EX1048

48-CHANNEL, PRECISION THERMOCOUPLE INSTRUMENT

USER'S MANUAL

P/N: 82-0105-000
Released February 23, 2007

VXI Technology, Inc.

2031 Main Street
Irvine, CA 92614-6509
(949) 955-1894

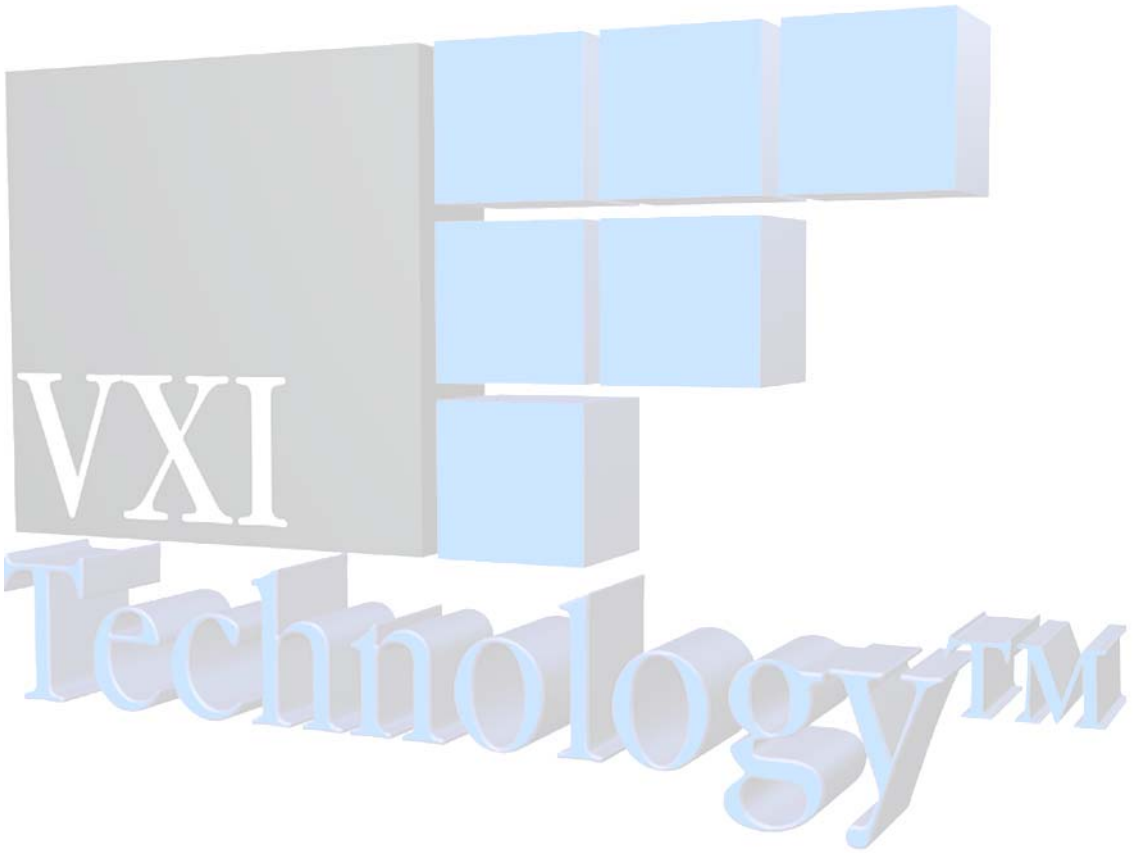
VXI
Technology™

TABLE OF CONTENTS

CERTIFICATION

VXI Technology, Inc. (VTI) certifies that this product met its published specifications at the time of shipment from the factory. VTI further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology (formerly National Bureau of Standards), to the extent allowed by that organization's calibration facility, and to the calibration facilities of other International Standards Organization members.

WARRANTY

The product referred to herein is warranted against defects in material and workmanship for a period of one year from the receipt date of the product at customer's facility. The sole and exclusive remedy for breach of any warranty concerning these goods shall be repair or replacement of defective parts, or a refund of the purchase price, to be determined at the option of VTI.

For warranty service or repair, this product must be returned to a VXI Technology authorized service center. The product shall be shipped prepaid to VTI and VTI shall prepay all returns of the product to the buyer. However, the buyer shall pay all shipping charges, duties, and taxes for products returned to VTI from another country.

VTI warrants that its software and firmware designated by VTI for use with a product will execute its programming when properly installed on that product. VTI does not however warrant that the operation of the product, or software, or firmware will be uninterrupted or error free.

LIMITATION OF WARRANTY

The warranty shall not apply to defects resulting from improper or inadequate maintenance by the buyer, buyer-supplied products or interfacing, unauthorized modification or misuse, operation outside the environmental specifications for the product, or improper site preparation or maintenance.

VXI Technology, Inc. shall not be liable for injury to property other than the goods themselves. Other than the limited warranty stated above, VXI Technology, Inc. makes no other warranties, express or implied, with respect to the quality of product beyond the description of the goods on the face of the contract. VTI specifically disclaims the implied warranties of merchantability and fitness for a particular purpose.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subdivision (b)(3)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

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

GENERAL SAFETY INSTRUCTIONS

Review the following safety precautions to avoid bodily injury and/or damage to the product. These precautions must be observed during all phases of operation or service of this product. Failure to comply with these precautions, or with specific warnings elsewhere in this manual, violates safety standards of design, manufacture, and intended use of the product.

Service should only be performed by qualified personnel.

TERMS AND SYMBOLS

These terms may appear in this manual:

WARNING Indicates that a procedure or condition may cause bodily injury or death.

CAUTION Indicates that a procedure or condition could possibly cause damage to equipment or loss of data.

These symbols may appear on the product:

ATTENTION - Important safety instructions

Frame or chassis ground

Indicates that the product was manufactured after August 13, 2005. This mark is placed in accordance with *EN 50419, Marking of electrical and electronic equipment in accordance with Article 11(2) of Directive 2002/96/EC (WEEE).* End-of-life product can be returned to VTI by obtaining an RMA number. Fees for take-back and recycling will apply if not prohibited by national law.

WARNINGS

Follow these precautions to avoid injury or damage to the product:

Use Proper Power Cord To avoid hazard, only use the power cord specified for this product.

Use Proper Power Source To avoid electrical overload, electric shock, or fire hazard, do not use a power source that applies other than the specified voltage.

Use Proper Fuse To avoid fire hazard, only use the type and rating fuse specified for this product.

WARNINGS (CONT.)

Avoid Electric Shock

To avoid electric shock or fire hazard, do not operate this product with the covers removed. Do not connect or disconnect any cable, probes, test leads, etc. while they are connected to a voltage source. Remove all power and unplug unit before performing any service. ***Service should only be performed by qualified personnel.***

Ground the Product

This product is grounded through the grounding conductor of the power cord. To avoid electric shock, the grounding conductor must be connected to earth ground.

Operating Conditions

To avoid injury, electric shock or fire hazard:

- Do not operate in wet or damp conditions.
- Do not operate in an explosive atmosphere.
- Operate or store only in specified temperature range.
- Provide proper clearance for product ventilation to prevent overheating.
- DO NOT operate if any damage to this product is suspected. ***Product should be inspected or serviced only by qualified personnel.***

Improper Use

The operator of this instrument is advised that if the equipment is used in a manner not specified in this manual, the protection provided by the equipment may be impaired.
Conformity is checked by inspection.

SUPPORT RESOURCES

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

VXI Technology
World Headquarters

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

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

VXI Technology
Cleveland Instrument Division

5425 Warner Road
Suite 13
Valley View, OH 44125

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

VXI Technology
Lake Stevens Instrument Division

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

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

Technical Support

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

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

SECTION 1

INTRODUCTION

OVERVIEW

The EX1048 is a 48-channel high-performance thermocouple measurement instrument. Its combination of measurement performance and integrity, configuration flexibility, package density, and network connectivity make it the most powerful, yet easy-to-use, instrument of its kind. The EX1048 is a complete, self-contained temperature measurement system that communicates over Ethernet. Unlike other data acquisition offerings in its class, the EX1048 offers a tightly integrated solution that frees the user from the complexity of marrying terminal blocks, signal conditioning cards, digitizer, power supply, and chassis together.

The EX1048 also provides a level of measurement integrity and channel independence that far exceeds the typical data acquisition system. Absent are caveats about scanning speed, channel order, and overload effects. In the EX1048, channels have no influence on each other, regardless of scanning speed or channel state. Excellent common mode rejection performance provides immunity from not only power line interference, but also high frequency noise. Moreover, it aids in maintaining overall system integrity by offering a high-performance open thermocouple detection system.

The EX1048 provides a high level of configuration flexibility as well. Each channel can be configured independently with regards to measurement function, hardware filter setting, and limit values. In addition to measuring all standard thermocouples, the EX1048 can be programmed with user-defined thermocouple polynomial equations or be used as a low-noise millivoltmeter. Scanning speed is programmable up to a maximum of 1 kHz, independent of the number of channels being scanned. This combination of filtering and scanning speed provides the EX1048 with the low noise performance required for sensitive applications as well as the speed necessary to measure fast, fine-gauge thermocouples.

For highest accuracy and stability, the EX1048 provides an embedded isothermal input section that is monitored by twelve precision thermistors, one for every four thermocouple channels. Moreover, it features an internal calibration source that can be used to self-calibrate the unit upon command. This extends the unit's high accuracy over a wide ambient temperature range. For maximum utility, detailed temperature accuracy specifications are provided over an ambient operating range of 15 °C to 35 °C, indicating accuracy limits with and without the use of self-calibration.

The EX1048 is as easy-to-use as it is powerful. An integrated web page provides a convenient way to instantly verify communications and instrument functionality, while industry standard VXI*plug&play* drivers provide a familiar application programming interface to reduce integration and program development time.

FEATURES

Channel Independence

Each of the EX1048's 48 differential input channels is an independent signal conditioning path, complete with amplification, programmable hardware filtering, and continuous open thermocouple detection. This independence frees the user from the problem of channel-to-channel crosstalk that is pervasive in most multi-channel data acquisition systems. In the EX1048, channels have no influence on each other, regardless of scanning speed or channel state. Specifically, open or significantly overloaded channels do not affect the measurement results of any other channels.

Measurement Range

The measurement range of the EX1048 in terms of temperature is a function of its input voltage range and the capabilities of the thermocouple sensors themselves. Specifically, the measurement range of the EX1048 for the standard thermocouple types is the following:

	Min (°C)	Max (°C)	Min (°F)	Max (°F)
Type J	-200	1200	-328	2192
Type K	-200	1372	-328	2502
Type T	-200	400	-328	752
Type E	-200	900	-328	1652
Type S	-50	1768	-58	3214
Type R	-50	1768	-58	3214
Type B	250	1820	482	3308
Type N	-200	1300	-328	2372

TABLE 1-1: EX1048 MEASUREMENT RANGE

For maximum flexibility, each channel can be independently configured with regards to its thermocouple conversion. Moreover, non-standard thermocouples are accommodated through the input of user-defined thermocouple polynomial coefficients.

Hardware Filter

Each EX1048 input channel can be individually configured with a hardware filter of 4 Hz or 1 kHz cutoff frequency. This feature allows the EX1048 to cover a diverse range of applications, even within the same unit. Suitable for most applications, the 4 Hz setting provides the lowest noise floor and exceptional common mode rejection. For higher speed applications, the 1 kHz setting will pass the output from even the fastest fine-gauge thermocouples with little distortion.

Open Thermocouple (OTC) Detection

While the integration of the EX1048 removes many of the reliability and connectivity problems typically faced by system designers, they still must contend with the reliability of the sensor connections. Fortunately, the EX1048 aids a great deal in that regard as well. Each input channel is biased with a very small current source. In the event of a broken sensor connection, this current deterministically drives the input amplifier to a state that registers as an out-of-bounds condition. Implemented in this way, open thermocouple detection is continuous and requires no discrete command on the part of the user to be activated. This offers more protection than a system that checks for an open on command, as a broken sensor can occur at any time during testing, not just at installation. The EX1048 additionally offers a front panel OTC LED for each channel that illuminates upon the recognition of a fault condition. This provides for quick and easy problem channel identification. Moreover, to insure that even intermittent problems are identified, the fault recognition is a latching mechanism, retaining the information of the current acquisition sequence until a new acquisition is initiated.

Cold Junction Compensation (CJC)

For highest accuracy and stability, the EX1048 provides an embedded isothermal input section that is monitored by twelve precision thermistors, one for every four thermocouple channels. To insure that the CJC information is current and time correlated with the input channels, the CJC channels are measured with every scan, providing a maximum time separation of less than 4 ms between the measurement of an input channel and its associated CJC measurement. For those users that prefer to employ an external cold junction, the EX1048 also allows for the programming input of up to forty-eight unique external cold junction temperatures, one for every input channel. Moreover, the use of internal and external CJC inputs can be mixed throughout the unit on a per channel basis.

Input Connector

The EX1048 employs an uncompensated (Cu-Cu) mini-thermocouple female jack as its input connector. This connector provides a solid, reliable connection that is also easily changeable. Since it is not thermocouple-type specific, different thermocouple types can be mixed throughout the unit without hardware modification.

Self-calibration

In order to deliver high measurement accuracy over a wide ambient operating temperature range, the EX1048 provides the ability to perform an instrument self-calibration. During self-calibration, the input signal conditioning paths are disconnected from the input jacks and connected instead to a calibration bus that is driven by an internal calibration source. Through measurement of the conditioning paths at multiple calibration source points, software compensation for circuitry drift since the last full calibration is conducted. This provides a significant improvement in the accuracy of the EX1048 without the burden of connecting external equipment. Moreover, the self-calibration process completes quickly and does not require removal of the actual input connections, making it convenient to run often.

Sampling Rate

The EX1048 can be configured for a sampling rate up to a maximum of 1 kHz, regardless of the number of channels included in the scan list. When the requested sampling rate is significantly less than 1 kHz, however, the EX1048 automatically takes and averages multiple samples. This offers improved noise performance, while maintaining the requested data output rate.

Digital I/O and Limits

The EX1048 provides two unique sets of programmable limits that are used for open thermocouple detection as well as general purpose input channel monitoring. These limits are programmable on a per channel basis and are evaluated with each completed scan. The output of limit evaluations is presented in three forms. The operation of the front panel LEDs is tied to the upper and lower limit values of one limit set. The operation of the digital I/O port can be optionally linked to any combination of the upper and lower limit values of either or both limit sets. Finally, the limit condition information is accessible through the instrument driver.

The EX1048 features an 8-channel digital I/O port on the rear panel of the instrument. This port can be used as an arm/trigger source, for presentation of limit evaluation information, and as a general purpose output device. As a general purpose output device, each DIO channel can be independently programmed with regards to its output functionality and its static level to assume when enabled as an output. For expanded and more automated operation, each DIO channel can be independently linked to one or multiple limit conditions on one or more input channels.

LXI Trigger Bus

The EX1048 features an 8-channel LXI (*LAN eXtensions for Instrumentation*) trigger bus on the rear panel of the instrument. This differential-pair LVDS bus consists of two identical ports connected in parallel. The primary use of the trigger bus is the transmission of high-speed signals for multiple-unit triggering and synchronization.

Triggering

The EX1048 supports a full function trigger model with a separate arm source and trigger source event structure. Trigger and arm source events can be independently programmed from a variety of sources including Immediate, Timer, Digital I/O, and the Trigger Bus.

Embedded Web Page

The EX1048 offers an embedded web page for easy remote operation without any programming. Virtually all of the EX1048's configuration controls and data collection options are available through the web page, and it also serves as the mechanism to read and set network parameters.

EX1048 SPECIFICATIONS

GENERAL SPECIFICATIONS	
NUMBER OF CHANNELS	
	48 differential inputs
FUNCTIONS	
	J, K, T, E, S, R, B, N, mV
SAMPLING RATE	
	1000 Sa/s per channel maximum
TEMPERATURE RESOLUTION	
	0.01 °C
TEMPERATURE ACCURACY	
	See Table 1-2 and Table 1-3 below
TEMPERATURE NOISE	
Peak-to-peak	0.08 °C typical (J, K, T, E)
VOLTAGE INPUT RANGE	
	±66 mV
VOLTAGE RESOLUTION	
	1 µV
VOLTAGE ACCURACY[1]	
With Self-Calibration	±(0.05% + 10 µV)
Without Self-Calibration	±(0.1% + 30 µV)
VOLTAGE OFFSET STABILITY	
	1 µV/°C typical
VOLTAGE GAIN STABILITY	
	25 ppm/°C typical
INPUT IMPEDANCE	
	40 MΩ differential
INPUT BIAS CURRENT	
	7.5 nA typical
COMMON MODE INPUT RANGE	
	±10 V
COMMON MODE REJECTION RATIO (CMRR)	
4 Hz Filter	
dc	100 dB minimum
(50/60) Hz	140 dB typical, 120 dB minimum
1 kHz Filter	
dc	100 dB minimum
(50/60) Hz	100 dB typical, 80 dB minimum
FILTER	
	4 Hz or 1 kHz (selectable per channel)
INPUT PROTECTION	
	±35 V
NETWORK CONNECTION	
	10/100 Base-T
INPUT CONNECTOR	
	Cu-Cu mini-TC jack
OPERATING TEMPERATURE	
	0 °C to 50 °C
POWER INPUT	
	(90 – 264) V ac, (50/60) Hz, 25 VA maximum
DIMENSIONS	
	1.75” H x 17.5” W x 14.4” D

TABLE 1-2: ENHANCED THERMOCOUPLE ACCURACY

	-100 °C	0 °C	100 °C	300 °C	500 °C	700 °C	900 °C	1100 °C	1400 °C
Type J	±0.60 °C	±0.45 °C	±0.45 °C	±0.50 °C	±0.60 °C	±0.65 °C	±0.75 °C	±0.90 °C	----------
Type K	±0.70 °C	±0.50 °C	±0.50 °C	±0.60 °C	±0.65 °C	±0.75 °C	±0.90 °C	±1.10 °C	----------
Type T	±0.75 °C	±0.50 °C	±0.45°C	±0.45 °C	----------	----------	----------	----------	----------
Type E	±0.60 °C	±0.45 °C	±0.40 °C	±0.45 °C	±0.50 °C	±0.60 °C	±0.75 °C	----------	----------
Type S	----------	±2.00 °C	±1.50 °C	±1.30 °C	±1.30 °C	±1.30 °C	±1.40 °C	±1.40 °C	±1.50 °C
Type R	----------	±2.00 °C	±1.50 °C	±1.20 °C	±1.20 °C	±1.20 °C	±1.20 °C	±1.30 °C	±1.40 °C
Type B	----------	----------	----------	±3.30 °C	±2.10 °C	±1.60 °C	±1.40 °C	±1.30 °C	±1.30 °C
Type N	±0.80 °C	±0.60 °C	±0.55 °C	±0.55 °C	±0.60 °C	±0.70 °C	±0.80 °C	±0.95 °C	----------

Conditions[1]
Guaranteed maximum limits. Typical errors are approximately ½ of maximum.
<30 days, ±2 °C from last self-calibration
15 °C to 35 °C, 1 year from full calibration
60 minute warm-up
Exclusive of thermocouple errors
Exclusive of noise
Vcm = 0

TABLE 1-3: BASIC THERMOCOUPLE ACCURACY

	-100 °C	0 °C	100 °C	300 °C	500 °C	700 °C	900 °C	1100 °C	1400 °C
Type J	±1.00 °C	±0.75 °C	±0.75 °C	±0.90 °C	±1.10 °C	±1.20 °C	±1.40 °C	±1.70 °C	----------
Type K	±1.30 °C	±0.90 °C	±0.90 °C	±1.10 °C	±1.30 °C	±1.50 °C	±1.80 °C	±2.10 °C	----------
Type T	±1.30 °C	±0.90 °C	±0.80 °C	±0.80 °C	----------	----------	----------	----------	----------
Type E	±1.00 °C	±0.70 °C	±0.65 °C	±0.75 °C	±0.90 °C	±1.10 °C	±1.40 °C	----------	----------
Type S	----------	±4.90 °C	±3.70 °C	±3.10 °C	±3.10 °C	±3.10 °C	±3.10 °C	±3.20 °C	±3.40 °C
Type R	----------	±5.00 °C	±3.60 °C	±2.90 °C	±2.80 °C	±2.80 °C	±2.80 °C	±2.80 °C	±3.00 °C
Type B	----------	----------	----------	±8.50 °C	±5.30 °C	±4.10 °C	±3.50 °C	±3.20 °C	±3.10 °C
Type N	±1.60 °C	±1.20 °C	±1.10 °C	±1.10 °C	±1.20 °C	±1.40 °C	±1.60 °C	±1.90 °C	----------

Conditions[1]
Guaranteed maximum limits. Typical errors are approximately ½ of maximum.
15 °C to 35 °C, 1 year from full calibration
60 minute warm-up
Exclusive of thermocouple errors
Exclusive of noise
Vcm = 0

Note
[1]*Applies to the 4 Hz filter setting. Add 10 µV for the 1 kHz filter setting.*

EXPLANATION OF SPECIFICATIONS

The base accuracy of the EX1048 is specified over an ambient operating temperature range of 15 ºC to 35 ºC and within one year of full calibration. This accuracy is shown in *Table 1-3: Basic Thermocouple Accuracy*. Significant performance improvement over the base accuracy can be realized, however, using periodic self-calibration. If self-calibration is employed, the *Enhanced Thermocouple Accuracy* in Table 1-2 is valid for 30 days and over a ±2 °C ambient temperature change. Note, however, that the base ambient operating temperature range and full calibration time interval restrictions still apply.

The EX1048's thermocouple accuracy tables provide the user with an easy-to-use description of the unit's capabilities at several points over the valid input dynamic range of the instrument. Within these tables, each specific thermocouple type is represented by a different row. Each column refers to an input temperature of that value. For example, in the *Enhanced Thermocouple Accuracy* table, an input temperature of 100 ºC on a type K thermocouple has a maximum instrument uncertainty of ±0.50 ºC, exclusive of thermocouple errors.

Exclusive of thermocouple errors

The "exclusive of thermocouple errors" qualification refers to two inherent sources of error present in all thermocouples to some extent. The first, and most obvious, error source is the accuracy of the thermocouple itself. This refers to the extent to which the specific thermocouple potentially deviates from the standardized International Temperature Scale of 1990 (ITS-90) characteristic for its thermocouple type. The second, less obvious error source relates to the resistance of the thermocouple wire, which creates a voltage drop against the input bias current of the EX1048. In most applications, this error effect is negligible. However, if the length and wire gauge of the employed thermocouple represents a resistance over 250 Ω, its effect should be analyzed for significance.

As an example, 650 ft of 24 gauge type T wire has a resistance of about 500 Ω. Against the EX1048's typical input bias current of 7.5 nA, this creates a voltage error of:

$$500\,\Omega \times 7.5\,\text{nA} = 3.75\,\mu\text{V}$$

To convert this error to its representative temperature error, it is then divided by the slope of the thermocouple characteristic at the temperature of interest. For example, the slope of the type T characteristic at 0 ºC is 39 µV/ºC. The error at this point is then:

$$3.75\,\mu\text{V} \div 39\,\mu\text{V}/^{\circ}\text{C} = 0.1\,^{\circ}\text{C}$$

Common mode rejection

The common mode rejection characteristics of the EX1048 are specified in terms of dB for both dc and (50/60) Hz interference. It should be noted that these specifications refer to the distortion of the measurement in terms of voltage, not temperature. The following example illustrates how to determine the possible temperature error for a T type thermocouple measuring 0 ºC due to a 1 V dc common mode signal.

The EX1048 has a minimum CMRR at dc of 100 dB. A 1 V common mode voltage creates a maximum differential voltage error of:

$$\frac{1\,\text{V}}{10^{\left(\frac{100}{20}\right)}} = 10\,\mu\text{V}$$

As before, this voltage error converts to a temperature error by dividing by the appropriate thermocouple slope to yield a maximum error of:

$$10\ \mu V \div 39\ \mu V/°C = 0.26\ °C$$

1 kHz filter setting

Finally, this same conversion is done to determine the additional uncertainty involved with using the 1 kHz filter setting (see *Note*). At this same measurement point, the 10 µV additional uncertainty translates into a temperature uncertainty of:

$$10\ \mu V \div 39\ \mu V/°C = 0.26\ °C$$

MAXIMIZING MEASUREMENT PERFORMANCE

This section discusses tips and procedures that can help maximize the actual performance realized with the EX1048 and aid the user in avoiding some common pitfalls associated with thermocouple measurement.

Utilize self-calibration

Self-calibration should be conducted as often as practical, especially if the ambient environment has changed significantly since the previous calibration. However, fast ambient environmental changes should ideally be followed by a period of thermal stabilization before conducting self-calibration. The self-calibration process completes quickly and does not require removal of the actual input connections, making it convenient to run often.

Allow for cold junction thermal stabilization

The performance of any thermocouple measurement instrument is largely determined by the stability of the cold junction sensing mechanism. For maximum accuracy and stability, the EX1048 is designed with a significant thermal mass that connects the input connectors to the cold junction sensing element. However, while this lowers the sensitivity to thermal disturbances, transient measurement errors are still possible under certain operating conditions. Awareness of these conditions will help achieve the maximum performance from the EX1048.

Step change in ambient temperature

If the EX1048 is subjected to a significant ambient temperature change over a short period of time, a transient measurement error will occur due to the inherent thermal time constant differences between the thermal mass and the actual cold junction made between the EX1048 input connector and the type-specific thermocouple connector. The magnitude of the error will be directly proportional to the rate of change of the ambient temperature. Because the actual cold junction is more closely tied to the external ambient conditions than the internal thermal mass, it will follow the ambient temperature change to a slightly better extent than the thermal mass. Consequently, when subjected to a drastic rise in ambient temperature, the actual cold junction will rise in temperature at a slightly faster rate than the thermal mass (and thus cold junction measurement). In this case, the channel measurements will show a transient error of negative polarity. Conversely, if the ambient temperature drastically falls, the channel measurements will show a transient error of positive polarity. In either case, as the rate of ambient temperature change decreases, thermal equilibrium is regained, and the error is eliminated. This error mechanism is not affected by calibration and is avoided by allowing for an appropriate thermal equilibrium delay.

NOTE	Significant ambient temperature changes should be followed by a thermal equilibrium delay before the initiation of measurements.

Initial insertion of a jack

A similar error mechanism is introduced when a male thermocouple jack is first inserted into an EX1048 input channel. A transient error is generated because the male thermocouple jack and mating female jack on the EX1048 are at different temperatures when they make first contact. If both the unit and male jack are at the same ambient temperature, this error typically dissipates to less than 0.1 ºC within 3 minutes. However, a jack at room temperature plugged into a unit that is at a substantially different ambient temperature will generate a significantly larger initial error that takes longer to decay to a level of insignificance. This error mechanism is not affected by calibration and is avoided by allowing for a thermal equilibrium delay after initial jack insertion.

NOTE	Initial jack insertion should be followed by a thermal equilibrium delay before the initiation of measurements.

Insertion of jacks into neighboring channels

A less obvious source of thermal destabilization is the insertion of jacks into neighboring channels. Because each CJC sensor is shared among four input channels, there is a thermal disturbance seen by it when jacks are inserted into any of the four channels that it is monitoring. The disturbance occurs because the newly inserted jack represents a thermal mass that is at a different temperature than the internal thermal mass. This instantly lowers the measured CJC temperature to a small extent. Since the same CJC temperature is used for three other channels, they exhibit a small transient error of negative polarity as a result. The magnitude of the error is proportional to the quantity of jacks that are inserted at a time, with three additional jacks being the worst-case scenario. The error decreases with time, as thermal equilibrium is gradually restored. Empirical testing has shown that the worst-case error typically dissipates to less than 0.1 ºC within 3 minutes. This error mechanism is only present on channels that share a CJC sensor. Moreover, it is essentially not present when jacks are removed, as no thermal disturbance is created. This error mechanism is not affected by calibration and is avoided by allowing for a thermal equilibrium delay after jack population changes. For reference, the association between CJC channels and input channels is shown in Table 1-4.

NOTE	Jack population changes should be followed by a thermal equilibrium delay before the initiation of measurements.

CJC #	Channel	Input Channel
CJC0	48	0-3
CJC1	49	4-7
CJC2	50	8-11
CJC3	51	12-15
CJC4	52	16-19
CJC5	53	20-23
CJC6	54	24-27
CJC7	55	28-31
CJC8	56	32-35
CJC9	57	36-39
CJC10	58	40-43
CJC11	59	44-47

TABLE 1-4: CJC CHANNEL / INPUT CHANNEL RELATIONSHIP

Select the proper hardware filter

Unless the bandwidth of the sensor requires a higher instrument bandwidth, the 4 Hz setting of the EX1048 hardware filter should be used, as it provides the greatest immunity to external electrical and magnetic interference.

Choose an appropriate sampling rate

For best instrument noise performance, the sampling rate should be set as low as the data collection requirements allow. For more details, see *Sampling Rate / Noise Performance* in Section 3.

Select the proper location

The EX1048 unit should be located away from sources of high or low temperature, strong air currents, and high magnetic fields. For more details, see *Installation Location* in Section 2.

Use the correct wiring

Best results will be achieved with the shortest and largest thermocouple wire that the physical requirements of the application can support. In addition, shielded thermocouples can be employed to raise the system's rejection of electrical interference. For more details, see *Input Connections / Wiring* in Section 2.

SECTION 2

PREPARATION FOR USE

UNPACKING

When the EX1048 is unpacked from its shipping carton, the contents should include the following items:

- EX1048 Precision Thermocouple Instrument
- Power line cord
- *EX1048 User's Manual* (this manual)
- *VXI Technology LXI User's Manuals and Drivers* CD

All components should be immediately inspected for damage upon receipt of the unit.

INSTALLATION LOCATION

The EX1048 is designed to be largely insensitive to external electrical, magnetic, and thermal disturbances. However, as with all precision instrumentation, certain precautions, if taken into consideration, can help achieve maximum performance.

1) The unit, particularly its front panel, should be located away from sources of high or low temperatures. When used in a rack-mount application with other heat-generating instruments, the EX1048 should be located as far away from the other instruments as possible, 1U minimum. Multiple EX1048s, however, can be stacked directly on top of one another without any performance degradation.
2) The front panel of the EX1048 should not be exposed to strong air currents. Typical problematic sources include building ventilation and instrument or cabinet fans.
3) The unit should be located away from sources of high magnetic fields such as motors, generators, and power transformers.

WARM-UP TIME

The specified warm-up time of the EX1048 is 60 minutes. If, however, the unit is being subjected to an ambient temperature change greater than 5 ºC, extra stabilization time is recommended to achieve maximum performance.

NETWORK CONFIGURATION

By default, the EX1048 will attempt to locate a DHCP server. If one is found, the IP address assigned by the DHCP server will be assumed. Otherwise, after a timeout of 20 seconds, the unit will attempt to obtain an IP address by using AutoIP.

AutoIP is a mechanism for finding an unused IP address in the range 169.254.X.Y where X is in the range 1 - 254 and Y is in the range 0 - 255. The device will first attempt to obtain the specific address 169.254.X.Y, where X and Y are the second-to-last and last octets of the device's MAC address. However, X will be set to 1 if it is 0 in the MAC address, and to 254 if it is 255 in the MAC address. If this address is already in use, the unit will attempt to obtain other IP addresses in a pseudorandom fashion until it finds one that is available.

To illustrate the AutoIP mechanism, Table 2-1 lists the AutoIP default address for some example MAC addresses.

MAC Address	AutoIP Default Address
00:0D:3F:01:00:01	169.254.1.1
00:0D:3F:01:01:01	169.254.1.1
00:0D:3F:01:A3:28	169.254.163.40
00:0D:3F:01:FE:FE	169.254.254.254
00:0D:3F:01:FF:FE	169.254.254.254

TABLE 2-1: AUTOIP DEFAULT ADDRESS ASSIGNMENT

If a static IP address assignment is preferred, one can be optionally assigned via the embedded web page interface. This is done by clicking the **Network Configuration** link, disabling DHCP, and then assigning a static IP address. For more information, see *Network Configuration* in Section 5.

However, a much more convenient and recommended way to obtain the benefits of a static IP address is to employ DHCP, but assign the instrument a reserved IP address in your company's DHCP server configuration. This reserved address, linked to the EX1048's MAC address on the DHCP server, would be assigned to the EX1048 at power up initialization without having to manually set it on the EX1048. The DHCP server configuration provides a centralized, controlled database of assigned IP addresses, preventing accidental assignment of the same IP address to multiple instruments. Consult your company's Information Technology department for assistance.

VXI-11 Device Discovery is also supported by the EX1048. This allows all EX1048s on a local network to be found without knowledge of their MAC address or IP address with the use of a broadcast message.

Reset button

The reset button on the rear panel of the EX1048 can be used to restore default network settings. This is useful for recovery from an incorrect or unknown network configuration. To perform a network reset:

1) Power off the EX1048.
2) Press and hold the reset button.
3) Power on the EX1048.
4) Continue to hold the reset button for at least 30 seconds.
5) Release the reset button.

The EX1048 will power up as usual, but will use the default network configuration (DHCP) instead of its previous settings.

TIME CONFIGURATION

The EX1048 will initially be configured to receive its time through SNTP, the Simple Network Time Protocol. Optionally, the time can be set manually by the user. This will be necessary if the

network environment is such that the unit cannot reach the Internet. For more information, see *Time Configuration* in Section 5.

INPUT CONNECTIONS / WIRING

The EX1048 employs an uncompensated (Cu-Cu) mini-thermocouple female jack as its input connector. This connector provides a solid, reliable connection that is also easily changeable. Since it is not thermocouple-type specific, different thermocouple types can be mixed throughout the unit without hardware modification. The input jack is polarized and will only accept its mating connector in one orientation. The mating connector is a standard mini-thermocouple male jack. A popular source is the SMPW series from Omega Engineering. For specified accuracy performance, the input connector must be of the same thermocouple type as the wire being connected.

Thermocouple wire is polarized, and it is critical to consider this polarity when connecting the thermocouple wire to the thermocouple jack. For reference, the color designations and polarizations of the most popular thermocouple types are listed in Table 2-2 for both ANSI (American) and IEC (European) standards.

ANSI Thermocouple Standard				
Thermocouple	**+**		**-**	
Type J	White		Red	
Type K	Yellow		Red	
Type T	Blue		Red	
Type E	Violet		Red	
Type S	Black		Red	
Type R	Black		Red	
Type B	Gray		Red	
Type N	Orange		Red	

IEC Thermocouple Standard				
Thermocouple	**+**		**-**	
Type J	Black		White	
Type K	Green		White	
Type T	Brown		White	
Type E	Violet		White	
Type S	Orange		White	
Type R	Orange		White	
Type B	Gray		White	
Type N	Pink		White	

TABLE 2-2: STANDARD THERMOCOUPLE REFERENCE DESIGNATIONS

In most applications, the length and gauge of the thermocouple wire do not affect the accuracy of the measurement. Due to the high input impedance and lack of dynamic switching in the signal conditioning circuitry of the EX1048, the resistance and capacitance of the thermocouple wire are normally not important factors. If, however, maximum system accuracy is desired, the resistance of the thermocouple wire must be considered as a system error source. As an example, 650 ft of 24 gauge type T wire has a resistance of about 500 Ω. Against the EX1048's typical input bias current of 7.5 nA, this creates a voltage error of:

$$500\,\Omega \times 7.5\,\text{nA} = 3.75\,\mu\text{V}$$

To convert this error to its representative temperature error, it is then divided by the slope of the thermocouple characteristic at the temperature of interest. For example, the slope of the type T characteristic at 0 ºC is 39 µV/ºC. The error at this point is then:

$$3.75\,\mu\text{V} \div 39\,\mu\text{V}/^{\circ}\text{C} = 0.1\,^{\circ}\text{C}$$

This example demonstrates how to evaluate the potential error that a specific wire installation represents. The user is encouraged to evaluate each individual application to insure that the error is within acceptable bounds. In general, best results will be achieved with the shortest and largest wire that the physical requirements of the application can support.

The EX1048 offers excellent noise rejection through its high common mode rejection and selectable bandwidth limiting, which allows for high integrity, noise-free measurements with even less-than-ideal wiring setups. However, some common instrumentation wiring practices can be used to achieve or insure maximum performance.

Shielded thermocouple wire can be used to raise the system's rejection of electrical interference. Shielded wire encloses the two thermocouple wires with a low impedance conductor that should be terminated by the user to a convenient earth ground. The EX1048 provides an external ground stud that can be used for this purpose, but any earth ground point is acceptable.

Magnetic interference, which is present wherever high currents are flowing, is conversely decreased by minimizing the loop area represented by the two thermocouple wires. That is, the wires should be run closely together from the thermocouple junction to the connections in the thermocouple jack. Fortunately, most thermocouple wire comes with a sheath that covers the two thermocouple conductors, inherently creating a small loop area.

Many test applications involve the monitoring of a test article in a chamber, requiring the routing of numerous types of signals through the chamber's cable access ports. It is recommended that the thermocouple wires be run through a separate port and conduit from cables carrying power or high frequency signals.

SECTION 3

BASIC OPERATION

INTRODUCTION

This section expands on the description of the EX1048's features and explains how to best use them.

ENGINEERING UNIT (EU) CONVERSION

Each EX1048 input channel can be individually configured for one of eleven different EU conversions. The selections and their definitions are:

Conversion	Parameter
Voltage	mV
Type J	J
Type K	K
Type T	T
Type E	E
Type S	S
Type R	R
Type B	B
Type N	N
User-defined 0	User0
User-defined 1	User1

TABLE 3-1: ENGINEERING UNIT CONVERSION SETTINGS

Voltage The instrument will return the raw voltage measured at its input with units of volts (V). It is unaffected by the measured or input CJC temperature for that channel.

Type J, K, T, E, S, R, B, N The instrument will return the compensated thermocouple temperature measured at its input with units of temperature (ºC or ºF). The thermocouple calculations are performed using the full-order polynomial equations and coefficients from the NIST ITS-90 Thermocouple Database.

User-defined 0, User-defined 1 The instrument will return the compensated thermocouple temperature measured at its input with units of temperature (ºC or ºF). The thermocouple calculations are performed using user-defined coefficients for the polynomial equations. More information on this is given under *User-defined Conversions*.

The default selection is voltage.

NOTE	In a mixed thermocouple system, it is very easy to accidentally mismatch the hardware setup and the software configuration setup. Care is especially warranted, as the resultant errors may not be large enough to obviously indicate a problem, but be significantly larger than the accuracy specification of the instrument.

HARDWARE FILTER

Each EX1048 input channel can be individually configured with a hardware filter of 4 Hz or 1 kHz cutoff frequency. Of the two settings, the 4 Hz setting is suitable for the majority of temperature measurement applications. It offers sufficient response to pass medium gauge thermocouple signals without distortion, while providing the highest degree of (50/60) Hz common mode rejection This setting also provides the lowest noise floor and is recommended for all applications that do not require a higher bandwidth. For those that do, however, the 1 kHz setting provides that capability. Its response will pass the output from even the fastest fine-gauge thermocouples with little distortion.

The default selection is 4 Hz.

MEASUREMENT RANGE / INPUT PROTECTION

The specified input voltage range of the EX1048 is ±66 mV. This level refers to the maximum differential voltage, or voltage that is applied between the + and – input terminals, that can be measured without distortion. The maximum common mode voltage, or voltage that is applied to the + and – inputs together, that can be applied without causing out-of-specification distortion of the differential measurement is ±10 V. Application of voltage beyond these levels will result in incorrect measurements, but no instrument damage, up to a maximum of ±35 V. Voltages beyond this limit can permanently damage the EX1048.

NOTE	The application of voltages beyond ±35 V can permanently damage the EX1048.

The measurement range of the EX1048 in terms of temperature is a function of its input voltage range and the capabilities of the thermocouple sensors themselves. Specifically, the measurement range of the EX1048 for the standard thermocouple types is the following:

	Min (°C)	Max (°C)	Min (°F)	Max (°F)
Type J	-200	1200	-328	2192
Type K	-200	1372	-328	2502
Type T	-200	400	-328	752
Type E	-200	900	-328	1652
Type S	-50	1768	-58	3214
Type R	-50	1768	-58	3214
Type B	250	1820	482	3308
Type N	-200	1300	-328	2372

TABLE 3-2: EX1048 MEASUREMENT RANGE

In addition to the standard thermocouple types, the EX1048 can accept and measure a custom thermocouple of any type. The effective measurement range in temperature is subsequently determined by the input voltage range of the EX1048 and the thermocouple transfer function of the custom thermocouple.

COLD JUNCTION COMPENSATION (CJC)

For highest accuracy and stability, the EX1048 provides an embedded isothermal input section that is monitored by twelve precision thermistors, one for every four thermocouple channels. To insure that the CJC information is current and time correlated with the input channels, the CJC channels are measured with every scan, providing a maximum time separation of less than 4 ms between the measurement of an input channel and its associated CJC measurement. The association between CJC channels and input channels is:

CJC #	Channel	Input Channel
CJC0	48	0-3
CJC1	49	4-7
CJC2	50	8-11
CJC3	51	12-15
CJC4	52	16-19
CJC5	53	20-23
CJC6	54	24-27
CJC7	55	28-31
CJC8	56	32-35
CJC9	57	36-39
CJC10	58	40-43
CJC11	59	44-47

TABLE 3-3: CJC CHANNEL / INPUT CHANNEL RELATIONSHIP

The user has configuration control over the reporting of the measured CJC data. This control only affects the display of their data, not the actual measurement of them. They are updated with every scan, regardless of their reporting status. If they are reported, their values are provided with units of ºC, unaffected by the °C/°F configuration setting of the input channels.

The EX1048 also accommodates the use of an external cold junction that is maintained and measured by the user. In this application, the cold junction temperature in ºC is entered into the EX1048 and enabled on a per channel basis. That is, the use of internal and user-defined CJC inputs can be mixed throughout the unit. This control is unaffected by the °C/°F configuration setting of the input channels.

The default selections are:

- CJC reporting is disabled
- User-defined CJC temperatures are 0.0
- User-defined CJC temperatures are disabled for all channels

TEMPERATURE UNITS

The EX1048 can output its temperature data with units of ºC or ºF. This is controlled on a global basis, such that all input channels are configured with one setting. This selection applies to input channel data only. CJC measurement data and the input values for external CJC temperatures are fixed at ºC. Similarly, input channels configured for an EU conversion of voltage are unaffected by this setting.

The default selection is °C.

SAMPLING RATE / NOISE PERFORMANCE

The EX1048 can be configured for a sampling rate up to a maximum of 1 kHz, regardless of the number of channels included in the scan list. The selected sampling rate refers to the frequency at which the entire scan list is measured. As such, all channels are measured at the same sampling rate. When the requested sampling rate is significantly less than 1 kHz, however, the EX1048 automatically takes and averages multiple samples. This offers improved noise performance, while maintaining the requested data output rate. For example, 100 readings of a K type thermocouple at 25 ºC are displayed with the following noise profiles at different selected sampling rates:

Sampling Rate	Noise (ºCp-p)
1 kHz	0.26
500 Hz	0.13
400 Hz	0.10
300 Hz	0.08
≤200 Hz	0.07

TABLE 3-4: TYPE K NOISE PROFILE VS. SAMPLING RATE

As implied by the table above, the maximum averaging occurs once the sampling rate reaches 200 Hz and represents the ultimate noise floor of the instrument. Decreasing the sampling rate further does not result in a lower noise profile. In practice, the noise performance delivered by the EX1048 is a function of the slope of the thermocouple characteristic for the sensor being used. That is, type E thermocouples (59 µV/ºC) will deliver inherently quieter measurements than type K thermocouples (39 µV/ºC). However, the relative relationship between noise and sampling rate shown above will be true for any thermocouple type. Consequently, for best instrument noise performance, the sampling rate should be set as low as the data collection requirements allow.

NOTE The described averaging mechanism occurs only if the trigger model is configured with a TRIG source of Timer. For more details, see *Maximizing Measurement Performance* in Section 4.

SCAN LIST CONFIGURATION

The EX1048 can be configured to include from 1 to all 48 of its input channels in the scan list. Because of the channel independence present in the EX1048 design, there are no accuracy, noise, or speed ramifications from the structure of the scan list. Its channel entries can consequently be solely dictated by the user's application requirements. A valid scan list consists of:

- at least one channel
- no more than 48 channels
- no repeated channels

A scan list entered via the embedded web page interface will be scanned in sequential order from the lowest to the highest numbered enabled channel. For greater control, the instrument driver permits the entry of channels into the scan list in an arbitrary order.

Each EX1048 input channel maintains its high input impedance and operational independence from the other channels regardless of its inclusion in the scan list. That is, the connection or lack of connection to a valid input signal of unscanned channels makes no difference.

SCAN LIST TIMING

Each scan sequence commences with the sequential measurement of the twelve CJC channels, regardless of the population of the scan list, the use of external CJC temperatures, or the EU conversions employed on the input channels. The enabled scan list channels are then individually

measured. In order to provide the tightest time correlation possible between channel measurements, the scan sequencer cycles through the scan list as fast as possible. Because of the averaging mechanism discussed in *Sampling Rate / Noise Performance*, the exact timing between channels depends on the selected sampling rate. For reference, the interchannel timings for different sampling rates are the following:

Sampling Rate	Interchannel Timing (µs)
1 kHz	16.6
500 Hz	32.1
400 Hz	39.8
300 Hz	51.5
≤200 Hz	74.7

TABLE 3-5: INTERCHANNEL TIMING VS. SAMPLING RATE

Applications that require maximum time correlation of transient signals will benefit from the following suggestions to configure the unit for maximum speed:

- set a sampling rate of 1 kHz
- eliminate unneeded channels from the scan list
- employ the 1 kHz setting of the hardware filter

SELF-CALIBRATION

In order to deliver high measurement accuracy over a wide ambient operating temperature range, the EX1048 provides the ability to perform an instrument self-calibration. During self-calibration, the input signal conditioning paths are disconnected from the input jacks and connected instead to a calibration bus that is driven by an internal calibration source. Through measurement of the conditioning paths at multiple calibration source points, software compensation for circuitry drift since the last full calibration is conducted. Once self-calibration is performed, the *Enhanced Thermocouple Accuracy* in Table 1-2 is valid for 30 days and over a ±2 ºC ambient temperature change. Note, however, that the unit must still undergo a full calibration at least once a year, regardless of the use of self-calibration.

Self-calibration should be conducted as often as practical, especially if the ambient environment has changed significantly since the previous calibration. However, fast ambient environmental changes should ideally be followed by a period of thermal stabilization before conducting self-calibration to allow the internal circuitry to stabilize to a new thermal operating condition. The self-calibration process completes quickly and does not require removal of the actual input connections, making it convenient to run often.

Similarly, self-calibration should only be performed after the EX1048 has been allowed to warm-up from a cold start for at least 60 minutes. To protect the user, the instrument will return a warning if self-calibration is initiated prior to the completion of this warm-up time. However, it is only a warning, and it can be overridden by repeating the calibration command. An override would be completely acceptable, for example, in cases where a) the unit is already fully warmed-up and is quickly moved from one physical location to another or b) instrument line power is briefly lost due to a facility power outage.

Self-calibration does not overwrite, modify, or take the place of the instrument's nonvolatile calibration constants generated by a full calibration. Instead, it generates an additional set of calibration constants that are applied to the measurement calculation after the full calibration constants. By default, self-calibration data is volatile, meaning that it is not saved through instrument resets or power cycles. This insures that the instrument always initializes with calibration constants generated by a full calibration. This feature is particularly important when

the instrument is being shared among multiple users. Each user is consequently sheltered from the actions of others.

Despite having the ability to conduct self-calibration at any time, there may be user applications that require the use of self-calibration, but demand that it create nonvolatile data. The EX1048 supports that operation as well. Once self-calibration is performed, the data can be stored to nonvolatile memory through a separate command. Similarly, previously stored self-calibration data can be loaded or cleared from nonvolatile memory.

Self-calibration offers a convenient way to mitigate the effects of time and temperature on the signal conditioning circuitry of the EX1048, resulting in significant performance improvement. However, it cannot compensate or correct for cold junction thermal stabilization errors that are created by excessive external sources of heat/cold or a population change of input connectors that is not followed by a thermal equilibrium delay. Similarly, any error created from the resistive drop of very long thermocouple wires is outside of the calibration loop and is not eliminated. For more details, see *Maximizing Measurement Performance* in Section 1.

Self-calibration operations can be performed through the embedded web page or instrument driver interfaces. For web page operations, see *Self Calibration* in Section 5. For programming operations, see *Performing Self-Calibration* in Section 6.

OPEN THERMOCOUPLE DETECTION / LIMITS

The EX1048 provides two unique sets of programmable limits that are used for open thermocouple detection as well as general purpose input channel monitoring. These limits, termed limit set 0 and limit set 1, are programmable on a per channel basis. Limit conditions are evaluated with each completed scan and are updated with a maximum latency of 25 ms.

Each limit set has an upper and lower limit value, and limit evaluations can be done against either or both values. The output of limit evaluations is presented in three forms. The operation of the front panel LEDs is tied to the upper and lower limit values of limit set 0. The operation of the digital I/O port can be optionally linked to any combination of the upper and lower limit values of either or both limit sets. Finally, the limit condition information is accessible through the instrument driver.

Because of its linkage to the front panel LEDs and the typical use of these LEDs as open thermocouple detection, limit set 0 has a unique operating feature not present in limit set 1. By default, the values in limit set 0 are set automatically, based on the EU conversion and units selection for each channel. Specifically, the upper and lower limit values are set to the upper and lower values of the EX1048 measurement range, as specified in Table 3-2. If desired, this automatic operation can be disabled, allowing the manual setting of the limit values. In manual operating mode, the limit values remain constant through EU conversion and unit changes. Limit set 1 operates in manual mode only.

As previously mentioned, limit evaluations can be linked to the operation of the digital I/O port. This linkage is termed a DIO Limit Event. For more information, see *Digital I/O and DIO Limit Events.*

The clearing of limit conditions is slightly different for each presentation mechanism. The front panel LEDs always operate in a latch configuration, meaning that a channel's LED remains illuminated even if subsequent channel readings within an acquisition sequence are not out of limit. This behavior is critical to identifying an open thermocouple that is intermittent in its failure. Clearing is done at the beginning of a new acquisition. DIO Limit Events can be programmed to operate in latch or non-latch mode. In either case, they are also cleared at the beginning of a new acquisition. Finally, the instrument driver provides information on the accumulated limit status and is cleared at the beginning of a new acquisition.

The default values for each limit set are their maximum values.

DIGITAL I/O AND DIO LIMIT EVENTS

The EX1048 features an 8-channel digital I/O port on the rear panel of the instrument. This port can be used as an arm/trigger source, for presentation of limit evaluation information, and as a general purpose output device. The digital I/O connector is a standard DB-9 with the following pin assignment:

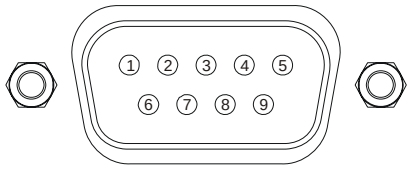

FIGURE 3-1: DIGITAL I/O DB-9 CONNECTOR

Pin	Function
1	DIO Channel 0
2	DIO Channel 1
3	DIO Channel 2
4	DIO Channel 3
5	DIO Channel 4
6	DIO Channel 5
7	DIO Channel 6
8	DIO Channel 7
9	GND

TABLE 3-6: DIGITAL I/O CONNECTOR PIN ASSIGNMENT

As a general purpose output device, each DIO channel can be independently programmed with regards to its output functionality and its static level to assume when enabled as an output. When not enabled as an output, a channel becomes tri-stated, preventing conflict with other potential voltage drivers. Reference the port's electrical specifications in Table 3-7 for voltage tolerance limits and output drive capabilities. Regardless of output functionality, each channel provides constant input functionality. That is, the input level on each channel can be accessed without a specific enable command. Moreover, the base functionality of the DIO channels is not affected by triggering, scanning, or any other instrument process. Unless linked to a limit condition, as discussed below, its operation is completely autonomous.

When enabled as an output, each channel also has the ability to generate a 1 µs pulse upon command. An example application of this pulse is to use the EX1048 to externally trigger another piece of test equipment. The specific operation of the pulse depends on the static level programmed for that channel. When a channel is programmed with a static level of high, the pulse will be low-going. When a channel is programmed with a static level of low, the pulse will be high-going. Each pulse generation requires a separate command.

For expanded and more automated operation, each DIO channel can be independently linked to one or multiple limit conditions on one or more input channels. This is termed a DIO Limit Event. For example, DIO channel 0 can be programmed to go high when the upper limit of set 0 for channel 2 or the lower limit of set 1 for channel 1 is exceeded. When linked as a limit event, a DIO channel will be cleared at the beginning of a new acquisition. Its state will then be updated with each scan according to the programmed limit evaluations. By default, the cleared state is low, but can be set on a per channel basis to be high through the use of the Invert setting. Similarly, the default operation of each channel is non-latch mode, but can be set on a per channel basis to be latch mode. In latch mode, a transition out of the cleared state would remain, regardless of future limit evaluations, until it is cleared at the beginning of a new acquisition.

It is important to note that the control of the DIO channels through DIO Limit Event assignment does not lock out control through the direct output mechanism. For example, even if a DIO channel has been set high by a DIO Limit Event, it could be asynchronously set low through direct output control. Because limits can only be evaluated as fast as data is being acquired, there could be an application that employs a slow sampling rate but requires the DIO channel to be reset sooner than the normal limit evaluation mechanism would do it. The direct DIO control provides this capability. Use of that control does not alter or disable the limit event mechanism controls; it simply asynchronously alters the level of the DIO channel output. Upon the next scan (and subsequent limit evaluation), the DIO channel will be updated normally per its DIO Limit Event configuration. In general, however, an application will employ only one control mechanism. For that reason, if a DIO channel is linked as a DIO Limit Event, this is noted in the direct control mechanism as a pseudo-warning to guard against accidental use.

The default selections for each DIO channel are:

- output enable is off
- output level is 0

The default selections for DIO Limit Events are:

- all channels are disabled
- invert is off
- latch is off

The electrical specifications for the digital I/O port are provided in Table 3-7. Particular note should be given to the V_{INPUT} specification of -0.5 V to 5.5 V. Exceeding that value with an external voltage source, even through a resistance, could permanently damage the EX1048.

Characteristic	Value
V_{INPUT}	-0.5 V to 5.5 V
V_{IH}	2 V min
V_{IL}	0.8 V max
V_{OH} (I_{OH} = -32 mA)	2 V min
V_{OL} (I_{OL} = 64 mA)	0.55 V max

TABLE 3-7: DIGITAL I/O PORT ELECTRICAL SPECIFICATIONS

LXI TRIGGER BUS

The EX1048 features an 8-channel LXI trigger bus (VTB) on the rear panel of the instrument. This differential-pair LVDS bus consists of two identical ports connected in parallel. The primary use of the trigger bus is the transmission of high-speed signals for multiple-unit triggering and synchronization, but it can also be controlled as a general purpose output device. However, it is only designed to interface with other LVDS ports, not general purpose digital I/O ports. The trigger bus connector is a micro DB-25 with the following pin assignment:

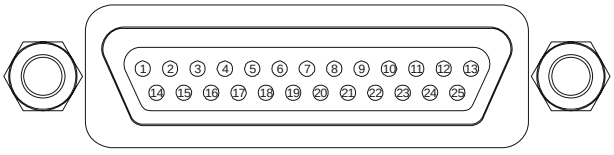

FIGURE 3-2: TRIGGER BUS DB-25 CONNECTOR

Pin	Function	Pin	Function
1	GND	14	VTB Channel 0 Pos
2	VTB Channel 0 Neg	15	GND
3	VTB Channel 1 Pos	16	VTB Channel 1 Neg
4	GND	17	VTB Channel 2 Pos
5	VTB Channel 2 Neg	18	GND
6	VTB Channel 3 Pos	19	VTB Channel 3 Neg
7	GND	20	VTB Channel 4 Pos
8	VTB Channel 4 Neg	21	GND
9	VTB Channel 5 Pos	22	VTB Channel 5 Neg
10	GND	23	VTB Channel 6 Pos
11	VTB Channel 6 Neg	24	GND
12	VTB Channel 7 Pos	25	VTB Channel 7 Neg
13	GND		

TABLE 3-8: TRIGGER BUS CONNECTOR PIN ASSIGNMENT[1]

Connection to the trigger bus connector is done with a VXI Technology trigger bus cable (P/N: 70-0307-001). For proper operation, each trigger bus connector must be populated with either a cable or a termination. Since the two connectors per unit are in parallel, they are electrically equivalent with respect to receiving the cable or termination. VXI Technology P/N: 70-0304-000 provides two termination assemblies, which is sufficient to connect any number of instruments, since a termination is only needed at each end. An example of a three-unit trigger bus installation is shown in Figure 3-3.

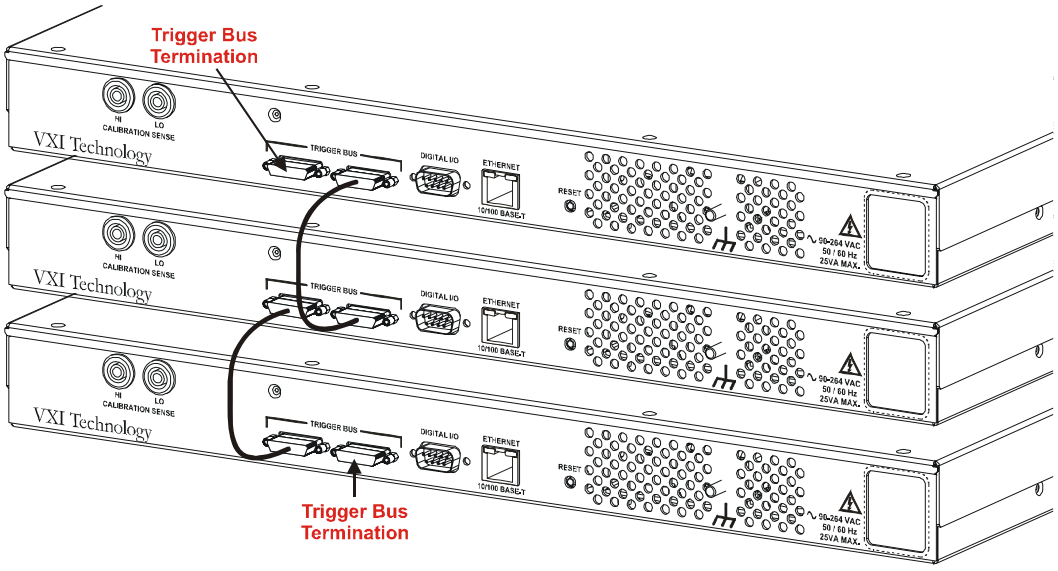

FIGURE 3-3: TRIGGER BUS CABLING EXAMPLE

As a general purpose output device, each VTB channel can be independently programmed with regards to its output functionality and its static level to assume when enabled as an output. When not enabled as an output, a channel becomes tri-stated, preventing conflict with other potential voltage drivers. Regardless of output functionality, each channel provides constant input functionality. That is, the input level on each channel can be accessed without a specific enable command. Moreover, the base functionality of the VTB channels is not affected by triggering, scanning, or any other instrument process.

[1] Note that these pin assignments are specific to the rev. A version of the EX1048. For rev. B pin assignments, please visit *www.vxitech.com* for the latest manual revision.

When enabled as an output, each channel also has the ability to generate a 1 µs pulse upon command. The specific operation of the pulse depends on the static level programmed for that channel. When a channel is programmed with a static level of high, the pulse will be low-going. When a channel is programmed with a static level of low, the pulse will be high-going. Each pulse generation requires a separate command.

A common application of the pulse generation is multiple-unit synchronization. Referring to Figure 3-3 above, each unit can be configured to trigger on the same VTB channel. Then one unit is selected as a master and is used to output a signal on the designated VTB channel line, serving as a common trigger source for all three instruments.

The default selections for each VTB channel are:

- output enable is off
- output level is 0

The electrical specifications for the trigger bus are provided in Table 3-9.

Characteristic	Value
Logic Type	M-LVDS Type 2
V_{IT+}	150 mV max
V_{IT-}	50 mV min
V_{OS}	1 V typical

TABLE 3-9: TRIGGER BUS ELECTRICAL SPECIFICATIONS

LOCKING

By default, the EX1048 allows unrestricted operation from multiple hosts, both through the web page and instrument driver interfaces. While this offers a high level of user flexibility, there are instances where protected operation is desirable, if not required. For these applications, the EX1048 can be "locked," meaning that it will accept commands from only the host IP address that issued the lock command. With the EX1048 in this mode, other host connections that attempt commands will be denied.

By design, the locking mechanism is able to be overridden by a secondary host that issues a break lock command. Thus, the lock provides a warning to other users that the unit is in a protected operation state, but not absolute security. This allows for instrument recovery if the locking IP address would become disabled.

Self-calibration requires the acquisition of a lock prior to its initiation.

TRIGGERING

The EX1048 supports a full function trigger model with a separate arm source and trigger source event structure. For a complete explanation of the trigger model, see Section 4. In summary, an acquisition sequence is enabled with a trigger initialize command. Scanning is then initiated upon the receipt of the programmed arm source event followed by the receipt of the programmed trigger source event. Trigger and arm source events can be independently programmed from a variety of sources including Immediate, Timer, Digital I/O, and the Trigger Bus.

DATA FORMAT

By default, the data returned during data retrieval is limited to the channel readings and the absolute time of scan initiation. If enabled, the EX1048 can also return the measured CJC temperatures and the delta timestamp of each channel reading from the scan initiation time.

ACQUIRING DATA

In general, the acquisition and retrieval of data on the EX1048 are conducted with discrete commands that are often separated in time to a large degree. The EX1048 utilizes a 20 MB FIFO memory storage to buffer acquisition data prior to retrieval. This reading buffer is cleared upon receipt of the trigger initialize command in preparation for reading storage. Then, upon fulfillment of the programmed trigger model conditions, the EX1048 initiates the scanning of the channel configuration that is defined in the scan list. It continues scanning and storing the acquisition data until the trigger and arm count quantities are reached or the acquisition is aborted. At that point, scanning ceases, and the trigger model is returned to the INIT layer.

The amount of scans that can be buffered within the memory is dependent on the number of channels in the scan list and the requested data format. For this reason, very long scan sequences (in terms of count) will benefit from a minimization of the data format, so as to maximize the number of scans that can be stored. Specifically, the number of scans that can be buffered (Page_Count) is determined by the following formula:

$$\text{Channel\_Size} = 4 + 2 \times \text{Timestamp\_Reporting}$$

$$\text{CJC\_Channel\_Size} = 4 + 2 \times \text{Timestamp\_Reporting}$$

$$\text{CJC\_Channel\_Count} = 16 \times \text{CJC\_Reporting}$$

$$\text{Page\_Size} = 108 + \text{Channel\_Count} \times \text{Channel\_Size} + \text{CJC\_Channel\_Count} \times \text{CJC\_Channel\_Size}$$

$$\text{Page\_Count} = 20971520 / \text{Page\_Size}$$

where: Channel_Count 1-48 (number of channels in scan list)
CJC_Reporting 0 or 1 (1 = YES, 0 = NO)
Timestamp_Reporting 0 or 1 (1 = YES, 0 = NO)

Page_Count sizes for some typical configurations are the following:

Channel_Count	CJC_Reporting	Timestamp_Reporting	Page_Count
1	0	0	187246
16	0	0	121927
32	0	0	88862
48	0	0	69905
48	1	0	57614
48	0	1	52958
48	1	1	42625

TABLE 3-10: EXAMPLE PAGE COUNT SIZES

If the reading buffer fills to its maximum capacity during scanning, the behavior with regards to additional readings is governed by the setting of the instrument's blocking mode. With blocking mode enabled, additional readings are discarded, leaving the contents of the buffer intact. With blocking mode disabled, the buffer becomes circular, in that additional readings overwrite the oldest readings.

The default selection for blocking mode is disabled.

NOTE	The reading buffer memory is volatile and is cleared upon an instrument reset or power cycle.

RETRIEVING DATA

In general, acquisition data is retrieved from the EX1048 upon the completion of the acquisition. This mode of operation is fully supported by the embedded web page as well as the instrument driver. In addition, data retrieval in the midst of an acquisition sequence is supported by the instrument driver. In either case, retrieved data is fully calibrated and compensated by the EX1048 and output directly in the requested units. No post-acquisition user manipulations are required. Data can be retrieved one page (scan) at a time or downloaded in multiple page blocks up to the maximum number of pages that exist on the instrument at the time of retrieval. In order to provide the maximum reading buffer capacity for future acquisitions, data is deleted from the FIFO memory upon retrieval.

USER-DEFINED CONVERSIONS

The EX1048 nominally accepts standard thermocouple types and performs its thermocouple calculations using polynomial coefficients from the NIST ITS-90 Thermocouple Database. In some applications, however, a user may want to override the embedded coefficients with a user-defined coefficient set. One reason to do this is if the transfer function of the specific thermocouple being used has been completely characterized to an accuracy level that exceeds standard thermocouple limits of error. Another reason to do this is if a non-standard thermocouple is used. Up to two unique sets of coefficients can be entered. Specifically, the use of custom thermocouple equations requires the user to know or generate the coefficients for two conversion polynomials.

The *forward conversion polynomial* is used to convert a CJC temperature into a compensating cold junction voltage and has the form of:

$$E = c_0 + c_1 * t^1 + c_2 * t^2 + \ldots + c_{12} * t^{12}$$

where E is in volts, t is in ºC, and c_0 - c_{12} are the coefficients.

The *inverse conversion polynomial* is used to convert a compensated input voltage into temperature and has the form of:

$$t = d_0 + d_1 * E^1 + d_2 * E^2 + \ldots + d_{12} * E^{12}$$

where E is in volts, t is in ºC, and d_0 - d_{12} are the coefficients.

The default values for the coefficients are 0.0.

NOTE	The entry of user-defined coefficients does not automatically enable their use. The enabling is done by setting the EU conversion to User0 or User1.

SECTION 4

TRIGGERING

OVERVIEW

The EX1048 supports a full function trigger model with a separate arm source and trigger source event structure. The trigger model is based on the industry standard SCPI 1999 Trigger Subsystem and is diagramed in Figure 4-1.

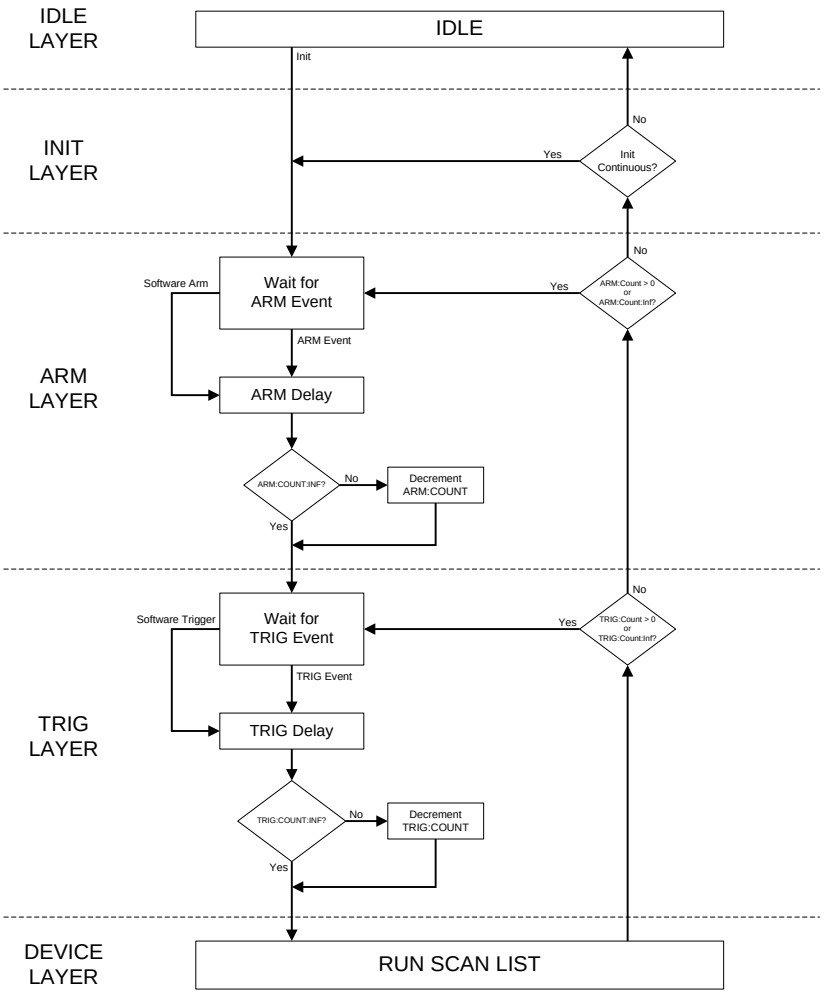

FIGURE 4-1: EX1048 TRIGGER MODEL

The trigger model is sectioned into 5 layers: IDLE, INIT, ARM, TRIG, and DEVICE. The EX1048 reset condition places it in the IDLE state. A trigger initialize command begins the acquisition sequence by transitioning the instrument through the INIT layer into the ARM layer. As this occurs, the reading buffer memory is cleared, and all enabled limit events are reset to their default states.

Upon entering the ARM layer, the ARM Count is reset to its specified value. The instrument remains in the ARM layer until the specified ARM event occurs or a Software Arm is issued. Once that occurs, the specified ARM Delay is waited, the ARM Count is decremented, and the instrument transitions into the TRIG layer.

Upon entering the TRIG layer, the TRIG Count is reset to its specified value. The instrument remains in the TRIG layer until the specified TRIG event occurs or a Software Trigger is issued. Once that occurs, the specified TRIG Delay is waited, the TRIG Count is decremented, and the instrument transitions into the DEVICE layer.

In the DEVICE layer, the scan list is measured, calibrated, and stored into local memory. In addition, limit evaluation is conducted, and enabled limit event states are updated.

If the TRIG Count remains nonzero, the instrument stays in the TRIG layer until the specified TRIG event (and subsequent device action) occurs enough times to decrement it to zero. Once the TRIG Count reaches zero, the instrument then evaluates the remaining ARM Count and repeats the ARM layer action if it is nonzero. However, since each transition into the TRIG layer resets the TRIG Count, each additional ARM layer action results in the full specified number of TRIG Count actions through the TRIG layer and DEVICE layer.

Once the ARM Count reaches zero, the instrument transitions back into the INIT layer. If Init Continuous mode is enabled, the ARM layer is automatically reentered without the issuance of a trigger initialize command. However, unlike with a trigger initialize, enabled limit events are not reset, and the reading buffer memory is not cleared. Conversely, if Init Continuous mode is disabled, the instrument is returned to the IDLE layer and requires the issuance of a new trigger initialize command to begin a new acquisition sequence.

EVENTS

There are two events that control the progress through the trigger model: the ARM event and the TRIG event. Each of these events can be programmed independently to be activated from any combination of the Trigger Bus, the Digital I/O port, and the internal system timer. In addition, each event can be programmed to be Immediate, creating a permanent satisfaction of the event monitor. Each event monitor can also be bypassed on command with the issuance of a Software Arm or Software Trigger, as appropriate. Software Arms and Software Triggers are always enabled, regardless of the other programmed event sources.

The Digital I/O Port and the Trigger Bus monitor the digital hardware ports on the rear panel of the instrument. An event can be controlled by any combination of the eight channels of each port. Moreover, each channel can be independently programmed to monitor a transition edge or a level.

The instrument's internal system timer generates ticks at the specified timer interval, where each tick represents a unique event. For example, a TRIG event source set to Timer at a timer interval of 0.005 s will generate internal triggers at a rate of 200 Hz.

As previously mentioned, an event condition can be designated to be any combination of Trigger Bus channel transitions or levels, Digital I/O channel transitions or levels, and Timer ticks. If multiple sources for an ARM event are specified, they are logically combined as follows:

ARM event = [(Timer tick event) AND (Digital I/O event) AND (Trigger Bus event)]

If multiple sources for a TRIG event are specified, they are logically combined as follows:

TRIG event = [(Timer tick event) AND (Digital I/O event) AND (Trigger Bus event)]

Within each digital hardware port, it is also possible to select multiple channels and multiple conditions for a specific channel. In that case, they are logically combined as follows:

Digital I/O event = (Ch 7 events) AND (Ch 6 events) ... AND (Ch 0 events)

Finally, within each channel, the four event conditions of Pos. Edge, Neg. Edge, Pos. Level, and Neg. Level are OR'ed together.

As an example, if a Digital I/O event is specified to be a combination of a Pos. Level on channel 3, a Pos. Edge on channel 6, and a Neg. Edge on channel 6, the event will be satisfied when a Pos. Level on channel 3 occurs simultaneously with a Pos. Edge or a Neg. Edge transition on channel 6. While the Digital I/O event structure was used as an example, the Trigger Bus event structure operates in the same manner.

NOTE	The extensive flexibility of the trigger model system permits the creation of very specialized trigger conditions, which is a powerful application tool. However, it also permits the creation of trigger conditions that would be very difficult to satisfy in practice. For example, if edge conditions are specified on multiple digital hardware channels, the edges must occur within 25 ns of each other to be recognized as having occurred simultaneously. Similarly, the timer source should not be combined with any digital hardware edge conditions.

MAXIMIZING MEASUREMENT PERFORMANCE

As explained in *Sampling Rate / Noise Performance* in Section 3, the EX1048 takes advantage of a sampling rate setting of less than 1 kHz to acquire and average multiple samples. This offers improved noise performance, while maintaining the requested data output rate. However, the trigger model must be configured in a specific manner to allow this to occur. Specifically, the TRIG event source must be configured for Timer. This is critical, because only under this condition does the EX1048 know exactly the rate and timing of the trigger events. That is, it knows the available time in which to complete the acquisition and consequently how many samples it can average and still complete the entire scan list. With any other trigger source, the triggers can occur at any time. In that case, the EX1048 completes the scan list in minimum time and does not average.

If the desired trigger model configuration requires the use of the digital I/O port and/or the trigger bus, it is thus best to configure the ARM source for the required digital event, leaving the TRIG source as Timer. The TRIG source should be changed from Timer only in the situation where triggering is required on a two-stage digital event. An example of this is arming on a digital I/O channel level and then triggering on a trigger bus channel edge.

MAXIMUM TRIGGER RATE

The ability to configure the TRIG event source to be an external event such as a Digital I/O channel level or a Trigger Bus channel edge is a powerful test feature, allowing measurements to be precisely correlated with external events. External events, however, could nominally be generated at a rate far faster than the EX1048 can completely and correctly execute the DEVICE layer. To prevent the instrument from being overdriven, the trigger model contains an internal pacer timer. Once the trigger event has been satisfied, this timer disables the recognition of all trigger events for a time period of 990 µs. Trigger events that occur during this 990 µs period are ignored, not buffered. This is similar in concept to the fact that, for example, additional ARM events are ignored if the trigger model is not specifically waiting for an ARM event. Due to this action, triggering the EX1048 with a burst of external pulses that exceeds 1 kHz will result in not

only fewer readings than trigger events, but also a reading rate that is potentially far less than the maximum rate.

Consider an example in which the trigger model is set to the following configuration:

Init Continuous	Disabled
Timer Interval	0.1
Arm Source	Immediate
Arm Count	1
Arm Delay	0
Trig Source	DIO Channel 0 Pos. Edge
Trig Count	1
Trig Delay	0

Once initialized, the EX1048 is driven by a burst of 100 positive edge transitions at a rate of 1.5 kHz. At 1.5 kHz, the time between trigger events is 667 µs. Since this time is less than 990 µs, every other trigger event will be ignored. Consequently, this burst will generate 50 readings at a rate of 750 Hz. Note that the actual reading output rate is far less than the 1 kHz maximum rate. To achieve the maximum output rate, the TRIG event source should be set to Immediate or Timer (with a Timer Interval of 0.001).

SECTION 5

WEB PAGE OPERATION

INTRODUCTION

The EX1048 offers an embedded web page for easy remote operation without any programming. Virtually all of the EX1048's configuration controls and data collection options are available through the web page, and it also serves as the mechanism to read and set network parameters.

OPENING THE WEB PAGE

Type the EX1048's assigned IP address into an internet browser application.

GENERAL WEB PAGE OPERATION

When initial connection is made to the EX1048, the instrument home page appears. This page displays instrument-specific information including:

- Serial number
- MAC address
- Firmware version
- Date of last full calibration
- Presence of nonvolatile self cal data
- Error status

This page is accessible from any other instrument page through the **EX1048 Main** link on the command menu.

All of the configuration, data acquisition, and data retrieval features available through the web page are accessed through the EX1048 command menu that is displayed on the left hand side of every internal web page. The entries on the command menu represent three types of pages:

Status This type of page performs no action and accepts no entries. It provides operational status and information only. The **EX1048 Main** page is an example of a status page.

Action This type of page initiates a command on the instrument, but does not involve parameter entry. The **Trigger…Initialize** page is an example of an action page.

Entry This type of page displays and accepts changes to the configuration controls of the instrument. The **Scan List…Configuration** page is an example of an entry page.

Use of the entry-type web pages in the EX1048 are governed by a common set of operational characteristics:

- Each page contains two standard buttons: *Apply* and *Cancel*.
- Pages initially load with the currently-entered parameters or selections displayed.
- Changes that are made after the initial load are not accepted until the *Apply* button is clicked. Once the *Apply* button is clicked, the page will reload, showing the new state of the page.
- Changes that are made after the initial load can be rejected by clicking the *Cancel* button. If cancelled, the parameters or selections will revert to the values that were last accepted. Leaving a page before applying any changes has the same effect as canceling the changes.
- Navigation through a parameter screen is done with the Tab key. The Enter key has the same function as clicking the *Apply* button and cannot be used for navigation.

TRIGGER MENU

The Trigger menu is used to configure and operate the EX1048's trigger system.

Configuration

This entry page is used to configure the arm and trigger sources of the EX1048. The page is designed with a collapsible and expandable menu structure that is retained between page visits. This allows the user to expose only as much as the trigger system as needed, making navigation through the configuration options easier. Detailed information on triggering is provided in Section 4. Most of the controls on this page are checkboxes to enable different levels of functionality. The rest are parameter entry fields that have the following valid input ranges:

Parameter	Min	Max	Resolution
Timer Interval (s)	0.001	4294	0.000001
Arm Count	1	$(2^{31}-1)$	1
Arm Delay (s)	0	4294	0.000001
Trig Count	1	$(2^{31}-1)$	1
Trig Delay (s)	0	4294	0.000001

TABLE 5-1: TRIGGER CONFIGURATION PARAMETER RANGES

In addition to the two standard buttons, this page also provides a *Default* button. This button provides a one-click method of returning all trigger configuration parameters to their reset values. It should be noted that the use of this button does not eliminate the need to click the *Apply* button to accept the changes.

Initialize

This action page begins an acquisition, transitioning the trigger model from the IDLE layer to the ARM layer. This command clears the FIFO reading memory and resets all limit indications.

Abort

This action page aborts the current acquisition and returns the trigger model to the IDLE layer. Once an acquisition has begun, most commands are not accessible. The abort command is used to prematurely end the acquisition in order to make configuration changes. Acquired data is not affected by the issuance of an abort.

Software Arm

This action page performs a software arm of the trigger model. Software arms are always enabled as an ARM source and can be used to perform a bypass of the configured ARM source. The software arm applies at the time it is issued and will be ignored if the system is not waiting for an ARM event. The issuance of a software arm has no effect on the ARM source configuration.

Software Trigger

This action page performs a software trigger of the trigger model. Software triggers are always enabled as a TRIG source and can be used to perform a bypass of the configured TRIG source. The software trigger applies at the time it is issued and will be ignored if the system is not waiting for a TRIG event. The issuance of a software trigger has no effect on the TRIG source configuration.

SCAN LIST MENU

The Scan List menu is used to configure the EX1048 input channels with regards to scan list inclusion, EU conversion, and hardware filter setting.

Configuration

This entry page is used to define the channel scan list and the EU conversion setting for each channel. An example of this page is shown below.

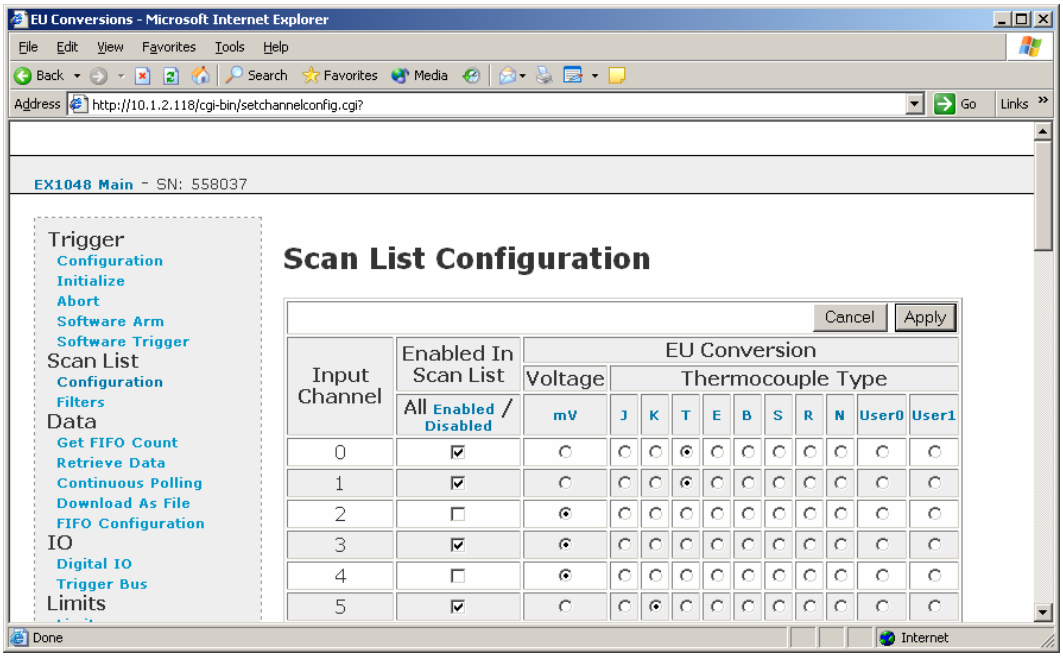

FIGURE 5-1: SCAN LIST CONFIGURATION PAGE

This screen displays that currently channels 0, 1, 3, and 5 are enabled in the scan list, with EU conversions of T, T, mV, and K, respectively.

Channels are enabled or disabled in the scan list by toggling the appropriate checkbox. Additionally, all channels can simultaneously be enabled or disabled by clicking on the *Enabled* or *Disabled* link. Each channel's EU conversion is set by clicking the appropriate radio button under the desired conversion type. Additionally, all channels can simultaneously be set to the same conversion type by clicking on the appropriate link. It should be noted that the use of these links does not eliminate the need to click the *Apply* button to accept the changes.

Filters

This entry page is used to define the hardware filter setting for each channel. Each channel's hardware filter is set by clicking the appropriate radio button under the desired frequency value. Additionally, all channels can simultaneously be set to the same frequency value by clicking on the appropriate link. It should be noted that the use of these links does not eliminate the need to click the *Apply* button to accept the changes.

DATA MENU

The Data menu is used to retrieve data from the EX1048 as well as configure the format of the data to be acquired.

Get FIFO Count

This status page displays how many pages (scans) worth of acquisition data have yet to be retrieved.

Retrieve Data

This action page extracts and displays one page (scan) worth of acquisition data on a first-in, first-out (FIFO) basis. If this page is invoked when the FIFO is empty, an error is returned. In order to provide the maximum reading buffer capacity for future acquisitions, data is deleted from the FIFO memory upon retrieval.

Continuous Polling

This action page is used to continuously trigger and retrieve one page (scan) of data from the EX1048 at a refresh rate of approximately once every 3 seconds. It is very useful for general data monitoring, installation debugging, and instrument control familiarization. It cannot, however, be used to test the trigger system, as it must alter the trigger configuration settings in order to acquire data. Once continuous polling is initiated, it can be stopped by clicking the *Stop Polling* button or a link to another page. Using the button preserves the most recent data screen.

NOTE	Use of the **Continuous Polling** page alters the trigger configuration settings. Any previously entered configuration settings must be reentered.

Download As File

This entry page is used to download all or part of the contents of the FIFO memory into a file. Supported file formats are CSV, Raw Binary, and MATLAB. By default, all scans will be downloaded. If a specific number of scans is desired, uncheck the **All** checkbox and enter in the number of scans. In order to provide the maximum reading buffer capacity for future acquisitions, data is deleted from the FIFO memory upon retrieval.

FIFO Configuration

This entry page is used to control the data format and overflow behavior of the FIFO memory.

The **Report CJC Temperatures** control enables the reporting of the CJC temperatures in the acquisition data. This control does not affect their measurement, only the display of the data. The CJC temperatures are taken with every scan, regardless of this control. Reported CJC temperatures are always in units of ºC, regardless of the **Temperature Units** selection.

The **Report Timestamps** control enables the reporting of delta timestamps per channel in the acquisition data. The timestamp represents the time, in increments of 100 ns, between the specific channel measurement and the scan initiation time.

The **Temperature Units** selection controls whether the input channel data is returned in units of ºC or ºF. This setting has no affect on the reported CJC data or data for input channels that are configured for an EU conversion of mV.

The **Blocking Mode** control governs the system behavior if the reading buffer fills to its maximum capacity during scanning. With blocking mode enabled, additional readings are discarded, leaving the contents of the buffer intact. With blocking mode disabled, the buffer becomes circular, in that additional readings overwrite the oldest readings.

IO MENU

The IO menu is used to receive and output data through the digital I/O port and trigger bus of the EX1048.

Digital IO

This entry page is used to monitor the state of the 8-channel digital I/O port and control it as a general purpose output device. Each DIO channel can be independently programmed with regards to its output functionality, its static level to assume when enabled as an output, and pulse operation.

The **Input State** status section displays the measured digital level (0 or 1) of the DIO channels each time the page is accessed or refreshed. The information does not update automatically on an input level transition or on a timer. Consequently, it should only be used to monitor static or slowly changing levels. The input monitoring functionality operates regardless of the state of the **Output Enable** control.

The **Output State** control provides a pull-down selection in which the output level of each DIO channel is specified. This setting represents the level that the channel will assume if/when enabled as an output. When a channel is not enabled, this selection has no effect.

The **Output Enable** control enables each DIO channel as an output. Once enabled, a channel will assume the level specified in the **Output State** control. When not enabled as an output, a channel becomes tri-stated.

NOTE	Before specifying a DIO channel as an output, insure that no other active voltage drivers are connected to the channel. Otherwise, incorrect operation or instrument damage could result.

The **Pulse** control is used to generate a single-shot 1 µs pulse. The specific operation of the pulse depends on the static level programmed for that channel. When a channel is programmed with a static level of high, the pulse will be low-going. When a channel is programmed with a static level of low, the pulse will be high-going. Each pulse generation requires a separate command. This control is only valid when the channel is enabled as an output. Normally, pulse operation is used once a channel's static output level is already defined. However, if selections are applied together to change the output state and output a pulse, the channel will first assume the new output state and then output the appropriate pulse.

The **Limit Event** status section indicates (Y or N) the linkage of any DIO channels to a limit evaluation condition. When linked, the output state of a channel will be primarily controlled by the limit conditions specified in the **DIO Limit Events** page. The linkage is noted here, because channels linked as DIO Limit Events are normally not used as direct control outputs. However, direct output control is not disabled and can be used to asynchronously reset a linked channel.

Trigger Bus

This entry page is used to monitor the state of the 8-channel trigger bus (VTB) and control it as a general purpose output device. Each VTB channel can be independently programmed with regards to its output functionality, its static level to assume when enabled as an output, and pulse operation.

The **Input State** status section displays the measured digital level (0 or 1) of the VTB channels each time the page is accessed or refreshed. The information does not update automatically on an input level transition or on a timer. Consequently, it should only be used to monitor static or slowly changing levels. The input monitoring functionality operates regardless of the state of the **Output Enable** control.

The **Output State** control provides a pull-down selection in which the output level of each VTB channel is specified. This setting represents the level that the channel will assume if/when enabled as an output. When a channel is not enabled, this selection has no effect.

The **Output Enable** control enables each VTB channel as an output. Once enabled, a channel will assume the level specified in the **Output State** control. When not enabled as an output, a channel becomes tri-stated.

NOTE	Before specifying a VTB channel as an output, insure that no other active voltage drivers are connected to the channel. Otherwise, incorrect operation or instrument damage could result.

The **Pulse** control is used to generate a single-shot 1 µs pulse. The specific operation of the pulse depends on the static level programmed for that channel. When a channel is programmed with a static level of high, the pulse will be low-going. When a channel is programmed with a static level of low, the pulse will be high-going. Each pulse generation requires a separate command. This control is only valid when the channel is enabled as an output. Normally, pulse operation is used once a channel's static output level is already defined. However, if selections are applied together to change the output state and output a pulse, the channel will first assume the new output state and then output the appropriate pulse.

LIMITS MENU

The Limits menu is used to configure the limit condition evaluation system of the EX1048.

Limits

This entry page is used to specify the values for the two programmable limit sets. These limits, termed limit set 0 and limit set 1, are programmable on a per channel basis. Only one set of limits is displayed at a time on the page. Navigation between the sets is provided by the **Set** control, a pull-down selection.

By default, the values in limit set 0 are set automatically, based on the EU conversion and units selection for each channel. Specifically, the upper and lower limit values are set to the upper and lower values of the EX1048 measurement range, as specified in Table 3-2. For example, if input channel 0 is configured for an E type conversion and a units selection of ºC, its upper and lower limit values will be +900 and -200, respectively. To override the automatic setting and set the values manually, check the **Manual** checkbox and enter new values. Values can be entered in decimal or scientific notation, but will always be displayed in scientific notation. Limit set 1 operates in manual mode only.

NOTE	Limit sets operating in manual mode do not automatically adjust for changes in EU conversion or units selection. For proper operation, limit values should be set after EU conversion and units selections are made and reset upon subsequent configuration changes.

DIO Limit Events

This entry page is used to link one or multiple limit conditions to the operation of a channel on the digital I/O port. Nominally, the results of limit set 0 evaluations are represented in the front panel LEDs, and the results of both limit set evaluations are accessible through the instrument driver. The DIO limit event mechanism offers the additional ability to precisely control a DIO channel through the limit evaluations on one or more input channels.

DIO limit events are programmable on a per channel basis; that is, each of the 8 DIO channels can be configured with a unique set of operational characteristics. Only one DIO channel's limit event controls is displayed at a time on the page. Navigation between the DIO channels is provided by the **DIO Channel** control, a pull-down selection.

The linkage of a DIO channel to a specific limit condition on a specific input channel is done by checking the appropriate checkbox. Multiple selections are allowed and are logically OR'ed together. For example, Figure 5-2 shows that DIO channel 0 is linked to limit conditions on input channels 0, 3, and 5. Specifically, DIO channel 0 will transition if any of the four specified limits is exceeded.

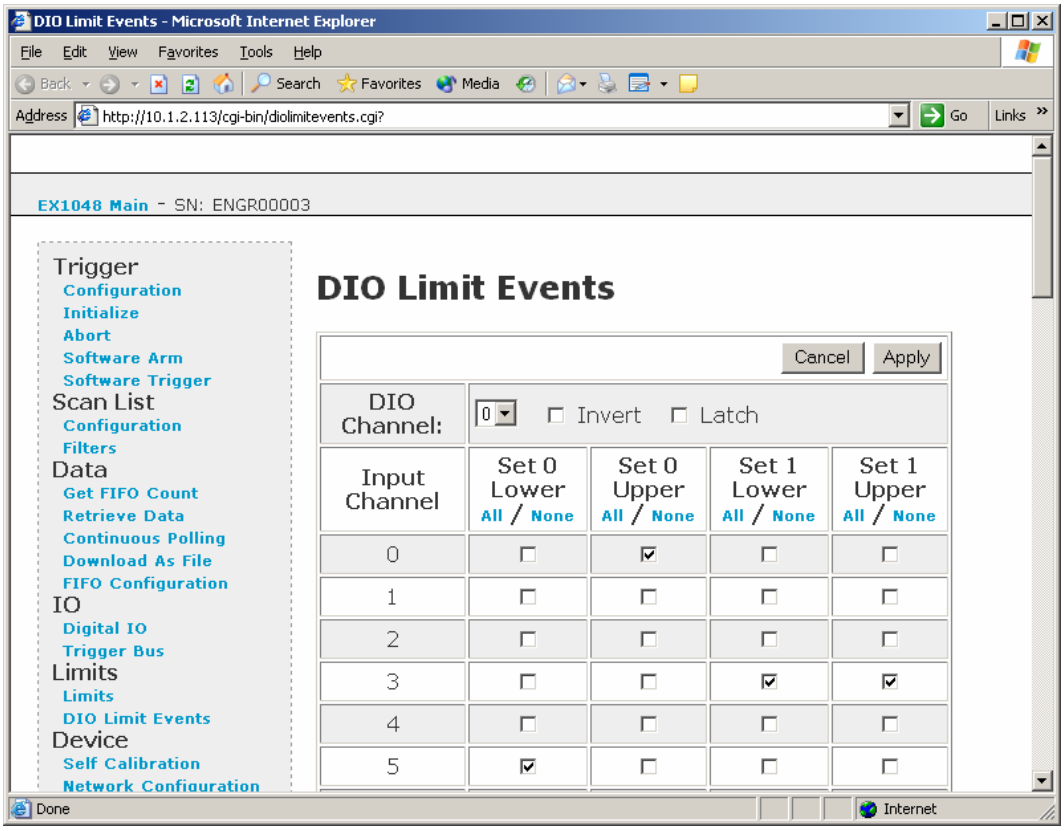

FIGURE 5-2: DIO LIMIT EVENTS PAGE

For easier entry of complex configurations, the *All* and *None* links can be used to enable or disable a limit set value across all input channels. It should be noted that the use of these links does not eliminate the need to click the *Apply* button to accept the changes.

The DIO limit events and the scan list configuration controls are autonomous. That is, specifying a limit condition in **DIO Limit Events** does not automatically enable it in the scan list. Similarly, removing an input channel from the scan list does not automatically remove it as a linked limit condition. DIO limit event linkages to unscanned channels do not cause an error condition and are simply ignored in the evaluation.

When linked as a limit event, a DIO channel will be cleared at the beginning of a new acquisition. Its state will then be updated with each scan according to the programmed limit evaluations. By default, the cleared state is low, but can be set on a per channel basis to be high through the selection of the **Invert** checkbox. Similarly, the default operation for each channel is non-latch mode, but can be set on a per channel basis to be latch mode through the selection of the **Latch** checkbox. In latch mode, a transition out of the cleared state would remain, regardless of future limit evaluations, until it is cleared at the beginning of a new acquisition.

DEVICE MENU

The Device menu is used to perform advanced configuration operations on the EX1048.

Self Calibration

This action page is used to perform operations related to self-calibration. As discussed in *Self-calibration* in Section 3, the measurement performance of the EX1048 is significantly improved through the periodic use of self-calibration. The self-calibration process completes quickly and does not require removal of the actual input connections or the use of external equipment, making it convenient to run often. In order to take full advantage of the self-calibration system, it is important to understand how the self-calibration constants are generated, used, and stored.

Self-calibration does not overwrite, modify, or take the place of the instrument's nonvolatile calibration constants generated by a full calibration. Instead, it generates an additional set of calibration constants that are applied to the measurement calculation after the full calibration constants. When self-calibration is performed, it places its calculated data in volatile memory. This data is termed the "current self cal data" and is used automatically by the instrument. No additional command is necessary. Because it is in volatile memory, however, it will be cleared upon an instrument reset or power cycle. There is the ability, however, to store the current self cal data into nonvolatile memory.

In order to perform a self-calibration, a lock on the instrument must first be acquired. For more information on locking, refer to the **Locking Menu** description in this section. If self-calibration is attempted without the acquisition of a lock, an error message will be displayed. Within this page, the self-calibration process is initiated by clicking on the *Perform Self Calibration* button. Once started, the page will confirm the initiation of the calibration process and then continuously refresh with a percentage completion status until completed. Once the completion status reaches 100%, the self-calibration process is complete.

To protect the user from performing a self-calibration while the instrument is warming up, a warning message will be issued and calibration will not be performed if self-calibration is attempted before the EX1048 has been powered on continuously for 60 minutes. This is only a warning, however, and it can be overridden with the *Override Self Calibration Uptime* button that appears when this warning is triggered. In general, this option should not be used, as performing self-calibration prior to complete instrument warm-up could result in degraded measurement performance. However, it would be appropriate to utilize the override in the case where the unit has actually been warmed up, but has simply been subjected to a quick power cycle or reboot.

As noted above, the self-calibration operation places its calculated data in volatile memory, termed "current self cal data." Once present, it can be cleared by utilizing the *Clear Current Self Cal Data* button. This operation clears the volatile data, but does not affect any self cal data that is stored in nonvolatile memory.

Despite having the ability to conduct self-calibration at any time, there may be user applications that require the use of self-calibration, but demand that it create nonvolatile data. The *Store Current Self Cal Data As Nonvolatile* button saves the current self cal data to nonvolatile memory, enabling it to be loaded upon instrument power cycle and reset. If this button is pressed when no current self cal data is present, an error is generated. Since the existence of nonvolatile self cal data represents a permanent (although revocable) change from the factory calibration settings, its presence is noted on the **EX1048 Main** and **Self Calibration** pages.

Previously saved nonvolatile self cal data can be cleared through the use of the *Clear Nonvolatile Self Cal Data* button. This operation does not clear the current self cal data, only that in nonvolatile memory. If this button is pressed when no nonvolatile self cal data is present, an error is generated. Alternatively, previously saved nonvolatile self cal data can be loaded as the current self cal data through the use of the *Load Nonvolatile Self Cal Data* button. If current self cal data previously existed, it is simply overwritten and need not be cleared in advance. If this button is pressed when no nonvolatile self cal data is present, an error is generated.

The ability to conduct a self-calibration is not affected by the existence of current self cal data or nonvolatile self cal data. A performed self-calibration will simply overwrite any current self cal data that already existed. It is not necessary to utilize the *Clear Current Self Cal Data* button prior to performing a new self-calibration. Similarly, the ability to store current self cal data as nonvolatile is not affected by the existence of previously saved nonvolatile data. The old data is simply overwritten and need not be cleared in advance.

NOTE	As part of the self-calibration process, the EX1048 is reset, returning all configuration parameters to their default values.

Network Configuration

This entry page is used to change the network configuration of the EX1048. By default, the EX1048 will attempt to locate a DHCP server. If one is found, the IP address assigned by the DHCP server will be assumed. Otherwise, after a timeout of 20 seconds, the unit will attempt to obtain an IP address by using AutoIP.

AutoIP is a mechanism for finding an unused IP address in the range 169.254.X.Y where X is in the range 1 - 254 and Y is in the range 0 - 255. The device will first attempt to obtain the specific address 169.254.X.Y, where X and Y are the second-to-last and last octets of the device's MAC address. However, X will be set to 1 if it is 0 in the MAC address, and to 254 if it is 255 in the MAC address. If this address is already in use, the unit will attempt to obtain other IP addresses in a pseudorandom fashion until it finds one that is available.

To illustrate the AutoIP mechanism, Table 5-2 lists the AutoIP default address for some example MAC addresses.

MAC Address	AutoIP Default Address
00:0D:3F:01:00:01	169.254.1.1
00:0D:3F:01:01:01	169.254.1.1
00:0D:3F:01:A3:28	169.254.163.40
00:0D:3F:01:FE:FE	169.254.254.254
00:0D:3F:01:FF:FE	169.254.254.254

TABLE 5-2: AUTOIP DEFAULT ADDRESS ASSIGNMENT

If a static IP address assignment is preferred, one can be optionally assigned via this page. Click on the *Disabled* radio button and then assign a static IP address. Following any changes, a **Hard Reboot** must be conducted to activate them. The network configuration settings are stored in nonvolatile memory and are unaffected by the **Reset Device** or **Hard Reboot** commands.

However, a much more convenient and recommended way to obtain the benefits of a static IP address is to employ DHCP, but assign the instrument a reserved IP address in your company's DHCP server configuration. This reserved address, linked to the EX1048's MAC address on the DHCP server, would be assigned to the EX1048 at power up initialization without having to manually set it on the EX1048. The DHCP server configuration provides a centralized, controlled database of assigned IP addresses, preventing accidental assignment of the same IP address to multiple instruments. Consult your company's Information Technology department for assistance.

VXI-11 Device Discovery is also supported by the EX1048. This allows all EX1048s on a local network to be found without knowledge of their MAC address or IP address with the use of a broadcast message.

Reset button

The reset button on the rear panel of the EX1048 can be used to restore default network settings. This is useful for recovery from an incorrect or unknown network configuration. To perform a network reset:

1) Power off the EX1048.
2) Press and hold the reset button.
3) Power on the EX1048.
4) Continue to hold the reset button for at least 30 seconds.
5) Release the reset button.

The EX1048 will power up as usual, but will use the default network configuration (DHCP) instead of its previous settings.

Time Configuration

This entry page is used to change the time configuration of the EX1048. By default, the instrument will receive its time through SNTP, the Simple Network Time Protocol. The **Zone** control provides a pull-down selection in which the user's specific time zone is selected. An instrument reboot is then required to activate the new selection. Optionally, the time can be set manually. This will be necessary if the network environment is such that the instrument cannot reach the Internet. Manual time is entered by selecting the appropriate **Source** radio button and entering in the time and date in the format specified by the page. Manual time entry is not affected by the **Zone** control and does not require an instrument reboot to be activated. However, manually-specified time is volatile, and therefore must be reentered upon an instrument reboot or power cycle. It is not, however, affected by the **Reset Device** command.

Upgrade Firmware

This action page is used to upgrade the embedded firmware of the EX1048. Prior to initiating the firmware upgrade process, a new, uncompressed firmware image must be obtained from VXI Technology and be accessible from the computer that is connected to the EX1048. Unless specifically noted by VXI Technology, firmware upgrades do not alter the calibration of the EX1048.

NOTE	Do not power cycle the EX1048 during the firmware upgrade process. If power is lost during the upgrade, the instrument may be put into an inoperable state, requiring return to the factory. An uninterruptible power supply may be used to avoid this risk.

Perform the following steps to conduct a firmware upgrade:

1) Perform a **Hard Reboot** or a power cycle.
2) Connect to the EX1048 via the embedded web page.
3) Click on the **Upgrade Firmware** link.

4) Click on the *Proceed* button.
5) Click on the *Browse* button and select the firmware image file to be uploaded to the instrument.
6) Click the *Upload* button to initiate the upgrade process.

NOTE	Once the upgrade process is initiated, do not attempt to reconnect to the instrument. Wait until the process is complete, as described below, before reconnecting.

During the upgrade process, 6 channel LEDs on the front panel will light sequentially from CH0 to CH5, indicating the progress of the upgrade sequence. Then all 48 channel LEDs will flash once, indicating an instrument reboot and the conclusion of the upgrade process. The process takes approximately 5 minutes to complete.

Once the instrument has rebooted, reconnect and confirm on the **EX1048 Main** page that the firmware revision level has been properly updated.

Reset Device

This action page is used to return all of the EX1048's acquisition configuration parameters to their default values. It is most commonly used to return the instrument to a known configuration state prior to the initiation of a new test sequence. Specific affected configuration parameters and their reset values are documented in Table 6-1. With regards to self-calibration data, an instrument reset will clear the current self cal data and then load, if it exists, the nonvolatile self cal data as the current self cal data. For more information on the generation and use of this data, refer to the **Self Calibration** page description in this section.

NOTE	An instrument reset clears the FIFO reading memory. All desired acquisition data must be retrieved from the FIFO prior to the issuance of this command.

Hard Reboot

This action page is used to perform a complete instrument reboot, equivalent to that which occurs when the instrument is power cycled. It is most commonly used to accept changes that are made to the network configuration or time configuration settings. In addition, it is suggested that a reboot be performed before conducting a firmware upgrade. Finally, it can be used to perform instrument recovery if the unit is suspected to be in a failed firmware state.

LOCKING MENU

The Locking menu is used to acquire, release, and break a lock on the EX1048's acquisition system.

Lock

This action page attempts to acquire a lock. It is good practice to lock the instrument whenever protected operation is desired, but it is mandatory in order to perform self-calibration. If the EX1048 has been previously locked by another host, the lock will fail, and an error message will be returned.

Unlock

This action page attempts to release a previously acquired lock. It will only release a lock that was acquired by the same host. It will not release a lock acquired by a different host. In that case, the unlock will fail, and an error message will be returned. An error message is also returned if the unlock command is issued when the instrument is not in the locked state.

Break Lock

This action page breaks the lock on the EX1048, regardless of the host that originally acquired it. It does not automatically acquire a lock in the same operation. If desired, that must be done with a separate **Lock** command.

ADVANCED MENU

The Advanced menu is used to configure the EX1048 to employ user-defined thermocouple polynomial equations or utilize user-defined CJC temperatures. Neither of these functions is required for normal operation with standard thermocouples.

User Conversions

This entry page is used to enter user-defined thermocouple polynomial coefficients. Up to two unique sets of coefficients can be entered. For more details, see *User-defined Conversions* in Section 3. The entry of user-defined coefficients does not automatically enable their use. The enabling is done by setting the EU conversion to User0 or User1 in the **Scan List…Configuration** page.

User CJC Temp

This entry page is used to enter and enable user-defined CJC temperatures. Each channel can be associated with a unique value and be independently enabled with regards to its use. If enabled, the user-defined CJC temperature will be used in the thermocouple calculations instead of the internally measured one. The entry of external CJC values and their enabling are disjoint functions. That is, the entry of a value does not automatically enable its use, and the disabling of a previously enabled channel does not clear the value. All entries must be in units of ºC.

NOTE	This feature must be used with care, as the input channel data contains no indication that it was calculated with an external CJC value instead of an internal one.

CONFIGURATION EXAMPLES

Example #1

This example demonstrates a typical setup sequence for a static monitoring application with an alarm. The requirements of the application are:

- Input connections are five type J thermocouples connected to channels 0-4 and five type K thermocouples connected to channels 5-9.
- The temperatures to be monitored are slow-moving, such that the 4 Hz filter frequency is suitable.
- Measurement data is required to be in units of ºF.
- CJC temperature reporting is desired.
- Data is to be acquired at a rate of 20 readings/second whenever DIO Channel 0 is high.
- Once initiated, the test should continue until aborted.
- DIO Channel 1 is to be used as an alarm indication, in which it needs to transition high and stay high if any of the 10 input channels exceeds 100 ºF.

The exact steps to be performed to satisfy this application are:

Configure the Scan List

The scan list is configured with the **Scan List…Configuration** page. Click the *Disabled* link to clear all previously enabled channels and then check the checkboxes of channels 0-9. With the appropriate radio button, set the EU conversion of channels 0-4 to J and channels 5-9 to K. Click the *Apply* button to accept the changes.

Configure the Filter Frequencies

The filters are configured with the **Scan List…Filters** page. Click the 4 Hz radio button for channels 0-9. Alternatively, since all enabled channels require the same filter setting, the *4 Hz* link can be used to simultaneously set all channels. Click the *Apply* button to accept the changes.

Configure the FIFO

The FIFO is configured with the **Data…FIFO Configuration** page. Select the **Temperature Units** selection of F. Check the **Report CJC Temperatures** checkbox. Click the *Apply* button to accept the changes.

Configure the Trigger Model

The trigger model is configured with the **Trigger…Configuration** page. Click the *Default* button. Expand the Arm branch to expose the Arm source. Deselect any currently checked sources. Expand the Digital IO branch and select Pos. Level of DIO Channel 0. Within the Trig source, select Timer. Enter a **Timer Interval** of 0.05. Check the **Init Continuous** checkbox. Click the *Apply* button to accept the changes.

Configure the Limit System

Limit conditions are configured with the **Limits…Limits** page. Using the **Set** pull-down selection, select Set 0. For channels 0-9, check the **Manual** checkbox and enter an upper limit value of 100. Click the *Apply* button to accept the changes. These limit conditions are linked to the DIO channel through the **Limits…DIO Limit Events** page. Using the **DIO Channel** pull-down selection, select channel 1. Check the Set 0 Upper checkboxes for Input Channels 0-9. Check the **Latch** checkbox. Click the *Apply* button to accept the changes.

Initiate the Trigger Model

The acquisition sequence is initiated with the **Trigger…Initialize** page.

Example #2

This example demonstrates a typical setup sequence for a transient monitoring application with an alarm. The requirements of the application are:

- Input connections are 48 type E thermocouples connected to channels 0-47.
- The temperatures to be monitored are transient in nature, such that the 1 kHz filter frequency is suitable.
- Measurement data is required to be in units of ºC.
- Timestamp reporting is desired, but CJC temperature reporting is not.
- Data is to be acquired at a rate of 1000 readings/second for 10 seconds, triggered by a positive edge transition on channel 0 of the trigger bus.
- Once the 10 seconds of data is collected, the test is complete.
- DIO Channel 7 is to be used as an alarm indication, in which it needs to transition low if any of the 48 channels exceeds 50 ºC.

The exact steps to be performed to satisfy this application are:

Configure the Scan List

The scan list is configured with the **Scan List…Configuration** page. Click the *Enabled* link to enable all 48 channels. Since all channels require the same EU conversion, click the *E* link. Click the *Apply* button to accept the changes.

Configure the Filter Frequencies

The filters are configured with the **Scan List…Filters** page. Since all channels require the same filter setting, click the *1 kHz* link. Click the *Apply* button to accept the changes.

Configure the FIFO

The FIFO is configured with the **Data…FIFO Configuration** page. Select the **Temperature Units** selection of C. Uncheck the **Report CJC Temperatures** checkbox. Check the **Report Timestamps** checkbox. Click the *Apply* button to accept the changes.

Configure the Trigger Model

The trigger model is configured with the **Trigger…Configuration** page. Click the *Default* button. Expand the Arm branch to expose the Arm source. Deselect any currently checked sources. Expand the VTI Trig. Bus branch and select Pos. Edge of VTB Channel 0. Within the Trig source, select Timer. Set a **TRIG Count** of 10000. Enter a **Timer Interval** of 0.001. Uncheck the **Init Continuous** checkbox. Click the *Apply* button to accept the changes.

Configure the Limit System

Limit conditions are configured with the **Limits…Limits** page. Using the **Set** pull-down selection, select Set 0. For channels 0-47, check the **Manual** checkbox and enter an upper limit value of 50. Click the *Apply* button to accept the changes. These limit conditions are linked to the DIO channel through the **Limits…DIO Limit Events** page. Using the **DIO Channel** pull-down selection, select channel 7. Since all channels are required in the limit evaluation, click the Set 0 Upper *All* link. Check the **Invert** checkbox. Uncheck the **Latch** checkbox. Click the *Apply* button to accept the changes.

Initiate the Trigger Model

The acquisition sequence is initiated with the **Trigger…Initialize** page.

SECTION 6

PROGRAMMING

INTRODUCTION

The EX1048 instrument driver is a VXI*plug&play* driver that is composed of top-level C functions.

DEFAULT SETTINGS

The default instrument settings after an instrument reset or a power cycle are listed in Table 6-1. Many programming applications do not require parameter changes from the default settings and can be made far simpler by the elimination of redundant commands. The EX1048 can be returned to the reset state at any time through the issuance of the **vtex1048_reset** command.

TRIGGER CONFIGURATION RESET VALUES	
CONFIGURATION PARAMETER	RESET VALUE
Init Continuous	Disabled
Timer Interval (seconds)	0.1
Arm Source	Immediate
Arm Count	1
Arm Delay (seconds)	0
Trig Source	Timer
Trig Count	1
Trig Delay (seconds)	0
SCAN LIST CONFIGURATION RESET VALUES	
CONFIGURATION PARAMETER	RESET VALUE
Scan List (enabled channels)	0
EU Conversions	mV (voltage)
Filter (hertz)	4.0
FIFO CONFIGURATION RESET VALUES	
CONFIGURATION PARAMETER	RESET VALUE
CJC Temperature Reporting	Disabled
Timestamp Reporting	Disabled
Temperature Units	°C
Blocking Mode	Disabled
DIGITAL I/O CONFIGURATION RESET VALUES	
CONFIGURATION PARAMETER	RESET VALUE
Output State	0
Output Enable	Disabled
TRIGGER BUS CONFIGURATION RESET VALUES	
CONFIGURATION PARAMETER	RESET VALUE
Output State	0
Output Enable	Disabled

TABLE 6-1: DEFAULT SETTINGS

LIMITS CONFIGURATION RESET VALUES	
CONFIGURATION PARAMETER	RESET VALUE
Lower Limit of Limit Set 0	-0.066
Upper Limit of Limit Set 0	0.066
Limit Set 0, Manual	Disabled
Lower Limit of Limit Set 1	-1e+38
Upper Limit of Limit Set 1	1e+38
DIO Limit Events	Disabled
DIO Limit Events, Invert	Disabled
DIO Limit Events, Latch	Disabled
DEVICE CONFIGURATION RESET VALUES	
CONFIGURATION PARAMETER	RESET VALUE
Current Self Cal Data	Cleared
Network Configuration	Unaffected
Time Configuration	Unaffected
Lock	Unaffected
ADVANCED CONFIGURATION RESET VALUES	
CONFIGURATION PARAMETER	RESET VALUE
User-defined Conversion Coefficients	0
User-defined CJC Temperatures (values)	0
User-defined CJC Temperatures (enable)	Disabled

TABLE 6-1 (CONTINUED): DEFAULT SETTINGS

GENERAL COMMANDS / QUERIES

These commands are used for general instrument setup and communication. Details on full command syntax are provided in Section 7.

Initializing the device

The **vtex1048_init** command is used to establish communication with the EX1048 and must be performed before any other command. The key parameter of this command is the IP address of the EX1048 to which to connect. Also specified within this command are options to confirm the identity of the connected instrument and to reset the instrument upon connecting. The output of this command is the session handle assigned to the connection, which is a unique identifier necessary in all subsequent communications. Consequently, sessions to multiple instruments can be opened within the same application, each uniquely identified by their session handle.

The **vtex1048_revisionQuery** query is used to obtain the release revisions of the instrument driver and the embedded firmware of the connected EX1048.

The **vtex1048_close** command is used to terminate an established communication link with the EX1048 and should be performed at the conclusion of the test application. Part of its execution is the unlocking of the instrument, leaving it in the proper state for the next application.

Resetting the device

The **vtex1048_reset** command is used to return all of the EX1048's acquisition configuration parameters to their default values. It is most commonly used to return the instrument to a known configuration state prior to the initiation of a new test sequence. Specific affected configuration parameters and their reset values are documented in Table 6-1. With regards to self-calibration data, an instrument reset will clear the current self cal data and then load, if it exists, the nonvolatile self cal data as the current self cal data. For more information on the generation and use of this data, refer to the **Performing Self-Calibration** description in this section.

NOTE	An instrument reset clears the FIFO reading memory. All desired acquisition data must be retrieved from the FIFO prior to the issuance of this command.

Performing Self-Calibration

As discussed in *Self-calibration* in Section 3, the measurement performance of the EX1048 is significantly improved through the periodic use of self-calibration. The self-calibration process completes quickly and does not require removal of the actual input connections or the use of external equipment, making it convenient to run often. In order to take full advantage of the self-calibration system, it is important to understand how the self-calibration constants are generated, used, and stored.

Self-calibration does not overwrite, modify, or take the place of the instrument's nonvolatile calibration constants generated by a full calibration. Instead, it generates an additional set of calibration constants that are applied to the measurement calculation after the full calibration constants. When self-calibration is performed, it places its calculated data in volatile memory. This data is termed the "current self cal data" and is used automatically by the instrument. No additional command is necessary. Because it is in volatile memory, however, it will be cleared upon an instrument reset or power cycle. There is the ability, however, to store the current self cal data into nonvolatile memory.

In order to perform a self-calibration, a lock on the instrument must first be acquired. For more information on locking, refer to the **Acquiring a Lock** description in this section. If self-calibration is attempted without the acquisition of a lock, an error will be generated. The self-calibration process is initiated with the **vtex1048_self_cal_init** command. Once started, its percentage completion status is accessible through the **vtex1048_self_cal_get_status** query. Other than this status check, additional instrument driver calls should not be performed during self-calibration. Once the completion status reaches 100%, the self-calibration process is complete.

To protect the user from performing a self-calibration while the instrument is warming up, an error will be generated and calibration will not be performed if self-calibration is attempted before the EX1048 has been powered on continuously for 60 minutes. This is only a warning, however, and it can be overridden. In general, this option should not be used, as performing self-calibration prior to complete instrument warm-up could result in degraded measurement performance. However, it would be appropriate to utilize the override in the case where the unit has actually been warmed up, but has simply been subjected to a quick power cycle or reboot. Details on the override mechanism are provided in the **vtex1048_self_cal_init** command description in Section 7.

NOTE	The self-calibration uptime requirement is in place to protect the measurement integrity of the instrument and should only be overridden with caution. In order to insure that the override is intentional, it is strongly recommended that user intervention be required in the software application to employ it.

As noted above, the self-calibration operation places its calculated data in volatile memory, termed "current self cal data." Once present, it can be cleared by utilizing the **vtex1048_self_cal_clear** command. This operation clears the volatile data, but does not affect any self cal data that is stored in nonvolatile memory.

Despite having the ability to conduct self-calibration at any time, there may be user applications that require the use of self-calibration, but demand that it create nonvolatile data. The **vtex1048_self_cal_store** command saves the current self cal data to nonvolatile memory, enabling it to be loaded upon instrument power cycle and reset. If this command is sent when no current self cal data is present, an error is generated. Since the existence of nonvolatile self cal data represents a permanent (although revocable) change from the factory calibration settings, its presence is able to be ascertained with the **vtex1048_self_cal_is_stored** query.

Previously saved nonvolatile self cal data can be cleared through the use of the **vtex1048_self_cal_clear_stored** command. This operation does not clear the current self cal data, only that in nonvolatile memory. If this command is sent when no nonvolatile self cal data is present, an error is generated. Alternatively, previously saved nonvolatile self cal data can be loaded as the current self cal data through the use of the **vtex1048_self_cal_load** command. If current self cal data previously existed, it is simply overwritten and need not be cleared in advance. If this command is sent when no nonvolatile self cal data is present, an error is generated.

The ability to conduct a self-calibration is not affected by the existence of current self cal data or nonvolatile self cal data. A performed self-calibration will simply overwrite any current self cal data that already existed. It is not necessary to utilize the **vtex1048_self_cal_clear** command prior to performing a new self-calibration. Similarly, the ability to store current self cal data as nonvolatile is not affected by the existence of previously saved nonvolatile data. The old data is simply overwritten and need not be cleared in advance.

NOTE	As part of the self-calibration process, the EX1048 is reset, returning all configuration parameters to their default values.

Acquiring a Lock

By default, the EX1048 allows unrestricted operation from multiple hosts. While this offers a high level of user flexibility, there are instances where protected operation is desirable, if not required. For these applications, the EX1048 can be "locked," meaning that it will accept commands from only the host IP address that issued the **vtex1048_lock** command. With the EX1048 in this mode, other host connections that attempt commands will be denied. Self-calibration, in fact, requires the acquisition of a lock prior to its initiation.

Once a lock has been acquired, it is released with the **vtex1048_unlock** command. The lock status is not affected by the **vtex1048_reset** command, and it cannot be used to release a lock.

By design, the locking mechanism is able to be overridden by a secondary host that issues a **vtex1048_break_lock** command. Thus, the lock provides a warning to other users that the unit is in a protected operation state, but not absolute security. This allows for instrument recovery if the locking IP address would become disabled. Breaking a lock, however, does not automatically acquire it. That must be done with a separate **vtex1048_lock** command.

The lock status of the instrument is queried with the **vtex1048_check_lock** query. The response to the query not only indicates whether the unit is locked, but also if the lock is owned by the issuer of the query.

PROGRAMMING SEQUENCE

The following flow chart describes the typical steps performed in programming an acquisition sequence on the EX1048. While the flow chart implies an order dependence to the commands, none truly exists. Each command operates in a disjoint manner. The flow chart, however, does present a logical flow of operations. For many applications, some of the detailed steps would not be required, either because the default parameters are sufficient or the associated functionality is simply not required.

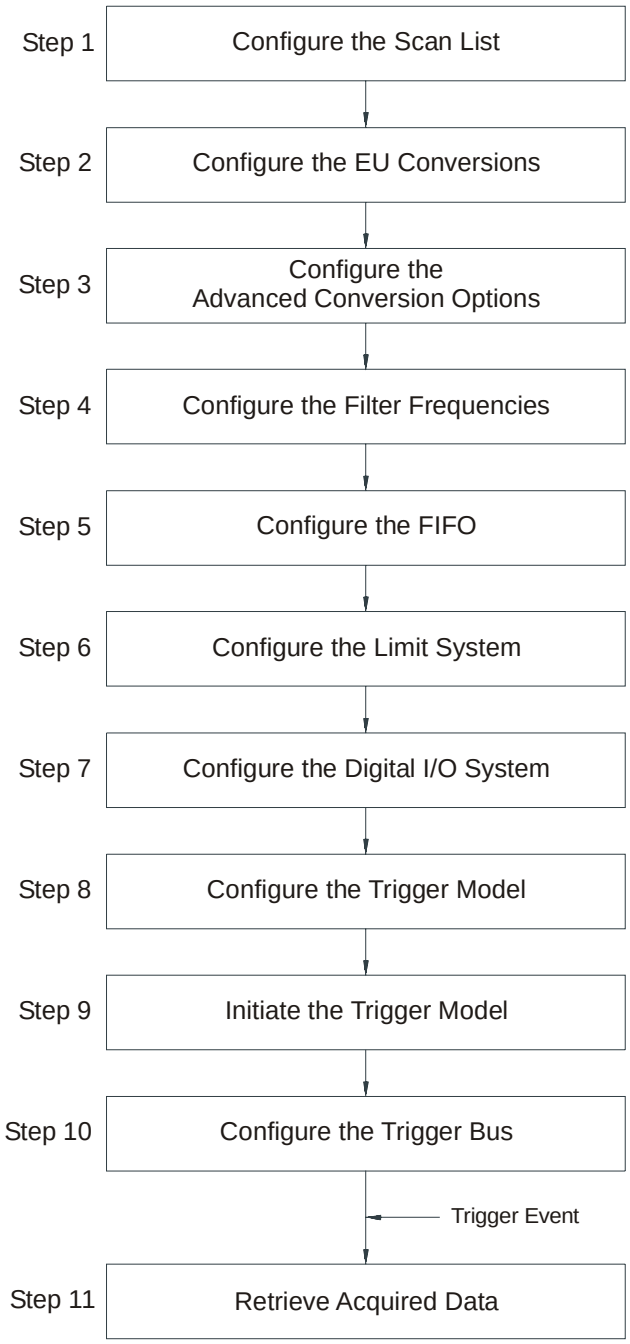

FIGURE 6-1: PROGRAMMING SEQUENCE DIAGRAM

MEASUREMENT COMMANDS / QUERIES

These commands are used for configuration of the acquisition system and the retrieval of acquisition data. Details on full command syntax are provided in Section 7.

Configure the Scan List

The scan list is defined with the **vtex1048_set_scanlist** command. The EX1048 can be configured to include from 1 to all 48 of its input channels in the scan list. A valid scan list consists of:

- at least one channel
- no more than 48 channels
- no repeated channels

The channels in the scan list can be listed in any order and will be scanned in exactly the order in which they are listed.

Example #1: This code block configures a five-channel scan of channels 0-4 in sequential order.

```
ViInt32 channels[5] = { 0, 1, 2, 3, 4 };
vtex1048_set_scanlist(vi, channels, 5);
```

Example #2: This code block configures a five-channel scan of channels 0-4 in reverse order.

```
ViInt32 channels[5] = { 4, 3, 2, 1, 0 };
vtex1048_set_scanlist(vi, channels, 5);
```

The current scan list is queried with the **vtex1048_get_scanlist** query.

Configure the EU Conversions

The EU conversion for each channel is configured with the **vtex1048_set_channel_conversion** command. A channel's EU conversion can be configured regardless of its inclusion in the scan list, and multiple channels can be assigned to a specific conversion type within one command. However, each unique conversion type must be set with a separate command.

Example: This code block configures channels 0-4 for thermocouple type E and channels 5-8 for thermocouple type T.

```
#define TYPE_E    0x04
#define TYPE_T    0x03
ViInt32 e_channels[5] = { 0, 1, 2, 3, 4 };
vtex1048_set_channel_conversion(vi, e_channels, 5, TYPE_E);
ViInt32 t_channels[4] = { 5, 6, 7, 8 };
vtex1048_set_channel_conversion(vi, t_channels, 4, TYPE_T);
```

The current EU conversion assignment for any channel is queried with the **vtex1048_get_channel_conversion** query.

Configure the Advanced Conversion Options

To facilitate its use, the EX1048 provides a high level of hardware and software integration. A high performance internal cold junction mechanism allows for direct connection of thermocouple wire, and embedded polynomial coefficients for all standard thermocouple types provides compensated temperature readings without external mathematical manipulation. It does, however, also support the advanced configuration options of employing an external cold junction and employing custom thermocouple conversions.

Employing an external cold junction

The EX1048 accommodates the use of external cold junctions that are maintained and measured by the user. In this application, the cold junction temperature in ºC is entered into the EX1048 and enabled on a per channel basis. That is, the use of internal and user-defined CJC inputs can be mixed throughout the unit. User-defined CJC temperatures are entered with the **vtex1048_set_user_cjc_temp** command. A channel's CJC temperature can be configured regardless of its inclusion in the scan list, and multiple channels can be assigned to the same temperature within one command. However, each unique temperature must be set with a separate command.

The use of a user-defined CJC temperature is enabled on a per channel basis with the **vtex1048_set_user_cjc_enable** command. If enabled, the user-defined CJC temperature will be used in the thermocouple calculations instead of the internally measured one. The entry of external CJC values and their enabling are disjoint functions. That is, the entry of a value does not automatically enable its use, and the disabling of a previously enabled channel does not clear the value.

NOTE	These commands should only be used when the thermocouple cold junction is made external to the EX1048. This feature must be used with care, as the input channel data contains no indication that it was calculated with an external CJC value instead of an internal one.

The current user-defined CJC temperature and enable status for any channel are queried with the **vtex1048_get_user_cjc_temp** and **vtex1048_get_user_cjc_enable** queries, respectively.

Employing custom thermocouple conversions

The EX1048 nominally accepts standard thermocouple types and performs its thermocouple calculations using polynomial coefficients from the NIST ITS-90 Thermocouple Database. In some applications, however, a user may want to override the embedded coefficients with a user-defined coefficient set. One reason to do this is if the transfer function of the specific thermocouple being used has been completely characterized to an accuracy level that exceeds standard thermocouple limits of error. Another reason to do this is if a non-standard thermocouple is used. Up to two unique sets of coefficients can be entered. Specifically, the use of custom thermocouple equations requires the user to know or generate the coefficients for two conversion polynomials.

The *forward conversion polynomial* is used to convert a CJC temperature into a compensating cold junction voltage and has the form of:

$$E = c_0 + c_1 * t^1 + c_2 * t^2 + \ldots + c_{12} * t^{12}$$

where E is in volts, t is in ºC, and c_0 - c_{12} are the coefficients.

The *inverse conversion polynomial* is used to convert a compensated input voltage into temperature and has the form of:

$$t = d_0 + d_1 * E^1 + d_2 * E^2 + \ldots + d_{12} * E^{12}$$

where E is in volts, t is in ºC, and d_0 - d_{12} are the coefficients.

The coefficient sets are entered with the **vtex1048_set_user_conversion** command. This command accepts the following parameters:

- an integer value of 9 or 10, corresponding to the EU conversion type (User0 or User1, respectively) to which the entered coefficients are assigned
- an array of forward conversion polynomial coefficients
- an integer value corresponding to the length of the forward coefficients array
- an array of inverse conversion polynomial coefficients
- an integer value corresponding to the length of the inverse coefficients array

Coefficients that are not specifically defined are automatically set to 0. The current values for each coefficient set are queried with the **vtex1048_get_user_conversion** query.

NOTE	The entry of user-defined coefficients does not automatically enable their use. The enabling is done by setting the EU conversion to User0 or User1.

Configure the Filter Frequencies

The hardware filter for each channel is configured with the **vtex1048_set_filt_freq** command. A channel's hardware filter can be configured regardless of its inclusion in the scan list, and multiple channels can be assigned to a specific filter frequency within one command. However, each unique filter frequency must be set with a separate command.

Example: This code block configures channels 0-4 for a 4 Hz filter and channels 5-8 for a 1 kHz filter.

```
ViInt32 low_channels[5] = { 0, 1, 2, 3, 4 };
vtex1048_set_filt_freq(vi, low_channels, 5, 4.0);
ViInt32 high_channels[4] = { 5, 6, 7, 8 };
vtex1048_set_filt_freq(vi, high_channels, 4, 1000.0);
```

The current hardware filter assignment for any channel is queried with the **vtex1048_get_filt_freq** query.

Configure the FIFO

The **vtex1048_set_fifo_config** command is used to set the data format and overflow behavior of the FIFO memory. By default, the data returned during data retrieval is limited to the input channel readings and the absolute time of scan initiation. While measured with every scan, the CJC temperature values are not returned unless enabled to do so. Reported CJC temperatures are always in units of ºC, regardless of the units of the channel readings. In addition, the returned data can be enabled to include delta timestamps per channel. Specifically, the timestamp represents the time, in increments of 100 ns, between the specific channel measurement and the scan initiation time.

By default, the channel readings are returned with units of ºC. Optionally, the units can be set to ºF. This setting has no effect on the reported CJC data or data for input channels that are configured for an EU conversion of mV.

In addition to data formatting, this command is used to specify the system behavior if the reading buffer fills to its maximum capacity during scanning. With blocking mode enabled, additional readings are discarded, leaving the contents of the buffer intact. With blocking mode disabled, the buffer becomes circular, in that additional readings overwrite the oldest readings.

The current FIFO configuration is queried with the **vtex1048_get_fifo_config** query.

NOTE	Regardless of their enabling, the CJC temperature and delta timestamp information is not accessible through the **vtex1048_read_fifo** command. They are available through the streaming interface, and the parameter settings apply for it.

Configure the Limit System

The EX1048 features two sets of programmable limits. These limits, termed limit set 0 and limit set 1, are programmable on a per channel basis. Manipulation of the limit set values is slightly different for each limit set, as limit set 0 offers enhanced functionality.

By default, the values in limit set 0 are set automatically, based on the EU conversion and units selection for each channel. Specifically, the upper and lower limit values are set to the upper and lower values of the EX1048 measurement range, as specified in Table 3-2. For example, if input channel 0 is configured for an E type conversion and a units selection of ºC, its upper and lower limit values will be +900 and -200, respectively. To override the automatic setting and set the values manually on a per channel basis, manual control must first be enabled with the **vtex1048_set_limit_set0_manual** command. A channel's manual entry control can be set regardless of its inclusion in the scan list, and multiple channels can be configured within one command. However, each unique control value must be set with a separate command.

Once enabled for manual entry, a channel's limit set 0 values are set with the **vtex1048_set_limit_set0** command. A channel's limit values can be set regardless of its inclusion in the scan list, and multiple channels can be assigned to the same limit values within one command. However, each unique combination of limit values must be set with a separate command.

Example: This code block configures channels 0-4 for manual limit values of 0 and 100.

```
ViInt32 e_channels[5] = { 0, 1, 2, 3, 4 };
vtex1048_set_limit_set0_manual(vi, e_channels, 5, 1);
vtex1048_set_limit_set0(vi, e_channels, 5, 0, 100);
```

Conversely, disabling manual limit control on a channel that had been enabled will automatically set the limit set 0 values for that channel, based on its current EU conversion and units selection.

The current manual entry control and limit set 0 values for any channel are queried with the **vtex1048_get_limit_set0_manual** and **vtex1048_get_limit_set0** queries, respectively.

In contrast, limit set 1 operates in manual mode only. A channel's limit set 1 values are set with the **vtex1048_set_limit_set1** command. A channel's limit values can be set regardless of its inclusion in the scan list, and multiple channels can be assigned to the same limit values within one command. However, each unique combination of limit values must be set with a separate command.

The current limit set 1 values for any channel are queried with the **vtex1048_get_limit_set1** query.

NOTE	Limit sets operating in manual mode do not automatically adjust for changes in EU conversion or units selection. For proper operation, limit values should be set after EU conversion and units selections are made and reset upon subsequent configuration changes.

Once an acquisition has been completed, the **vtex1048_get_accum_limit_status** query is used to obtain the accumulated limit status of all 48 channels. The response to this query is four arrays of 48 Boolean values each. The four arrays correspond to the set 0 lower limit, set 0 upper limit, set 1 lower limit, and set 1 upper limit, respectively. The returned values represent, on a per channel basis, any excursion of the measurement data over their respective limits since the last trigger initialize. Specifically, a returned "1" indicates that the limit was exceeded, while a returned "0"

indicates that it was not. As implied, limit status is cleared as part of the **vtex1048_init_imm** command. Limit status is returned for all channels, regardless of their inclusion in the scan list. The limit status for unscanned channels is always 0.

Configure the Digital I/O System

The digital I/O port on the EX1048 can be used for input and output operations. Each DIO channel can be independently programmed with regards to its output functionality, its static level to assume when enabled as an output, and pulse operation.

The **vtex1048_get_dio_input** query is used to obtain the measured digital level (0 or 1) of the DIO channels. The input monitoring functionality operates regardless of the simultaneous use of the channels as outputs. The response to this query is a decimal value from 0 to 255 that represents the 8-bit value of the port. Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0.

Example: This code block queries the state of the I/O port and reports the level of bits 7, 4, and 0.

```
ViInt32 dio_in;
vtex1048_get_dio_input(vi, &dio_in);
if (dio_in & 0x80)
      printf("Bit 7 is high");
else printf("Bit 7 is low");
if (dio_in & 0x10)
      printf("Bit 4 is high");
else printf("Bit 4 is low");
if (dio_in & 0x01)
      printf("Bit 0 is high");
else printf("Bit 0 is low");
```

The **vtex1048_set_dio_output** command is used to set the static level that each channel of the digital I/O port will assume if/when enabled as an output. Enabling, however, is done with the **vtex1048_set_dio_output_enable** command (see below). This command accepts a value that represents the desired state of the 8-bit port, specified in either decimal (0 - 255) or hex (0x00 - 0xFF). Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0.

The **vtex1048_set_dio_output_enable** command enables or disables the output functionality of each channel of the digital I/O port. Once enabled, a channel will assume the level specified with the **vtex1048_set_dio_output** command (see above). When not enabled as an output, a channel becomes tri-stated. Input functionality on each channel is constant regardless of its output functionality. This command accepts a value that represents the desired output enable state of the 8-bit port, specified in either decimal (0 - 255) or hex (0x00 - 0xFF). Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0.

Example: This code block sets bit 7 (high) and bit 6 (low) of the I/O port and then enables them as outputs.

```
vtex1048_set_dio_output(vi, 0x80);
vtex1048_set_dio_output_enable(vi, 0xC0);
```

The **vtex1048_get_dio_output** query is used to obtain the programmed output state of the DIO channels. The response to this query is a decimal value from 0 to 255 that represents the programmed output state of the 8-bit port. However, since the outputs must be enabled, it does not necessarily represent the true output state. The **vtex1048_get_dio_output_enable** query is used to obtain the output enable state of each channel. The response to this query is a decimal value from 0 to 255 that represents the output enable state of the 8-bit port. Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0.

Example: This code block queries the output state and output enable state of the I/O port and reports the states of bit 4.

```
ViInt32 dio_out;
ViInt32 dio_outen;
vtex1048_get_dio_output(vi, &dio_out);
vtex1048_get_dio_output_enable(vi, &dio_outen);
if (dio_out & 0x10)
      printf("Bit 4 is set high");
else printf("Bit 4 is set low");
if (dio_outen & 0x10)
      printf("Bit 4 is enabled");
else printf("Bit 4 is not enabled");
```

The **vtex1048_set_dio_pulse** command is used to generate a 1 μs pulse on selected channels of the digital I/O port. The pulse will occur only if the selected channels are enabled as outputs. When a channel is programmed with a static level of high, the pulse will be low-going. When a channel is programmed with a static level of low, the pulse will be high-going. Each pulse generation requires a separate command. This command accepts a value that represents the channels to be pulsed within the 8-bit port, specified in either decimal (0 - 255) or hex (0x00 - 0xFF). Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0.

Example: This code block sets bit 7 to a static level of low and then generates a high-going pulse.

```
vtex1048_set_dio_output(vi, 0x00);
vtex1048_set_dio_output_enable(vi, 0x80);
vtex1048_set_dio_pulse(vi, 0x80);
```

Configure DIO Limit Events

In addition to performing simple input and output operations, the digital I/O port can also be linked to reflect the state of input channel limit evaluations. This linkage is termed a DIO limit event. DIO limit events are programmable on a per channel basis; that is, each of the 8 DIO channels can be configured with a unique set of operational characteristics. In nominal operation, a DIO channel that is linked to an input channel's limit evaluation will transition from low to high whenever the limit is exceeded.

The linkage of a DIO channel to a specific limit condition on a specific input channel is accomplished with the **vtex1048_set_dio_limit_event** command. Multiple linkages per DIO channel are allowed and are logically OR'ed together. That is, a DIO channel that is linked to four input channel limit evaluations will transition whenever any of the four limits are exceeded. Multiple linkages can be created on the same input channel and/or spanning multiple input channels.

This command accepts the following parameters:

- an integer value representing the DIO channel number (0 – 7)
- an array of 48 4-bit integer values representing, on a per input channel basis, the linking of limit evaluations to any of the 4 limit conditions. Within the 4-bit field, the order of the values is: limit set 0 lower, limit set 0 upper, limit set 1 lower, limit set 1 upper. The values can be specified in either decimal (0 – 15) or hex (0x00 – 0x0F). Channel 0 through channel 47 are represented in array elements [0] through [47], respectively.

Example: This code block links DIO channel 7 to set 1 lower limits on input channels 12 and 14 and set 0 upper limit on input channel 1

```
ViUInt8 limit_masks[48] =
{0,4,0,0,0,0,0,0,0,0,0,0,2,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
vtex1048_set_dio_limit_event(vi, 7, limit_masks);
```

The DIO limit event and the scan list configuration controls are autonomous. That is, specifying a linkage to a specific input channel does not automatically enable it in the scan list. Similarly, removing an input channel from the scan list does not automatically remove it as a linked limit condition. DIO limit event linkages to unscanned channels do not cause an error condition and are simply ignored in the evaluation.

When linked as a limit event, a DIO channel will be cleared at the beginning of a new acquisition. Its state will then be updated with each scan according to the programmed limit evaluations. By default, the cleared state is low, but can be set on a per channel basis to be high through the **vtex1048_set_dio_limit_event_invert** command. Similarly, the default operation for each channel is non-latch mode, but can be set on a per channel basis to be latch mode through the **vtex1048_set_dio_limit_event_latch** command. In latch mode, a transition out of the cleared state would remain, regardless of future limit evaluations, until it is cleared at the beginning of a new acquisition.

The current state of the DIO limit event mechanism is queried with the **vtex1048_get_dio_limit_event**, **vtex1048_get_dio_limit_event_invert**, and **vtex1048_get_dio_limit_event_latch** queries.

Configure the Trigger Model

The EX1048 supports a full function trigger model with a separate arm source and trigger source event structure. For a complete explanation of the trigger model, see Section 4. In summary, an acquisition sequence is enabled with a trigger initialize command. Scanning is then initiated upon the receipt of the programmed arm source event followed by the receipt of the programmed trigger source event. Trigger and arm source events can be independently programmed from a variety of sources including Immediate, Timer, Digital I/O, and the Trigger Bus.

Configure the ARM event system

The ARM source is configured with the **vtex1048_set_arm_source** command. The ARM event can be specified to be any combination of Trigger Bus channel transitions or levels, Digital I/O channel transitions or levels, and Timer ticks, or simply be set to Immediate. If multiple sources for an ARM event are specified, they are logically combined as follows:

ARM event = [(Timer tick event) AND (Digital I/O event) AND (Trigger Bus event)]

Within each digital hardware port, it is also possible to select multiple channels and multiple conditions for a specific channel. In that case, they are logically combined as follows:

Digital I/O event = (Ch 7 events) AND (Ch 6 events) … AND (Ch 0 events)

Finally, within each channel, the four event conditions of Pos. Edge, Neg. Edge, Pos. Level, and Neg. Level are OR'ed together.

As an example, if a Digital I/O event is specified to be a combination of a Pos. Level on channel 3, a Pos. Edge on channel 6, and a Neg. Edge on channel 6, the event will be satisfied when a Pos. Level on channel 3 occurs simultaneously with a Pos. Edge or a Neg. Edge transition on channel 6. While the Digital I/O event structure was used as an example, the Trigger Bus event structure operates in the same manner.

This command accepts the following parameters:

- an array of four 8-bit values representing the enabling of events from any of the 8 channels of the trigger bus. The order of the values is: positive edge, negative edge, positive level, negative level. Each value is specified in either decimal (0 - 255) or hex (0x00 - 0xFF). Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0.
- an array of four 8-bit values representing the enabling of events from any of the 8 channels of the digital I/O port. The order of the values is: positive edge, negative edge, positive level, negative level. Each value is specified in either decimal (0 - 255) or hex (0x00 - 0xFF). Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0.
- a Boolean value indicating the enable status of the Timer event
- a Boolean value indicating the enable status of the Immediate event.

Regardless of this setting, software arms are always enabled.

Example #1: This code block enables arm on the timer only.

```
ViUInt8 vtb_masks[4] = {0,0,0,0};
ViUInt8 dio_masks[4] = {0,0,0,0};
vtex1048_set_arm_source(vi, vtb_masks, dio_masks, 1, 0);
```

Example #2: This code block enables arm on a positive level on DIO channels 0-3 and a negative edge on VTB channel 6.

```
ViUInt8 vtb_masks[4] = {0,64,0,0};
ViUInt8 dio_masks[4] = {0,0,0x0F,0};
vtex1048_set_arm_source(vi, vtb_masks, dio_masks, 0, 0);
```

Example #3: This code block enables software arm only.

```
ViUInt8 vtb_masks[4] = {0,0,0,0};
ViUInt8 dio_masks[4] = {0,0,0,0};
vtex1048_set_arm_source(vi, vtb_masks, dio_masks, 0, 0);
```

The current ARM source is queried with the **vtex1048_get_arm_source** query.

The ARM count is configured with the **vtex1048_set_arm_count** command, specified with a value from 1 to (2^{31}-1). This value is reset with each trigger initialize or automatically upon reaching zero when init continuous is enabled.

Example: This code block sets an ARM count of 10.

```
vtex1048_set_arm_count(vi, 10);
```

The current ARM count is queried with the **vtex1048_get_arm_count** query.

Alternatively, the ARM count can be set to infinity, overriding any manual setting of ARM count, with the **vtex1048_set_arm_infinite** command. This command simply accepts a Boolean value indicating the enable status of an infinite ARM count.

The current setting of an infinite ARM count is queried with the **vtex1048_get_arm_infinite** query.

The ARM delay is configured with the **vtex1048_set_arm_delay** command, specified with a value in seconds from 0 to 4294 with a resolution of 0.000001 (1 µs). The ARM delay is the time between the recognition of the ARM event and the transition into the TRIG layer of the trigger model.

Example: This code block sets an ARM delay of 5 ms.

```
vtex1048_set_arm_delay(vi, 0.005);
```

The current ARM delay is queried with the **vtex1048_get_arm_delay** query.

Configure the TRIG event system

The TRIG source is configured with the **vtex1048_set_trigger_source** command. The TRIG event can be specified to be any combination of Trigger Bus channel transitions or levels, Digital I/O channel transitions or levels, and Timer ticks, or simply be set to Immediate. If multiple sources for a TRIG event are specified, they are logically combined as follows:

TRIG event = [(Timer tick event) AND (Digital I/O event) AND (Trigger Bus event)]

Within each digital hardware port, it is also possible to select multiple channels and multiple conditions for a specific channel. In that case, they are logically combined as follows:

Digital I/O event = (Ch 7 events) AND (Ch 6 events) … AND (Ch 0 events)

Finally, within each channel, the four event conditions of Pos. Edge, Neg. Edge, Pos. Level, and Neg. Level are OR'ed together.

As an example, if a Digital I/O event is specified to be a combination of a Pos. Level on channel 3, a Pos. Edge on channel 6, and a Neg. Edge on channel 6, the event will be satisfied when a Pos. Level on channel 3 occurs simultaneously with a Pos. Edge or a Neg. Edge transition on channel 6. While the Digital I/O event structure was used as an example, the Trigger Bus event structure operates in the same manner.

This command accepts the following parameters:

- an array of four 8-bit values representing the enabling of events from any of the 8 channels of the trigger bus. The order of the values is: positive edge, negative edge, positive level, negative level. Each value is specified in either decimal (0 - 255) or hex (0x00 - 0xFF). Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0.
- an array of four 8-bit values representing the enabling of events from any of the 8 channels of the digital I/O port. The order of the values is: positive edge, negative edge, positive level, negative level. Each value is specified in either decimal (0 - 255) or hex (0x00 - 0xFF). Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0.
- a Boolean value indicating the enable status of the Timer event
- a Boolean value indicating the enable status of the Immediate event.

Regardless of this setting, software triggers are always enabled.

Example #1: This code block enables trigger on the timer only.

```
ViUInt8 vtb_masks[4] = {0,0,0,0};
ViUInt8 dio_masks[4] = {0,0,0,0};
vtex1048_set_trigger_source(vi, vtb_masks, dio_masks, 1, 0);
```

Example #2: This code block enables trigger on a positive level on DIO channels 0-3 and a negative edge on VTB channel 6.

```
ViUInt8 vtb_masks[4] = {0,64,0,0};
ViUInt8 dio_masks[4] = {0,0,0x0F,0};
vtex1048_set_trigger_source(vi, vtb_masks, dio_masks, 0, 0);
```

Example #3: This code block enables software trigger only.

```
ViUInt8 vtb_masks[4] = {0,0,0,0};
ViUInt8 dio_masks[4] = {0,0,0,0};
vtex1048_set_trigger_source(vi, vtb_masks, dio_masks, 0, 0);
```

The current TRIG source is queried with the **vtex1048_get_trigger_source** query.

The TRIG count is configured with the **vtex1048_set_trigger_count** command, specified with a value from 1 to (2^{31}-1). This value is reset with each ARM event.

Example: This code block sets a TRIG count of 10.

```
vtex1048_set_trigger_count(vi, 10);
```

The current TRIG count is queried with the **vtex1048_get_trigger_count** query.

Alternatively, the TRIG count can be set to infinity, overriding any manual setting of TRIG count, with the **vtex1048_set_trigger_infinite** command. This command simply accepts a Boolean value indicating the enable status of an infinite TRIG count.

The current setting of an infinite TRIG count is queried with the **vtex1048_get_trigger_infinite** query.

The TRIG delay is configured with the **vtex1048_set_trigger_delay** command, specified with a value in seconds from 0 to 4294 with a resolution of 0.000001 (1 µs). The TRIG delay is the time between the recognition of the TRIG event and the execution of the scan list.

Example: This code block sets a TRIG delay of 5 ms.

```
vtex1048_set_trigger_delay(vi, 0.005);
```

The current TRIG delay is queried with the **vtex1048_get_trigger_delay** query.

Configure general trigger model parameters

The timer interval for Timer source events is configured with the **vtex1048_set_trigger_timer** command, specified with a value in seconds from 0.001 (1 ms) to 4294 with a resolution of 0.000001 (1 µs). The timer interval controls the frequency of the Timer event source. The same value is used for both arm and trigger events.

Example: This code block sets a timer interval of 50 ms.

```
vtex1048_set_trigger_timer(vi, 0.05);
```

The current timer interval is queried with the **vtex1048_get_trigger_timer** query.

Normally, each acquisition sequence is initiated by a trigger initialize command. However, this requirement can be circumvented through the use of init continuous mode. Specifically, init continuous returns the trigger model to the entrance of the ARM layer without the requirement of a new trigger initialize command. This allows for a specific acquisition sequence to be restarted automatically. The enabling of init continuous mode is done with the **vtex1048_set_init_cont** command. This command simply accepts a Boolean value indicating the enable status of init continuous mode.

The current setting of init continuous mode is queried with the **vtex1048_get_init_cont** query.

Resetting the trigger model configuration

The **vtex1048_reset_trigger_arm** command is used to return all of the EX1048's trigger configuration parameters to their default values. It is similar to the **vtex1048_reset** command, except that only trigger configuration parameters are affected. These parameters and their reset values are documented in Table 6-1.

NOTE	The extensive flexibility of the trigger model system permits the creation of very specialized trigger conditions, which is a powerful application tool. However, it also permits the creation of trigger conditions that would be very difficult to satisfy in practice. For example, if edge conditions are specified on multiple digital hardware channels, the edges must occur within 25 ns of each other to be recognized as having occurred simultaneously. Similarly, the timer source should not be combined with any digital hardware edge conditions.

Initiate the Trigger Model

A trigger initialize is performed with the **vtex1048_init_imm** command. Upon the receipt of this command, the trigger model transitions out of the IDLE layer and begins waiting for the ARM event. This command clears the FIFO reading memory and resets all limit indications.

Once the trigger model has transitioned out of the IDLE layer, most instrument configuration controls are disabled. An acquisition can, however, be aborted at any time with the **vtex1048_abort** command. An abort does not affect any stored acquisition data.

Configure the Trigger Bus

The primary use of the trigger bus is the transmission of high-speed signals for multiple-unit triggering and synchronization. Accordingly, it can be used for input and output operations. Each VTB channel can be independently programmed with regards to its output functionality, its static level to assume when enabled as an output, and pulse operation.

The **vtex1048_get_vtb_input** query is used to obtain the measured digital level (0 or 1) of the VTB channels. The input monitoring functionality operates regardless of the simultaneous use of the channels as outputs. The response to this query is a decimal value from 0 to 255 that represents the 8-bit value of the bus. Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0.

Example: This code block queries the state of the trigger bus and reports the level of bits 7, 4, and 0.

```
ViInt32 vtb_in;
vtex1048_get_vtb_input(vi, &vtb_in);
if (vtb_in & 0x80)
      printf("Bit 7 is high");
else printf("Bit 7 is low");
if (vtb_in & 0x10)
      printf("Bit 4 is high");
else printf("Bit 4 is low");
if (vtb_in & 0x01)
      printf("Bit 0 is high");
else printf("Bit 0 is low");
```

The **vtex1048_set_vtb_output** command is used to set the static level that each channel of the trigger bus will assume if/when enabled as an output. Enabling, however, is done with the **vtex1048_set_vtb_output_enable** command (see below). This command accepts a value that represents the desired state of the 8-bit bus, specified in either decimal (0 - 255) or hex (0x00 - 0xFF). Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0.

The **vtex1048_set_vtb_output_enable** command enables or disables the output functionality of each channel of the trigger bus. Once enabled, a channel will assume the level specified with the **vtex1048_set_vtb_output** command (see above). When not enabled as an output, a channel becomes tri-stated. Input functionality on each channel is constant regardless of its output functionality. This command accepts a value that represents the desired output enable state of the 8-bit bus, specified in either decimal (0 - 255) or hex (0x00 - 0xFF). Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0.

Example: This code block sets bit 7 (high) and bit 6 (low) of the trigger bus and then enables them as outputs.

```
vtex1048_set_vtb_output(vi, 0x80);
vtex1048_set_vtb_output_enable(vi, 0xC0);
```

The **vtex1048_get_vtb_output** query is used to obtain the programmed output state of the trigger bus. The response to this query is a decimal value from 0 to 255 that represents the programmed output state of the 8-bit bus. However, since the outputs must be enabled, it does not necessarily represent the true output state. The **vtex1048_get_vtb_output_enable** query is used to obtain the output enable state of each channel. The response to this query is a decimal value from 0 to 255 that represents the output enable state of the 8-bit bus. Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0.

Example: This code block queries the output state and output enable state of the trigger bus and reports the states of bit 4.

```
ViInt32 vtb_out;
ViInt32 vtb_outen;
vtex1048_get_vtb_output(vi, &vtb_out);
vtex1048_get_vtb_output_enable(vi, &vtb_outen);
if (vtb_out & 0x10)
      printf("Bit 4 is set high");
else printf("Bit 4 is set low");
if (vtb_outen & 0x10)
      printf("Bit 4 is enabled");
else printf("Bit 4 is not enabled");
```

The **vtex1048_set_vtb_pulse** command is used to generate a 1 μs pulse on selected channels of the trigger bus. The pulse will occur only if the selected channels are enabled as outputs. When a channel is programmed with a static level of high, the pulse will be low-going. When a channel is programmed with a static level of low, the pulse will be high-going. Each pulse generation requires a separate command. This command accepts a value that represents the channels to be pulsed within the 8-bit bus, specified in either decimal (0 - 255) or hex (0x00 - 0xFF). Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0.

Example: This code block sets bit 7 to a static level of low and then generates a high-going pulse.

```
vtex1048_set_vtb_output(vi, 0x00);
vtex1048_set_vtb_output_enable(vi, 0x80);
vtex1048_set_vtb_pulse(vi, 0x80);
```

Trigger Event

An acquisition sequence will be conducted once the programmed arm and trigger events are recognized. In addition to the programmed event sources, software arms and software triggers are always enabled. These are used to put the trigger model under direct software program control.

A software arm is performed by the **vtex1048_soft_arm** command. The software arm applies at the time it is issued and will be ignored if the system is not waiting for an ARM event. The issuance of a software arm has no effect on the ARM source configuration.

A software trigger is performed by the **vtex1048_soft_trigger** command. The software trigger applies at the time it is issued and will be ignored if the system is not waiting for a TRIG event. The issuance of a software trigger has no effect on the TRIG source configuration.

Retrieve Acquired Data

The EX1048 utilizes a 20 MB FIFO memory storage to buffer acquisition data prior to retrieval. The amount of scans that can be buffered within the memory is dependent on the number of channels in the scan list and the requested data format. See *Acquiring Data* in Section 3 for specific formulas and examples. The number of data pages (scans) that have yet to be retrieved from FIFO memory is obtained with the **vtex1048_get_fifo_count** query.

Acquisition data is retrieved from the EX1048 with the **vtex1048_read_fifo** command. Through the parameters of this command, there is control over how many scans of data are requested and the allowed time during which data retrieval is attempted. Data returned by this command includes input channel measurement data and the start time of each scan. CJC channel measurement data and individual channel timestamp data are not returned. If those quantities are required, the streaming interface must be used.

The syntax of this command is complex, and it is important for the user to understand the application of its various input parameters and return values. Another important concept is that

this command is a blocking command. That is, once it is sent by an application, no further commands can be executed within that application until the **vtex1048_read_fifo** command has completed its execution.

There are two ways for the execution to complete. The most obvious way is for the requested number of scans to actually be delivered by the EX1048. This is considered to be successful execution. The other way is for a specified timeout period to elapse. In this case, the full number of requested scans was not delivered; this is considered an execution error. However, there are applications where this is the expected and desired behavior, and all data returned is completely valid, despite this being technically an error condition. These concepts are clarified below.

This command accepts the following parameters:

- the maximum number of scans to return (*maxscans*). This represents the desired number of scans to retrieve. If FIFO count ≥ *maxscans* at the time of command issuance, then *maxscans* are returned, and the command completes. If, however, FIFO count < *maxscans* at the time of command issuance, the command will continue to poll the EX1048 for data until *maxscans* are returned or the timeout period (*to_secs*) is reached (see below).
- a return array of scan start times (*ts_secs[]*), specified in seconds since the epoch (Jan. 1, 1970). To ensure data integrity, the declared dimension of *ts_secs[]* must be at least as large as *maxscans*.
- a return array of scan start times (*ts_usecs[]*), specified in seconds since the last full second represented in *ts_secs[]*. To ensure data integrity, the declared dimension of *ts_usecs[]* must be at least as large as *maxscans*.
- a return value indicating the actual number of scans retrieved (*numscans*). If the command completed by reaching *maxscans*, then *numscans* will be equal to *maxscans*. If, however, the command completed by the timeout period, *numscans* will be less than *maxscans*.
- the maximum length of the return data array (*maxdata*). This should specifically be the declared dimension of the *data[]* array (see below). To ensure data integrity, *maxdata* must be no less than *maxscans* multiplied by the number of channels in the scan list. Since data is retrieved from the FIFO according to the *maxscans* parameter and not the *maxdata* parameter, if *maxdata* is set too low, there is the possibility of data being permanently lost.
- a return array of sample data (*data[]*).
- a return value indicating the actual number of samples retrieved (*numdata)*. This will be equal to the lesser of *maxdata* or *numscans* multiplied by the number of channels in the scan list.
- a timeout period in seconds (*to_secs*) that represents how long to poll the EX1048 for data (in an attempt to retrieve *maxscans*). A value of 0 is used to indicate an infinite timeout period.

The importance of the *to_secs* parameter depends on the trigger model configuration employed. In a typical application where the ARM source is set to Immediate and the TRIG source is set to Timer, the amount of data to be acquired and the time required to acquire that data are deterministic. Specifically, upon the trigger initialize command, there will be a quantity of scans acquired that equals TRIG count, acquired over a total time that is TRIG count multiplied by the timer interval. For example, 500 scans taken with a timer interval of 5 ms would require 2.5 s. In this case, *to_secs* would normally be set to an arbitrary value greater than 2.5. In this case, command completion via the timeout period does indeed represent an error condition.

In a contrasting example, consider an application where data is desired from the EX1048 only once a certain test condition has occurred. Accordingly, the ARM source is set to a specific condition of the digital I/O port. Moreover, this digital event is created asynchronously by a stimulus system being controlled by the same application that is taking EX1048 data. The logic of the control application is such that the stimulus system is increased to a new value and then the EX1048 is polled for data. If data is returned, then the EX1048 was triggered, and no further increase in the stimulus system is required. If no data is returned after a specified period, then the stimulus system is increased further. In this case, the timeout period value is important, as it determines the frequency at which the stimulus system is updated. Moreover, it is completely

expected that the **vtex1048_read_fifo** command initially completes due to the timeout period, and so this does not represent an error condition that requires intervention.

Example: This code block reads the FIFO after a timer-based scan.

```
#define NUM_CHANNELS        5
#define MAX_SCANS           20
#define MAX_DATA            (NUM_CHANNELS * MAX_SCANS)
#define TIMER_INTERVAL      0.01

ViInt32    channels[NUM_CHANNELS] = {0, 1, 2, 3, 4};
ViReal64 ts_secs[MAX_SCANS], ts_usecs[MAX_SCANS];
ViInt32    num_scans;
ViReal64 data[MAX_DATA];
ViInt32    num_data;

// set the scanlist
vtex1048_set_scanlist(vi, channels, NUM_CHANNELS);

// set the trigger source to be timer
vtex1048_set_trig_source_timer(vi, TIMER_INTERVAL);

// set trigger count
vtex1048_set_trigger_count(vi, MAX_SCANS);

// trigger initialize
vtex1048_init_imm(vi);

// retrieve the data, nominally ready in 2 seconds
vtex1048_read_fifo(vi, MAX_SCANS, ts_secs, ts_usecs, &num_scans, MAX_DATA, data, &num_data,
3);
```

Data correlation

The data returned by the **vtex1048_read_fifo** command is written into three one-dimensional arrays, termed data[], ts_secs[], and ts_usecs[] in this example. For a five-channel scan, the data from the first scan will be in elements data[0] to data[4] with a start of scan time indicated by ts_secs[0] + ts_usecs[0]. Data from the second scan will be in elements data[5] to data[9] with a start of scan time indicated by ts_secs[1] + ts_usecs[1], and so on. Within each scan, the first data element represents the first declared entry in the scan list. The second data element represents the second declared entry in the scan list, and so on.

NOTE	In order to provide the maximum reading buffer capacity for future acquisitions, data is deleted from the FIFO memory upon retrieval.

EXAMPLE PROGRAM

```
#include <ansi_c.h>
#include <stdio.h>
#include <windows.h>
#include <vtex1048.h>

#define INSTR_RESRC_STR         "TCPIP::192.168.0.127::INSTR"
#define TYPE_T                  3
#define TYPE_E                  4
#define NUM_CHANNELS            11
#define NUM_E_CHANNELS          6
#define NUM_T_CHANNELS          5
#define TRIG_TIMER              0.2     /* (5 readings per sec) */
#define MAX_SCANS               1000
#define MAX_DATA                (NUM_CHANNELS * MAX_SCANS)

int main( int argc, char **argv )
{
   ViSession        vi;
   ViStatus         status;
   /* scanlist entries */
   ViInt32          channels[NUM_CHANNELS] = {0,1,3,6,10,15,24,30,31,32,35};
   ViInt32          e_channels[NUM_E_CHANNELS] = {0,1,3,6,10,15};
   ViInt32          t_channels[NUM_T_CHANNELS] = {24,30,31,32,35};
   ViInt32          i, j;
   ViReal64         ts_secs[MAX_SCANS], ts_usecs[MAX_SCANS];
   ViInt32          numscans;
   ViReal64         data[MAX_DATA];
   ViInt32          numdata;

   /* open a session */
   status = vtex1048_init(INSTR_RESRC_STR, VI_ON, VI_ON, &vi);
   if(status != VI_SUCCESS)
   {
      printf("ERROR OPENING CONNECTION\n");
      return –1;
   }
   printf("Connection opened to %s\n", INSTR_RESRC_STR);

   /* configure the scan list */
   status = vtex1048_set_scanlist(vi, channels, NUM_CHANNELS);
   if(status != VI_SUCCESS)
   {
      printf("ERROR CONFIGURING SCAN LIST\n");
      return –1;
   }

   /* configure the EU conversions */
   status = vtex1048_set_channel_conversion(vi, e_channels, NUM_E_CHANNELS, TYPE_E);
   if(status != VI_SUCCESS)
   {
      printf("ERROR CONFIGURING EU CONVERSIONS\n");
      return –1;
   }
   status = vtex1048_set_channel_conversion(vi, t_channels, NUM_T_CHANNELS, TYPE_T);
   if(status != VI_SUCCESS)
```

```
   {
      printf("ERROR CONFIGURING EU CONVERSIONS\n");
      return –1;
   }

   /* configure the filter frequencies */
   status = vtex1048_set_filt_freq(vi, channels, NUM_CHANNELS, 4.0);
   if(status != VI_SUCCESS)
   {
      printf("ERROR CONFIGURING FILTER FREQUENCIES\n");
      return –1;
   }

   /* configure the FIFO (deg F, blocking mode) */
   status = vtex1048_set_fifo_config(vi, 0, 0, 0, 0, 1);
   if(status != VI_SUCCESS)
   {
      printf("ERROR CONFIGURING FIFO\n");
      return –1;
   }

   /* configure the trigger model */

   /* reset the trigger model to default settings */
   status = vtex1048_reset_trigger_arm(vi);

   /* set the trigger timer */
   status = vtex1048_set_trigger_timer(vi, TRIG_TIMER);
   if(status != VI_SUCCESS)
   {
      printf("ERROR CONFIGURING TIMER\n");
      return –1;
   }

   /* set the trigger count */
   status = vtex1048_set_trigger_count(vi, MAX_SCANS);
   if(status != VI_SUCCESS)
   {
      printf("ERROR CONFIGURING COUNT\n");
      return –1;
   }

   /* initialize the acquisition */
   status = vtex1048_init_imm(vi);
   if(status != VI_SUCCESS)
   {
      printf("ERROR INITIATING TRIGGER\n");
      return –1;
   }

   /* read acquisition data */
   status = vtex1048_read_fifo(vi, MAX_SCANS, ts_secs, ts_usecs, &numscans, MAX_DATA, data, &numdata, (ViInt32)
(MAX_SCANS * TRIG_TIMER + 0.1));
   if(status != VI_SUCCESS)
   {
      printf("ERROR READING DATA\n");
      return –1;
```

```
    }

    /* print acquisition data */
    for(i = 0; i < numscans; i++)
    {
        printf("%.0f.%06.0f: ", ts_secs[i], ts_usecs[i] * 1e6);
        for(j = 0; j < NUM_CHANNELS; j++)
        {
            printf("%6.2f ", data[i * NUM_CHANNELS + j]);
        }
        printf("\n");
    }

    /* close the session */
    status = vtex1048_close(vi);

    return 0;
}
```

SECTION 7

COMMAND DICTIONARY

INTRODUCTION

This section presents the instrument command set. It begins with an alphabetical list of all of the commands supported by the EX1048. With each command is a brief description of its function. The remainder of this section is devoted to describing each command, one per page, in detail. Each command entry provides the exact command and/or query syntax, the use and range of parameters, and a description of the command's purpose.

ALPHABETICAL COMMAND LIST

The following table provides a summary of the commands used by the EX1048.

Command	Description
vtex1048_abort	Aborts the current acquisition.
vtex1048_break_lock	Breaks a lock on the instrument.
vtex1048_check_lock	Queries the lock status of the instrument.
vtex1048_close	Closes an instrument programming session.
vtex1048_get_accum_limit_status	Queries the accumulated limit status of all 48 channels.
vtex1048_get_arm_count	Queries the arm count value.
vtex1048_get_arm_delay	Queries the arm delay.
vtex1048_get_arm_infinite	Queries the enabled status of an infinite arm count.
vtex1048_get_arm_source	Queries the enabled arm source events.
vtex1048_get_channel_conversion	Queries the engineering units (EU) conversion of a specified channel.
vtex1048_get_dio_input	Queries the current input state of the digital I/O port.
vtex1048_get_dio_limit_event	Queries the enabled DIO limit events.
vtex1048_get_dio_limit_event_invert	Queries the inverted operation of a DIO channel linked as a limit event.
vtex1048_get_dio_limit_event_latch	Queries the latch operation of a DIO channel linked as a limit event.
vtex1048_get_dio_output	Queries the programmed output state of the digital I/O port.
vtex1048_get_dio_output_enable	Queries the output enable state of the digital I/O port.
vtex1048_get_fifo_config	Queries the data format and overflow behavior of the FIFO memory.
vtex1048_get_fifo_count	Queries the number of data pages (scans) in the FIFO memory.
vtex1048_get_filt_freq	Queries the hardware filter frequency of a specified channel.
vtex1048_get_init_cont	Queries the enabled status of init continuous mode.
vtex1048_get_limit_set0	Queries the limit set 0 values of a specified channel.
vtex1048_get_limit_set0_manual	Queries the manual entry control of limit set 0 values of a specified channel.
vtex1048_get_limit_set1	Queries the limit set 1 values of a specified channel.
vtex1048_get_scanlist	Queries the current scan list.
vtex1048_get_trigger_count	Queries the trigger count value.
vtex1048_get_trigger_delay	Queries the trigger delay.
vtex1048_get_trigger_infinite	Queries the enabled status of an infinite trigger count.
vtex1048_get_trigger_source	Queries the enabled trigger source events.
vtex1048_get_trigger_timer	Queries the timer interval for the timer source event.
vtex1048_get_user_cjc_enable	Queries the enabled status of a user-defined CJC temperature of a specified channel.
vtex1048_get_user_cjc_temp	Queries the user-defined CJC temperature of a specified channel.
vtex1048_get_user_conversion	Queries the user-defined conversion polynomials.
vtex1048_get_vtb_input	Queries the current input state of the trigger bus.

Command	Description
vtex1048_get_vtb_output	Queries the programmed output state of the trigger bus.
vtex1048_get_vtb_output_enable	Queries the output enable state of the trigger bus.
vtex1048_init	Opens an instrument programming session.
vtex1048_init_imm	Performs a trigger initialize.
vtex1048_lock	Attempts to acquire a lock on the instrument.
vtex1048_read_fifo	Retrieves acquisition data.
vtex1048_reset	Performs an instrument reset.
vtex1048_reset_fifo	Clears the FIFO memory.
vtex1048_reset_trigger_arm	Performs a reset of the trigger configuration parameters.
vtex1048_revisionQuery	Queries the release revision of the instrument driver and embedded firmware.
vtex1048_self_cal_clear	Clears the current self cal data.
vtex1048_self_cal_clear_stored	Clears self cal data from nonvolatile memory.
vtex1048_self_cal_get_status	Queries the completion status of self-calibration.
vtex1048_self_cal_init	Performs an instrument self-calibration.
vtex1048_self_cal_is_stored	Queries the presence of self cal data in nonvolatile memory.
vtex1048_self_cal_load	Loads nonvolatile self cal data as the current self cal data.
vtex1048_self_cal_store	Stores the current self cal data into nonvolatile memory.
vtex1048_set_arm_count	Sets the arm count value.
vtex1048_set_arm_delay	Sets the arm delay.
vtex1048_set_arm_infinite	Enables or disables the use of an infinite arm count.
vtex1048_set_arm_source	Sets the arm source events.
vtex1048_set_channel_conversion	Sets the engineering units (EU) conversion for the specified channels.
vtex1048_set_dio_limit_event	Links limit evaluations to the operation of the digital I/O port.
vtex1048_set_dio_limit_event_invert	Enables or disables inverted operation of a DIO channel linked as a limit event.
vtex1048_set_dio_limit_event_latch	Enables or disables latch operation of a DIO channel linked as a limit event.
vtex1048_set_dio_output	Sets the static level that each channel of the digital I/O port will assume if enabled.
vtex1048_set_dio_output_enable	Enables or disables the output functionality of each channel of the digital I/O port.
vtex1048_set_dio_pulse	Generates a 1 µs pulse on selected channels of the digital I/O port.
vtex1048_set_fifo_config	Sets the data format and overflow behavior of the FIFO memory.
vtex1048_set_filt_freq	Sets the hardware filter frequency for the specified channels.
vtex1048_set_init_cont	Enables or disables the use of init continuous mode.
vtex1048_set_limit_set0	Sets the limit set 0 values manually for the specified channels.
vtex1048_set_limit_set0_manual	Enables or disables manual entry of limit set 0 values for the specified channels.
vtex1048_set_limit_set1	Sets the limit set 1 values for the specified channels.
vtex1048_set_scanlist	Sets the scan list to be acquired.
vtex1048_set_trig_source_timer	Sets a trigger source of timer only and sets the timer interval.
vtex1048_set_trigger_count	Sets the trigger count value.
vtex1048_set_trigger_delay	Sets the trigger delay.
vtex1048_set_trigger_infinite	Enables or disables the use of an infinite trigger count.
vtex1048_set_trigger_source	Sets the trigger source events.
vtex1048_set_trigger_timer	Sets the timer interval for the timer source event.
vtex1048_set_user_cjc_enable	Enables or disables the use of a user-defined CJC temperature for the specified channels.
vtex1048_set_user_cjc_temp	Sets the user-defined CJC temperature for the specified channels.
vtex1048_set_user_conversion	Sets the user-defined conversion polynomials.
vtex1048_set_vtb_output	Sets the static level that each channel of the trigger bus will assume if enabled.
vtex1048_set_vtb_output_enable	Enables or disables the output functionality of each channel of the trigger bus.
vtex1048_set_vtb_pulse	Generates a 1 µs pulse on selected channels of the trigger bus.
vtex1048_soft_arm	Performs a software arm.
vtex1048_soft_trigger	Performs a software trigger.
vtex1048_unlock	Releases a lock on the instrument.

COMMAND DICTIONARY

The remainder of this section is devoted to the actual command dictionary. Each command is fully described on its own page. In defining how each command is used, the following items are described:

Purpose	Describes the purpose of the command.
Type	Describes the type of command such as an event or setting.
Command Syntax	Details the exact command format.
Command Parameters	Describes the parameters sent with the command and their legal range.
Reset Value	Describes the values assumed when a **vtex1048_reset** command is sent.
Query Response	Describes the format of the query response and the valid range of output.
Description	Describes in detail what the command does and refers to additional sources.
Examples	Presents the proper use of each command/query.

vtex1048_abort

Purpose	Aborts the current acquisition.
Type	Event
Command Syntax	vtex1048_abort(**vi**)
Command Parameters	**vi** = session ID
Reset Value	N/A
Query Response	N/A
Description	Aborts the current acquisition, transitioning the trigger model back to the IDLE layer.
Examples	

vtex1048_break_lock

Purpose	Breaks a lock on the instrument.
Type	Event
Command Syntax	vtex1048_break_lock(**vi**)
Command Parameters	**vi** = session ID
Reset Value	N/A
Query Response	N/A
Description	Breaks a lock on the instrument. This releases a lock on the instrument, regardless of its owner. This allows for instrument recovery if the locking IP address would become disabled. **NOTE: Breaking a lock on the instrument does not automatically acquire it. That must be done with a separate *vtex1048_lock* command.**
Examples	

vtex1048_check_lock

Purpose	Queries the lock status of the instrument.
Type	Query
Command Syntax	vtex1048_check_lock(**vi, &locked, &mine**)
Command Parameters	**vi** = session ID **locked** = a Boolean value indicating the lock status of the instrument. **mine** = a Boolean value indicating whether the lock is owned by the host IP address that issued the query.
Reset Value	N/A
Query Response	N/A
Description	Queries the lock status of the instrument. When locked, the EX1048 will accept commands from only the host IP address that issued the lock command.
Examples	

vtex1048_close

Purpose	Closes an instrument programming session.
Type	Event
Command Syntax	vtex1048_close(**vi**)
Command Parameters	**vi** = session ID
Reset Value	N/A
Query Response	N/A
Description	Closes an instrument programming session. This command should be performed at the conclusion of the test application. Part of its execution is the unlocking of the instrument, leaving it in the proper state for the next application.
Examples	

vtex1048_get_accum_limit_status

Purpose	Queries the accumulated limit status of all 48 channels.
Type	Query
Command Syntax	vtex1048_get_accum_limit_status(**vi**, **set0_lower[]**, **set0_upper[], set1_lower[], set1_upper[]**)
Command Parameters	**vi** = session ID **set0_lower[]** = a return array of Boolean values representing the accumulated limit status of all 48 channels with respect to each channel's limit set 0 lower limit. Channel 0 through channel 47 are represented in array elements [0] through [47], respectively. **set0_upper[]** = a return array of Boolean values representing the accumulated limit status of all 48 channels with respect to each channel's limit set 0 upper limit. Channel 0 through channel 47 are represented in array elements [0] through [47], respectively. **set1_lower[]** = a return array of Boolean values representing the accumulated limit status of all 48 channels with respect to each channel's limit set 1 lower limit. Channel 0 through channel 47 are represented in array elements [0] through [47], respectively. **set1_upper[]** = a return array of Boolean values representing the accumulated limit status of all 48 channels with respect to each channel's limit set 1 upper limit. Channel 0 through channel 47 are represented in array elements [0] through [47], respectively.
Reset Value	N/A
Query Response	**set0_lower**: each array element is 0 (not tripped) or 1 (tripped). **set0_upper**: each array element is 0 (not tripped) or 1 (tripped). **set1_lower**: each array element is 0 (not tripped) or 1 (tripped). **set1_upper**: each array element is 0 (not tripped) or 1 (tripped).
Description	Queries the accumulated limit status of all 48 channels. Limit status is returned for all channels, regardless of their inclusion in the scan list. The returned values represent, on a per channel basis, any excursion of the measurement data over their respective limits since the last trigger initialize. As implied, limit status is cleared as part of the *vtex1048_init_imm* command. The limit status for unscanned channels is always 0.
Examples	

vtex1048_get_arm_count

Purpose	Queries the arm count value.
Type	Query
Command Syntax	vtex1048_get_arm_count(**vi**, **&count**)
Command Parameters	**vi** = session ID **count** = an integer output value indicating the arm count.
Reset Value	N/A
Query Response	**count**: an integer in the range of 1 to (2^{31}-1).
Description	Queries the arm count value.
Examples	

vtex1048_get_arm_delay

Purpose	Queries the arm delay.
Type	Query
Command Syntax	vtex1048_get_arm_delay(**vi**, **&arm_delay**)
Command Parameters	**vi** = session ID **arm_delay** = a real output value (in seconds) indicating the arm delay.
Reset Value	N/A
Query Response	**arm_delay**: a real number in the range of 0 to 4294.
Description	Queries the arm delay, the time between the recognition of the arm event and the transition into the TRIG layer of the trigger model.
Examples	

vtex1048_get_arm_infinite

Purpose	Queries the enabled status of an infinite arm count.
Type	Query
Command Syntax	vtex1048_get_arm_infinite(**vi**, **&arminf**)
Command Parameters	**vi** = session ID **arminf** = a Boolean value indicating whether the use of an infinite arm count is enabled.
Reset Value	N/A
Query Response	**arminf**: 0 or 1.
Description	Queries the enabled status of an infinite arm count.
Examples	

vtex1048_get_arm_source

Purpose	Queries the enabled arm source events.
Type	Query
Command Syntax	vtex1048_get_arm_source(**vi**, **vtb_masks[]**, **dio_masks[]**, **&timer_enable**, **&immediate**)
Command Parameters	**vi** = session ID **vtb_masks[]** = a return array of four 8-bit integer values representing the enabled state of arm events from the 8 channels of the trigger bus. The order of the values is: positive edge, negative edge, positive level, negative level. Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0. **dio_masks[]** = a return array of four 8-bit integer values representing the enabled state of arm events from the 8 channels of the digital I/O port. The order of the values is: positive edge, negative edge, positive level, negative level. Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0. **timer_enable** = a Boolean value indicating whether the timer is enabled as an arm event. **immediate** = a Boolean value indicating whether immediate is enabled as an arm event.
Reset Value	N/A
Query Response	**vtb_masks**: each array element is an integer in the range of 0 to 255. **dio_masks**: each array element is an integer in the range of 0 to 255. **timer_enable**: 0 or 1. **immediate**: 0 or 1.
Description	Queries the enabled arm source events. Regardless of the response, software arms are always enabled.
Examples	

vtex1048_get_channel_conversion

Purpose	Queries the engineering units (EU) conversion of a specified channel.
Type	Query
Command Syntax	vtex1048_get_channel_conversion(**vi**, **channel**, **&eu_conv**)
Command Parameters	**vi** = session ID **channel** = the channel for which the EU conversion value is desired. Value must be an integer in the range of 0 to 47. **eu_conv** = an integer output value representing the EU conversion, which maps to these functions: **Integer Value** — **Function** 0 — Voltage 1 — Type J 2 — Type K 3 — Type T 4 — Type E 5 — Type B 6 — Type S 7 — Type R 8 — Type N 9 — User-defined 0 10 — User-defined 1
Reset Value	N/A
Query Response	**eu_conv**: an integer in the range of 0 to 10.
Description	Queries the engineering units (EU) conversion of a specified channel.
Examples	

vtex1048_get_dio_input

Purpose	Queries the current input state of the digital I/O port.
Type	Query
Command Syntax	vtex1048_get_dio_input(**vi**, **&dio_in**)
Command Parameters	**vi** = session ID **dio_in** = an integer output value in decimal representing the 8-bit value of the port. Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0.
Reset Value	N/A
Query Response	**dio_in**: an integer in the range of 0 to 255.
Description	Queries the current input state of the digital I/O port.

Examples	// check state of DIO bits 7, 4, and 0
	ViInt32 dio_in;
	vtex1048_get_dio_input(vi, &dio_in);
	if (dio_in & 0x80)
	printf("Bit 7 is high");
	else printf("Bit 7 is low");
	if (dio_in & 0x10)
	printf("Bit 4 is high");
	else printf("Bit 4 is low");
	if (dio_in & 0x01)
	printf("Bit 0 is high");
	else printf("Bit 0 is low");

vtex1048_get_dio_limit_event

Purpose	Queries the enabled DIO limit events.
Type	Query
Command Syntax	vtex1048_get_dio_limit_event(**vi**, **dio_channel**, **limit_masks[]**)
Command Parameters	**vi** = session ID **dio_channel** = the channel of the digital I/O port for which the DIO limit event status is desired. Value must be an integer in the range of 0 to 7. **limit_masks[]** = a return array of 48 4-bit integer values representing, on a per input channel basis, the linking of limit evaluations to any of the 4 limit conditions. Within the 4-bit field, the order of the values is: limit set 0 lower, limit set 0 upper, limit set 1 lower, limit set 1 upper. Channel 0 through channel 47 are represented in array elements [0] through [47], respectively.
Reset Value	N/A
Query Response	**limit_masks**: each array element is an integer in the range of 0 to 15.
Description	Queries the enabled DIO limit events.
Examples	

vtex1048_get_dio_limit_event_invert

Purpose	Queries the inverted operation of a DIO channel linked as a limit event.
Type	Query
Command Syntax	vtex1048_get_dio_limit_event_invert(**vi**, **dio_channel, &invert**)
Command Parameters	**vi** = session ID **dio_channel** = the channel of the digital I/O port to be queried. Value must be an integer in the range of 0 to 7. **invert** = a Boolean value indicating whether the specified DIO channel is operating in invert mode.
Reset Value	N/A
Query Response	**invert**: 0 or 1.
Description	Queries the inverted operation of a DIO channel linked as a limit event.
Examples	

vtex1048_get_dio_limit_event_latch

Purpose	Queries the latch operation of a DIO channel linked as a limit event.
Type	Query
Command Syntax	vtex1048_get_dio_limit_event_latch(**vi**, **dio_channel, &latch**)
Command Parameters	**vi** = session ID **dio_channel** = the channel of the digital I/O port to be queried. Value must be an integer in the range of 0 to 7. **latch** = a Boolean value indicating whether the specified DIO channel is operating in latch mode.
Reset Value	N/A
Query Response	**latch**: 0 or 1.
Description	Queries the latch operation of a DIO channel linked as a limit event.
Examples	

vtex1048_get_dio_output

Purpose	Queries the programmed output state of the digital I/O port.
Type	Query
Command Syntax	vtex1048_get_dio_output(**vi**, **&dio_out**)
Command Parameters	**vi** = session ID **dio_out** = an integer output value in decimal that represents the programmed output state of the 8-bit port. Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0.
Reset Value	N/A
Query Response	**dio_out**: an integer in the range of 0 to 255.
Description	Queries the programmed output state of the digital I/O port. This simply returns the programmed setting. Since the outputs must be enabled, it does not necessarily represent the true output state.

Examples	// query status of DIO bit 4
	ViInt32 dio_out;
	ViInt32 dio_outen;
	vtex1048_get_dio_output(vi, &dio_out);
	vtex1048_get_dio_output_enable(vi, &dio_outen);
	if (dio_out & 0x10)
	printf("Bit 4 is set high");
	else printf("Bit 4 is set low");
	if (dio_outen & 0x10)
	printf("Bit 4 is enabled");
	else printf("Bit 4 is not enabled");

vtex1048_get_dio_output_enable

Purpose	Queries the output enable state of the digital I/O port.
Type	Query
Command Syntax	vtex1048_get_dio_output_enable(**vi**, **&out_enable**)
Command Parameters	**vi** = session ID **out_enable** = an integer output value in decimal that represents the output enable state of the 8-bit port. Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0.
Reset Value	N/A
Query Response	**out_enable**: an integer in the range of 0 to 255.
Description	Queries the output enable state of the digital I/O port.

Examples	// query status of DIO bit 4
	ViInt32 dio_out;
	ViInt32 dio_outen;
	vtex1048_get_dio_output(vi, &dio_out);
	vtex1048_get_dio_output_enable(vi, &dio_outen);
	if (dio_out & 0x10)
	printf("Bit 4 is set high");
	else printf("Bit 4 is set low");
	if (dio_outen & 0x10)
	printf("Bit 4 is enabled");
	else printf("Bit 4 is not enabled");

vtex1048_get_fifo_config

Purpose	Queries the data format and overflow behavior of the FIFO memory.
Type	Query
Command Syntax	vtex1048_get_fifo_config(**vi, &report_cjc, &report_timestamp, &report_celsius, &blocking_mode**)
Command Parameters	**vi** = session ID **report_cjc** = a Boolean value indicating whether CJC temperatures will be reported in the acquisition data. Reported CJC temperatures are always in units of ºC, regardless of the value of the **report_celsius** parameter. **report_timestamp** = a Boolean value indicating whether delta timestamps per channel will be reported in the acquisition data. **report_celsius** = a Boolean value indicating whether the input channel data will be reported in units of ºC (true) or ºF (false). **blocking_mode** = a Boolean value indicating the FIFO behavior upon reaching maximum capacity during scanning. With blocking mode enabled, additional readings are discarded, leaving the contents of the buffer intact. With blocking mode disabled, the buffer becomes circular, in that additional readings overwrite the oldest readings.
Reset Value	N/A
Query Response	**report_cjc**: 0 or 1. **report_timestamp**: 0 or 1. **report_celsius**: 0 or 1. **blocking_mode**: 0 or 1.
Description	Queries the data format and overflow behavior of the FIFO memory. **NOTE: Regardless of the settings of the report_cjc and report_timestamp parameters, the CJC temperature and delta timestamp information is not accessible through the *vtex1048_read_fifo* command. They are available through the streaming interface, and the parameter settings apply for it.**
Examples	

vtex1048_get_fifo_count

Purpose	Queries the number of data pages (scans) in the FIFO memory.
Type	Query
Command Syntax	vtex1048_get_fifo_count(**vi**, **&count**)
Command Parameters	**vi** = session ID **count** = an integer output value indicating the number of data pages.
Reset Value	N/A
Query Response	**count**: an integer in the range of 0 to 187246.
Description	Queries the number of data pages (scans) in the FIFO memory.
Examples	

vtex1048_get_filt_freq

Purpose	Queries the hardware filter frequency of a specified channel.
Type	Query
Command Syntax	vtex1048_get_filt_freq(**vi**, **channel**, **&filt_freq**)
Command Parameters	**vi** = session ID **channel** = the channel for which the hardware filter frequency value is desired. Value must be an integer in the range of 0 to 47. **filt_freq** = a real output value representing the frequency setting.
Reset Value	N/A
Query Response	**filt_freq**: 4.0 or 1000.0
Description	Queries the hardware filter frequency of a specified channel.
Examples	

vtex1048_get_init_cont

Purpose	Queries the enabled status of init continuous mode.
Type	Query
Command Syntax	vtex1048_get_init_cont(**vi**, **&init_cont_mode**)
Command Parameters	**vi** = session ID **init_cont_mode** = a Boolean value indicating whether init continuous mode is enabled.
Reset Value	N/A
Query Response	**init_cont_mode**: 0 or 1.
Description	Queries the enabled status of init continuous mode.
Examples	

vtex1048_get_limit_set0

Purpose	Queries the limit set 0 values of a specified channel.
Type	Query
Command Syntax	vtex1048_get_limit_set0(**vi**, **channel**, **&lower_limit, &upper_limit**)
Command Parameters	**vi** = session ID **channel** = the channel for which the limit set 0 values are desired. Value must be an integer in the range of 0 to 47. **lower_limit** = a real output value indicating the lower limit. **upper_limit** = a real output value indicating the upper limit.
Reset Value	N/A
Query Response	**lower_limit**: a real number in the range of -3e+38 to 3e+38. **upper_limit**: a real number in the range of -3e+38 to 3e+38.
Description	Queries the limit set 0 values of a specified channel.
Examples	

vtex1048_get_limit_set0_manual

Purpose	Queries the manual entry control of limit set 0 values of a specified channel.
Type	Query
Command Syntax	vtex1048_get_limit_set0_manual(**vi**, **channel**, **&manual**)
Command Parameters	**vi** = session ID **channel** = the channel for which the manual entry control value is desired. Value must be an integer in the range of 0 to 47. **manual** = a Boolean value indicating whether manual entry is enabled.
Reset Value	N/A
Query Response	**manual**: 0 or 1.
Description	Queries the manual entry control of limit set 0 values of a specified channel.
Examples	

vtex1048_get_limit_set1

Purpose	Queries the limit set 1 values of a specified channel.
Type	Query
Command Syntax	vtex1048_get_limit_set1(**vi**, **channel**, **&lower_limit, &upper_limit**)
Command Parameters	**vi** = session ID **channel** = the channel for which the limit set 1 values are desired. Value must be an integer in the range of 0 to 47. **lower_limit** = a real output value indicating the lower limit. **upper_limit** = a real output value indicating the upper limit.
Reset Value	N/A
Query Response	**lower_limit**: a real number in the range of -3e+38 to 3e+38. **upper_limit**: a real number in the range of -3e+38 to 3e+38.
Description	Queries the limit set 1 values of a specified channel.
Examples	

vtex1048_get_scanlist

Purpose	Queries the current scan list.
Type	Query
Command Syntax	vtex1048_get_scanlist(**vi**, **channels[]**, **&numChannels**)
Command Parameters	**vi** = session ID **channels[]** = an array to receive the requested scan list. **numChannels** = an integer output value indicating how many channels are in the scan list.
Reset Value	N/A
Query Response	**channels[]**: an array of up to 48 integer values in the range of 0 to 47. **numChannels**: an integer in the range of 1 to 48.
Description	Queries the current scan list.
Examples	

vtex1048_get_trigger_count

Purpose	Queries the trigger count value.
Type	Query
Command Syntax	vtex1048_get_trigger_count(**vi**, **&count**)
Command Parameters	**vi** = session ID **count** = an integer output value indicating the trigger count.
Reset Value	N/A
Query Response	**count**: an integer in the range of 1 to (2^{31}-1).
Description	Queries the trigger count value.
Examples	

vtex1048_get_trigger_delay

Purpose	Queries the trigger delay.
Type	Query
Command Syntax	vtex1048_get_trigger_delay(**vi**, **&trig_delay**)
Command Parameters	**vi** = session ID **trig_delay** = a real output value (in seconds) indicating the trigger delay.
Reset Value	N/A
Query Response	**trig_delay**: a real number in the range of 0 to 4294.
Description	Queries the trigger delay, the time between the recognition of the trigger event and the execution of the scan list.
Examples	

vtex1048_get_trigger_infinite

Purpose	Queries the enabled status of an infinite trigger count.
Type	Query
Command Syntax	vtex1048_get_trigger_infinite(**vi**, **&triginf**)
Command Parameters	**vi** = session ID **triginf** = a Boolean value indicating whether the use of an infinite trigger count is enabled.
Reset Value	N/A
Query Response	**triginf**: 0 or 1.
Description	Queries the enabled status of an infinite trigger count.
Examples	

vtex1048_get_trigger_source

Purpose	Queries the enabled trigger source events.
Type	Query
Command Syntax	vtex1048_get_trigger_source(**vi**, **vtb_masks[]**, **dio_masks[]**, **&timer_enable, &immediate**)
Command Parameters	**vi** = session ID **vtb_masks[]** = a return array of four 8-bit integer values representing the enabled state of trigger events from the 8 channels of the trigger bus. The order of the values is: positive edge, negative edge, positive level, negative level. Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0. **dio_masks[]** = a return array of four 8-bit integer values representing the enabled state of trigger events from the 8 channels of the digital I/O port. The order of the values is: positive edge, negative edge, positive level, negative level. Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0. **timer_enable** = a Boolean value indicating whether the timer is enabled as a trigger event. **immediate** = a Boolean value indicating whether immediate is enabled as a trigger event.
Reset Value	N/A
Query Response	**vtb_masks**: each array element is an integer in the range of 0 to 255. **dio_masks**: each array element is an integer in the range of 0 to 255. **timer_enable**: 0 or 1. **immediate**: 0 or 1.
Description	Queries the enabled trigger source events. Regardless of the response, software triggers are always enabled.
Examples	

vtex1048_get_trigger_timer

Purpose	Queries the timer interval for the timer source event.
Type	Query
Command Syntax	vtex1048_get_trigger_timer(**vi**, **&trig_timer**)
Command Parameters	**vi** = session ID **trig_timer** = a real output value indicating the time (in seconds) between timer ticks.
Reset Value	N/A
Query Response	**trig_timer**: a real number in the range of 0.001 to 4294.
Description	Queries the timer interval for the timer source event. The same value is used for both arm and trigger events.
Examples	

vtex1048_get_user_cjc_enable

Purpose	Queries the enabled status of a user-defined CJC temperature of a specified channel.
Type	Query
Command Syntax	vtex1048_get_user_cjc_enable(**vi**, **channel**, **&usercjc**)
Command Parameters	**vi** = session ID **channel** = the channel for which the enabled status is desired. Value must be an integer in the range of 0 to 47. **usercjc** = a Boolean value indicating whether the use of a user-defined CJC temperature is enabled.
Reset Value	N/A
Query Response	**usercjc**: 0 or 1.
Description	Queries the enabled status of a user-defined CJC temperature of a specified channel.
Examples	

vtex1048_get_user_cjc_temp

Purpose	Queries the user-defined CJC temperature of a specified channel.
Type	Query
Command Syntax	vtex1048_get_user_cjc_temp(**vi**, **channel**, **&cjc_temp**)
Command Parameters	**vi** = session ID **channel** = the channel for which the user-defined CJC temperature is desired. Value must be an integer in the range of 0 to 47. **cjc_temp** = a real output value indicating the CJC temperature in ºC.
Reset Value	N/A
Query Response	**cjc_temp**: a real number in the range of -3e+38 to 3e+38.
Description	Queries the user-defined CJC temperature of a specified channel.
Examples	

vtex1048_get_user_conversion

Purpose	Queries the user-defined conversion polynomials.
Type	Query
Command Syntax	vtex1048_get_user_conversion(**vi, eu_conv, fwdcoeff[], invcoeff[]**)
Command Parameters	**vi** = session ID **eu_conv** = the polynomial set to be obtained. Value must be an integer equal to 9 (User0) or 10 (User1). **fwdcoeff[]** = a return array of forward conversion polynomial coefficients. Coefficients c_0 through c_{12} are represented in array elements [0] through [12], respectively. **invcoeff[]** = a return array of inverse conversion polynomial coefficients. Coefficients d_0 through d_{12} are represented in array elements [0] through [12], respectively.
Reset Value	N/A
Query Response	**fwdcoeff[]**: each array element is a real number in the range of -3e+38 to 3e+38. **invcoeff[]**: each array element is a real number in the range of -3e+38 to 3e+38.
Description	Queries the user-defined conversion polynomials. All twelve elements of the arrays are returned, regardless of how many were set with the *vtex1048_set_user_conversion* command.
Examples	

vtex1048_get_vtb_input

Purpose	Queries the current input state of the trigger bus.
Type	Query
Command Syntax	vtex1048_get_vtb_input(**vi**, **&vtb_in**)
Command Parameters	**vi** = session ID **vtb_in** = an integer output value in decimal representing the 8-bit value of the port. Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0.
Reset Value	N/A
Query Response	**vtb_in**: an integer in the range of 0 to 255.
Description	Queries the current input state of the trigger bus.

Examples	// check state of VTB bits 7, 4, and 0
	ViInt32 vtb_in;
	vtex1048_get_vtb_input(vi, &vtb_in);
	if (vtb_in & 0x80)
	printf("Bit 7 is high");
	else printf("Bit 7 is low");
	if (vtb_in & 0x10)
	printf("Bit 4 is high");
	else printf("Bit 4 is low");
	if (vtb_in & 0x01)
	printf("Bit 0 is high");
	else printf("Bit 0 is low");

vtex1048_get_vtb_output

Purpose	Queries the programmed output state of the trigger bus.
Type	Query
Command Syntax	vtex1048_get_vtb_output(**vi**, **&vtb_out**)
Command Parameters	**vi** = session ID **vtb_out** = an integer output value in decimal that represents the programmed output state of the 8-bit port. Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0.
Reset Value	N/A
Query Response	**vtb_out**: an integer in the range of 0 to 255.
Description	Queries the programmed output state of the trigger bus. This simply returns the programmed setting. Since the outputs must be enabled, it does not necessarily represent the true output state.

Examples	// query status of VTB bit 4
	ViInt32 vtb_out;
	ViInt32 vtb_outen;
	vtex1048_get_vtb_output(vi, &vtb_out);
	vtex1048_get_vtb_output_enable(vi, &vtb_outen);
	if (vtb_out & 0x10)
	printf("Bit 4 is set high");
	else printf("Bit 4 is set low");
	if (vtb_outen & 0x10)
	printf("Bit 4 is enabled");
	else printf("Bit 4 is not enabled");

vtex1048_get_vtb_output_enable

Purpose	Queries the output enable state of the trigger bus.
Type	Query
Command Syntax	vtex1048_get_vtb_output_enable(**vi**, **&out_enable**)
Command Parameters	**vi** = session ID **out_enable** = an integer output value in decimal that represents the output enable state of the 8-bit port. Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0.
Reset Value	N/A
Query Response	**out_enable**: an integer in the range of 0 to 255.
Description	Queries the output enable state of the trigger bus.

Examples	// query status of VTB bit 4
	ViInt32 vtb_out;
	ViInt32 vtb_outen;
	vtex1048_get_vtb_output(vi, &vtb_out);
	vtex1048_get_vtb_output_enable(vi, &vtb_outen);
	if (vtb_out & 0x10)
	printf("Bit 4 is set high");
	else printf("Bit 4 is set low");
	if (vtb_outen & 0x10)
	printf("Bit 4 is enabled");
	else printf("Bit 4 is not enabled");

vtex1048_init

Purpose	Opens an instrument programming session.
Type	Event
Command Syntax	vtex1048_init(**resourceName, IDQuery, resetDevice, &vi**)
Command Parameters	**resourceName** = the VISA resource string. It has the form of "TCPIP::<W.X.Y.Z>::INSTR", where W.X.Y.Z represents the IP address to which to connect. **IDQuery** = a Boolean value indicating whether to perform confirmation that the connected instrument is an EX1048. **resetDevice** = a Boolean value indicating whether to reset the instrument upon connecting. **vi** = the session handle (ID), unique to each connection instance.
Reset Value	N/A
Query Response	N/A
Description	Opens an instrument programming session. This command must be successfully performed in order to communicate with the EX1048. Sessions to multiple instruments can be opened within the same application, each uniquely identified by their session handle.

Examples	#include <vtex1048.h>
	
	#define INSTR_RESRC_STR "TCPIP::192.168.0.127::INSTR"
	
	ViSession vi;
	
	// open a session to EX1048 at IP address 192.168.0.127
	vtex1048_init(INSTR_RESRC_STR, 1, 1, &vi);

vtex1048_init_imm

Purpose	Performs a trigger initialize.
Type	Event
Command Syntax	vtex1048_init_imm(**vi**)
Command Parameters	**vi** = session ID
Reset Value	N/A
Query Response	N/A
Description	Performs a trigger initialize, transitioning the trigger model out of the IDLE layer.
Examples	

vtex1048_lock

Purpose	Attempts to acquire a lock on the instrument.
Type	Event
Command Syntax	vtex1048_lock(**vi**)
Command Parameters	**vi** = session ID
Reset Value	N/A
Query Response	N/A
Description	Attempts to acquire a lock on the instrument. When locked, the EX1048 will accept commands from only the host IP address that issued the lock command. A lock can only be acquired if the instrument is not already locked by another user. By design, the locking mechanism is able to be overridden by a secondary host that issues a *vtex1048_break_lock* command. Thus, the lock provides a warning to other users that the unit is in a protected operation state, but not absolute security. The lock status of the instrument is unaffected by the *vtex1048_reset* command. Self-calibration requires the acquisition of a lock prior to its initiation.
Examples	

vtex1048_read_fifo

Purpose	Retrieves acquisition data.
Type	Query
Command Syntax	vtex1048_read_fifo(**vi, maxscans, ts_secs[], ts_usecs[], &numscans, maxdata, data[], &numdata, to_secs**)
Command Parameters	**vi** = session ID **maxscans** = the maximum number of scans to return. Value must be an integer in the range of 1 to (2^{31}-1). **ts_secs[]** = a return array of scan start times, specified in seconds since the epoch (Jan. 1, 1970). **ts_usecs[]** = a return array of scan start times, specified in seconds since the last full second represented in ts_secs[]. **numscans** = an integer output value indicating the actual number of scans retrieved. **maxdata** = the maximum length of the return data array. Value must be an integer in the range of 1 to (2^{31}-1). **data[]** = a return array of sample data. **numdata** = an integer output value indicating the actual number of samples retrieved. **to_secs** = the timeout period (in seconds), indicating how long to poll the EX1048 for data. Value must be an integer in the range of 0 to (2^{31}-1), where 0 represents an infinite timeout period.
Reset Value	N/A
Query Response	**ts_secs[]**: an array of real numbers. **ts_usecs[]**: an array of real numbers. **numscans**: an integer in the range of 0 to (2^{31}-1). **data[]**: an array of real numbers. **numdata**: an integer in the range of 0 to (2^{31}-1).
Description	Retrieves acquisition data. Data returned by this command includes input channel measurement data and the start time of each scan. **NOTE: In order to provide the maximum reading buffer capacity for future acquisitions, data is deleted from the FIFO memory upon retrieval.**

Examples	ViInt32 channels[5] = {0, 1, 2, 3, 4};
	ViReal64 ts_secs[20], ts_usecs[20], data[100];
	ViInt32 num_scans, num_data;
	
	vtex1048_set_scanlist(vi, channels, 5);
	vtex1048_set_trig_source_timer(vi, 0.01);
	vtex1048_set_trigger_count(vi, 20);
	vtex1048_init_imm(vi);
	vtex1048_read_fifo(vi, 20, ts_secs, ts_usecs, &num_scans, 100, data, &num_data, 3);

vtex1048_reset

Purpose	Performs an instrument reset.
Type	Event
Command Syntax	vtex1048_reset(**vi**)
Command Parameters	**vi** = session ID
Reset Value	N/A
Query Response	N/A
Description	Performs an instrument reset, returning all of the EX1048's acquisition configuration parameters to their default values. **NOTE: An instrument reset clears the FIFO reading memory. All desired acquisition data must be retrieved from the FIFO prior to the issuance of this command.**
Examples	

vtex1048_reset_fifo

Purpose	Clears the FIFO memory.
Type	Event
Command Syntax	vtex1048_reset_fifo(**vi**)
Command Parameters	**vi** = session ID
Reset Value	N/A
Query Response	N/A
Description	Clears the FIFO memory. Since the memory is cleared upon the receipt of a trigger initialize command and made available as data is retrieved, this command is normally not needed. That is, the FIFO need not be specifically cleared before a new acquisition is initiated.
Examples	

vtex1048_reset_trigger_arm

Purpose	Performs a reset of the trigger configuration parameters.
Type	Event
Command Syntax	vtex1048_reset_trigger_arm(**vi**)
Command Parameters	**vi** = session ID
Reset Value	N/A
Query Response	N/A
Description	Performs a reset of the trigger configuration parameters to default values, while not affecting any other acquisition parameters. In contrast, the *vtex1048_reset* command resets all of the acquisition parameters to their default values.
Examples	

vtex1048_revisionQuery

Purpose	Queries the release revision of the instrument driver and embedded firmware.
Type	Query
Command Syntax	vtex1048_revisionQuery(**vi, driverRev[], instrRev[]**)
Command Parameters	**vi** = session ID **driverRev[]** = a return array representing the release revision of the instrument driver. **instrRev[]** = a return array representing the release revision of the embedded firmware.
Reset Value	N/A
Query Response	**driverRev[]**: an array of characters. **instrRev[]**: an array of characters.
Description	Queries the release revision of the instrument driver and embedded firmware.
Examples	ViChar driverRev[256], instrRev[256]; vtex1048_revisionQuery(vi, driverRev, instrRev);

vtex1048_self_cal_clear

Purpose	Clears the current self cal data.
Type	Event
Command Syntax	vtex1048_self_cal_clear(**vi**)
Command Parameters	**vi** = session ID
Reset Value	N/A
Query Response	N/A
Description	Clears the current self cal data. This operation clears the volatile data, but does not affect any self cal data that is stored in nonvolatile memory.
Examples	

vtex1048_self_cal_clear_stored

Purpose	Clears self cal data from nonvolatile memory.
Type	Event
Command Syntax	vtex1048_self_cal_clear_stored(**vi**)
Command Parameters	**vi** = session ID
Reset Value	N/A
Query Response	N/A
Description	Clears self cal data from nonvolatile memory. This operation does not clear the current self cal data, only that in nonvolatile memory. If this command is sent when no nonvolatile self cal data is present, an error is generated.
Examples	

vtex1048_self_cal_get_status

Purpose	Queries the completion status of self-calibration.
Type	Query
Command Syntax	vtex1048_self_cal_get_status(**vi, &cal_percent**)
Command Parameters	**vi** = session ID **cal_percent** = an integer output value in decimal that represents the percentage completion status of self-calibration.
Reset Value	N/A
Query Response	**cal_percent**: an integer in the range of 0 to 100.
Description	Queries the completion status of self-calibration. **NOTE: Additional instrument driver calls should not be performed until the completion status reaches 100 percent.**
Examples	

vtex1048_self_cal_init

Purpose	Performs an instrument self-calibration.
Type	Event
Command Syntax	vtex1048_self_cal_init(**vi, &override**)
Command Parameters	**vi** = session ID **override** = an integer output value in decimal that represents an override code. If self-calibration is attempted before the EX1048 has been powered on continuously for 60 minutes, an error will be generated, and an integer value will be placed in the **override** variable. If appropriate, resending the command will override the error and initiate self-calibration.
Reset Value	N/A
Query Response	N/A
Description	Performs an instrument self-calibration. In general, self-calibration should not be performed until the EX1048 has been powered on continuously for 60 minutes. In fact, if self-calibration is attempted prior to the required uptime, an error will be generated. When appropriate, as described below, this error can be overridden by resending the command. **NOTE: In order to perform a self-calibration, a lock on the instrument must first be acquired. Attempting to self-calibrate without the acquisition of a lock will generate an error that is not able to be overridden. See the *vtex1048_lock* command.** **NOTE: The self-calibration uptime requirement is in place to protect the measurement integrity of the instrument. Overriding the requirement must only be done when the operating conditions allow it. An example of this is where the unit has actually been warmed up, but has simply been subjected to a quick power cycle or reboot. In order to insure that the override is intentional, it is strongly recommended that user intervention be required in the software application to employ it.** **NOTE: Once self-calibration has been successfully initiated, its percentage completion status is accessible through the *vtex1048_self_cal_get_status* command. Additional instrument driver calls should not be performed until the completion status reaches 100 percent.**
Examples	

vtex1048_self_cal_is_stored

Purpose	Queries the presence of self cal data in nonvolatile memory.
Type	Query
Command Syntax	vtex1048_self_cal_is_stored(**vi, &stored**)
Command Parameters	**vi** = session ID **stored** = a Boolean value indicating whether self cal data is stored in nonvolatile memory.
Reset Value	N/A
Query Response	**stored**: 0 or 1.
Description	Queries the presence of self cal data in nonvolatile memory. Nonvolatile self cal data is automatically loaded and used upon an instrument power cycle or reset. It is stored with the *vtex1048_self_cal_store* command.
Examples	

vtex1048_self_cal_load

Purpose	Loads nonvolatile self cal data as the current self cal data.
Type	Event
Command Syntax	vtex1048_self_cal_load(**vi**)
Command Parameters	**vi** = session ID
Reset Value	N/A
Query Response	N/A
Description	Loads nonvolatile self cal data as the current self cal data. If current self cal data previously existed, it is simply overwritten and need not be cleared in advance. If this command is sent when no nonvolatile self cal data is present, an error is generated.
Examples	

vtex1048_self_cal_store

Purpose	Stores the current self cal data into nonvolatile memory.
Type	Event
Command Syntax	vtex1048_self_cal_store(**vi**)
Command Parameters	**vi** = session ID
Reset Value	N/A
Query Response	N/A
Description	Stores the current self cal data into nonvolatile memory, enabling it to be loaded upon instrument power cycle and reset. If this command is sent when no current self cal data is present, an error is generated. Since the existence of nonvolatile self cal data represents a permanent (although revocable) change from the factory calibration settings, its presence is able to be queried. See the *vtex1048_self_cal_is_stored* query.
Examples	

vtex1048_set_arm_count

Purpose	Sets the arm count value.
Type	Setting
Command Syntax	vtex1048_set_arm_count(**vi**, **count**)
Command Parameters	**vi** = session ID **count** = the value for arm count. Value must be an integer in the range of 1 to (2^{31}-1).
Reset Value	1
Query Response	N/A
Description	Sets the arm count value. This value is reset with each trigger initialize or automatically upon reaching zero when init continuous is enabled.

Examples	// set an arm count of 10
	vtex1048_set_arm_count(vi, 10);

vtex1048_set_arm_delay

Purpose	Sets the arm delay.
Type	Setting
Command Syntax	vtex1048_set_arm_delay(**vi**, **arm_delay**)
Command Parameters	**vi** = session ID **arm_delay** = the delay value (in seconds). Value must be in the range of 0 to 4294 (71.5 minutes) with a resolution of 0.000001 (1 µs).
Reset Value	0
Query Response	N/A
Description	Sets the arm delay, the time between the recognition of the arm event and the transition into the TRIG layer of the trigger model.

Examples	// set an arm delay of 5 ms
	vtex1048_set_arm_delay(vi, 0.005);

vtex1048_set_arm_infinite

Purpose	Enables or disables the use of an infinite arm count.
Type	Setting
Command Syntax	vtex1048_set_arm_infinite(**vi**, **arminf**)
Command Parameters	**vi** = session ID **arminf** = a Boolean value indicating whether to set the arm count to infinite.
Reset Value	0
Query Response	N/A
Description	Enables or disables the use of an infinite arm count. Enabling overrides any manual setting of arm count.
Examples	

vtex1048_set_arm_source

Purpose	Sets the arm source events.
Type	Setting
Command Syntax	vtex1048_set_arm_source(**vi**, **vtb_masks[]**, **dio_masks[]**, **timer_enable**, **immediate**)
Command Parameters	**vi** = session ID **vtb_masks[]** = an array of four 8-bit integer values representing the enabling of arm events from any of the 8 channels of the trigger bus. The order of the values is: positive edge, negative edge, positive level, negative level. The values must be in the range of 0-255 (decimal), 0x00-0xFF (hex). Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0. **dio_masks[]** = an array of four 8-bit integer values representing the enabling of arm events from any of the 8 channels of the digital I/O port. The order of the values is: positive edge, negative edge, positive level, negative level. The values must be in the range of 0-255 (decimal), 0x00-0xFF (hex). Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0. **timer_enable** = a Boolean value indicating whether to enable the timer as an arm event. **immediate** = a Boolean value indicating whether to enable immediate as an arm event.
Reset Value	The arm source is set to immediate.
Query Response	N/A
Description	Sets the arm source events. Regardless of this setting, software arms are always enabled.

Examples	// enable timer arm only
	ViUInt8 vtb_masks[4] = {0,0,0,0};
	ViUInt8 dio_masks[4] = {0,0,0,0};
	vtex1048_set_arm_source(vi, vtb_masks, dio_masks, 1, 0);
	
	// enable arm on a positive level on DIO channels 0-3 and a negative edge on VTB channel 6
	ViUInt8 vtb_masks[4] = {0,64,0,0};
	ViUInt8 dio_masks[4] = {0,0,0x0F,0};
	vtex1048_set_arm_source(vi, vtb_masks, dio_masks, 0, 0);
	
	// enable software arm only
	ViUInt8 vtb_masks[4] = {0,0,0,0};
	ViUInt8 dio_masks[4] = {0,0,0,0};
	vtex1048_set_arm_source(vi, vtb_masks, dio_masks, 0, 0);

vtex1048_set_channel_conversion

Purpose	Sets the engineering units (EU) conversion for the specified channels.
Type	Setting
Command Syntax	vtex1048_set_channel_conversion(**vi**, **channels[]**, **numChannels**, **eu_conv**)
Command Parameters	**vi** = session ID **channels[]** = the list of channels to which to apply the EU conversion selection. The list of channels can include channels not currently in the scan list and can be a subset of the channels in the scan list. **numChannels** = the length of the channels array. Value must be an integer in the range of 1 to 48. **eu_conv** = one of the following integer values:
Reset Value	**eu_conv** = VTEX1048_CONV_MV for all channels
Query Response	N/A
Description	Sets the EU conversion for the specified channels.

Examples	// channels 0-4 are E, channels 5-8 are T
	#define TYPE_E 0x04
	#define TYPE_T 0x03
	ViInt32 e_channels[5] = { 0, 1, 2, 3, 4 };
	vtex1048_set_channel_conversion(vi, e_channels, 5, TYPE_E);
	ViInt32 t_channels[4] = { 5, 6, 7, 8 };
	vtex1048_set_channel_conversion(vi, t_channels, 4, TYPE_T);

Value	Integer Value	Function
VTEX1048_CONV_MV	0	Voltage
VTEX1048_CONV_THERMO_TYPE_J	1	Type J
VTEX1048_CONV_THERMO_TYPE_K	2	Type K
VTEX1048_CONV_THERMO_TYPE_T	3	Type T
VTEX1048_CONV_THERMO_TYPE_E	4	Type E
VTEX1048_CONV_THERMO_TYPE_B	5	Type B
VTEX1048_CONV_THERMO_TYPE_S	6	Type S
VTEX1048_CONV_THERMO_TYPE_R	7	Type R
VTEX1048_CONV_THERMO_TYPE_N	8	Type N
VTEX1048_CONV_USER_DEF0	9	User-defined 0
VTEX1048_CONV_USER_DEF1	10	User-defined 1

vtex1048_set_dio_limit_event

Purpose	Links limit evaluations to the operation of the digital I/O port.
Type	Setting
Command Syntax	vtex1048_set_dio_limit_event(**vi**, **dio_channel**, **limit_masks[]**)
Command Parameters	**vi** = session ID **dio_channel** = the channel of the digital I/O port to be linked to limit evaluations. Value must be an integer in the range of 0 to 7. **limit_masks[]** = an array of 48 4-bit integer values representing, on a per input channel basis, the linking of limit evaluations to any of the 4 limit conditions. Within the 4-bit field, the order of the values is: limit set 0 lower, limit set 0 upper, limit set 1 lower, limit set 1 upper. The values must be in the range of 0-15 (decimal), 0x00-0x0F (hex). Channel 0 through channel 47 are represented in array elements [0] through [47], respectively.
Reset Value	**limit_masks[]** = 0 in all array elements for all DIO channels
Query Response	N/A
Description	Links limit evaluations to the operation of the digital I/O port. In nominal operation, a DIO channel that is linked to an input channel's limit evaluation will transition from low to high whenever the limit is exceeded. Multiple linkages per DIO channel are allowed and are logically OR'ed together. That is, a DIO channel that is linked to four input channel limit evaluations will transition whenever any of the four limits are exceeded. Multiple linkages can be created on the same input channel and/or spanning multiple input channels. The nominal transition of a DIO channel is from low to high whenever the linked limit is exceeded. Optionally, the transition can be specified to be from high to low with the *vtex1048_set_dio_limit_event_invert* command. The nominal operation of a linked DIO channel is to reflect the latest limit evaluation. That is, it will be updated with every scan. Optionally, the behavior can be set to latch mode with the *vtex1048_set_dio_limit_event_latch* command. In latch mode, a transition out of the cleared state would remain, regardless of future limit evaluations, until it is cleared at the beginning of a new acquisition.

Examples	// link DIO channel 7 to set 1 lower limits on ch 12 and 14, set 0 upper limit on ch 1
	ViUInt8 limit_masks[48] =
	{0,4,0,0,0,0,0,0,0,0,0,0,2,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
	
	vtex1048_set_dio_limit_event(vi, 7, limit_masks);

vtex1048_set_dio_limit_event_invert

Purpose	Enables or disables inverted operation of a DIO channel linked as a limit event.
Type	Setting
Command Syntax	vtex1048_set_dio_limit_event_invert(**vi**, **dio_channel**, **invert**)
Command Parameters	**vi** = session ID **dio_channel** = the channel of the digital I/O port to be affected. Value must be an integer in the range of 0 to 7. **invert** = a Boolean value indicating whether to operate in invert mode.
Reset Value	**invert** = 0 for all DIO channels
Query Response	N/A
Description	Enables or disables inverted operation of a DIO channel linked as a limit event. The nominal transition of a DIO channel is from low to high whenever the linked limit is exceeded. In invert mode, it is from high to low.
Examples	

vtex1048_set_dio_limit_event_latch

Purpose	Enables or disables latch operation of a DIO channel linked as a limit event.
Type	Setting
Command Syntax	vtex1048_set_dio_limit_event_latch(**vi, dio_channel, latch**)
Command Parameters	**vi** = session ID **dio_channel** = the channel of the digital I/O port to be affected. Value must be an integer in the range of 0 to 7. **latch** = a Boolean value indicating whether to operate in latch mode.
Reset Value	**latch** = 0 for all DIO channels
Query Response	N/A
Description	Enables or disables latch operation of a DIO channel linked as a limit event. The nominal operation of a linked DIO channel is to reflect the latest limit evaluation. That is, it will be updated with every scan. In latch mode, a transition out of the cleared state would remain, regardless of future limit evaluations, until it is cleared at the beginning of a new acquisition.
Examples	

vtex1048_set_dio_output

Purpose	Sets the static level that each channel of the digital I/O port will assume if enabled.
Type	Setting
Command Syntax	vtex1048_set_dio_output(**vi**, **dio_out**)
Command Parameters	**vi** = session ID **dio_out** = the value that represents the desired state of the 8-bit port. The value must be an integer in the range of 0-255 (decimal), 0x00-0xFF (hex). Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0.
Reset Value	0
Query Response	N/A
Description	Sets the static level that each channel of the digital I/O port will assume if enabled. Enabling is done with the *vtex1048_set_dio_output_enable* command.

Examples	// set DIO bit 7 (high) and DIO bit 6 (low)
	vtex1048_set_dio_output(vi, 0x80);
	// enable them as outputs
	vtex1048_set_dio_output_enable(vi, 0xC0);

vtex1048_set_dio_output_enable

Purpose	Enables or disables the output functionality of each channel of the digital I/O port.
Type	Setting
Command Syntax	vtex1048_set_dio_output_enable(**vi**, **out_enable**)
Command Parameters	**vi** = session ID **out_enable** = the value that represents the desired output enable state of the 8-bit port. The value must be an integer in the range of 0-255 (decimal), 0x00-0xFF (hex). Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0.
Reset Value	0
Query Response	N/A
Description	Enables or disables the output functionality of each channel of the digital I/O port. Input functionality on each channel is constant regardless of its output functionality.

Examples	// set DIO bit 7 (high) and DIO bit 6 (low)
	vtex1048_set_dio_output(vi, 0x80);
	// enable them as outputs
	vtex1048_set_dio_output_enable(vi, 0xC0);

vtex1048_set_dio_pulse

Purpose	Generates a 1 µs pulse on selected channels of the digital I/O port.
Type	Event
Command Syntax	vtex1048_set_dio_pulse(**vi**, **dio_pulse**)
Command Parameters	**vi** = session ID **dio_pulse** = the value that represents the channels to be pulsed within the 8-bit port. The value must be an integer in the range of 0-255 (decimal), 0x00-0xFF (hex). Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0.
Reset Value	N/A
Query Response	N/A
Description	Generates a 1 µs pulse on selected channels of the digital I/O port. The pulse will occur only if the selected channels are enabled as outputs. When a channel is programmed with a static level of high, the pulse will be low-going. When a channel is programmed with a static level of low, the pulse will be high-going.
Examples	

	// set DIO bit 7 low and then pulse high
	vtex1048_set_dio_output(vi, 0x00);
	vtex1048_set_dio_output_enable(vi, 0x80);
	vtex1048_set_dio_pulse(vi, 0x80);

vtex1048_set_fifo_config

Purpose	Sets the data format and overflow behavior of the FIFO memory.
Type	Setting
Command Syntax	vtex1048_set_fifo_config(**vi, report_cjc, report_timestamp, report_celsius, blocking_mode**)
Command Parameters	**vi** = session ID **report_cjc** = a Boolean value indicating whether to report the CJC temperatures in the acquisition data. This control does not affect their measurement, only the display of the data. The CJC temperatures are taken with every scan, regardless of this setting. Reported CJC temperatures are always in units of ºC, regardless of the value of the **report_celsius** parameter. **report_timestamp** = a Boolean value indicating whether to report delta timestamps per channel in the acquisition data. The timestamp represents the time, in increments of 100 ns, between the specific channel measurement and the scan initiation time. **report_celsius** = a Boolean value indicating whether to report the input channel data in units of ºC (true) or ºF (false). **blocking_mode** = a Boolean value that governs the system behavior if the reading buffer fills to its maximum capacity during scanning. With blocking mode enabled, additional readings are discarded, leaving the contents of the buffer intact. With blocking mode disabled, the buffer becomes circular, in that additional readings overwrite the oldest readings.
Reset Value	**report_cjc** = 0 **report_timestamp** = 0 **report_celsius** = 1 **blocking_mode** = 0
Query Response	N/A
Description	Sets the data format and overflow behavior of the FIFO memory. **NOTE: Regardless of the settings of the report_cjc and report_timestamp parameters, the CJC temperature and delta timestamp information is not accessible through the *vtex1048_read_fifo* command. They are available through the streaming interface, and the parameter settings apply for it.**
Examples	

vtex1048_set_filt_freq

Purpose	Sets the hardware filter frequency for the specified channels.
Type	Setting
Command Syntax	vtex1048_set_filt_freq(**vi**, **channels[]**, **numChannels**, **filt_freq**)
Command Parameters	**vi** = session ID **channels[]** = the list of channels to which to apply the filter frequency selection. The list of channels can include channels not currently in the scan list and can be a subset of the channels in the scan list. **numChannels** = the length of the channels array. Value must be an integer in the range of 1 to 48. **filt_freq** = 4.0 or 1000.0 (value in Hz)
Reset Value	**filt_freq** = 4.0 Hz for all channels
Query Response	N/A
Description	Sets the hardware filter frequency for the specified channels.

Examples	// set channels 0-4 for 4 Hz
	ViInt32 low_channels[5] = { 0, 1, 2, 3, 4 };
	vtex1048_set_filt_freq(vi, low_channels, 5, 4.0);
	
	// set channels 5-8 for 1 kHz
	ViInt32 high_channels[4] = { 5, 6, 7, 8 };
	vtex1048_set_filt_freq(vi, high_channels, 4, 1000.0);

vtex1048_set_init_cont

Purpose	Enables or disables the use of init continuous mode.
Type	Setting
Command Syntax	vtex1048_set_init_cont(**vi**, **init_cont_mode**)
Command Parameters	**vi** = session ID **init_cont_mode** = a Boolean value indicating whether to enable init continuous mode.
Reset Value	0
Query Response	N/A
Description	Enables or disables the use of init continuous mode. Init continuous returns the trigger model to the entrance of the ARM layer without the requirement of a new trigger initialize command.
Examples	

vtex1048_set_limit_set0

Purpose	Sets the limit set 0 values manually for the specified channels.
Type	Setting
Command Syntax	vtex1048_set_limit_set0(**vi**, **channels[]**, **numChannels, lower_limit, upper_limit**)
Command Parameters	**vi** = session ID **channels[]** = the list of channels to which to apply the limit values. The list of channels can include channels not currently in the scan list and can be a subset of the channels in the scan list. **numChannels** = the length of the channels array. Value must be an integer in the range of 1 to 48. **lower_limit** = the value for the lower reading limit. The limit will be tripped if the applicable channel reading is less than the lower reading limit. For proper operation, the limit value must be entered in the same units as the channels with which it is associated. **upper_limit** = the value for the upper reading limit. The limit will be tripped if the applicable channel reading is greater than the upper reading limit. For proper operation, the limit value must be entered in the same units as the channels with which it is associated.
Reset Value	**lower_limit** = -6.6e-2 for all channels **upper_limit** = 6.6e-2 for all channels
Query Response	N/A
Description	Sets the limit set 0 values manually for the specified channels. A channel's limit values can be set regardless of its inclusion in the scan list, and multiple channels can be assigned to the same limit values within one command. However, each unique combination of limit values must be set with a separate command. By default, the limit values for limit set 0 are set automatically, based on the EU conversion and units selection for each channel. If manual limit control has been enabled with the *vtex1048_set_limit_set0_manual* command, user defined limit values can be entered. If manual limit control is not enabled, execution of this command will not generate an error, but the specified limit values will be ignored. For proper operation, limit values must be entered and maintained in the same units as the channels with which they are associated. Once entered, manual limit values are not automatically converted by subsequent changes in EU conversion or units designations.

Examples	// set channels 0-4 to manual limits of 0 and 100
	ViInt32 e_channels[5] = { 0, 1, 2, 3, 4 };
	vtex1048_set_limit_set0_manual(vi, e_channels, 5, 1);
	vtex1048_set_limit_set0(vi, e_channels, 5, 0, 100);

vtex1048_set_limit_set0_manual

Purpose	Enables or disables manual entry of limit set 0 values for the specified channels.
Type	Setting
Command Syntax	vtex1048_set_limit_set0_manual(**vi**, **channels[]**, **numChannels, manual**)
Command Parameters	**vi** = session ID **channels[]** = the list of channels for which to enable manual limit value entry. The list of channels can include channels not currently in the scan list and can be a subset of the channels in the scan list. **numChannels** = the length of the channels array. Value must be an integer in the range of 1 to 48. **manual** = a Boolean value indicating whether to enable manual limit value entry.
Reset Value	0 for all channels
Query Response	N/A
Description	Enables or disables manual entry of limit set 0 values for the specified channels. A channel's manual entry control can be set regardless of its inclusion in the scan list, and multiple channels can be configured within one command. However, each unique control value must be set with a separate command. By default, the limit values for limit set 0 are set automatically, based on the EU conversion and units selection for each channel. If manual limit control is desired, it must first be enabled with this command. Disabling manual limit control on a channel that had been enabled will automatically set the limit set 0 values for that channel, based on its current EU conversion and units selection.
Examples	

vtex1048_set_limit_set1

Purpose	Sets the limit set 1 values for the specified channels.
Type	Setting
Command Syntax	vtex1048_set_limit_set1(**vi**, **channels[]**, **numChannels, lower_limit, upper_limit**)
Command Parameters	**vi** = session ID **channels[]** = the list of channels to which to apply the limit values. The list of channels can include channels not currently in the scan list and can be a subset of the channels in the scan list. **numChannels** = the length of the channels array. Value must be an integer in the range of 1 to 48. **lower_limit** = the value for the lower reading limit. The limit will be tripped if the applicable channel reading is less than the lower reading limit. For proper operation, the limit value must be entered in the same units as the channels with which it is associated. **upper_limit** = the value for the upper reading limit. The limit will be tripped if the applicable channel reading is greater than the upper reading limit. For proper operation, the limit value must be entered in the same units as the channels with which it is associated.
Reset Value	**lower_limit** = -3e+38 for all channels **upper_limit** = 3e+38 for all channels
Query Response	N/A
Description	Sets the limit set 1 values for the specified channels. A channel's limit values can be set regardless of its inclusion in the scan list, and multiple channels can be assigned to the same limit values within one command. However, each unique combination of limit values must be set with a separate command. Unlike limit set 0, the limit values for limit set 1 are always set manually. For proper operation, limit values must be entered and maintained in the same units as the channels with which they are associated. Once entered, limit values are not automatically converted by subsequent changes in EU conversion or units designations.
Examples	

vtex1048_set_scanlist

Purpose	Sets the scan list to be acquired.
Type	Setting
Command Syntax	vtex1048_set_scanlist(**vi**, **channels[]**, **numChannels**)
Command Parameters	**vi** = session ID **channels[]** = the list of channels to include in the scan list. Elements must be unique integers in the range of 0 to 47. The order of the elements will correspond to the order that the data is returned. **numChannels** = the length of the channels array. Value must be an integer in the range of 1 to 48.
Reset Value	**channels[0]** = 0 **numChannels** = 1
Query Response	N/A
Description	Sets the scan list to be acquired. A valid scan list consists of: • at least one channel • no more than 48 channels • no repeated channels

Examples	// sequential order five-channel scan
	ViInt32 channels[5] = { 0, 1, 2, 3, 4 };
	vtex1048_set_scanlist(vi, channels, 5);
	
	// reverse order five-channel scan
	ViInt32 channels[5] = { 4, 3, 2, 1, 0 };
	vtex1048_set_scanlist(vi, channels, 5);

vtex1048_set_trig_source_timer

Purpose	Sets a trigger source of timer only and sets the timer interval.
Type	Setting
Command Syntax	vtex1048_set_trig_source_timer(**vi**, **trig_timer**)
Command Parameters	**vi** = session ID **trig_timer** = the time value (in seconds) between timer ticks. Value must be in the range of 0.001 (1 ms) to 4294 (71.5 minutes) with a resolution of 0.000001 (1 µs).
Reset Value	N/A
Query Response	N/A
Description	Sets a trigger source of timer only and sets the timer interval. This is a one-command combination of the *vtex1048_set_trigger_source* and *vtex1048_set_trigger_timer* commands.
Examples	

vtex1048_set_trigger_count

Purpose	Sets the trigger count value.
Type	Setting
Command Syntax	vtex1048_set_trigger_count(**vi**, **count**)
Command Parameters	**vi** = session ID **count** = the value for trigger count. Value must be an integer in the range of 1 to (2^{31}-1).
Reset Value	1
Query Response	N/A
Description	Sets the trigger count value. This value is reset with each arm event.

Examples	// set a trigger count of 10
	vtex1048_set_trigger_count(vi, 10);

vtex1048_set_trigger_delay

Purpose	Sets the trigger delay.
Type	Setting
Command Syntax	vtex1048_set_trigger_delay(**vi**, **trig_delay**)
Command Parameters	**vi** = session ID **trig_delay** = the delay value (in seconds). Value must be in the range of 0 to 4294 (71.5 minutes) with a resolution of 0.000001 (1 µs).
Reset Value	0
Query Response	N/A
Description	Sets the trigger delay, the time between the recognition of the trigger event and the execution of the scan list.

Examples	// set a trigger delay of 5 ms
	vtex1048_set_trigger_delay(vi, 0.005);

vtex1048_set_trigger_infinite

Purpose	Enables or disables the use of an infinite trigger count.
Type	Setting
Command Syntax	vtex1048_set_trigger_infinite(**vi**, **triginf**)
Command Parameters	**vi** = session ID **triginf** = a Boolean value indicating whether to set the trigger count to infinite.
Reset Value	0
Query Response	N/A
Description	Enables or disables the use of an infinite trigger count. Enabling overrides any manual setting of trigger count.
Examples	

vtex1048_set_trigger_source

Purpose	Sets the trigger source events.
Type	Setting
Command Syntax	vtex1048_set_trigger_source(**vi**, **vtb_masks[]**, **dio_masks[]**, **timer_enable, immediate**)
Command Parameters	**vi** = session ID **vtb_masks[]** = an array of four 8-bit integer values representing the enabling of trigger events from any of the 8 channels of the trigger bus. The order of the values is: positive edge, negative edge, positive level, negative level. The values must be in the range of 0-255 (decimal), 0x00-0xFF (hex). Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0. **dio_masks[]** = an array of four 8-bit integer values representing the enabling of trigger events from any of the 8 channels of the digital I/O port. The order of the values is: positive edge, negative edge, positive level, negative level. The values must be in the range of 0-255 (decimal), 0x00-0xFF (hex). Within the 8-bit field, the MSB corresponds to DIO channel 7, and the LSB corresponds to DIO channel 0. **timer_enable** = a Boolean value indicating whether to enable the timer as a trigger event. **immediate** = a Boolean value indicating whether to enable immediate as a trigger event.
Reset Value	The trigger source is set to timer.
Query Response	N/A
Description	Sets the trigger source events. Regardless of this setting, software triggers are always enabled.

Examples	// enable timer trigger only
	ViUInt8 vtb_masks[4] = {0,0,0,0};
	ViUInt8 dio_masks[4] = {0,0,0,0};
	vtex1048_set_trigger_source(vi, vtb_masks, dio_masks, 1, 0);
	
	// enable trigger on a positive level on DIO channels 0-3 and a negative edge on VTB channel 6
	ViUInt8 vtb_masks[4] = {0,64,0,0};
	ViUInt8 dio_masks[4] = {0,0,0x0F,0};
	vtex1048_set_trigger_source(vi, vtb_masks, dio_masks, 0, 0);
	
	// enable software trigger only
	ViUInt8 vtb_masks[4] = {0,0,0,0};
	ViUInt8 dio_masks[4] = {0,0,0,0};
	vtex1048_set_trigger_source(vi, vtb_masks, dio_masks, 0, 0);

vtex1048_set_trigger_timer

Purpose	Sets the timer interval for the timer source event.
Type	Setting
Command Syntax	vtex1048_set_trigger_timer(**vi**, **trig_timer**)
Command Parameters	**vi** = session ID **trig_timer** = the time value (in seconds) between timer ticks. Value must be in the range of 0.001 (1 ms) to 4294 (71.5 minutes) with a resolution of 0.000001 (1 µs).
Reset Value	0.1 (100 ms)
Query Response	N/A
Description	Sets the timer interval for the timer source event. The same value is used for both arm and trigger events.

Examples	// set a timer interval of 50 ms
	vtex1048_set_trigger_timer(vi, 0.05);

vtex1048_set_user_cjc_enable

Purpose	Enables or disables the use of a user-defined CJC temperature for the specified channels.
Type	Setting
Command Syntax	vtex1048_set_user_cjc_enable(**vi**, **channels[]**, **numChannels, usercjc**)
Command Parameters	**vi** = session ID **channels[]** = the list of channels to which to apply the selection. The list of channels can include channels not currently in the scan list and can be a subset of the channels in the scan list. **numChannels** = the length of the channels array. Value must be an integer in the range of 1 to 48. **usercjc** = a Boolean value indicating whether to enable the user-defined CJC temperature.
Reset Value	**usercjc** = 0 for all channels
Query Response	N/A
Description	Enables or disables the use of a user-defined CJC temperature for the specified channels. If enabled, the user-defined CJC temperature will be used in the thermocouple calculations instead of the internally measured one. Each channel can be associated with a unique value and be independently enabled with regards to its use. The entry of external CJC values and their enabling are disjoint functions. That is, the entry of a value does not automatically enable its use, and the disabling of a previously enabled channel does not clear the value. Entry of the CJC temperature is done through the *vtex1048_set_user_cjc_temp* command. **NOTE: This command should only be used when the thermocouple cold junction is made external to the EX1048. This feature must be used with care, as the input channel data contains no indication that it was calculated with an external CJC value instead of an internal one.**

Examples	// CJC for channels 0-4 are held externally at 0.2 C
	ViInt32 ext_channels[5] = { 0, 1, 2, 3, 4 };
	vtex1048_set_user_cjc_temp(vi, ext_channels, 5, 0.2);
	vtex1048_set_user_cjc_enable(vi, ext_channels, 5, 1);

vtex1048_set_user_cjc_temp

Purpose	Sets the user-defined CJC temperature for the specified channels.
Type	Setting
Command Syntax	vtex1048_set_user_cjc_temp(**vi**, **channels[]**, **numChannels, cjc_temp**)
Command Parameters	**vi** = session ID **channels[]** = the list of channels to which to apply the CJC temperature. The list of channels can include channels not currently in the scan list and can be a subset of the channels in the scan list. **numChannels** = the length of the channels array. Value must be an integer in the range of 1 to 48. **cjc_temp** = the value for the CJC temperature. The value is entered with units of ºC, regardless of the units selection of the input channels.
Reset Value	**cjc_temp** = 0 for all channels
Query Response	N/A
Description	Sets the user-defined CJC temperature for the specified channels. If enabled, this entered temperature will be used in the thermocouple calculations instead of the internally measured one. Each channel can be associated with a unique value and be independently enabled with regards to its use. The entry of external CJC values and their enabling are disjoint functions. That is, the entry of a value does not automatically enable its use, and the disabling of a previously enabled channel does not clear the value. Enabling is done through the *vtex1048_set_user_cjc_enable* command. **NOTE: This command should only be used when the thermocouple cold junction is made external to the EX1048. This feature must be used with care, as the input channel data contains no indication that it was calculated with an external CJC value instead of an internal one.**

Examples	// CJC for channels 0-4 are held externally at 0.2 C
	ViInt32 ext_channels[5] = { 0, 1, 2, 3, 4 };
	vtex1048_set_user_cjc_temp(vi, ext_channels, 5, 0.2);
	vtex1048_set_user_cjc_enable(vi, ext_channels, 5, 1);

vtex1048_set_user_conversion

Purpose	Sets the user-defined conversion polynomials.
Type	Setting
Command Syntax	vtex1048_set_user_conversion(**vi, eu_conv, fwdcoeff[], numFwd, invcoeff[], numInv**)
Command Parameters	**vi** = session ID **eu_conv** = the polynomial set to be defined. Value must be an integer equal to 9 (User0) or 10 (User1). **fwdcoeff[]** = an array of forward conversion polynomial coefficients. Coefficients c_0 through c_{12} are represented in array elements [0] through [12], respectively. **numFwd** = the length of the forward coefficients array. Value must be an integer in the range of 1 to 13. **invcoeff[]** = an array of inverse conversion polynomial coefficients. Coefficients d_0 through d_{12} are represented in array elements [0] through [12], respectively. **numInv** = the length of the inverse coefficients array. Value must be an integer in the range of 1 to 13.
Reset Value	**fwdcoeff[]** = 0 in all array elements for both polynomial sets **invcoeff[]** = 0 in all array elements for both polynomial sets
Query Response	N/A
Description	Sets the user-defined conversion polynomials. The forward conversion polynomial is used to convert a CJC temperature into a compensating cold junction voltage and has the form of: $$E = c_0 + c_1 * t^1 + c_2 * t^2 + \ldots + c_{12} * t^{12}$$ where E is in volts, t is in ºC, and $c_o - c_{12}$ are the coefficients. The inverse conversion polynomial is used to convert a compensated input voltage into temperature and has the form of: $$t = d_0 + d_1 * E^1 + d_2 * E^2 + \ldots + d_{12} * E^{12}$$ where E is in volts, t is in ºC, and $d_o - d_{12}$ are the coefficients. Undefined coefficients are automatically set to 0. **NOTE: The entry of user-defined coefficients does not automatically enable their use. The enabling is done by setting the EU conversion to User0 or User1 through the *vtex1048_set_channel_conversion* command.**
Examples	

vtex1048_set_vtb_output

Purpose	Sets the static level that each channel of the trigger bus will assume if enabled.
Type	Setting
Command Syntax	vtex1048_set_vtb_output(**vi**, **vtb_out**)
Command Parameters	**vi** = session ID **vtb_out** = the value that represents the desired state of the 8-bit port. The value must be an integer in the range of 0-255 (decimal), 0x00-0xFF (hex). Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0.
Reset Value	0
Query Response	N/A
Description	Sets the static level that each channel of the trigger bus will assume if enabled. Enabling is done with the *vtex1048_set_vtb_output_enable* command.

Examples	// set VTB bit 7 (high) and VTB bit 6 (low)
	vtex1048_set_vtb_output(vi, 0x80);
	// enable them as outputs
	vtex1048_set_vtb_output_enable(vi, 0xC0);

vtex1048_set_vtb_output_enable

Purpose	Enables or disables the output functionality of each channel of the trigger bus.
Type	Setting
Command Syntax	vtex1048_set_vtb_output_enable(**vi**, **out_enable**)
Command Parameters	**vi** = session ID **out_enable** = the value that represents the desired output enable state of the 8-bit port. The value must an integer be in the range of 0-255 (decimal), 0x00-0xFF (hex). Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0.
Reset Value	0
Query Response	N/A
Description	Enables or disables the output functionality of each channel of the trigger bus. Input functionality on each channel is constant regardless of its output functionality.

Examples	// set VTB bit 7 (high) and VTB bit 6 (low)
	vtex1048_set_vtb_output(vi, 0x80);
	// enable them as outputs
	vtex1048_set_vtb_output_enable(vi, 0xC0);

vtex1048_set_vtb_pulse

Purpose	Generates a 1 µs pulse on selected channels of the trigger bus.
Type	Event
Command Syntax	vtex1048_set_vtb_pulse(**vi**, **vtb_pulse**)
Command Parameters	**vi** = session ID **vtb_pulse** = the value that represents the channels to be pulsed within the 8-bit port. The value must be an integer in the range of 0-255 (decimal), 0x00-0xFF (hex). Within the 8-bit field, the MSB corresponds to VTB channel 7, and the LSB corresponds to VTB channel 0.
Reset Value	N/A
Query Response	N/A
Description	Generates a 1 µs pulse on selected channels of the trigger bus. The pulse will occur only if the selected channels are enabled as outputs. When a channel is programmed with a static level of high, the pulse will be low-going. When a channel is programmed with a static level of low, the pulse will be high-going.

Examples	// set VTB bit 7 low and then pulse high
	vtex1048_set_vtb_output(vi, 0x00);
	vtex1048_set_vtb_output_enable(vi, 0x80);
	vtex1048_set_vtb_pulse(vi, 0x80);

vtex1048_soft_arm

Purpose	Performs a software arm.
Type	Event
Command Syntax	vtex1048_soft_arm(**vi**)
Command Parameters	**vi** = session ID
Reset Value	N/A
Query Response	N/A
Description	Performs a software arm. Software arms are always enabled, regardless of the state of the arm source.
Examples	

vtex1048_soft_trigger

Purpose	Performs a software trigger.
Type	Event
Command Syntax	vtex1048_soft_trigger(**vi**)
Command Parameters	**vi** = session ID
Reset Value	N/A
Query Response	N/A
Description	Performs a software trigger. Software triggers are always enabled, regardless of the state of the trigger source.
Examples	

vtex1048_unlock

Purpose	Releases a lock on the instrument.
Type	Event
Command Syntax	vtex1048_unlock(**vi**)
Command Parameters	**vi** = session ID
Reset Value	N/A
Query Response	N/A
Description	Releases a lock on the instrument. It can be successfully executed only by the host IP address that originally acquired the lock. In order to break a lock owned by another user, the *vtex1048_break_lock* command must be used. As the lock status of the instrument is unaffected by the *vtex1048_reset* command, it cannot be used to release a lock.
Examples	

SECTION 8

THEORY OF OPERATION

INTRODUCTION

The block diagram in Figure 8-1 illustrates the key components of the analog circuitry. Each of the main blocks is described below.

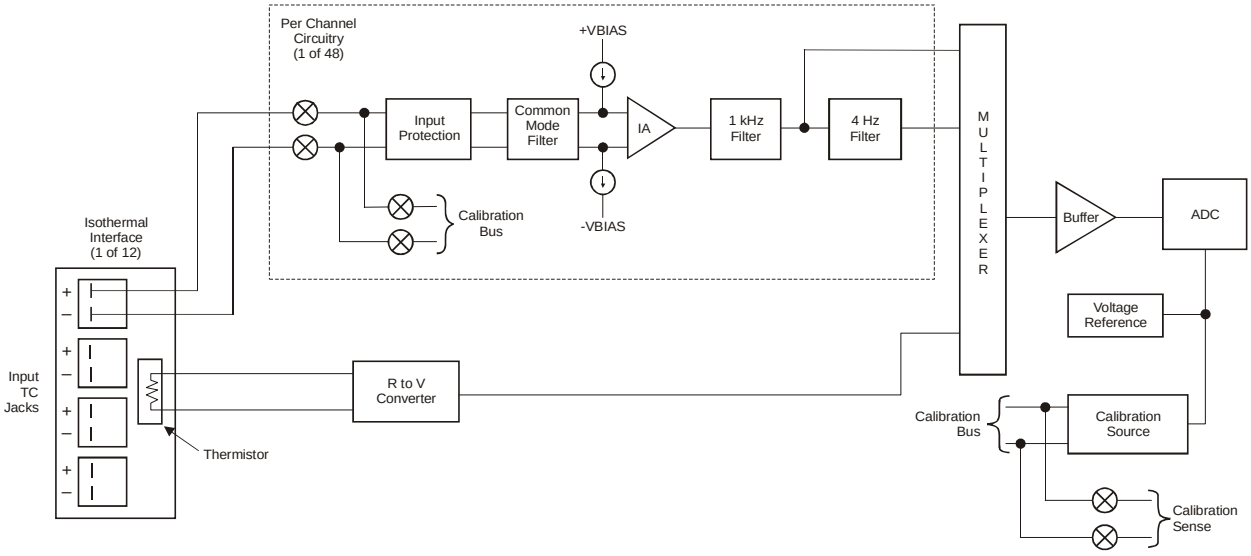

FIGURE 8-1: EX1048 ANALOG CIRCUITRY BLOCK DIAGRAM

SIGNAL CONDITIONING CIRCUITRY

Each input channel is conditioned by the block labeled “Per Channel Circuitry.” The core of this block is the instrumentation amplifier (IA) that provides amplification of the differential input signal and rejection of any common mode input signal. The IA provides excellent CMRR, especially at dc and (50/60) Hz, but its rejection characteristic decreases as the interference frequency increases. Moreover, IAs have the tendency to shift their dc offset in the presence of very high frequency signals. This effect would be particularly problematic, as the filters that follow the IA would not attenuate it. However, the presence of the common mode filter in front of the IA attenuates high frequency interference before it reaches the IA. This decreases the possibility of dc rectification and provides the system with excellent CMRR characteristics at all frequencies.

Open thermocouple detection is provided via nanoamp-level current sources on each input path. These current sources are small enough not to affect measurement accuracy, but large enough to drive the high impedance input of the IA deterministically into saturation in the event of an open condition. As these current sources are continually connected to the input path, they provide continuous monitoring of the input and will generate an open indication even when the open is intermittent in nature. Moreover, current sources are provided on both inputs so that an open condition is registered even in the case where only one lead of the input is open and the other is connected to earth ground.

For optimum noise performance, the IA output is routed through two analog filters. The user then selects, on a per channel basis, which filter output to measure. Having two filter selections is a powerful application tool, as it allows the bandwidth of the signal conditioning path to be matched to the characteristics of the attached sensor. Of particular note, the filters used are two discrete fixed filters that are continuously connected. Contrasting approaches utilize one filter whose characteristics are modified based on user selection. The disadvantage to that approach is that there is an inherent latency when the filter changes state before the data is valid. This requires the user to impose a potentially long delay in his measurement system. The two-filter approach used in the EX1048 has no such characteristic and requires no delay.

COLD JUNCTION COMPENSATION

Cold junction compensation is accomplished via twelve precision thermistors that each monitor the isothermal interface of four input connectors. A thermistor is an element whose resistance represents its applied temperature. Each thermistor resistance is converted to a representative voltage that is measured at the beginning of each scan. Each voltage is then transformed into a temperature value through the application of the thermistor's Steinhart-Hart coefficients. This temperature value is then used to generate a thermocouple type-specific compensating voltage that is mathematically added to the measured input voltage of the appropriate channels.

CALIBRATION

Calibration in the EX1048 takes two forms: full calibration and self-calibration. In both cases, the input signal conditioning paths are disconnected from the input jacks and connected instead to a calibration bus that is driven by an internal calibration source. Through measurement of the conditioning paths at multiple calibration source points, the voltage gain and offset of each path is calculated. Full calibration involves the additional steps of measuring the calibration source with a precision voltmeter. Thus, self-calibration is a subset of full calibration. Because of the internal input disconnection mechanism, the user does not have to remove the actual input connections to perform calibration.

THERMOCOUPLE CALCULATIONS

There are two thermocouple type-specific calculations that are performed in the EX1048. The first transforms the CJC temperature into a compensating voltage that is mathematically added to the measured input voltage. The second transforms this total voltage into its final thermocouple temperature. For maximum accuracy, both of these calculations are performed using the full-order polynomial equations and coefficients from the NIST ITS-90 Thermocouple Database, not from lookup tables or piecewise linear approximations.

INDEX

V

W