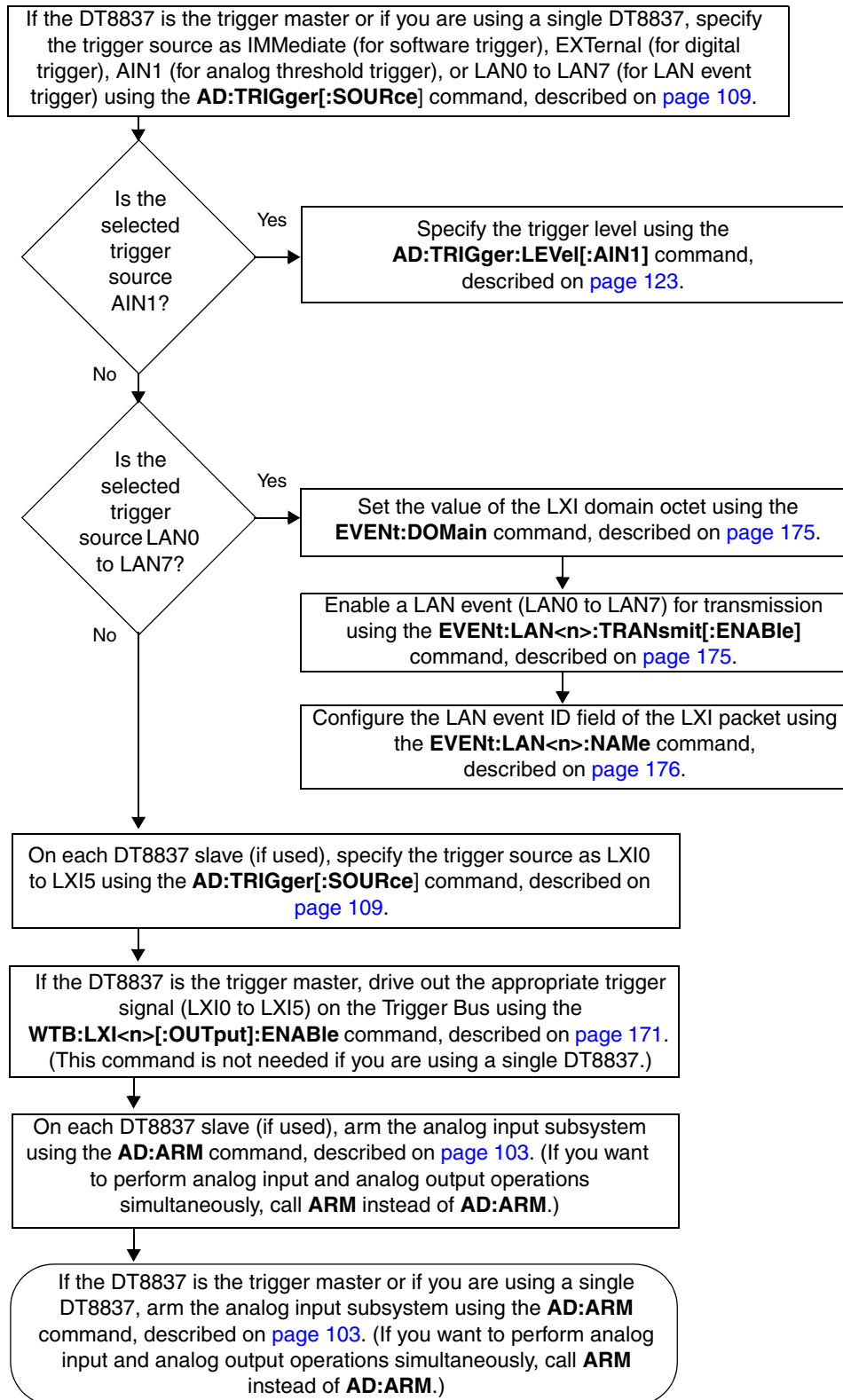
Setting up the Clock for Input Operations



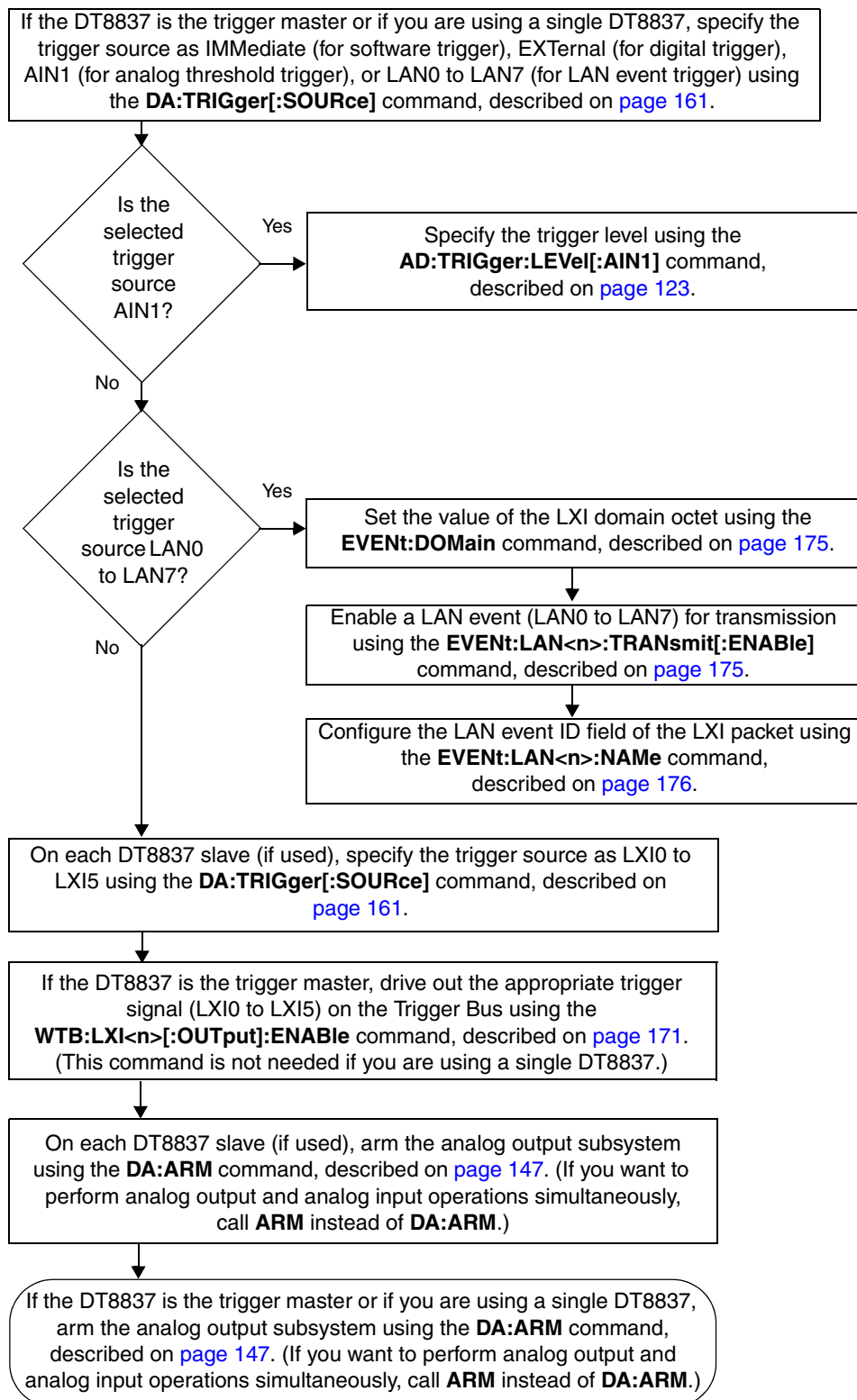
Setting up the Clock for Analog Output Operations



Setting up the Trigger for Input Operations



Setting up the Trigger for Analog Output Operations





Product Support

Should you experience problems using SCPI to program a DT8837 LXI instrument module, follow these steps:

1. Read all the appropriate sections of this manual. Make sure that you have added any “Read This First” information to your manual and that you have used this information.
2. Check for a README file on the DT8837 CD. If present, read this file for the latest installation and usage information.
3. Check that you have installed your hardware devices properly. For information, refer to the documentation supplied with your devices.
4. Check that you have installed the device drivers for your hardware devices properly. For information, refer to the documentation supplied with your devices.
5. Check that you have installed your software properly.

If you are still having problems, Data Translation’s Technical Support Department is available to provide technical assistance. To request technical support, go to our web site at <http://www.mccdaq.com> and click on the Support link.

When requesting technical support, be prepared to provide the following information:

- Your product serial number
- The hardware/software product you need help on
- The version of the CD you are using
- Your contract number, if applicable

If you are located outside the USA, contact your local distributor; see our web site (www.mccdaq.com) for the name and telephone number of your nearest distributor.



Errors

Error Codes 202

Troubleshooting Errors 204

Error Codes

Use the **SYSTem:ERRor:ALL?** command, described on [page 91](#), to return the errors that have occurred.

Errors are returned in your program as follows:

```
-110, "Command header error"
```

The number represents the error code and the string that follows represents the error description. Note that there is no space between the comma after error code and the quotation mark before the error description.

[Table 16](#) lists the error codes that DT8837 LXI instrument modules can return.

Table 16: SCPI Error Codes that DT8837 LXI Instrument Modules Can Return

Error Code	Error Message	Description
0	No error	Normal operation; no error occurred.
-100	Command error	A command is missing a parameter, or the command contains too many parameters or too many dimensions.
-102	Syntax error	An unrecognized command or data type was encountered; for example, a string was received when the instrument module does not accept strings.
-104	Data type error	A data element different than the one allowed was encountered; for example, numeric data was expected but string data was encountered.
-110	Command header error	The command header is invalid. See page 204 for information on troubleshooting this error.
-114	Header suffix out of range	The value for the header suffix is out of range.
-115	Unexpected number of parameters	The number of parameters received does not correspond to the number of parameters that were expected.
-120	Numeric data error	The value of a parameter overflowed, has the wrong dimensions, or contains an invalid value.
-131	Invalid suffix	The suffix does not follow the syntax described in IEEE 488.2, 7.7.3.2, or the suffix is inappropriate for this instrument module.
-200	Execution error	A <PROGRAM DATA> element following a header was evaluated by the instrument module as outside of its legal input range or is otherwise inconsistent with the instrument module's capabilities. A valid program message could not be executed properly due to some condition of the instrument module.
-221	Settings conflict	The specified socket cannot be configured because it is already connected.

Table 16: SCPI Error Codes that DT8837 LXI Instrument Modules Can Return (cont.)

Error Code	Error Message	Description
–222	Data out of range	A legal program data element was parsed but could not be executed because the interpreted value was outside the legal range for the instrument module.
–284	Program currently running	Certain operations dealing with programs are illegal while the program is running; for example, you cannot configure/reconfigure an operation while a scan is in progress.
–350	Queue overflow	The error queue is full; subsequent errors can not be added to the queue. Use the *CLS command, described on page 70 , to clear the error queue as well as the Status Byte register
–410	Query interrupted	The query did not complete. See page 205 for information on troubleshooting this error.

Troubleshooting Errors

This section describes how to troubleshoot the following frequently encountered errors:

- -110,"Command header error"
- -410,"Query interrupted"

Error -110 Command Header Error

This error indicates that the command you sent was not recognized by the instrument module as a valid command name. Here are the most likely causes for receiving this error:

- A space is missing between the command and its parameter. At least one space (blank) must exist between the command and its parameter.

For example, this command will not be recognized and will return error -110:

`:AD:BUFFer:MODEWRAP` *wrong*

This command, however, is correct and will not return an error:

`:AD:BUFFer:MODE WRAP` *right*

- The command was specified with the improper short or long form.

For example, you can specify `:AD:BUFF:MODE` but not `:AD:BUF:MODE` for the `:AD:BUFFer:MODE` command.

Refer to [Chapter 3](#) starting on [page 69](#) and [Chapter 4](#) starting on [page 79](#), for the correct command name.

- A space (blank) is inserted within the command name. Spaces are not permitted within the command name.

For example, this command is incorrect and will return error -110:

`:SYST: ERR?` *Wrong*

This command, however, is correct and will not return an error:

`:SYST:ERR?` *Right*

- The instrument module did not return data. To return data from the instrument module, the client must send a valid query to the instrument module. For example, to return scan data, the client must issue the `:AD:FETCh?` command once the scan has been initiated.
- A client sent a valid query following an invalid command. This can occur when you send multiple commands or queries within one program message. When it detects an error in a program message, the instrument module discards all further commands in the program message until the end of the string.

For example, consider the following program message:

`*IDN?;*xyz;*STB?`

Because the `*xyz` command generates error -110, all commands on this program line fail, even though `*IDN?` and `*STB?` are valid queries.

Error –410 Query Interrupted

This error applies to :INSTR connections, not to :SOCKET connections.

Usually, this error occurs when a client sends a valid query to the instrument module, and then sends another command or query to the instrument module before reading the response from the first query.

For example, the following sequence of commands will cause error –410 because the response from :SYST:ERR? is not read before the *OPC command is sent to the instrument module:

```
:SYST:ERR?  
*OPC?
```




Registers

Status Byte Register (STB).....	208
Standard Event Status Enable Register (ESE)	209
Standard Event Status Register (ESR)	211
Operation Status Register	212

Status Byte Register (STB)

Table 17 lists the bits of the Status Byte register.

Table 17: Status Byte Register

Bit	Name	Description
0	Unused	The value of this bit is always 0.
1	Unused	The value of this bit is always 0.
2	Error / Event Queue Summary (EAV)	<p>During an operation, the DT8837 LXI instrument module stores error conditions as they occur in an Error / Event Queue.</p> <p>If this bit is 1, the queue is not empty. To read the error message and empty the queue, use the SYST:ERR? command, described on page 91.</p> <p>If this bit is 0, the Error / Event Queue is empty.</p>
3	Questionable Register Summary	DT8837 LXI instrument modules do not implement the Questionable Data/Signal Status Register register. Therefore, the value of this bit is always 0.
4	Message Available (MAV)	<p>The Output Queue stores response messages until they are read.</p> <p>If this bit is 1, an unread message exists. The DT8837 LXI instrument module places Data Byte and END messages into the Output Queue in response to query commands. These messages are removed from the Output Queue as they are read by the controller. As long as the Output Queue contains an unread message, the value of the MAV bit is 1.</p> <p>If this bit is 0, an unread message does not exist in the Output Queue.</p>
5	Standard Event Status Bit (ESB) Summary	<p>If this bit is 1, one or more bits in the Standard Event Status register, described on page 211, is set.</p> <p>If this bit is 0, no bits are in Standard Event Status register are set.</p>
6	Request Service	DT8837 LXI instrument modules do not implement the Request Service register. Therefore, the value of this bit is always 0.
7	Operation Status Register Summary	<p>If this bit is 1, one or more bits in the device-dependent Operation Status register, described on page 212, is set.</p> <p>If this bit is 0, no bits in the Operation Status register are set.</p>

Standard Event Status Enable Register (ESE)

Figure 21 shows the relationship between the Standard Event Status Enable and Standard Event Status registers.

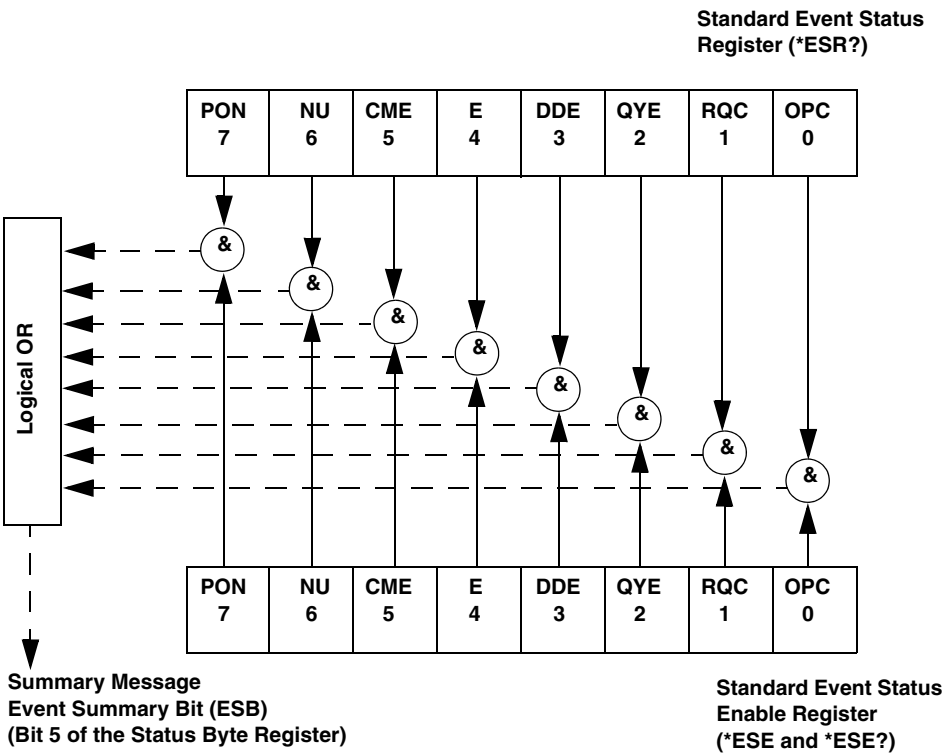


Figure 21: Standard Event Status Enable (ESE) and Standard Event Status Registers (ESR)

Table 18 lists the bits of the Standard Event Status Enable (ESE) register.

Table 18: Standard Event Status Enable (ESE) Register

Bit	Name	Description
0	Operation Complete (OPC)	<p>This bit is always 1 on DT8837 LXI instrument modules, since the instrument module is always enabled to complete all pending overlapped commands.</p> <p>Overlapped commands are executed in parallel with subsequent commands that are sent to the DT8837 LXI instrument module.</p>
1	Request Control (RQC)	This bit is always 0 on DT8837 LXI instrument modules since the instrument module is not configured to control GPIB operation.
2	Query Error (QYE)	<p>If this bit is set to 1, the Query Error bit (bit 2) of the Standard Event Status register is enabled.</p> <p>If this bit is set to 0, the Query Error bit (bit 2) of the Standard Event Status register is disabled.</p>
3	Device Dependent Error (DDE)	<p>If this bit is set to 1, the Device Dependent Error bit (bit 3) of the Standard Event Status register is enabled.</p> <p>If this bit is set to 0, the Device Dependent Error bit (bit 3) of the Standard Event Status register is disabled.</p>
4	Execution Error (E)	<p>If this bit is set to 1, the Execution Error bit (bit 4) of the Standard Event Status register is enabled.</p> <p>If this bit is set to 0, the Execution Error bit (bit 4) of the Standard Event Status register is disabled.</p>
5	Command Error (CME)	<p>If this bit is set to 1, the Command Error bit (bit 5) of the Standard Event Status register is enabled.</p> <p>If this bit is set to 0, the Command Error bit (bit 5) of the Standard Event Status register is disabled.</p>
6	Not Used (NU)	The value of this bit is always 0.
7	Power ON (PON)	<p>If this bit is set to 1, the Power ON bit (bit 7) of the Standard Event Status register is enabled.</p> <p>If this bit is set to 0, the Power ON bit (bit 7) of the Standard Event Status register is disabled.</p>

Standard Event Status Register (ESR)

Table 19 lists the bits of the Standard Event Status register; see Figure 21 on page 209 for more information on how the Standard Event Status Enable register is used with this register.

Table 19: Standard Event Status (ESR) Register

Bit	Name	Description
0	Operation Complete (OPC)	<p>If this bit is set to 1, all pending operations are complete.</p> <p>If this bit is set to 0, all pending operations have not been completed.</p>
1	Request Control (RQC)	Not supported. DT8837 LXI instrument modules are not configured to control GPIB operation. Therefore, the value of this bit is always 0.
2	Query Error (QYE)	<p>If this bit is 1, a query error was detected indicating that an attempt was made to read data from the output queue when no data was present, or that data in the output queue was lost (an overflow condition occurred).</p> <p>If this bit is 0, a query error did not occur.</p>
3	Device Dependent Error (DDE)	<p>If this bit is 1, a device-dependent error was detected. Refer to Appendix A starting on page 201 for a list of device-dependent errors that can be returned.</p> <p>If this bit is 0, a device-dependent error did not occur.</p>
4	Execution Error (E)	<p>If this bit is 1, an Execution Error was detected indicating that a <PROGRAM DATA> element was outside the legal range or was inconsistent with the operation of the DT8837 LXI instrument module, or that the DT8837 LXI instrument module could not execute a valid command due to some internal condition.</p> <p>If this bit is 0, an Execution Error did not occur.</p>
5	Command Error (CME)	<p>If this bit is 1, a Command Error was detected indicating that the DT8837 LXI instrument module received a command that did not follow proper syntax or was misspelled, or that the DT8837 LXI instrument module received a command that was not implemented.</p> <p>If this bit is 0, a Command Error did not occur.</p>
6	Not Used (NU)	The value of this bit is always 0.
7	Power ON (PON)	<p>If this bit is 1, power to the DT8837 LXI instrument module was turned OFF and then ON since the last time the Power ON register was read.</p> <p>If this bit is 0, power to the DT8837 LXI instrument module was always ON between reads of the Power ON register.</p>

Operation Status Register

Table 20 lists the bits of the Operation Status register. The logical OR of all bits is reported in bit 7 (Operation Status Register summary) of the Status Byte (STB) register, described on [page 208](#).

Table 20: Operation Status Register

Bit	Name	Description
0	Unused	The value of this bit is always 0.
1	Unused	The value of this bit is always 0.
2	Unused	The value of this bit is always 0.
3	Unused	The value of this bit is always 0.
4	A/D Measuring	If this bit is 1, the analog input subsystem is actively measuring. If this bit is 0, the analog input subsystem is not actively measuring.
5	A/D Waiting for Trigger	If this bit is 1, the analog input subsystem is waiting for a trigger. If this bit is 0, the analog input subsystem is not waiting for a trigger.
6	A/D Waiting for Arm	If this bit is 1, the analog input subsystem is waiting to be armed. If this bit is 0, the analog input subsystem is not waiting to be armed.
7	Unused	The value of this bit is always 0.
8	D/A Outputting	If this bit is 1, the analog output system is actively outputting data. If this bit is 0, the analog output subsystem is not actively outputting data.
9	D/A Waiting for Trigger	If this bit is 1, the analog output subsystem is waiting for a trigger. If this bit is 0, the analog output subsystem is not waiting for a trigger.
10	D/A Waiting for Arm	If this bit is 1, the analog output subsystem is waiting to be armed. If this bit is 0, the analog output subsystem is not waiting to be armed.
11	Unused	The value of this bit is always 0.
12	Unused	The value of this bit is always 0.

Table 20: Operation Status Register (cont.)

Bit	Name	Description
13	Unused	The value of this bit is always 0.
14	Unused	The value of this bit is always 0.
15	Unused	The value of this bit is always 0.



Examples

Features of the DT8837 SCPI Example Program	217
Opening and Running the DT8837 SCPI Example Program	218

The DT8837 SCPI example program illustrates how to use SCPI commands to program a DT8837 instrument module.

This example was written using Visual C++ in Microsoft Visual Studio.NET 2003, 2005, and 2008, and Visual C# in Microsoft Visual Studio.NET 2005 and 2008.

Features of the DT8837 SCPI Example Program

Using the DT8837 SCPI example program, you can do the following:

- Connect to a DT8837 instrument module by specifying an IP address.
- Configure the following parameters for the analog input subsystem:
 - Enable or disable the ability to acquire data from analog input channels 1 to 4.
 - Specify a gain of 1 or 10 for each analog input channel.
 - Specify AC or DC coupling for each analog input channel.
 - Save the analog input configuration.
- Configure the following parameters for the analog output subsystem:
 - Enable or disable the ability to output data from the analog output channel.
 - Enable or disable the ability to read the value of the analog output channel in the analog input data stream.
 - Load a DA buffer that contains the waveform values that you want to output from the analog output channel.

Ensure that this plain text file (such as .TXT or .CSV) includes one voltage per line, where the number of lines represents the number of points/values that are entered in the buffer. Voltage values should be within the ± 10 V range.
 - Save the analog output configuration.
- Configure the following parameters for the clock:
 - Specify the scan frequency for the analog input subsystem.
 - Specify the output frequency for the analog output subsystem.
 - Save the clock configuration.
- Start acquisition and see the results of each scan.

Note: The tachometer, counters, and the current source for the analog input channels are explicitly disabled in this application. You can enable and configure these features by modifying the application.

Opening and Running the DT8837 SCPI Example Program

To open and run the DT8837 SCPI example program, do the following:

1. Start Microsoft Visual Studio .NET.
2. Click **File**, click **Open**, and then click **Project**.
3. If you are using Visual C# .NET, from the Windows **Start** menu, select the following:
Programs -> Data Translation, Inc -> Instruments -> DT8837 -> SCPI Support -> Examples -> C# -> DT8837 SCPI Examples. sln

If you are using Visual C++ .NET, from the Windows **Start** menu, select the following:
Programs -> Data Translation, Inc -> Instruments -> DT8837 -> SCPI Support -> Examples -> CPP -> DT8837 SCPI Examples. sln

4. From the main menu of Microsoft Visual Studio .NET, click **Build**, and then click **Build Solution** to build the project.
5. To run the example, click **Debug** from the main menu, and then click **Start**.
The example program starts.
6. Use the capabilities of the example program to see how it operates.
7. When you are finished using the example program, click **Debug** from the main menu, then click **Stop Debugging**.
8. View the user interface of the example program by clicking the appropriate [Design] tab on the main window.
9. View the source code for the example program by clicking the appropriate tab (such as example.cs) on the main window.



Using HyperTerminal to Send and Receive SCPI Commands

If you want to use the standard Windows HyperTerminal application to send and receive SCPI commands, do the following:

1. From the **Start** menu, click **Programs -> Accessories -> Communications -> HyperTerminal**.

The HyperTerminal application opens and the following screen appears:

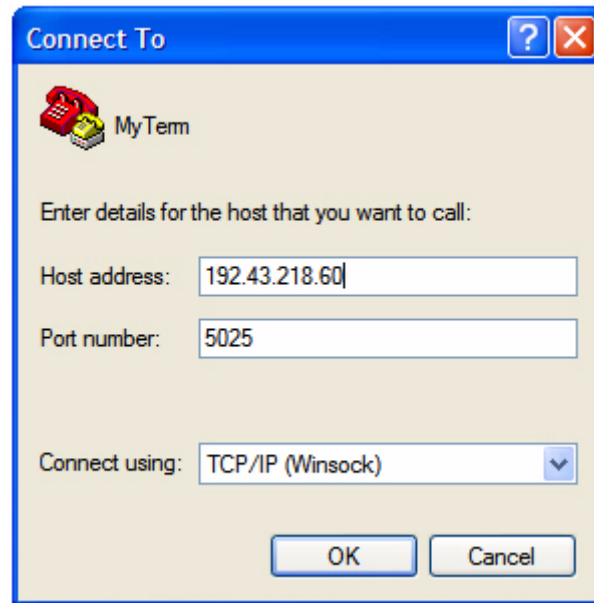


2. Enter a name for the connection, such as **MyTerm**, and click **OK**.
The Connect To dialog appears:

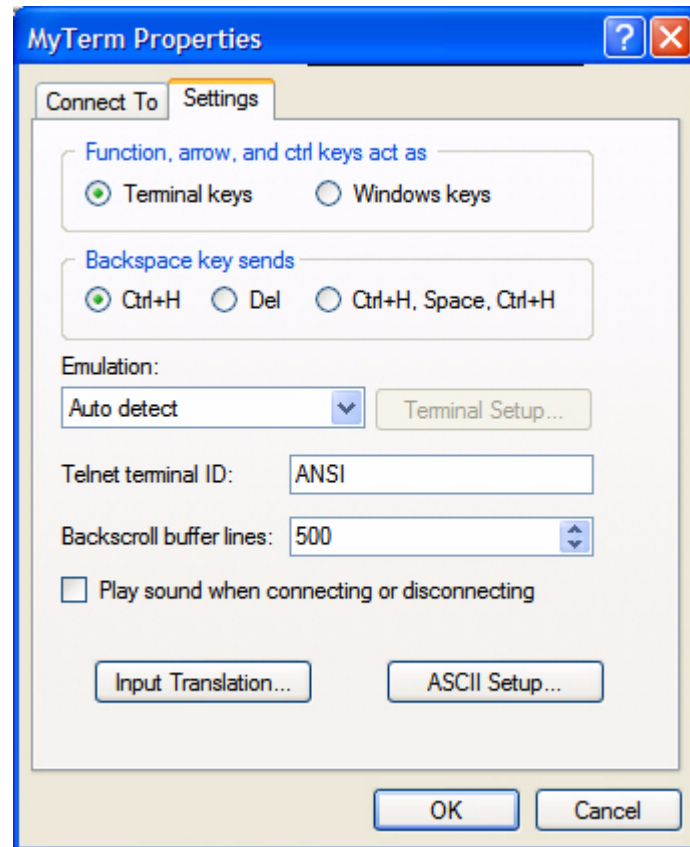


3. In the **Connect using** field, select **TCP/IP (Winsock)**. The other fields on the screen change accordingly.

4. Enter the IP address of the instrument module (which you can locate using the Eureka Discovery Utility), and then enter **5025** as the port number.
The screen appears as follows:

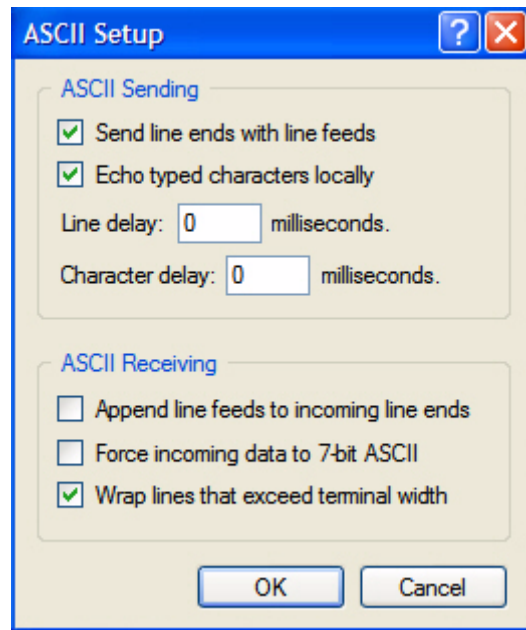


5. Click **OK**.
6. From the File menu, click **Properties**, and then click **Settings**.
The following screen appears:



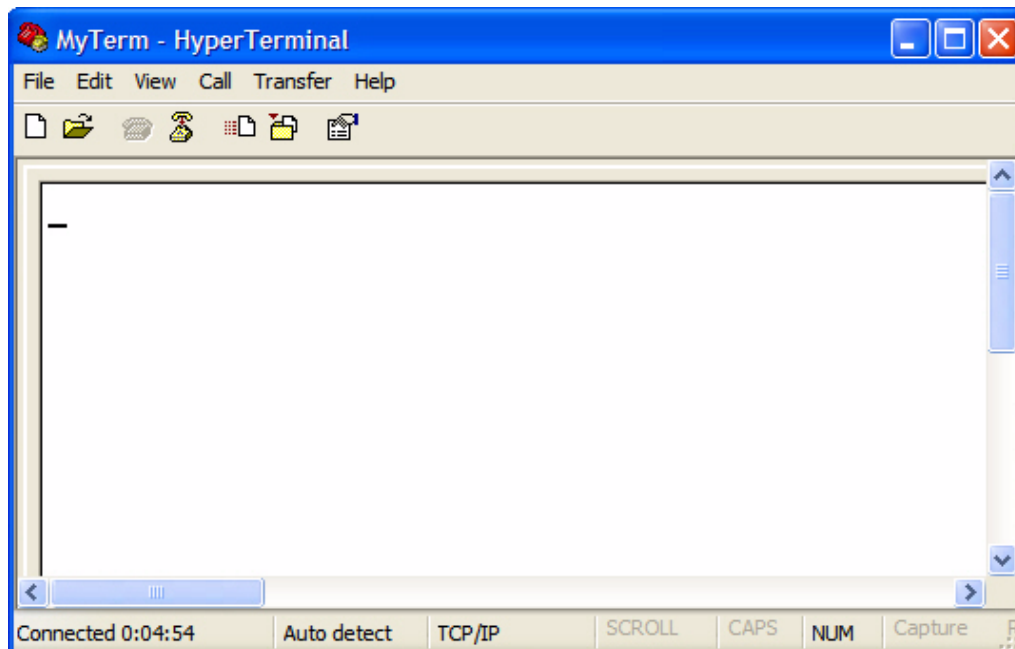
7. Click **ASCII Setup...**
8. Configure this dialog box so that the following options are checked: **Send line ends with line feeds**, **Echo typed characters locally**, and **Wrap lines that exceed terminal width**), and then click **OK**.

The screen should look as follows:



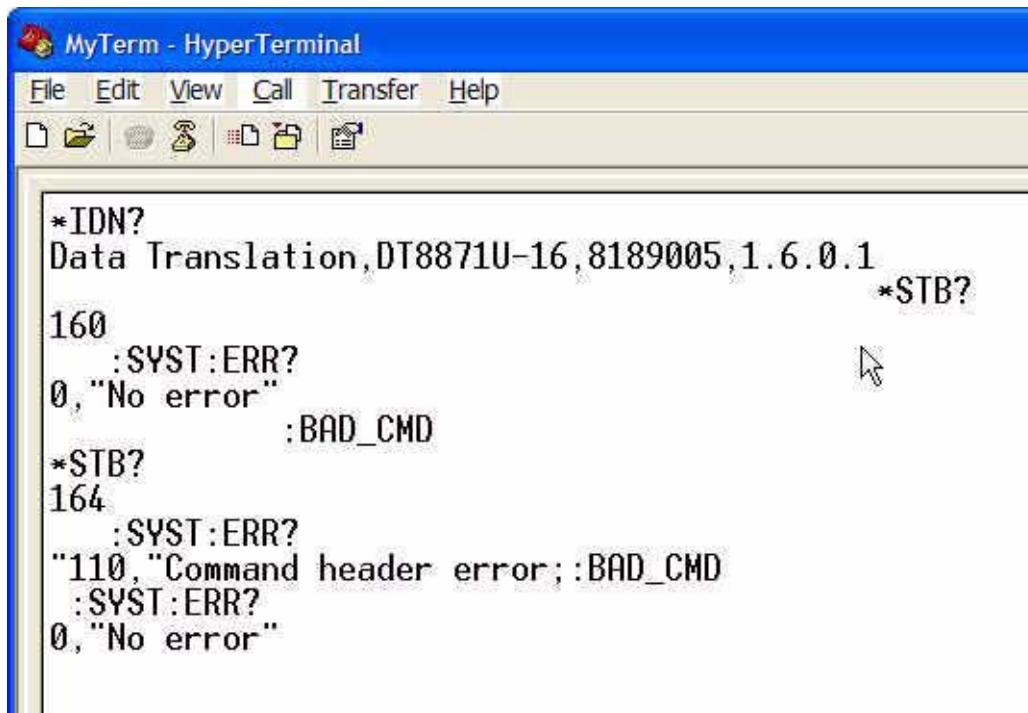
9. Click **OK** to close the Property dialog box.

The following screen appears:



10. From the **Call** menu, click **Call**.

11. Enter any of the documented SCPI commands or queries. The following screen shows an example:



```
*IDN?  
Data Translation,DT8871U-16,8189005,1.6.0.1  
*STB?  
160  
:SYST:ERR?  
0,"No error"  
:BAD_CMD  
*STB?  
164  
:SYST:ERR?  
"110,"Command header error; :BAD_CMD  
:SYST:ERR?  
0,"No error"
```

12. When you are finished using the SCPI commands, select the **Call** menu, and then click **Disconnect** to terminate your connection with the instrument module.

Symbols

[AD:]CLOCK:SOURce 45, 46, 109, 194
[AD:]CLOCK:SOURce? 47, 110
[DA:]ABORt 59, 65
[DA:]ARM 64
[DA:]CLOCK:SOURce 59, 60, 157, 195
[DA:]CLOCK:SOURce? 60, 158
[DA:]INITiate 64
[DA:]TRIGger:SOURce 61, 62, 63
[DA:]TRIGger:SOURce? 61
*CLS 70
*ESE 70
*ESE? 71
*ESR? 71
*IDN? 73
*OPC 74
*OPC? 74
*RST 74
*SRE 75
*SRE? 75
*STB? 76
*TST? 75
*WAI 77

A

A/D sample complete 42
ABORt Analog Input Operation command 102
ABORt Analog Output Operation command 146
AD subsystem commands
 [AD:]CLOCK:SOURce 109
 [AD:]CLOCK:SOURce? 110
 AD:ABORt 102
 AD:ARM 103
 AD:BIAS:CURRent:ENABle 112
 AD:BIAS:CURRent:ENABle? 113
 AD:BUFFer:MODE 104
 AD:BUFFer:MODE? 105
 AD:BUFFer:SIZE[:SCAns]? 106
 AD:CLOCK:FREQuency[:CONFigure] 115
 AD:CLOCK:FREQuency[:CONFigure]? 116
 AD:COUPling[:CONFigure] 111
 AD:COUPling[:CONFigure]? 111
 AD:ENABle 107
 AD:ENABle? 108
 AD:FETCh? 125

 AD:GAIN[:CONFigure] 114
 AD:GAIN[:CONFigure]? 114
 AD:INITiate 128
 AD:STATus:SCAn? 117
 AD:STATus? 118
 AD:SYNc:SOURce? 104
 AD:TRIGger:LEVel[:AIN1] 123
 AD:TRIGger:LEVel[:AIN1]? 124
 AD:TRIGger[:SOURce] 118
 AD:TRIGger[:SOURce]? 121
AD:ABORt 44, 57, 102, 191
AD:ARM 53, 103, 196
AD:BIAS:CURRent:ENABle 40, 112, 189
AD:BIAS:CURRent:ENABle? 40, 113
AD:BUFFer:MODE 44, 104, 190
AD:BUFFer:MODE? 45, 105
AD:BUFFer:SIZE[:SCAns]? 45, 56, 106
AD:CLOCK:FREQuency[:CONFigure] 46, 47, 115, 194
AD:CLOCK:FREQuency[:CONFigure]? 47, 116
AD:COUPling[:CONFigure] 40, 111, 189
AD:COUPling[:CONFigure]? 40, 111
AD:ENABle 39, 107, 189
AD:ENABle? 39, 108
AD:FETCh? 54, 125, 191
AD:GAIN[:CONFigure] 40, 114, 189
AD:GAIN[:CONFigure]? 40, 114
AD:INITiate 53, 128, 191
AD:STATus:SCAn? 57
AD:STATus? 54, 118
AD:SYNc:SOURce? 47, 104
AD:TRIGger:LEVel[:AIN1] 49, 52, 62, 63, 123, 196, 197
AD:TRIGger:LEVel[:AIN1]? 124
AD:TRIGger[:SOURce] 48, 49, 50, 52, 118, 196
AD:TRIGger[:SOURce]? 48, 121
ADC sync signal 45
ADC Synchronization Source Query command 104
AIN1 analog threshold trigger 49, 62
aliasing 47
analog input 39
 arming the operation 53
 configuring IEPE functions 40
 configuring the buffer 44
 configuring the clock 45
 configuring the gain 40
 configuring the input range 40

- configuring the tachometer [40](#)
 - configuring the trigger [48](#)
 - enabling channels [39](#)
 - enabling the analog output readback channel [44](#)
 - enabling the counter/timers [42](#)
 - enabling the tachometer [40](#)
 - retrieving data from the buffer [54](#)
 - starting the operation [53](#)
 - stopping the operation [57](#)
 - Analog Input Buffer Mode Configuration command [104](#)
 - Analog Input Buffer Mode Query command [105](#)
 - Analog Input Buffer Size Query command [106](#)
 - Analog Input Channel 1 Trigger Level Configuration command [123](#)
 - Analog Input Channel 1 Trigger Level Query command [124](#)
 - Analog Input Channel Enable command [107](#)
 - Analog Input Channel Enable Query command [108](#)
 - Analog Input Clock Source Configuration command [109](#)
 - Analog Input Clock Source Query command [110](#)
 - Analog Input Coupling Configuration command [111](#)
 - Analog Input Coupling Query command [111](#)
 - Analog Input Current Source Enable command [112](#)
 - Analog Input Current Source Enable Query command [113](#)
 - Analog Input Gain Configuration command [114](#)
 - Analog Input Gain Query command [114](#)
 - Analog Input Sampling Frequency Configuration command [115](#)
 - Analog Input Sampling Frequency Query command [116](#)
 - Analog Input Scan Status Query [117](#)
 - Analog Input Status Bits Query command [118](#)
 - Analog Input Trigger Source Configuration command [118](#)
 - Analog Input Trigger Source Query command [121](#)
 - analog output [58](#)
 - arming the operation [64](#)
 - configuring the buffer [58](#)
 - configuring the clock [59](#)
 - configuring the trigger [61](#)
 - enable the channel [58](#)
 - readback [44](#)
 - starting the operation [64](#)
 - stopping the operation [65](#)
 - Analog Output Buffer Mode Configuration command [149](#)
 - Analog Output Buffer Mode Query command [150](#)
 - Analog Output Buffer Read All Values command [150](#)
 - Analog Output Buffer Set Values command [151](#)
 - Analog Output Buffer Size Configuration command [153](#)
 - Analog Output Buffer Size Query command [155](#)
 - Analog Output Channel Enable command [155](#)
 - Analog Output Channel Enable Query command [156](#)
 - Analog Output Clock Source Configuration command [157](#)
 - Analog Output Clock Source Query command [158](#)
 - Analog Output Frequency Configuration command [166](#)
 - Analog Output Frequency Query command [167](#)
 - Analog Output Readback Enable command [159](#)
 - Analog Output Readback Enable Query command [160](#)
 - Analog Output Status Bits Query command [160](#)
 - Analog Output Trigger Source Configuration command [161](#)
 - Analog Output Trigger Source Query command [164](#)
 - ARM Analog Input Operation command [103](#)
 - ARM Analog Output Operation command [147](#)
 - arming operations
 - input [53](#)
 - output [64](#)
 - Auto-IP [37](#)
-
- ## B
- Block data types [27](#)
 - Boolean data types [27](#)
 - braces [22](#)
 - brackets [22](#)
-
- ## C
- channel lists [30](#)
 - character data types [24](#)
 - Clear Status command [70](#)
 - clients [54](#)
 - clock signal [45](#), [59](#)
 - clocks
 - analog input [45](#)
 - analog output [59](#)
 - command header error [204](#)
 - command hierarchy [80](#)

common commands

[*CLS 70](#)
[*ESE 70](#)
[*ESE? 71](#)
[*ESR? 71](#)
[*IDN? 73](#)
[*OPC 74](#)
[*OPC? 74](#)
[*RST 74](#)
[*SRE 75](#)
[*SRE? 75](#)
[*STB? 76](#)
[*TST? 75](#)
[*WAI 77](#)

configuring LAN settings of an instrument [188](#)

configuring LAN settings of an instrument module
[37](#)

configuring the counter/timers [42](#)

continuous wrap mode [44](#), [59](#)

COUNter subsystem commands

COUNter[<n>]:EDGE:{START|STOP}
 [:CONFigure] [139](#)

COUNter[<n>]:EDGE:{START|STOP}
 [:CONFigure]? [141](#)

COUNter[<n>]:ENABLE [138](#)

COUNter[<n>]:ENABLE? [138](#)

COUNter[<n>]:SCLR[:CONFigure] [142](#)

COUNter[<n>]:SCLR[:CONFigure]? [143](#)

COUNter[<n>]:EDGE:{START|STOP}
 [:CONFigure] [43](#), [139](#), [190](#)

COUNter[<n>]:EDGE:{START|STOP}
 [:CONFigure]? [43](#)

COUNter[<n>]:EDGE:{START|STOP}
 [:CONFigure]? [43](#), [141](#)

COUNter[<n>]:ENABLE [43](#), [138](#), [190](#)

COUNter[<n>]:ENABLE? [43](#), [138](#)

COUNter[<n>]:SCLR[:CONFigure] [43](#), [142](#), [190](#)

COUNter[<n>]:SCLR[:CONFigure]? [43](#), [143](#)

Counter/Timer Channel Enable command [138](#)

Counter/Timer Channel Enable Query command
[138](#)

Counter/Timer Edge Configuration command [139](#)

Counter/Timer Edge Query command [141](#)

Counter/Timer Self Clear Flag Configuration
 command [142](#)

Counter/Timer Self Clear Flag Query command
[143](#)

counter/timers [42](#)

coupling type [40](#)

current source [40](#)

D

DA subsystem commands

[DA:]CLOCK:SOURce [157](#)

[DA:]CLOCK:SOURce? [158](#)

DA:ABORt [146](#)

DA:ARM [147](#)

DA:BUFFer:MODE [149](#)

DA:BUFFer:MODE? [150](#)

DA:BUFFer:SCAns? [155](#)

DA:BUFFer:SIze[:SCAns] [153](#)

DA:BUFFer[:DATA] [151](#)

DA:BUFFer[:DATA]? [150](#)

DA:CLOCK:FREQuency[:CONFigure] [166](#)

DA:CLOCK:FREQuency[:CONFigure]? [167](#)

DA:ENABle [155](#)

DA:ENABle? [156](#)

DA:INITiate [168](#)

DA:READ:ENABle [159](#)

DA:READ:ENABle? [160](#)

DA:STATus? [160](#)

DA:TRIGger[:SOURce] [161](#)

DA:TRIGger[:SOURce]? [164](#)

DA:ABORt [146](#), [192](#)

DA:ARM [147](#), [197](#)

DA:BUFFer:MODE [59](#), [149](#), [192](#)

DA:BUFFer:MODE? [59](#), [150](#)

DA:BUFFer:SCAns? [155](#)

DA:BUFFer:SIze [59](#), [66](#)

DA:BUFFer:SIze[:SCAns] [58](#), [153](#), [192](#)

DA:BUFFer[:DATA] [58](#), [151](#), [192](#)

DA:BUFFer[:DATA]? [59](#), [150](#)

DA:CLOCK:FREQuency[:CONFigure] [60](#), [166](#), [195](#)

DA:CLOCK:FREQuency[:CONFigure]? [61](#), [167](#)

DA:ENABle [58](#), [155](#), [192](#)

DA:ENABle? [58](#), [156](#)

DA:INITiate [168](#), [192](#)

DA:READ:ENABle [44](#), [159](#), [190](#)

DA:READ:ENABle? [44](#), [160](#)

DA:STATus? [64](#), [160](#)

DA:TRIGger[:SOURce] [161](#), [197](#)

DA:TRIGger[:SOURce]? [164](#)

DAC readback [44](#)

Data Interchange Format (DIF) expressions [32](#)

data types [24](#)

Block [27](#)

Boolean [27](#)

character [24](#)

NR1 [25](#)

NR2 [25](#)

NRf [26](#)

- NRr 25
- string 24
- DATE Query command 90, 91
- description of the device 38
- device description 38
- DHCP 37
- digital output 67
- Digital Output AND Output Values command 182
- Digital Output OR Output Values command 183
- Digital OUTput Query State command 184
- Digital Output Set State command 184
- digital trigger 49, 61
- documentation 36
- domain 50
- DOUTput 67, 184, 193
- DOUTput subsystem commands
 - DOUTput 184
 - DOUTput:AND 182
 - DOUTput:OR 183
 - DOUTput? 184
- DOUTput:AND 67, 182
- DOUTput:OR 67, 183
- DOUTput? 67, 184
- DT8837 SCPI example 216

E

- edge for counter/timer measurement
 - starting 43
 - stopping 43
- enabling channels
 - analog input 39
 - analog output 58
- Error Query All Codes command 92
- Error Query All command 91
- ERRor Query Count command 93
- Error Query Next Code command 95
- Error Query Next command 94
- errors 201, 202, 207, 208, 210, 211, 212, 215, 219
 - 110 204
 - 410 205
 - error codes 202
 - troubleshooting 204
- event ID 50
- event name 50
- Event Status (ESR) register 211
- Event Status Enable (ESE) register 209
- Event Status Register Query command 71
- EVENT subsystem commands
 - EVENT:DOMain 175
 - EVENT:DOMain? 175

- EVENT:LAN[<n>]:TRANsmit[:ENABle]? 179
- EVENT:LAN<n>:NAME 176
- EVENT:LAN<n>:NAME? 177
- EVENT:LAN<n>:TRANsmit[:ENABle] 178
- EVENT:DOMain 50, 52, 62, 63, 175, 196, 197
- EVENT:DOMain? 175
- EVENT:LAN[<n>]:TRANsmit[:ENABle]? 179
- EVENT:LAN<n>:NAME 50, 51, 52, 62, 63, 176, 196, 197
- EVENT:LAN<n>:NAME? 177
- EVENT:LAN<n>:TRANsmit[:ENABle] 50, 52, 62, 63, 178, 196, 197
- examples 36, 216
- excitation current source 40
- expression types 29
 - channel lists 30
 - Data Interchange Format (DIF) 32
 - instrument-specifier 33
 - numeric 29
 - numeric lists 31
- EXTernal digital trigger 49, 61

F

- FIFO
 - input 44, 54
 - output 58
- flowcharts
 - analog input 189
 - analog output 192
 - clock for analog input 194
 - clock for analog output 195
 - digital output 193
 - trigger for analog input 196
 - trigger for analog output 197
- frequency
 - analog input operations 46, 47
 - analog output operations 60

G

- gain 40
- gate input 42
- group delay 48

H

- help 10
- hierarchy of commands 80
- HyperTerminal 219, 220

I

Identification query 73
 IEPE features 40
 IMMEDIATE software trigger 48, 61
 INITiate Analog Input Operation command 128
 INITiate Analog Output Operation command 168
 initiating operations
 input 53
 output 64
 input buffer
 configuring 44
 retrieving data 54
 input ranges 40
 Input Trigger LED 48
 installation 36
 instrument-specifier expressions 33
 internal clock 45, 59
 internet protocols, UDP 49
 IP address 37, 38

L

LAN configuration 37
 flowchart 188
 returning information about the instrument 38
 LAN Event Domain Configuration command 175
 LAN Event Domain Query command 175
 LAN Event Name Configuration command 176
 LAN Event Name Query command 177
 LAN Event Transmission Enable command 178
 LAN Event Transmission Enable Query command 179
 LAN Static IP Address Configuration command 96
 LAN Static IP Address Query command 97
 LAN Subnet Mask Configuration command 97
 LAN Subnet Mask Query command 98
 LAN trigger packet 49, 62
 LED
 Input Trigger 48
 Output Trigger 61
 LXI6 ADC Sync signal 45
 LXI7 clock signal 45, 59

M

mask, subnet 37, 38
 master clock 45, 59
 messages
 program 20
 response 23
 mnemonics 22

multicast 49

N

no wrap mode 44, 59
 NR1 data types 25
 NR2 data types 25
 NRf data types 26
 NRr data types 25
 numeric expression types 29
 numeric lists 31
 Nyquist Theorem 47

O

Operation Complete command 74
 Operation Complete Query command 74
 Operation Condition (OCR) register 212
 Operation Condition Register Query command 84
 Operation Event Register Enable command 85
 Operation Event Register Query command 86
 Operation Status Register Enable Query command 85
 output buffer, configuring 58
 output clock 59
 Output Trigger LED 61

P

period type 41
 port 5025 12
 Presetting registers command 86
 program messages 20
 protocols, UDP 49

Q

query interrupted error 205
 Questionable Condition Register Query command 86
 Questionable Enable Register command 87
 Questionable Event Register Query command 88
 Questionable Register Enable Query command 87

R

ranges, analog input 40
 Read Status Byte query 76
 readback of analog output 44
 reference clock 45, 59

- registers
 - Operation Condition (OCR) [212](#)
 - Standard Event Status (ESR) [211](#)
 - Standard Event Status Enable (ESE) [209](#)
 - Status Byte (STB) [208](#)
- related documents [10](#)
- Reset command [74](#)
- response messages [23](#)
- retrieving input data [54](#)

S

- sample clock [45](#)
- SCPI clients [54](#)
- SCPI data types [24](#)
- SCPI expression types [29](#)
 - channel lists [30](#)
 - Data Interchange Format (DIF) expressions [32](#)
 - instrument-specifier expressions [33](#)
 - numeric expressions [29](#)
 - numeric lists [31](#)
- SCPI support files [36](#)
- SCPI Version Query command [98](#)
- Self Clear flag
 - counter/timer [43](#)
 - tachometer [41](#)
- Self-Test Query command [75](#)
- service and support procedure [200](#)
- Service Request Enable command [75](#)
- Service Request Enable Query command [75](#)
- sockets [12](#)
- software trigger [48](#), [61](#)
- Stale Value flag [41](#)
- Standard Event Status (ESR) register [211](#)
- Standard Event Status Enable (ESE) register [209](#)
- Standard Event Status Enable Register command [70](#)
- Standard Event Status Enable Register Query command [71](#)
- starting operations
 - input [53](#)
 - output [64](#)
- starting signal edge [43](#)
- static IP [37](#)
- Status Byte (STB) register [208](#)
- STATus subsystem commands
 - STATus:OPERation:CONDition? [84](#)
 - STATus:OPERation:ENABle [85](#)
 - STATus:OPERation:ENABle? [85](#)
 - STATus:OPERation:EVENT? [86](#)
 - STATus:PRESet [86](#)
 - STATus:QUEStionable:CONDition? [86](#)
 - STATus:QUEStionable:ENABle [87](#)
 - STATus:QUEStionable:ENABle? [87](#)
 - STATus:QUEStionable:EVENT? [88](#)
 - SYSTem:VERSion? [98](#)
- STATus:OPERation:CONDition? [84](#)
- STATus:OPERation:ENABle [85](#)
- STATus:OPERation:ENABle? [85](#)
- STATus:OPERation:EVENT? [86](#)
- STATus:PRESet [86](#)
- STATus:QUEStionable:CONDition? [86](#)
- STATus:QUEStionable:ENABle [87](#)
- STATus:QUEStionable:ENABle? [87](#)
- STATus:QUEStionable:EVENT? [88](#)
- stopping operations
 - input [57](#)
 - output [65](#)
- stopping signal edge [43](#)
- string data types [24](#)
- subnet mask [37](#), [38](#)
- synchronizing tachometer, counter/timer, and analog output data with analog input data [48](#)
- syntax conventions [11](#)
 - braces [22](#)
 - brackets [22](#)
 - common SCPI commands [21](#)
 - program messages [20](#)
 - response messages [23](#)
 - SCPI data types [24](#)
 - SCPI expression types [29](#)
 - SCPI subsystem commands [21](#)
 - short- and long-form mnemonics [22](#)
 - vertical bars [22](#)
- SYSTem subsystem commands
 - SYSTem:COMMunicate:NETwork:IPAddr [96](#)
 - SYSTem:COMMunicate:NETwork:IPAddr? [97](#)
 - SYSTem:COMMunicate:NETwork:MASk [97](#)
 - SYSTem:COMMunicate:NETwork:MASk? [98](#)
 - SYSTem:DATE? [90](#), [91](#)
 - SYSTem:ERRor:ALL? [91](#)
 - SYSTem:ERRor:CODE:ALL? [92](#)
 - SYSTem:ERRor:CODE[:NEXT]? [95](#)
 - SYSTem:ERRor:COUNt? [93](#)
 - SYSTem:ERRor[:NEXT]? [94](#)
 - SYSTem:TIME? [99](#)
 - SYSTem:TZONE? [100](#)
- SYSTem:COMMunicate:NETwork:IPAddr [37](#), [96](#), [188](#)
- SYSTem:COMMunicate:NETwork:IPAddr? [38](#), [97](#)
- SYSTem:COMMunicate:NETwork:MASk [37](#), [97](#), [188](#)

SYSTem:COMMunicate:NETwork:MASk? 38, 98
 SYSTem:DATE? 90, 91
 SYSTem:DESCRiption 37
 SYSTem:DESCRiption? 38
 SYSTem:ERRor:ALL? 91
 SYSTem:ERRor:CODE:ALL? 92
 SYSTem:ERRor:CODE[:NEXT]? 95
 SYSTem:ERRor:COUNt? 93
 SYSTem:ERRor[:NEXT]? 94
 SYSTem:TIME? 99
 SYSTem:VERSion? 98

T

Tachometer Channel Enable command 131
 Tachometer Channel Enable Query command 131
 tachometer input 40
 Tachometer Period Type Configuration command 132
 Tachometer Period Type Query command 133
 Tachometer Self Clear Flag Configuration command 133
 Tachometer Self Clear Flag Query command 134
 Tachometer Stale Value Flag Configuration command 135
 Tachometer Stale Value Query command 135
 TACHometer subsystem commands
 TACHometer:ENABle 131
 TACHometer:ENABle? 131
 TACHometer:PERiod[:CONFigure] 132
 TACHometer:PERiod[:CONFigure]? 133
 TACHometer:SCLR[:CONFigure] 133
 TACHometer:SCLR[:CONFigure]? 134
 TACHometer:SFLAG[:CONFigure] 135
 TACHometer:SFLAG[:CONFigure]? 135
 TACHometer:ENABle 40, 131, 189
 TACHometer:ENABle? 40, 131
 TACHometer:PERiod[:CONFigure] 41, 132, 189
 TACHometer:PERiod[:CONFigure]? 41, 133
 TACHometer:SCLR[:CONFigure] 41, 133, 189
 TACHometer:SCLR[:CONFigure]? 41, 134
 TACHometer:SFLAG[:CONFigure] 41, 135
 TACHometer:SFLAG[:CONFigure]? 41, 135
 TCP port 5025 12
 technical support 10, 200
 threshold trigger 49, 62
 TIME Query command 99
 time stamp 50
 Time Zone Query 100

Trigger Bus
 clock signal 45, 59
 trigger signal 51, 62
 triggers
 AIN1 analog threshold 49, 62
 analog input 48
 analog output 61
 EXternal digital 49, 61
 IMMediate software 48, 61
 LAN0 to LAN7 trigger packet 49, 62
 Trigger Bus 51, 62
 troubleshooting checklist 200
 troubleshooting errors 204
 troubleshooting procedure 200
 TTL trigger 49, 61

U

UDP 49
 using HyperTerminal 219

V

vertical bars 22
 VISA 12
 voltage ranges 40
 VXI-11 clients 54

W

Wait-to-Continue command 77
 wrap mode 44, 59
 WTB subsystem commands
 WTB:LXI<n>[:OUTput]:ENABle 171
 WTB:LXI<n>[:OUTput]:ENABle? 172
 WTB:LXI<n>[:OUTput]:ENABle 46, 52, 53, 60, 63,
 171, 194, 195, 196, 197
 WTB:LXI<n>[:OUTput]:ENABle? 172

