EX1200-1538

MULTIFUNCTION COUNTER/TIMER

USER'S MANUAL

P/N: 82-0127-003
Released May 4, 2011

VTI Instruments Corp.

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

TABLE OF CONTENTS

Table of Contents....2
Certification....4
Warranty....4
Limitation of Warranty....4
Trademarks....4
Restricted Rights Legend....4
GENERAL SAFETY INSTRUCTIONS....5
Terms and Symbols....5
Warnings....5
SUPPORT RESOURCES....7
SECTION 1....9
INTRODUCTION....9
Overview....9
Features....10
EX1200-1538 Specifications....11
Frequency Measurement Using Digital Input....13
Frequency Measurement Using Analog Input....13
Period Measurement Using Digital Input....13
Period Measurement Using Analog Input....13
Pulse Width Measurement Accuracy Using Digital and Analog Inputs....14
Duty Cycle Measurement....14
Time Interval Measurement Accuracy Using Analog and Digital Inputs....14
SECTION 2....15
USING THE INSTRUMENT....15
Unpacking....15
Determine System Power Requirements....15
Plug-in Module Installation....15
Warm-up Time....16
Connector Pin/Signal Assignment....16
Front Panel Connector Pins Description....17
EX1200-TB104P-1 Terminal Block....18
Terminal Block Receiver....18
Calibration....19
SECTION 3....21
COUNTER/TIMER OPERATION....21
Overview....21
Inputs....22
Input Coupling....22
Signal Conversion....23
Polarity Conversion....24
Functions....24
Totalizing....24
Edge Counting....25
Period Measurements....25
Pulse Width Measurements....27
Duty Cycle Measurement....28
Frequency Measurement....28
RPM Measurement....29
Time Interval Measurement....30
Phase Measurements....31
Quadrature Measurements....31

SECTION 4 ..... **33**
DIGITAL I/O AND ANALOG OUTPUT OPERATION 33
Digital I/O Operation 33
Analog Output Operation 34
Static Update Mode 34
Dynamic Update Mode 34
Parallel Operation 35
Circuit Protection 35
SECTION 5 ..... **37**
PROGRAMMING THE INSTRUMENT 37
Related Software Components 37
Using the Driver 37
Initializing\Closing the Instrument 37
Option Strings 38
Totalize Function 39
Edge Counting Function 42
Frequency Function 42
RPM Function 45
Time Interval Function 48
Phase Difference Function 51
Quadrature Encoder Function 51
Digital I/O Function 54
Analog Output Function 55
SECTION 6 ..... **57**
SFP OPERATION 57
Introduction 57
General Web Page Operation 58
VTI Instruments Logo 59
EX1200-1538 Soft Front Panel 60
Counter Control Page 61
DIO Control Page 63
DAC Control Page 64
Monitor Page 65
LED Panel 65
DIO Status Section 65
DAC Status Section 65
Data Log Table 66
Data Acquisition Section 66
Lock/Unlock Button 66
Reset Button 66
Device Information Page 67
INDEX **69**

CERTIFICATION

VTI Instruments Corp. (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 VTI Instruments 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.

VTI Instruments Corp. shall not be liable for injury to property other than the goods themselves. Other than the limited warranty stated above, VTI Instruments Corp. 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.

TRADEMARKS

Java Runtime Environment™ are trademarks or registered trademarks of Sun Microsystems, Inc. or its subsidiaries in the United States and other countries. LabVIEW™ and LabWindows/CVI™ are trademarks of National Instruments Corporation. Visual Basic®, Windows®, and Internet Explorer® are registered trademarks of the Microsoft Corporation or its subsidiaries. Linux® is a registered trademark of the Linux Foundation. IVI™ is a trademark of the IVI Foundation. Bonjour™ is a trademark of Apple, Inc.

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.

VTI Instruments Corp.
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. Note that this product contains no user serviceable parts or spare parts.

Service should only be performed by qualified personnel. Disconnect all power before servicing.

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.

The mains outlet that is used to power the equipment must be within 3 meters of the device and shall be easily accessible.

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 VTI Instruments customer support centers.

VTI Instruments Corp.
World Headquarters

VTI Instruments Corp.
2031 Main Street
Irvine, CA 92614-6509

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

VTI Instruments
Cleveland Instrument Division

5425 Warner Road
Suite 13
Valley View, OH 44125

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

VTI Instruments
Lake Stevens Instrument Division

3216 Wetmore Avenue, Suite 1
Everett, WA 98201

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

VTI Instruments, Pvt. Ltd.
Bangalore Instrument Division

135, II & III Floors
Infantry Road
Bangalore – 560 001
India

Phone: +91 80 4040 7900
Phone: +91 80 4162 0200
Fax: +91 80 4170 0200

Technical Support

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

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

SECTION 1

INTRODUCTION

OVERVIEW

The EX1200-1538 is a high-performance multifunction card designed to provide frequency measurement, digital I/O (DIO), and digital-to-analog conversion (DAC) output capability on a single card. This allows users the ability to accommodate a wide range of mixed signals into a standard EX1200 series mainframe. Combining the EX1200-1538 with other cards/instruments in the EX1200 series allows for the creation of a complete measurement system in as small as a 1U rack space.

The EX1200-1538 provides eight channels of independent 32-bit counters, sixteen channels of isolated DIO, and two DAC channels with isolated analog output. While the refined electronic counter functions enhance the accuracy of time and frequency domain measurements, configurable DIO and analog output channels offer flexibility to measure and control various industrial systems. A wide range of measurement functions make this card suitable for both electronic functional test using ATE, as well as precision data acquisition applications. The electronic counter utilizes a high-stability (1 ppm) 50 MHz, TCXO base clock oscillator, along with a reciprocal counting method, to achieve a wide frequency measurement ranges spanning from 0.05 Hz to 1 MHz.

The following functionality is provided by the EX1200-1538:

- Frequency/Counter
  - Frequency measurements
  - RPM measurements
  - Pulse width measurements
  - Edge count/totalize functions
  - Duty cycle measurement
- Frequency/Counter (2-channel measurement)
  - Time interval measurement
  - Phase difference measurement
  - Quadrature measurements
- Digital I/O
  - Configurable direction per channel
  - Read and write discrete channels directly
  - Isolated inputs/outputs
- Analog Outputs (DAC)
  - Programmable, 16-bit DAC
  - Isolated outputs
  - Frequency to voltage/current mode

The electronic 32-bit counter measures the time and frequency domain parameters of repetitive and non-repetitive waveforms. The reciprocal counting technique used ensures high resolution and accuracy even when the input signals are low frequency and not synchronized with the aperture window. Counter channels accept both analog and digital inputs. The analog counter channels accept inputs up to ±48 V true differential voltages making it suitable to use with almost any real-

world signal without the need for external signal conditioners. Programmable hysteresis and threshold levels over the entire input voltage range help to extract the fundamental frequency from even the noisiest of analog input signals. Electronic counter channels can directly measure the RPM of tooth wheel and similar sensors. The EX1200-1538 provides a unique functionality that prevents the frequency bumps caused by the missing/extra tooth used for marking a tooth wheel sensor's reference. Additionally, counter channels can measure position and speed from Quadrature encoder signal pairs, including index channel (A, B, and Z).

The onboard memory of EX1200-1538 can store up to 256,000 measurement readings and supports the unified EX1200 triggering system. By utilizing IEEE 1588 time stamps, data samples can be easily correlated with other systems. Measurements can be paced at a constant rate so that time differential parameters, such as acceleration, can be calculated.

The EX1200-1538 DIO channels can be configured as inputs or outputs on a per channel basis. Each channel is isolated from the other and accepts voltages between 2.5V and 60 V. The output channels use solid-state switches that work in any polarity. Setting the output logic levels and reading the input logic states are fully programmable.

DAC output channels are configurable as either constant voltage or current mode and are independent of each other. The output range is fixed (±10 V in voltage mode, and ±20 mA in current mode) and its levels are programmable with 16-bit resolution. Both channels are isolated from each other and fully protected, providing the capability to be connected in series or parallel for an even wider output range. The channel's output can be statically updated using the API/SFP or can be used to convert frequency measured from a counter channel to voltage/current output.

FEATURES

- 195 kΩ input impedance and AC/DC coupling for counter channels
- Single frequency measurement range that works from 0.05 Hz to 1 MHz
- Stable TCXO base clock (50 MHz ±1 ppm)
- Wide differential input voltage range (±48 V) with up to 250 V working common mode voltage
- Programmable threshold and hysteresis levels with 1 mV resolution
- Support for Quadrature encoder
- Isolated DIO channels with up to 60 V compliance
- Isolated, independent 16-bit DAC channels configurable for voltage or current output

EX1200-1538 SPECIFICATIONS

GENERAL SPECIFICATIONS	
FRONT PANEL CONNECTOR	
	104-pin high-density D-sub
POWER CONSUMPTION	
3.3 V	0.380 A
5 V	0.0081 A
24 V	0.15 A
COUNTER INPUT SPECIFICATIONS	
NUMBER OF CHANNELS	
	8 channels (analog/digital)
DIGITAL INPUT SIGNAL RANGE	
	TTL
ANALOG INPUT SIGNAL RANGE	
	±48 V (differential)
SENSITIVITY	
	±500 mV
THRESHOLD & HYSTERESIS	
	Programmable, 1 mV step
INPUT IMPEDANCE	
	195 kΩ
INPUT COUPLING	
	AC/DC
COMMON MODE INPUT	
	250 V_{PEAK}
SIGNAL FREQUENCY RANGE	
DC coupling mode	0.05 Hz to 1 MHz
AC coupling mode	3 Hz to 1 MHz
MAIN TIME BASE CLOCK	
	50 MHz (TCXO)
TIME BASE CLOCK STABILITY	
	±1 ppm
COUNTER TYPE	
	32-bit, reciprocal counting type
MAXIMUM TOTALIZE TICK COUNT	
	2^{32}
MINIMUM DETECTABLE PULSE	
Digital channels	50 ns
Analog channels	600 ns
RPM MEASUREMENT RANGE	
	3 RPM (min) to 90,000 RPM (max) – single range
SAMPLE DATA CORRELATION	
	IEEE 1588 time stamp
ONBOARD MEMORY	
	256,000 readings
REAL-TIME DATA OPERATIONS	
	Time based and pulse count based averaging (256 sample depth)
AVERAGING METHODS	
	Moving average and simple average
APERTURE TIME WINDOW	
	1 ms to 30 s (1 ms programming step)
MAX DATA SAMPLING SPEED	
	1,000,000 samples/s (into onboard buffer)
TRIGGERING	
	Software, Immediate, EX1200-based LXI triggers
QUADRATURE MEASUREMENT	
	Two channels to be paired for each encoder input
UTILITY 24V POWER SUPPLY (PIN #104)	
	Regulated power supply, 20 mA (Fused) maximum

DIO Specifications	
Number of Channels	
	16
DIO Input Signal Level	
Logical High	2.5 V to 60 V
Logical Low	< 2.5 V
DIO Isolation	
	Channel-to-channel, optical isolation
DIO Output Signals	
	Optically isolated solid-state switch
Output Signal Compatibility	
	50 mA sink/source, up to 60 V (AC/DC)
Update Control	
	Software paced
Analog Output (DAC) Specifications	
Number of Channels	
	2
Output Type	
	Constant voltage or constant current
Voltage Mode Range	
	±10 V (bipolar) can supply up to 20 mA per channel
Current Mode Range	
	±20 mA (bipolar) can drive up to 250 Ω load(10V compliance)
Output Resolution	
	16-bit
Isolation	
	Channel-to-channel, galvanic
Output Mode	
	Static Mode or Dynamic mode (frequency to voltage/current conversion)
Protection	
	Open and short circuit for continuous duration of time

Accessories	
Mating Connector (No Crimp)	
Description	104-pin HD D-sub mating connector with hood and pins, fixed contacts (no crimp tool required)
VTI part number	27-0389-104
Manufacturer/part number	Positronics ODD104M210GEX
Mating Connector (Crimp-Style)	
Description	104-pin HD D-sub mating connector, backshell and pins, crimp style
VTI part number	27-0390-104
Manufacturer/part number	Positronics ODD104M10Y0X
Crimp Tool	
Description	Crimp tooling, includes handle and positioner, 22 AWG
VTI part number	70-0297-001
Manufacturer/part number	Positronics 9507 (tool) and 9502-4-0-0 (positioner)
Pre-assembled, Unterminated Wiring Harness	
Description	104-pin HD D-sub mating connector and backshell, with 3 ft unterminated 22 AWG wire
VTI part number	70-0363-501
Terminal Block	
Description	EX1200-TB104P-1, single-ended
VTI part number	70-0367-011

ACCURACY CALCULATIONS

This section explains accuracy calculations for different measurements.

Frequency Measurement Using Digital Input

$$\mp F_{error} = F_{in} - \frac{F_{in} \times T_{apt}}{F_{base} + F_{base} \times F_{aqu} \times 10^{-6} \times T_{apt} + 1} \times F_{base}$$

Where:
F_{error} = Absolute frequency error in measurement (in Hz)
F_{in} = Input frequency (in Hz)
T_{apt} = Aperture time (in s)
F_{base} = Base clock frequency (in Hz)
F_{aqu} = Base clock accuracy (in ppm)

Frequency Measurement Using Analog Input

$$\mp Fa_{error} = F_{error} + F_{in} - \frac{1}{\frac{1}{F_{in}} + \frac{Trg_{error}}{N}}$$

$$\mp Trg_{error} = \sqrt{2} \times \frac{X_n^2 + e_n^2}{S_r^2}$$

Where:
F_{error} = Absolute frequency error in measurement (in Hz)
F_{in} = Input frequency (in Hz)
Fa_{error} = Analog input absolute frequency error in measurement (in Hz)
Trg_{error} = Trigger error (in ns)
X_n = System noise (in V, maximum 60 x 10^{-3} V)
e_n = Source noise (in V)
S_r = Slew rate at trigger point (V/ns) (Typically 8.30 x 10^{-3} V/ns)
T_{apt} = Aperture time (in Sec)
N = Number of samples = $F_{in} \times \boldsymbol{T_{apt}}$

Period Measurement Using Digital Input

$$\mp P_{error} = \frac{1}{F_{in}} - \frac{1}{F_{in} + F_{error}}$$

Where:
P_{error} = Absolute period measurement error (in s)
F_{error} = Absolute frequency error in measurement (in Hz)
F_{in} = Input frequency (in Hz)

Period Measurement Using Analog Input

$$\mp Pa_{error} = \frac{1}{F_{in}} - \frac{1}{F_{in} + Fa_{error}}$$

Where:
Pa_{error} = Absolute period measurement error using analog inputs (in s)

Pulse Width Measurement Accuracy Using Digital and Analog Inputs

For digital inputs:

$$20 + \ T_P \times 10^4 \quad ns$$

Where:
T_P = Input pulse width (in s)

For analog inputs:

$$20 + \ T_P \times 10^4 \ + \ \mathbf{2} * Trg_{error} \quad ns$$

Where :
T_P = Input pulse width (in s)
Trg_{error} = Trigger error (in ns)

Duty Cycle Measurement

Measured $\frac{T_{ON}}{T_{ON}+T_{OFF}} \times 100\ \%$

Where:
T_{ON} = Input pulse ON time (in s)
T_{OFF} = Input pulse OFF time (in s)

For T_{ON} and T_{OFF}, measurement accuracy, refer to the *Pulse Width Measurement Accuracy Using Digital and Analog Inputs* calculations.

Time Interval Measurement Accuracy Using Analog and Digital Inputs

Refer to the *Pulse Width Measurement Accuracy Using Digital and Analog Inputs* calculations.

Addition of 1.48 x 10^{-7} s offset to the pulse width measurement error for comparator slew rate compensation. Slew rate of the comparator is 400 ns (0 V - 3.3 V LVTTL threshold 1.2 V to detect as high).

Positive offset if analog input is the reference channel and negative offset if digital channel is the reference.

SECTION 2

USING THE INSTRUMENT

UNPACKING

When an EX1200-1538 is unpacked from its shipping carton, the contents should include the following items:

- An EX1200-1538
- LXI Quick Start Guide
- *EX1200-1538 User's Manual* (this manual)

All components should be immediately inspected for damage upon receipt of the unit. ESD precautions should be observed while unpacking and installing the instrument into an EX1200 series mainframe.

DETERMINE SYSTEM POWER REQUIREMENTS

The power requirements of the EX1200-1538 is provided in the *Specifications* section of *Section 1*. It is imperative that the EX1200 mainframe provides adequate power for the modules installed. For more information on EX1200 mainframe power consumption, please refer *Appendix B* of the *EX1200 Series User's Manual* (P/N: 82-0127-000). The user should confirm that the power budget for the system (for the chassis and all modules installed therein) is not exceeded on any voltage line.

It should be noted that if the mainframe cannot provide adequate power to the module, the instrument might not perform to specification and possibly damage the power supply. In addition, if adequate cooling is not provided, the reliability of the instrument will be jeopardized and permanent damage may occur. Damage found to have occurred due to inadequate cooling will void the warranty on the instrument in question.

PLUG-IN MODULE INSTALLATION

Before installing a plug-in module into an EX1200 system, make sure that the mainframe is powered down. Insert the module into the base unit by orienting the module so that the metal cover of the module can be inserted into the slot of the base unit. Position the cover so that it fits into the module's slot groove. Once the module is properly aligned, push the module back and firmly insert it into the backplane connector. See Figure 2-1 for guidance.

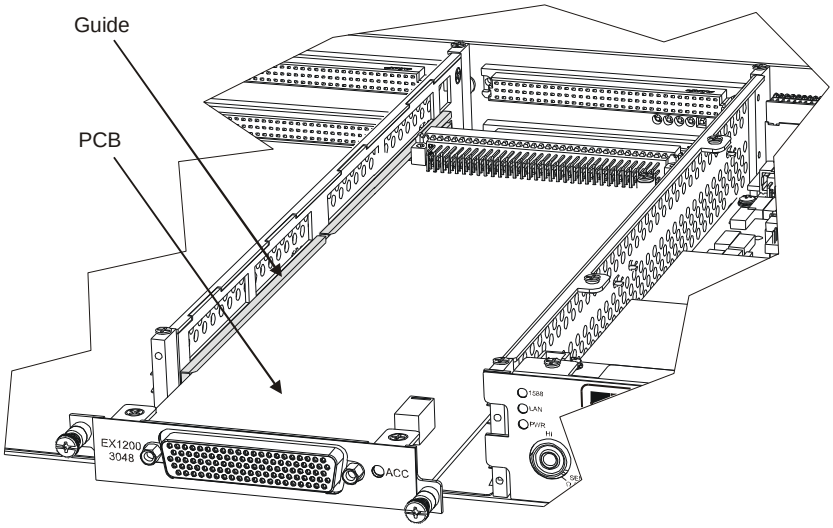

FIGURE 2-1: MODULE INSTALLATION (EX1200-3048 USED AS EXAMPLE)

WARM-UP TIME

The specified warm-up time for an EX1200 system is 30 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.

CONNECTOR PIN/SIGNAL ASSIGNMENT

The EX1200-1538 uses a 104-pin, high-density D-type connector for front panel signal interface. The tables below provides signal and connector pin assignment for the EX1200-1538. For mating connector information, please refer to the *EX1200-1538 Specifications* in this manual.

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	CH1_P	**22**	CH1_N	**43**	GND	**64**	USR_SHIELD	**85**	GND
2	CH2_P	**23**	CH2_N	**44**	GND	**65**	UNUSED	**86**	GND
3	CH3_P	**24**	CH3_N	**45**	GND	**66**	GND	**87**	GND
4	CH4_P	**25**	CH4_N	**46**	GND	**67**	UNUSED	**88**	GND
5	CH5_P	**26**	CH5_N	**47**	GND	**68**	UNUSED	**89**	GND
6	CH6_P	**27**	CH6_N	**48**	GND	**69**	QUAD_INDEX2	**90**	GND
7	CH7_P	**28**	CH7_N	**49**	GND	**70**	DIGI_SE2	**91**	GND
8	CH8_P	**29**	CH8_N	**50**	QUAD_INDEX1	**71**	QUAD_INDEX3	**92**	QUAD_INDEX4
9	GND	**30**	GND	**51**	DIGI_SE1	**72**	DIGI_SE3	**93**	DIGI_SE4
10	DIO1+	**31**	DIO1-	**52**	DIO9+	**73**	DIO9-	**94**	DIGI_SE5
11	DIO2+	**32**	DIO2-	**53**	DIO10+	**74**	DIO10-	**95**	DIGI_SE6
12	DIO3+	**33**	DIO3-	**54**	DIO11+	**75**	DIO11-	**96**	DIGI_SE7
13	DIO4+	**34**	DIO4-	**55**	DIO12+	**76**	DIO12-	**97**	DIGI_SE8
14	DIO5+	**35**	DIO5-	**56**	DIO13+	**77**	DIO13-	**98**	GND
15	DIO6+	**36**	DIO6-	**57**	DIO14+	**78**	DIO14-	**99**	UNUSED
16	DIO7+	**37**	DIO7-	**58**	DIO15+	**79**	DIO15-	**100**	GND
17	DIO8+	**38**	DIO8-	**59**	DIO16+	**80**	DIO16-	**101**	GND
18	AGND_DAC1	**39**	UNUSED	**60**	AGND_DAC1	**81**	GND	**102**	GND
19	AOUT_CH1-	**40**	AOUT_CH1+	**61**	AGND_DAC1	**82**	UNUSED	**103**	GND_C
20	AGND_DAC2	**41**	AGND_DAC1	**62**	AGND_DAC2	**83**	UNUSED	**104**	24V_OUTPUT
21	AOUT_CH2+	**42**	AOUT_CH2-	**63**	AGND_DAC2	**84**	GND		

TABLE 2-1: EX1200-1538 CONNECTOR PIN SIGNAL ASSIGNMENT

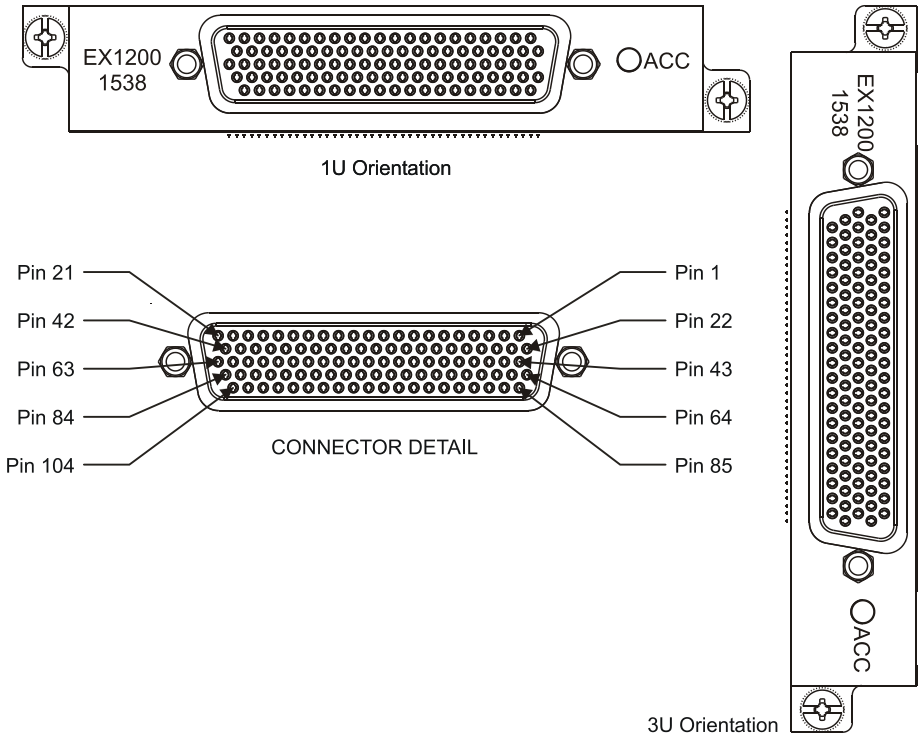

FIGURE 2-2: EX1200-1538 FRONT PANEL DETAIL

Front Panel Connector Pins Description

Pin Name	Description
CHx+/-	Analog input differential channels (P-Positive; N-Negative)
AOUT_CHx_+/-	Analog output channels (P-Positive; N-Negative)
AGND_DACx	Isolated GND for analog output
DIGI_SEx	Digital single ended channels
QUAD _INDEXx	Quadrature encoder's index signals
DIO_x+/-	Digital input/output channels
GND	Common ground
GND_C	Chassis ground
24V_OUTPUT	24 V power supply output. Limited to 24 mA.

EX1200-TB104P-1 TERMINAL BLOCK

VTI offers a single-ended terminal block for the EX1200-1538 (P/N: 70-0367-011). The terminal block simplifies cabling by providing screw-terminal blocks for user wiring. Signal pin mapping for the EX1200-1538 can be seen in Table 2-2.

TB Ref	Signal	Conn Pin	TB Ref	Signal	Conn Pin	TB Ref	Signal	Conn Pin	TB Ref	Signal	Conn Pin
T1	CH1_N	22	T31	DIGI_SE4	93	T61	DIO11+	54	T91	USR_SHIELD	64
T2	CH1_P	1	T32	GND	85	T62	DIO11-	75	T92	GND	102
T3	GND	9	T33	DIGI_SE5	94	T63	DIO12+	55	T93	UNUSED	UNUSED
T4	GND	48	T34	GND	86	T64	DIO12-	76	T94	UNUSED	UNUSED
T5	CH2_N	23	T35	DIGI_SE6	95	T65	DIO13+	56	T95	UNUSED	UNUSED
T6	CH2_P	2	T36	GND	87	T66	DIO13-	77	T96	UNUSED	UNUSED
T7	CH3_N	24	T37	DIGI_SE7	96	T67	DIO14+	57	T97	UNUSED	UNUSED
T8	CH3_P	3	T38	GND	88	T68	DIO14-	78	T98	UNUSED	UNUSED
T9	GND	49	T39	DIGI_SE8	97	T69	DIO15+	58	T99	UNUSED	UNUSED
T10	GND	30	T40	GND	89	T70	DIO15-	79	T100	UNUSED	UNUSED
T11	CH4_N	25	T41	DIO1+	10	T71	DIO16+	59	T101	UNUSED	UNUSED
T12	CH4_P	4	T42	DIO1-	31	T72	DIO16-	80	T102	UNUSED	UNUSED
T13	CH5_N	26	T43	DIO2+	11	T73	QUAD_INDEX1	50	T103	UNUSED	UNUSED
T14	CH5_P	5	T44	DIO2-	32	T74	GND	90	T104	UNUSED	UNUSED
T15	GND	66	T45	DIO3+	12	T75	QUAD_INDEX2	69	T105	UNUSED	UNUSED
T16	GND	44	T46	DIO3-	33	T76	GND	91	T106	UNUSED	UNUSED
T17	CH6_N	27	T47	DIO4+	13	T77	QUAD_INDEX3	71	T107	UNUSED	UNUSED
T18	CH6_P	6	T48	DIO4-	34	T78	GND	98	T108	UNUSED	UNUSED
T19	CH7_N	28	T49	DIO5+	14	T79	QUAD_INDEX4	92	T109	UNUSED	UNUSED
T20	CH7_P	7	T50	DIO5-	35	T80	GND	100	T110	UNUSED	UNUSED
T21	GND	45	T51	DIO6+	15	T81	AGND_DAC1	18	T111	UNUSED	UNUSED
T22	GND	81	T52	DIO6-	36	T82	AOUT_CH1+	40	T112	UNUSED	UNUSED
T23	CH8_N	29	T53	DIO7+	16	T83	AOUT_CH1-	19	T113	UNUSED	UNUSED
T24	CH8_P	8	T54	DIO7-	37	T84	AOUT_CH2+	21	T114	UNUSED	UNUSED
T25	DIGI_SE1	51	T55	DIO8+	17	T85	AOUT_CH2-	42	T115	UNUSED	UNUSED
T26	GND	46	T56	DIO8-	38	T86	AGND_DAC2	20	T116	UNUSED	UNUSED
T27	DIGI_SE2	70	T57	DIO9+	52	T87	GND	43	T117	UNUSED	UNUSED
T28	GND	47	T58	DIO9-	73	T88	24V_OUTPUT	104	T118	UNUSED	UNUSED
T29	DIGI_SE3	72	T59	DIO10+	53	T89	GND	101	T119	UNUSED	UNUSED
T30	GND	84	T60	DIO10-	74	T90	GND_C	103	T120	UNUSED	UNUSED

TABLE 2-2: EX1200-1538 TO EX1200-104P-1 PIN AND SIGNAL MAPPING

Terminal Block Receiver

The EX1200-TBR chassis is a 1U receiver capable of housing six terminal blocks. The EX1200-TBR ships with rubber feet for tabletop installations, but may be fitted with rackmount ears for installation into a test rack (P/N: 70-0367-010).

To install a terminal block into the EX1200-TBR, insert the flanges on the side of the terminal block into the guide rails of the desired slot. Continue to push the terminal block into the receiver until it is secured by the rear-locking latch of the receiver. To remove the terminal block from the EX1200-TBR, hold the center thumbscrew on the terminal block, then pull the terminal block from the receiver.

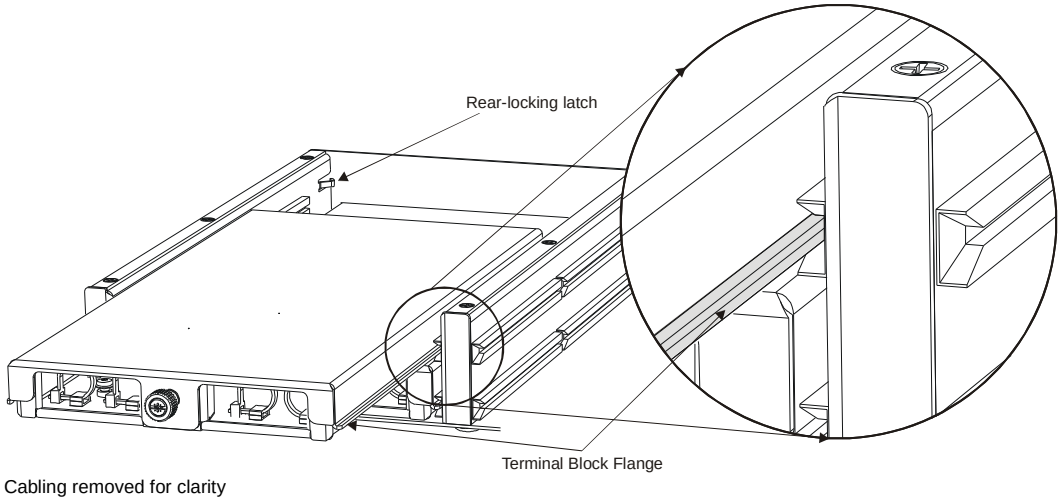

FIGURE 2-3: TERMINAL BLOCK INSTALLATION INTO THE EX1200-TBR

CALIBRATION

Every EX1200-1538 is factory calibrated using NIST-traceable standards. Optionally, the EX1200-1538 can be returned the factory for a complete factory calibration. **VTI recommends annual factory calibration of the EX1200-1538.**

SECTION 3

COUNTER/TIMER OPERATION

OVERVIEW

The EX1200-1538's electronic counter can be used to measure many time and frequency domain signal parameters for time-continuous waveforms and non-continuous (burst/pulse train) waveforms. These parameters include:

1) Frequency
2) RPM
3) Pulse width
4) Counter/totalize functions
5) Quadrature input
6) Duty cycle

The EX1200-1538 counter mechanism uses reciprocal counting technique (always makes a period measurement on the input signal). The benefit of the counting method is that it is based on an internal clock, so errors are dependent on the clock and not external sources. Hence, for a noiseless input signal and assuming negligible trigger and time base error, the resolution of the reciprocal counter would also be independent of the input signal frequency.

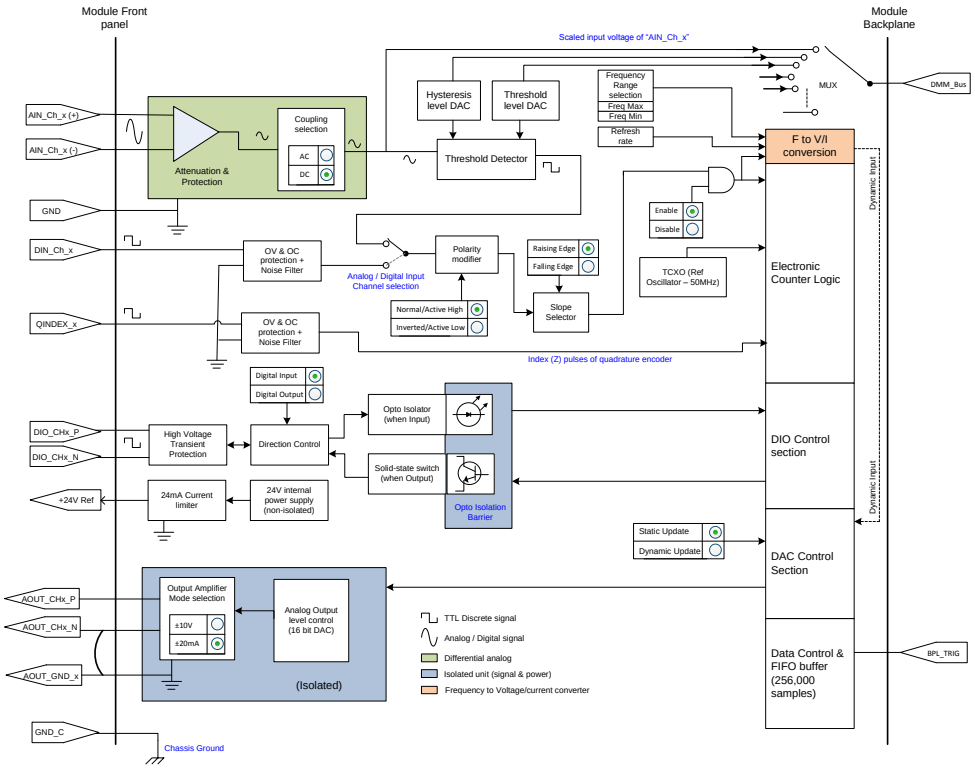

FIGURE 3-1: COUNTER BLOCK DIAGRAM

INPUTS

The EX1200-1538 counter has eight differential digital and eight analog input channels. The digital input channels can receive ground-referenced TTL signals (+5 V maximum) and allow for fast operating speeds.

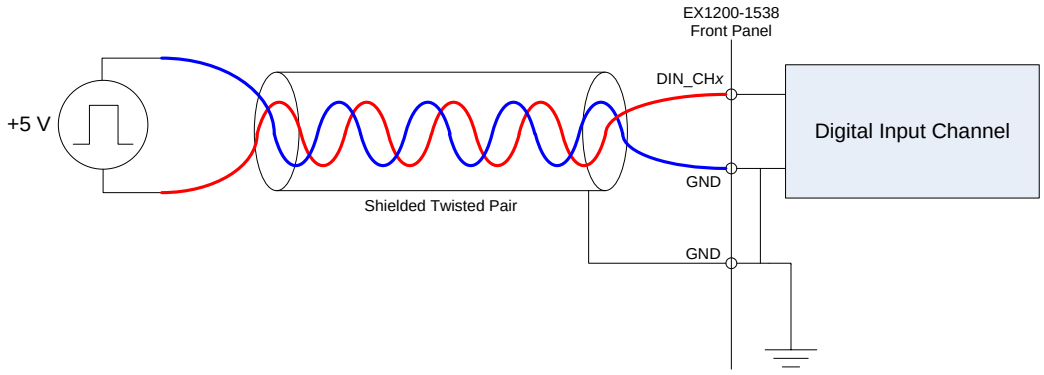

FIGURE 3-2: COUNTER DIGITAL INPUT CHANNEL CONNECTIVITY

Analog channels offer the threshold-level control and voltage compliance required to interface real-world sensors and transducers. The analog inputs are positive and negative differential pairs and receive differential voltage waveforms (up to ±48 V maximum). Analog input lines must be connected to the device under test (DUT) as detailed in Figure 3-3.

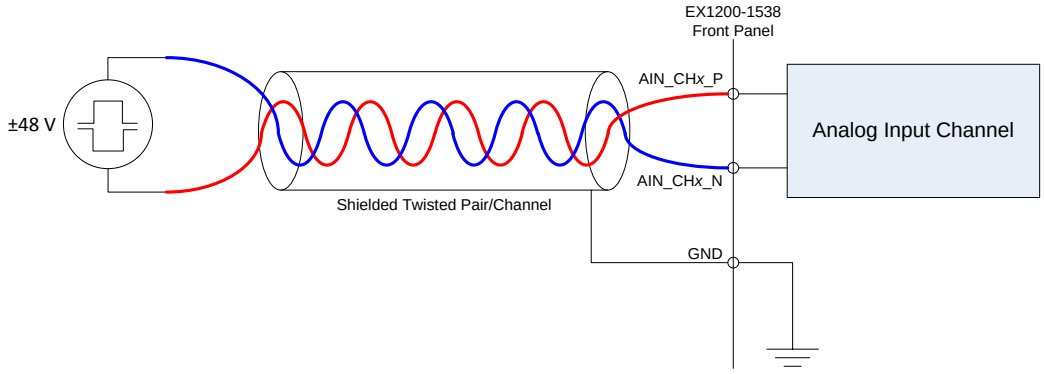

FIGURE 3-3: COUNTER ANALOG INPUT CHANNEL CONNECTIVITY

Input Coupling

AC/DC coupling is selectable on a per channel basis for analog inputs. In DC coupling mode, the signal is passed "as is" without modification. DC coupling is suitable for most applications. However, if the input signal contains a large, unpredictable DC offset, AC coupling mode may be used. In this mode, each channel contains a series capacitor (0.47 µF) which helps block the DC component from the input signal, while allowing the AC component of the signal to pass.

NOTE	The digital input channels are always DC coupled.

Signal Conversion

A programmable threshold detector is used to convert the analog input signals to digital states. The high and low trigger levels of a signal can be programmed using the level and hysteresis parameters as shown in Figure 3-4.

$V_{High} = V_{Level} + V_{HYSTERESIS}$
$V_{Low} = V_{Level} - V_{HYSTERESIS}$

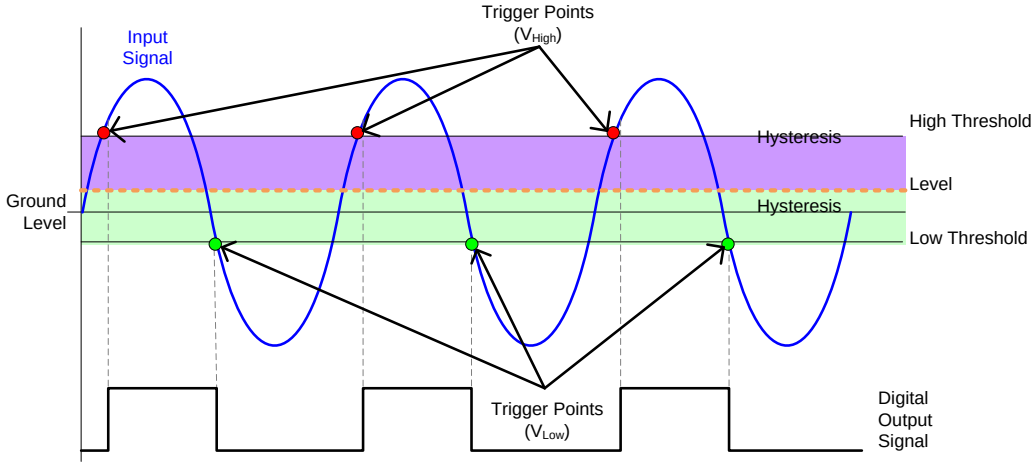

FIGURE 3-4: THRESHOLD AND HYSTERESIS LEVELS

Level and hysteresis are programmed in 1 mV steps over the entire voltage input range. These values should not exceed the minimum and maximum voltage limits of the input range (±48 V) ($V_{THRESHOLD} + V_{HYSTERESIS} < \pm 48$ V). When setting the level and hysteresis, it is important to note improper settings can adversely affect measurements. In Figure 3-5, the level and hysteresis are improperly set for the given input signal, resulting in noisy digital conversion.

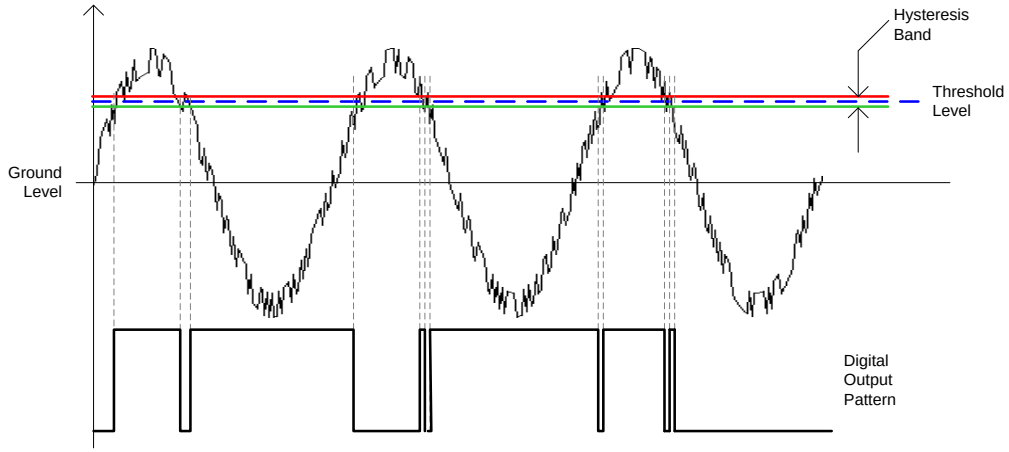

FIGURE 3-5: INCORRECT THRESHOLD AND HYSTERESIS USE

By properly adjusting the level and hysteresis, the effects of noise in input signal can be mitigated. Figure 3-6 shows how the fundamental frequency is extracted from the same noisy waveform with proper level and hysteresis setting.

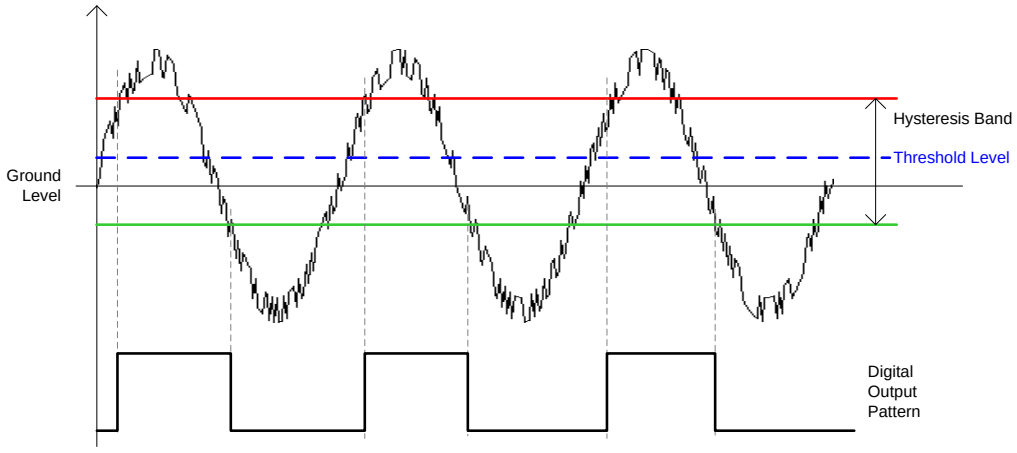

FIGURE 3-6: CORRECT THRESHOLD AND HYSTERESIS USE

Polarity Conversion

Polarity selection allows for the digital signal to be inverted prior to being sent to the electronic counter. By default, the signal is passed as an active High. When Inverted is selected, the signal is sent as an active Low.

FUNCTIONS

Totalizing

The **Totalize** function counts the total number of rising and falling edge transitions from an input signal. Counting begins as soon as the card is **Armed** and continues until its operation is **Aborted**. Any pulse with a width greater than 20 ns can be counted.

FIGURE 3-7: TOTALIZE FUNCTION

Typically, counting begins from zero (0) and can continue up to 4,294,967,295 (2^{32}-1) counts. If, however, it is necessary to begin counting at some positive number other than 0, the **Preset Count** parameter can be used. Should the EX1200-1538 exceed the maximum count number, the user can use one of three overflow modes for controlling how the data is managed: **STOP**, **PRESET**, and **WRAPAROUND**.

- **STOP** mode: the instrument stops counting once it "rolls over" (i.e. exceeds the maximum count number) and returns an invalid value (NaN) as a result.
- **PRESET** mode: the instrument counter rolls over and begins counting again from the defined **Preset Count** parameter.
- **WRAPAROUND** mode: the instruments counter rolls over and begins counting again from zero.

Edge Counting

By using the **Edge Count** function, the EX1200-1538 can count the number of rising or falling edge transitions. The **Slope** parameter determines whether rising or falling transitions are counted. If set to **Positive**, the EX1200-1538 only counts rising transitions, while it counts falling transitions when set to **Negative**.

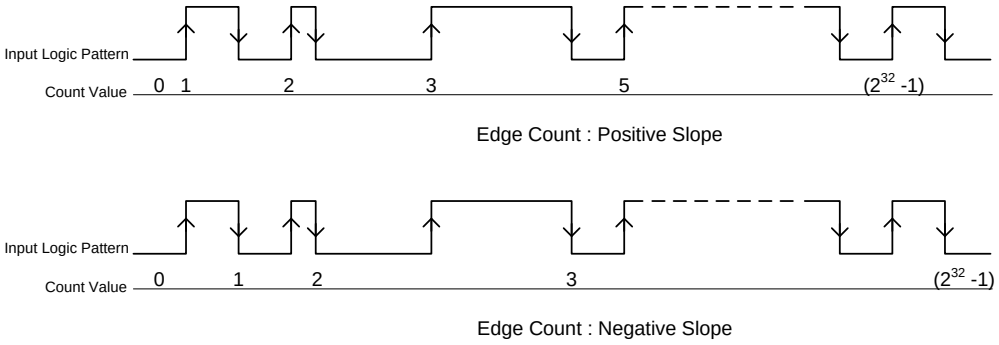

FIGURE 3-8: EDGE COUNTS FOR RISING AND FALLING SIGNAL TRANSITIONS

Edge Count also utilizes the same start/stop control, preset count, and overflow behaviors as the **Totalize** function.

Period Measurements

The period (T) of input signal can be measured using the EX1200-1538 counter. The data acquired from period measurements are the result of using one of two different averaging modes: **Aperture Time** mode and **Average Count** mode.

Aperture Time Mode

In **Aperture Time** mode, periods from all cycles contained within the aperture window are averaged and this is returned as the measurement result once the defined aperture time window elapses. This is referred to as “simple averaging”. This window has 1 ms resolution and starts immediately after the instrument is armed. Once armed, the window is continuous and remains until the operation is aborted. In this method, periods of each input cycle within aperture time window will be averaged to determine the period, which may be stored in the FIFO.

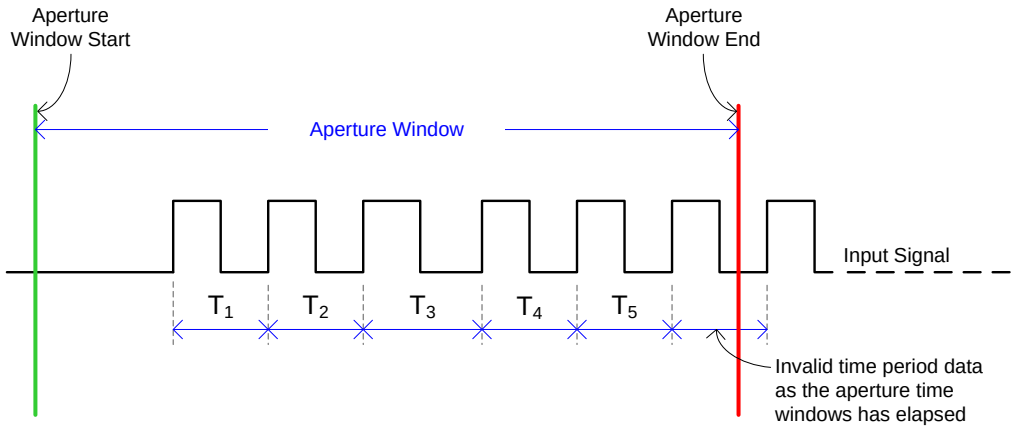

FIGURE 3-9: PERIOD MEASUREMENT USING APERTURE TIME AVERAGING

For the input signal in Figure 3-9, the period (T) of the input signal is calculated as follows:

$$T = \frac{T1+T2+T3+T4+T5}{5}$$

Average Count Mode

When **Average Count** mode is used, the period of the input signal is determined by averaging a user-defined number of input cycles. This is referred to as a "moving average". The **Average Count** parameter defines the number of cycles that are used when this calculation is performed. Figure 3-11 shows an input signal where the **Average Count** method is being used to calculate the period. Here, **Average Count** is set to "3". When **Average Repeat** is set to true, then every three consecutive cycles will be averaged. When set to **False**, the EX1200-1538 uses consecutive, overlapping cycles to calculate the period.

When **Average Repeat** is set to **False:**

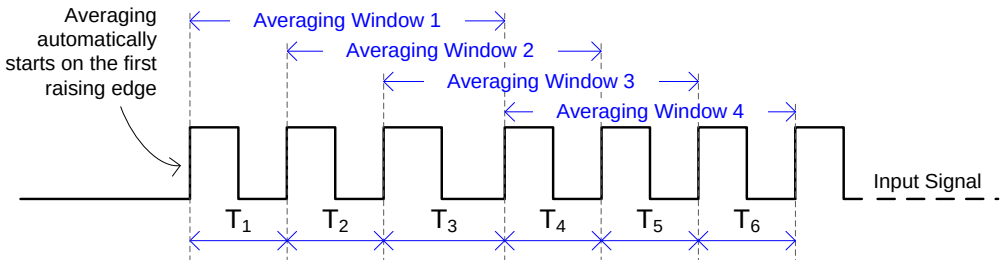

FIGURE 3-10: PULSE COUNT AVERAGING (AVERAGE REPEAT = FALSE)

If $n \geq N$ and after every cycle,

$$Average\ Window_x = \frac{T_{n-N+1} + T_{n-N+2} + .. + T_n}{N}$$

T_1, T_2 = Measured period for each cycle
N = Average count
n = Number of cycles occurred
x = Averaging window count

When **Average Repeat** is set to **True:**

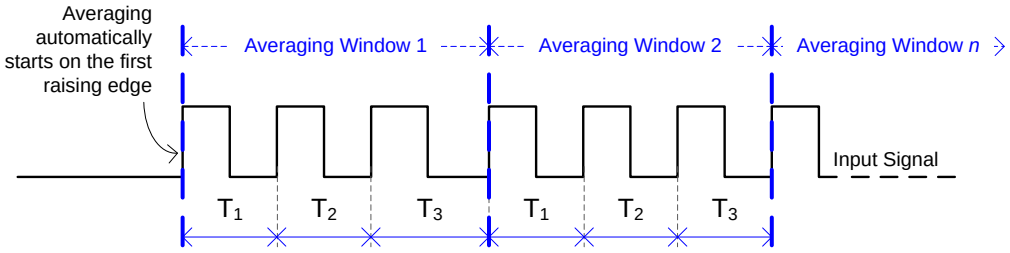

FIGURE 3-11: PULSE COUNT AVERAGING (AVERAGE REPEAT = TRUE)

Here, the period is determined using consecutive, *non-overlapping* samples.

If $n \geq N$ and for every N^{th} cycle,

$$Average\ Window_x = \frac{T_{n-N+1} + T_{n-N+2} + .. + T_n}{N}$$

T_1, T_2 = Measured period for each cycle
N = Average count
n = Number of cycles occurred
x = Averaging window count

Pulse Width Measurements

Pulse Width measurements measure the duration of an input signal's high cycles (T_{HC}). The EX1200-1538 only measures the time between a positive and negative slope. Thus, to determine a high cycle's duration, the polarity parameter should be set to **Normal** (high). To measure an input signal's low cycle, the polarity must be set to **Inverted** (low). As a result, both measurements cannot be performed simultaneously.

NOTE	Positive cycle width and negative cycle width cannot be measured simultaneously. However, by using two separate channels, one for measuring positive cycle width and other one for measuring low cycle width, this limitation can be surmounted

As a derivative of the **Period** function, pulse width measurements use the same averaging methodologies (i.e., **Aperture Time** and **Average Count** mode). The implementation of these methods is shown in the figures below.

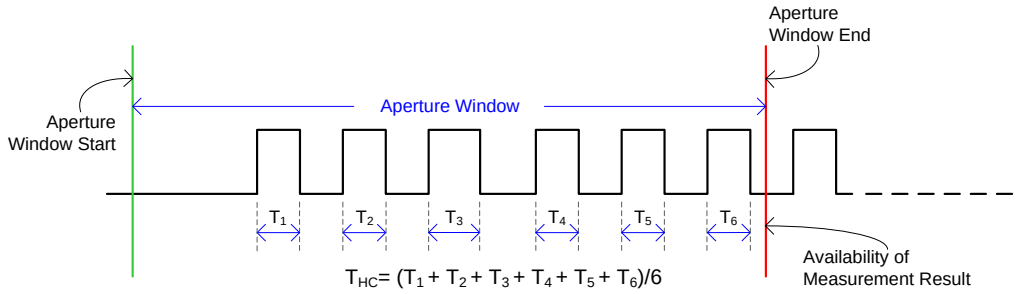

FIGURE 3-12: PULSE WIDTH MEASUREMENT USING APERTURE TIME AVERAGING

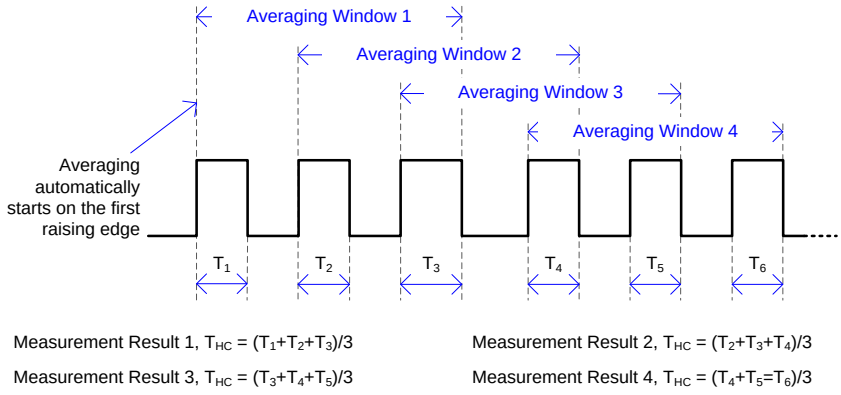

FIGURE 3-13: PULSE WIDTH MEASUREMENT (AVERAGE REPEAT = FALSE)

If $n \geq N$, and after every cycle:

$$Average\ Window_x = \frac{T_{n-N+1} + T_{n-N+2} + .. + T_n}{N}$$

T_1, T_2 = Measured pulse width for each cycle
N = Average count
n = Number of cycles occurred
x = Averaging window count

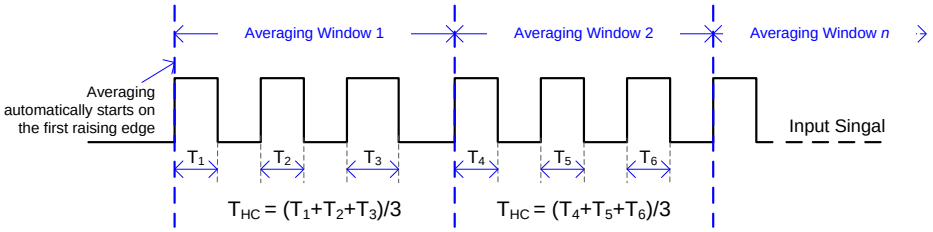

FIGURE 3-14: PULSE WIDTH MEASUREMENT (AVERAGE REPEAT = TRUE)

If $n \geq N$ and, for every N[th] cycle:

$$Average\ Window_x = \frac{T_{n-N+1} + T_{n-N+2} + .. + T_n}{N}$$

T_1, T_2 = Measured pulse width for each cycle
N = Average count
n = Number of cycles occurred
x = Averaging window count

Duty Cycle Measurement

Duty cycle is defined as the ratio between the high cycle duration and the total period for one input logic cycle as a percentage. EX1200-1538 counter directly calculates the duty cycle of the input signal and makes the results available at the end of every averaging cycle. As this measurement is derived from the period function, all configuration parameters including averaging modes for period measurements apply to duty cycle measurements as well.

Frequency Measurement

As the inverse of the **Period** function, the EX1200-1538 can also make **Frequency** (*f*) measurements. The reciprocal counting technique is used to calculate the frequency of an input signal. To do so, the period of incoming signal is measured and the inverse of the result is reported provide a frequency result.

As is true with the period measurements, frequency measurements typically utilize averaging methods in their calculations. Frequency measurements, however, can be made without averaging results. To do so, set the **Averaging Mode** to the **Average Count**, then set the **Average Count** value to "1". By doing so, the EX1200-1538 returns a frequency measurement for each incoming pulse.

When measuring frequency, a period measurement is made from one rising edge to the next. If the rise time of an input is long, however, frequency measurement accuracy can be affected by the jitter in the transition time. To obtain the most accurate frequency measurement, jitter on the rising edges should be reduced as much as possible.

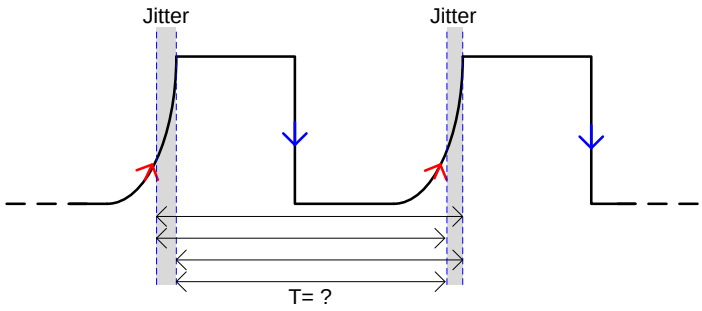

FIGURE 3-15: EFFECT OF JITTER ON FREQUENCY MEASUREMENTS

In Figure 3-15, the logic pattern has a long rise time and sharp fall time. When the signal is read by the analog/digital counter input channels, the jitter creates uncertainty when trying to identify the point at which the logical high begins. To reduce the effect of jitter, the signal can be inverted, as shown in Figure 3-16, to utilize the sharp fall time of the signal as the rise time on which the measurement is based.

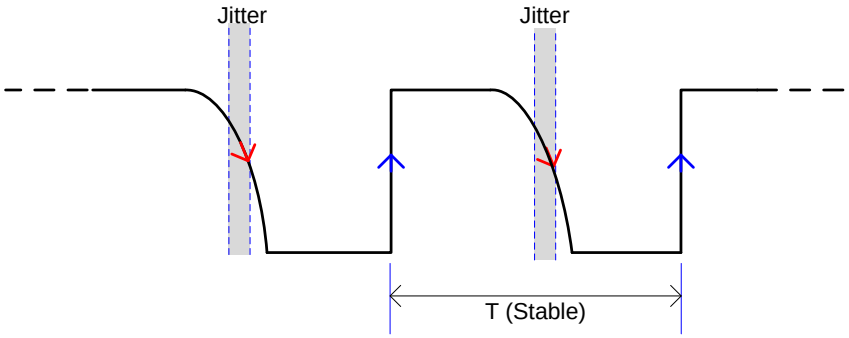

FIGURE 3-16: SIGNAL INVERSION TO DECREASE THE EFFECT OF JITTER

RPM Measurement

The **RPM** (revolutions per minute) function measures the rotational velocity of a toothed-wheel sensor. This function measures the tooth-to-tooth period (similar to **Frequency** mode) and converts it into units of revolutions per minute (RPM). In-line with frequency measurements, RPM measurements use averaging methods to determine the returned value.

The RPM measurement algorithm compensates for toothed-wheels that have a missing or extra tooth to mark their index position. Without this feature, RPM measurement would have bumps or sags. When configuring the **Tooth Count** parameter, the nominal number of teeth should be used. For example, a wheel nominally has twelve teeth, but has a missing tooth (A) or an extra tooth (B) to provide an index position as shown in Figure 3-17. In this case, the **Tooth Count** parameter should be set to“12” and **RpmToothState** should be set to **CounterRpmToothStateExtra** or **CounterRpmToothStateMissing** and the measurement will be compensated accordingly.

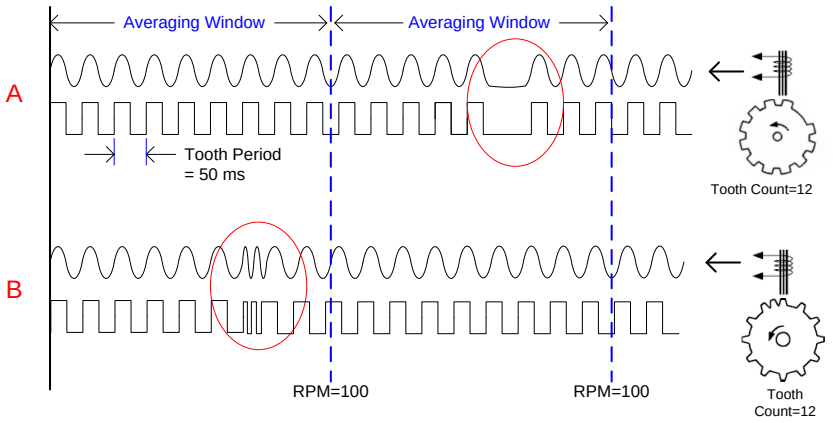

FIGURE 3-17: NOMINAL TOOTH COUNT FOR RPM MEASUREMENTS

When using the **RPM** function, ramp up time, ramp down time, and derivative parameters, such as acceleration, deceleration, etc., can be measured by tweaking the averaging window, sample trigger, etc.

Time Interval Measurement

Time Interval, or pulse-edge separation, measurements determine the time elapsed between the transition states of two signals on two different channels as shown in Figure 3-18. When the **Slope** of the signal is set to **Positive**, the **Time Interval** function uses the positive-going state transition, while the negative transition state is used when the **Slope** parameter is **Negative**. Since the **Time Interval** function is comparative, one channel must be defined as the **Reference** channel while the other channel is referred to as the **Measurement** channel. If a **Measurement** channel is not defined, the EX1200-1538 automatically uses the next adjacent channel as the reference channel.

If the input signal is not present or if the input signal does not crosses the configured threshold limit, the measurement will be waiting continuously for the signal. To prevent this, **Aperture Time** should be defined for this measurement, so that the instrument indicates the averaged measurement reading (if at least one complete measurement was acquired) and resets itself at the end of each aperture time window.

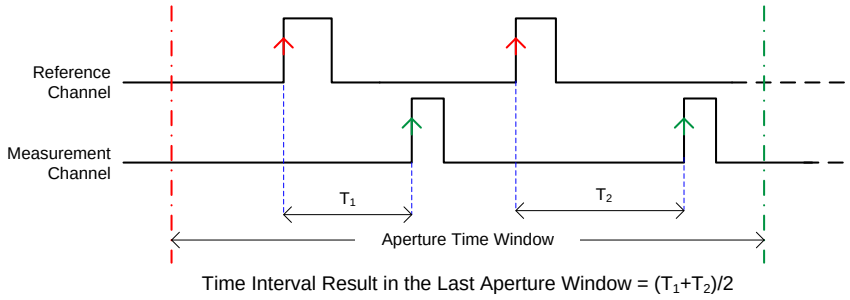

FIGURE 3-18: TIME INTERVAL MEASUREMENTS

The initial time interval measurements are always based on the first pulse detected on **Reference** channel. Should one or more pulses occur on the **Reference** channel prior to the first pulse occurring on the **Measurement** channel, the time interval value returned will be the time between the first pulse on the **Reference** channel and the first on the **Measurement** channel. If any pulses occur on the **Measurement** channel prior to the first pulse on the **Reference** channel, these pulses are ignored. If multiple pulses occur in the aperture window on both the **Reference** and **Measurement** channels, the average of these time intervals will be returned as the result of the

measurement. The measured value can be stored in the FIFO memory of the card at the end of each aperture window.

NOTE	The results for time interval measurements are only for the **Measurement** channel. Hence, the **Reference** channel can be used for other measurements as well (such as frequency measurements, pulse width, etc.).

Phase Measurements

Phase measurements are used to determine the phase shift/angular velocity between the **Reference** and **Measurement** channels. For phase measurement, the frequency of the **Reference** and **Measurement** signals must be same.

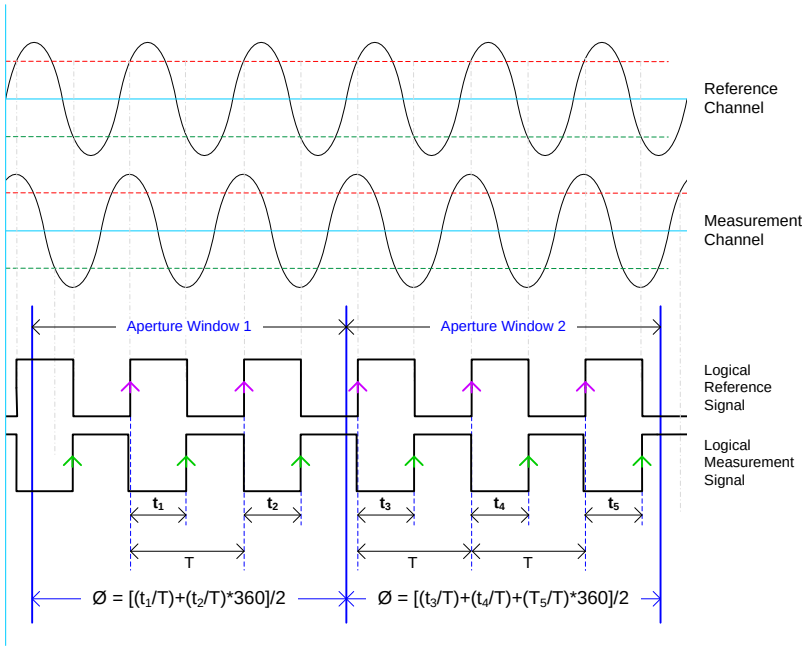

FIGURE 3-19: PHASE DIFFERENCE MEASUREMENTS

NOTE	The threshold levels of the **Measurement** and **Reference** channels have an effect on the overall accuracy of the measurement. A difference of ±20 mV can be expected between two channels,.

Quadrature Measurements

The EX1200-1538 can resolve a digital quadrature signal pair into an absolute 32-bit count. The quadrature position function increases/decreases the counter each time there is a transition on quadrature channel pair. When the **Measurement** channel's signal *leads* the **Reference** channel, the function counts up; when it *lags* the reference channel, the function counts down.

Quadrature position measurements use two channels: one as the **Reference** channel (A) and one as the **Measurement** channel (B). Any two free channels can be selected and defined as a pair. Optionally, an **Index** channel (Z) can be used to mark the reference position.

NOTE	Index pulse input channels must be digital. Take all appropriate grounding precautions while using analog signals for the measurement and reference channels.

The EX1200-1538 supports high-resolution **X2** and **X4** encoding methods. By enabling the **Index** pulse input, the counter is automatically set to **X4** mode. When an **Index** pulse is not used, the counter is set to **X2** mode by default.

In **X2** mode, the EX1200-1538 increments/decrements (dependent on which signal leads or lags) the count on the rising and falling edge of the **Reference** channel (A). As such, each cycle results in two increments or decrements. X2 mode behavior is shown in Figure 3-21 and X4 behavior is shown in Figure 3-20.

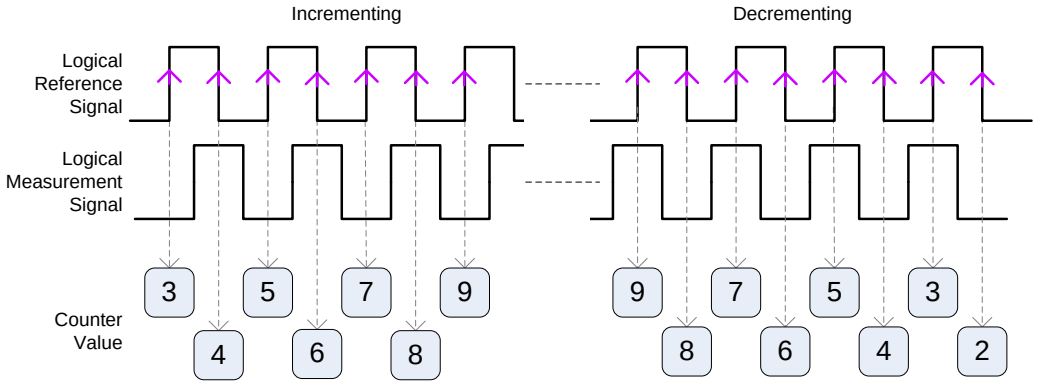

FIGURE 3-21: X2 MODE COUNTER INCREMENTS

In **X4** mode, the counter increments or decrements similarly on each edge of **Reference** and **Measurement** channels (A and B). Counter increments or decrements depends on which channel leads the other. Each cycle results in four increments or decrements, increasing the resolution by four times.

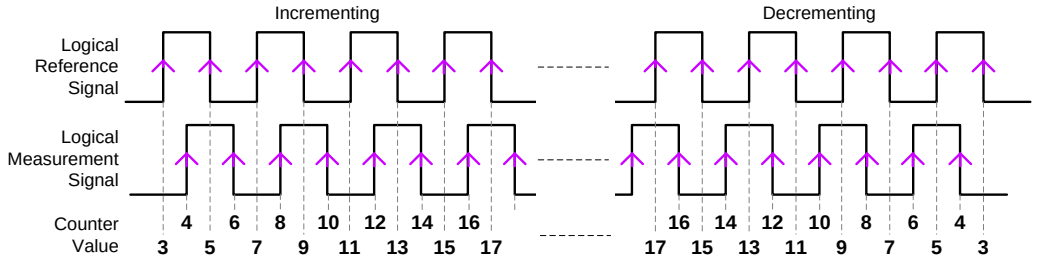

FIGURE 3-22: X4 MODE COUNTER INCREMENTS

NOTE	The results are only for the **Measurement** channel. Hence, the **Reference** channel can be used for other measurements as well (such as frequency measurements, pulse width, etc.).

SECTION 4

DIGITAL I/O AND ANALOG OUTPUT OPERATION

DIGITAL I/O OPERATION

In addition to its counter/timer functionality, the EX1200-1538 has sixteen independent digital input/output (DIO) lines that can be used independent of counter operation. Each channel can serve as either input or output.

When configured as a digital input, the logical state of the digital line is read via software. Digital input channels are isolated, differential pairs (positive and negative). Any differential voltage between 2.5 V and 60 V is detected as a logical High, while voltages lower than 2.5 V are treated as logical Low. To improve noise immunity, the grounds of the digital input channels use internal optical isolation.

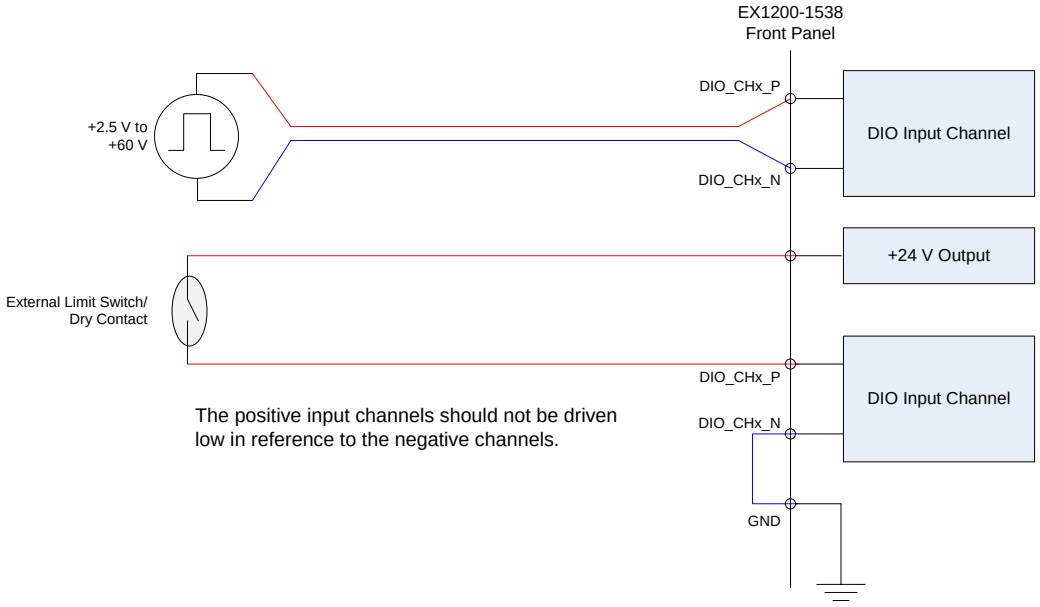

FIGURE 4-1: DIGITAL INPUT CHANNEL CONNECTIVITY

When configured as digital outputs, the logical state of the digital lines are controlled via software. The digital output channels use solid-state switches and are isolated from each other. Each output channel can handle voltages up to 60 V and currents up to 50 mA. Unlike the digital inputs, the output channels use a switch, instead of an open collector, so the polarity of the outputs do not need to be strictly maintained, meaning that positive and negative terminals can be interchanged if desired.

The EX1200-1538 also provides a 24 V dc output. This can be used for reading dry relay contacts and limit switches and removes the need for an additional external power supply. This output current is limited to 24 mA and referenced to instrument ground.

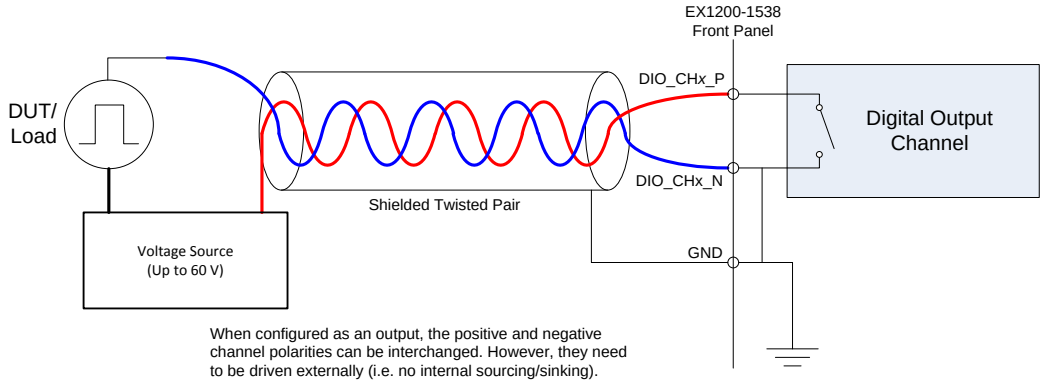

FIGURE 4-2: DIGITAL OUTPUT CHANNEL CONNECTIVITY

ANALOG OUTPUT OPERATION

The EX1200-1538 has two, on-board 16-bit digital-to-analog converters (DACs) that are capable of producing either voltage or current output. The outputs are bipolar and can be configured either as a ±10 V voltage source or as a 20 mA current source. The analog output updates can either be dynamic (frequency to voltage/current mode) or static mode.

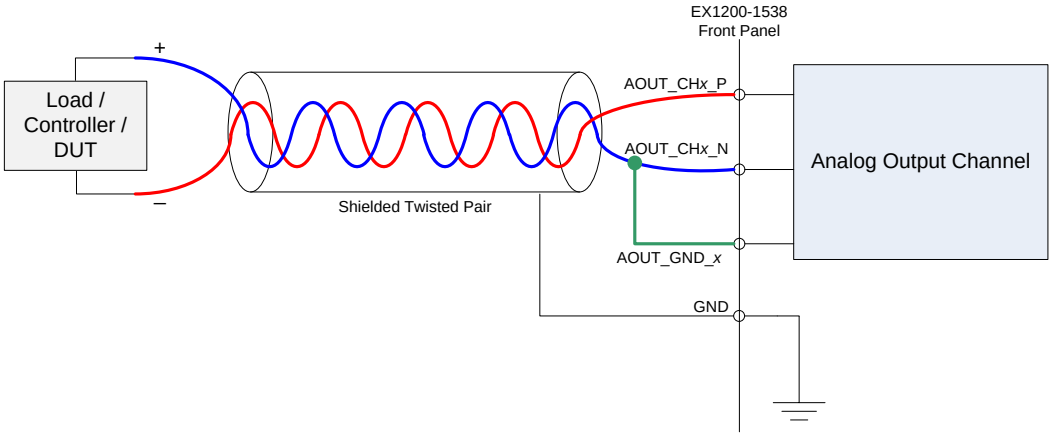

FIGURE 4-3: ANALOG (DAC) OUTPUT CHANNEL CONNECTIVITY

Static Update Mode

Counter channels can be configured as static outputs that generate a scalar voltage or current value on the specified digital I/O channel. As a static output, the value is controlled by the application software only and cannot respond to triggers.

Dynamic Update Mode

Any counter channel can be configured for dynamic update, allowing the channel's frequency to be converted into either a current (0 mA to 20 mA) or voltage (0 V to 10 V) output based on the drive mode of the DAC channel. The conversion formulae are based on the Lower Frequency Limit, Upper Frequency Limit of the counter channel, and the DAC channel's Refresh rate.

Voltage Formula

$$O_V\ V\ = \frac{\frac{C_O}{RFr}\ - F_{LO}}{(F_{HI} - F_{LO})} \times 10$$

NOTE	If the output value goes above 10 V or below 0 V as per the above equation, then the actual voltage will be saturated at 10 V and 0 V, respectively.

Current Formula

$$O_C\ mA\ = \frac{\frac{C_O}{RFr}\ - F_{LO}}{(F_{HI} - F_{LO})} \times 20$$

NOTE	If the output value goes above 20 mA or below 0 mA as per the above equation, then the actual current will be saturated at 20 mA and 0 mA respectively.

Where:

O_V = Voltage output from DAC in volts.
O_C = Current output from DAC in mA.
C_O = Number of pulses counted within Refresh rate window
RFr = Refresh rate setting of DAC
F_{HI} = Upper frequency limit of the counter channel
F_{LO} = Lower frequency limit of the counter channel

The dynamic update mode and the counter channel's measurement function are independent of each other. For example, the measurement function of a counter channel can be configured for pulse width or edge counting measurements, but can still be used for frequency to voltage/current conversion. The refresh interval defines the time span at which the frequency is averaged and updated into DAC.

Parallel Operation

The analog outputs are isolated, as each channel has its own ground reference and can be programmed independently. It is also possible to interconnect the channels to generate a larger voltage/current. When both channels are configured as voltage and connected in series, it can yield voltages up to ±20 V. When both channels are configured as current and connected in parallel, it can generate current signal up to ±40 mA.

NOTE	While connecting external voltage or current sources, the voltage difference between any pin to ground is not exceeding the card's safety limits.

The analog outputs are internally protected from short and open circuits continuously. When in voltage mode, the short circuit current is limited to 20 mA. In current mode, open circuit voltage (compliance voltage) is limited to 10 V.

CIRCUIT PROTECTION

All input and output channels are protected against intermittent voltage spikes. For the counter's analog input channels, the common mode voltage should not exceed 250 V_{peak} with respect the ground terminal. The maximum differential voltage across the analog input terminals should not exceed 300 V_{peak}. The digital input channels, which are intended to utilize TTL signals, are limited to +7.5 V from the ground terminal and inverse protection voltage is limited to 2.5 V. Beyond this will permanently damage the instrument.

WARNING	The absolute maximum voltage for digital counter channel's is limited to 25 V. Beyond 25 V, the EX1200-1538 will be permanently damaged. Additionally, the inputs should never be driven below the ground potential or damage may occur.

The DIO channels allow for the maximum voltage of 60 V per channel. When configured as a digital output channel, the current flowing in either direction should be limited to 50 mA.

Analog outputs have internal open and short circuit protected. Any external voltage applied should not exceed 25 V irrespective of whether channel enabled or not.

SECTION 5

PROGRAMMING THE INSTRUMENT

RELATED SOFTWARE COMPONENTS

IVI-COM Driver
IVI-C Driver
LabView Driver
Linux C++ Driver

USING THE DRIVER

The EX1200-1538 may be used in a variety of environments including: Visual Basic, C#, C++, LabView. VTI Instruments provides an IVI-C and IVI-COM compliant driver as well as a shared object that can be used on Linux systems that comply with the Linux Standard Base (Version 3.1).

Here is how to use the driver in each environment:

1) **Visual Studio C++**
```
#import "IviDriverTypeLib.dll" no_namespace
#import "VTEXMultifunction.dll" no_namespace
```

2) **C#**
Add a reference to VTEXMultifunction.dll in the project. Include the following at the top of any code file that will access the driver:

```
using VTI.VTEXFgen.Interop;
```

3) **C/C++ on Windows**
Link against VTEXFgen.lib and include VTEXMultifunction.h in the file.

4) **C++ on Linux**
Link against /opt/vti/lib/libmultifunction.so and include all the headers in /opt/vti/include in the source file.

5) **LabView**
Copy the driver package to the <Labview>/instr.lib directory and access all relevant VIs

INITIALIZING\CLOSING THE INSTRUMENT

The base interface of the EX1200-1538 IVI driver, VTEXMultifunction (LibMultifunction on Linux), is used to open and close connections to the instrument as well as containing pointers to all other interfaces to access the functionality of the instrument.

Visual Studio C++

```
#import "IviDriverTypeLib.dll" no_namespace
#import "VTEXMultifunction.dll" no_namespace

int main()
```

```
{
  //Windows driver creation
  ::CoInitialize(NULL); //Start the COM layer
  try
  {
    IVTEXMultifunctionPtr mfunction(__uuidof(VTEXMultifunction));

    /*The driver is given an empty options string. If more than one FGEN card in
   included in the mainframe, an option such as a slot number must be provided.
   This is because the Multifunction driver does not support more than one card per
driver instance. Note that the reset flag is also set so that the unit is started
   clean.*/

    mfunction ->Initialize("TCPIP::10.20.1.5::INSTR", VARIANT_TRUE, VARIANT_TRUE, "");

    // Use the Driver
    mfunction ->Close();
  }
  catch(...)
  {
    // Handle any exceptions
  }
  return 0;
}
```

Option Strings

The VTEX drivers provide option strings that can be used when initializing an instrument. The option string values exist to change the behavior of the driver. The following options strings are available on VTI IVI drivers:

- **Simulate**: Allows the user to run a program without commanding switch card or instruments. This option is useful as a debugging tool.
- **Cache**: Per the IVI specification, this option "specifies whether or not to cache the value of attributes." Caching allows IVI drivers to maintain certain instrument settings to avoid sending redundant commands. The standard allows for certain values to be cached always or never. In VTI IVI-drivers, all values used are of one of these types. As such, any values entered will have no effect on functionality.
- **QueryInstrumentStatus**: Queries the instrument for errors after each call is made. As implemented in the VTI IVI drivers, instruments status is always queried regardless of the value of this property.
- **DriverSetup**: Must be last, and contains the following properties:
  - **Logfile**: Allows the user to specify a file to which the driver can log calls and other data.
  - **Logmode**: Specifies the mode in which the log file is opened. The allowed modes are:
- **w**: truncate s the file to zero length or creates a text file for writing.
- **a**: opens the file for adding information to the end of the file. The file is created if it does not exist. The stream is positioned at the end of the file.
  - **LogLevel**: Allows the user to determine the severity of a log message by providing a level-indicator to the log entry.
  - **Slots**: This is the most commonly used option and it allows for a slot number or a slot number and a card model to be specified.
    "Slots=(2)" - Just slot 2.
    "Slots=(2=EX1200_1538)" - slot and card model
    "Slots=(2,3)" - Multiple slots

TOTALIZE FUNCTION

This example counts (totalize) the rising and falling edges of a ±19 V sine wave.

Visual Studio C++

```
// Totalize Measurement.cpp : Defines the entry point for the console application.

#include "stdafx.h"

#import "IviDriverTypeLib.dll" no_namespace
#import "VTEXMultifunction.dll" no_namespace

//This example counts the Rising and Falling edges of the sinewave ±19 volts.
//This example can be also used for Edge count measurement
int _tmain(int argc, _TCHAR* argv[])
{
   ::CoInitialize(NULL);

   try
   {
      IVTEXMultifunctionPtr mfunction(__uuidof(VTEXMultifunction));

      try
      {
           mfunction->Initialize("TCPIP::10.20.11.158::INSTR", VARIANT_TRUE,
VARIANT_TRUE, "");
         //Differential measurement
           //EX1200-1538 has only two voltage ranges - 48V and 100mV. Since the sine
wave is of ±19 volts, 48V range is being used.
           //This example uses AC coupling to cutoff offset voltage in the sinewave.
           //Analog channels Configuration
           mfunction->Counter->Channels->Item["CH2"]->ConfigureInput(VARIANT_TRUE,
VTEXMultifunctionCounterInputTypeAnalog,
               VTEXMultifunctionCounterInputModeDifferential, 48,
VTEXMultifunctionCounterCouplingAC);


           mfunction->Counter->Channels->Item["CH2"]->Function =
VTEXMultifunctionCounterFunctionTotalize;
           //Totalize measurement setting
           /* You can also work with the following functions by changing the channel
function
            * VTEXMultifunctionCounterFunctionEdgeCount
            */

           //±5V hysteresis for measuring the sine wave.
           mfunction->Counter->Channels->Item["CH2"]->ConfigureThreshold(0, 5,
VTEXMultifunctionCounterSlopePositive);
           //Polarity setting will NOT have any effet on totalize measurement, since
it counts both the rising and falling edges.
           //But in case of Edgecount measurement, normal polarity will count rising
edges and inverse polarity will count falling edges.
           mfunction->Counter->Channels->Item["CH2"]->Polarity =
VTEXMultifunctionCounterPolarityNormal;
           //This measures the frequency between 5000Hz and 1Hz.
           mfunction->Counter->Channels->Item["CH2"]->UpperFrequencyLimit = 5000;
//Maximum frequency of 5000Hz
           mfunction->Counter->Channels->Item["CH2"]->LowerFrequencyLimit = 1;
//Lower frequency of 1Hz
         //Enable Fifo for the channels
         mfunction->Counter->Channels->Item["CH2"]->FifoEnabled   = VARIANT_TRUE;
```

```
            //Aperature time setting is not required for Edge count and totalize
measurement. This setting will NOT have any effect on totalize and edge count
measurements.

            //measuring frequency in a Digital channel
            //Only single measurement possible with EX1200 - 1538
            //EX1200-1538 has only TTL logic for Digital channels. So range setting is
ignored.


            //Digital channels configuration
            mfunction->Counter->Channels->Item["CH3"]->ConfigureInput(VARIANT_TRUE,
VTEXMultifunctionCounterInputTypeDigital,
                VTEXMultifunctionCounterInputModeSingleEnded, 0,
VTEXMultifunctionCounterCouplingDC);


            mfunction->Counter->Channels->Item["CH3"]->Function =
VTEXMultifunctionCounterFunctionTotalize;
            //Totalize measurement setting
            /* You can also work with the following functions by changing the channel
function
             * VTEXMultifunctionCounterFunctionEdgeCount
             */

            // Threshold limits setting for Digital signal measurements are not
required.

            //This measures the frequency between 2000Hz and 2Hz.
            mfunction->Counter->Channels->Item["CH3"]->UpperFrequencyLimit = 2000;
//Maximum frequency of 2000Hz
            mfunction->Counter->Channels->Item["CH3"]->LowerFrequencyLimit = 2;
//Lower frequency of 2Hz


            //Polarity setting will NOT have any effet on totalize measurement, since
it counts both the rising and falling edges.
            //But in case of Edgecount measurement, normal polarity will count rising
edges and inverse polarity will count falling edges.
            //Alternatively Slope can also be used for the measurement
            mfunction->Counter->Channels->Item["CH3"]->Polarity =
VTEXMultifunctionCounterPolarityInverse;

          //Enable Fifo for the channels
          mfunction->Counter->Channels->Item["CH3"]->FifoEnabled   = VARIANT_TRUE;
            //Aperature time setting is not required for Edge count and totalize
measurement.This setting will NOT have any effect on totalize and edge count
measurements.

            //Here we are using software trigger. Using immediate trigger will store
the value for every edge.
            //Since we need the count say after 2 seconds we are using software
trigger
            mfunction->Trigger->Source = VTEXMultifunctionTriggerSourceSoftware;

             //Start measurement
             mfunction->Measurement->Initiate();
             int fifocount;
             do
             {
              //Wait for 2 seconds for collecting no. of edges(rising and falling).
              printf("Waiting for 2 seconds\n");
              Sleep(2000);
              printf("sending software trigger\n");
```

```
             //send a software trigger for transferring data to get a snap short of
totalize count
             mfunction->Measurement->SendSoftwareTrigger();
             fifocount = mfunction->Measurement->FifoCount;
             } while (fifocount < 10);

             //No. of enabled channels(here it is two. Channel 1 and channel 2)
             int no_of_channels_enabled = 2;
             /*For each trigger all channels data will be stored. In our case 2
channels is enabled and 10 readings are taken.
               So we will get 20 data points in the data array*/
             SAFEARRAY *data = NULL;
             SAFEARRAY *time = NULL;
             SAFEARRAY *time_fraction = NULL;
             //Read the measurements
             mfunction->Measurement->ReadFifo(0, fifocount, &data, &time,
&time_fraction);
             for (long scan = 0; scan<fifocount; scan++)
             {
              double dSecond, dFraction;
              SafeArrayGetElement(time, &scan, (void *)&dSecond);
              SafeArrayGetElement(time_fraction, &scan, (void *)&dFraction);
                 fprintf(stderr, "@Time = %f\n", dSecond + dFraction);
                 for (int channel = 0; channel < no_of_channels_enabled; channel++)
                 {
                      long dataindex = scan * no_of_channels_enabled + channel;
                  double dData;
                  SafeArrayGetElement(data, &dataindex, (void *)&dData);
                      fprintf(stderr, "Channel  = %d, Totalize = %f\n", channel+1,
dData);
                  //Channel 1 is measuring time interval and channel 2 is measuring
Frequency
                 }

             }
             //Abort the measurement.
             mfunction->Measurement->Abort();

       }
       catch (_com_error& e)
       {
          ::MessageBox(NULL, e.Description(), e.ErrorMessage(), MB_ICONERROR);
       }

       if (mfunction != NULL && mfunction->Initialized)
       {
          // Close driver
          mfunction->Close();
       }
   }
   catch (_com_error& e)
   {
       ::MessageBox(NULL, e.Description(), e.ErrorMessage(), MB_ICONERROR);
   }

   ::CoUninitialize();

   printf("\nDone - Press Enter to Exit");
    getchar();

   return 0;
}
```

EDGE COUNTING FUNCTION

Totalize example can also be used for edge counting functionality. To do so, change the function configured to "VTEXMultifunctionCounterFunctionEdgeCount" (for example, mfunction-> Counter->Channels->Item["CH2"]->Function = VTEXMultifunctionCounterFunctionEdgeCount).

FREQUENCY FUNCTION

The following code shows how to configure the EX1200-1538 to make a frequency measurement on an analog channel and a digital channel.

Visual Studio C++

```
// Program.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"

#import "IviDriverTypeLib.dll" no_namespace
#import "VTEXMultifunction.dll" no_namespace

//This example measures the frequency of a sinewave ±10 volts and a TTL digital
signal.
int _tmain(int argc, _TCHAR* argv[])
{
   ::CoInitialize(NULL);

   try
   {
      //Create Multifunction object
      IVTEXMultifunctionPtr mfunction(__uuidof(VTEXMultifunction));

      try
      {
         //Initialize new session
           mfunction->Initialize("TCPIP::10.30.1.16::INSTR", VARIANT_TRUE,
VARIANT_TRUE, "");



           //measuring frequency in a anaglog channel
           //Differential measurement
           //EX1200-1538 has only two voltage ranges - 48V and 100mV. Since the sine
wave is of ±10 volts, 48V range is being used.
           //This example uses AC coupling to cutoff offset voltage in the sinewave.
           //Analog channels Configuration
           mfunction->Counter->Channels->Item["CH1"]->ConfigureInput(VARIANT_TRUE,
                      VTEXMultifunctionCounterInputTypeAnalog,
                      VTEXMultifunctionCounterInputModeDifferential,
                      48, VTEXMultifunctionCounterCouplingAC);

           //5V±1V treshold for measuring the sine wave.
           mfunction->Counter->Channels->Item["CH1"]->Function =
VTEXMultifunctionCounterFunctionFrequency;
           //Frequency measurement setting
           /* You can also work with the following functions by changing the channel
function
            * VTEXMultifunctionCounterFunctionFrequency
            * VTEXMultifunctionCounterFunctionPulseWidth
            * VTEXMultifunctionCounterFunctionDutyCycle
            * VTEXMultifunctionCounterFunctionPeriod
            */
            mfunction->Counter->Channels->Item["CH1"]->ConfigureThreshold(0, 2.5,
VTEXMultifunctionCounterSlopePositive);
```

```
            //Measure on all the falling edges by inversing the polarity.
            mfunction->Counter->Channels->Item["CH1"]->Polarity =
VTEXMultifunctionCounterPolarityInverse;
            //This measures the frequency between 1000Hz and 1Hz.
            mfunction->Counter->Channels->Item["CH1"]->UpperFrequencyLimit = 10000;
//Maximum frequency of 1000Hz
            mfunction->Counter->Channels->Item["CH1"]->LowerFrequencyLimit = 1;
//Lower frequency of 1Hz


            //Aperture time of 2 senconds, since the minimum frequency to be measures
is 1Hz i.e 1second period. It averages the frequency of the signal for timespan of
2seconds
            mfunction->Counter->Channels->Item["CH1"]->AverageMode =
VTEXMultifunctionCounterAverageModeApertureTime;
            mfunction->Counter->Channels->Item["CH1"]->ApertureTime = 1;
//aperature time in seconds

          //Enable Fifo for the channels
          mfunction->Counter->Channels->Item["CH1"]->FifoEnabled   = VARIANT_TRUE;
            //measuring frequency in a Digital channel
            //Only single measurement possible with EX1200 - 1538
            //EX1200-1538 has only TTL logic for Digital channels. So range setting
is ignored.


            //Digital channels configuration
            mfunction->Counter->Channels->Item["CH2"]->ConfigureInput(VARIANT_TRUE,
                       VTEXMultifunctionCounterInputTypeDigital,
                       VTEXMultifunctionCounterInputModeSingleEnded,
                       0, VTEXMultifunctionCounterCouplingDC);

            //Frequency measurement setting
            mfunction->Counter->Channels->Item["CH2"]->Function =
VTEXMultifunctionCounterFunctionFrequency;
            //Frequency measurement setting
            /* You can also work with the following functions by changing the channel
function
             * VTEXMultifunctionCounterFunctionFrequency
             * VTEXMultifunctionCounterFunctionPulseWidth
             * VTEXMultifunctionCounterFunctionDutyCycle
             * VTEXMultifunctionCounterFunctionPeriod
             */

            // Threshold limits setting for Digital signal measurements are not
required

            //This measures the frequency between 5000Hz and 1Hz.
            mfunction->Counter->Channels->Item["CH2"]->UpperFrequencyLimit = 5000;
//Maximum frequency of 5000Hz
            mfunction->Counter->Channels->Item["CH2"]->LowerFrequencyLimit = 1;
//Lower frequency of 1Hz

            //Measure on all the Rising edges
            mfunction->Counter->Channels->Item["CH2"]->Polarity =
VTEXMultifunctionCounterPolarityNormal;

            //Aperture time of 10 seconds, say the minimum frequency to be measures
is 1Hz i.e 1second period. This setting averages the measured frequency for timespan
of 10seconds
            mfunction->Counter->Channels->Item["CH2"]->AverageMode =
VTEXMultifunctionCounterAverageModeApertureTime;
          mfunction->Counter->Channels->Item["CH2"]->ApertureTime = 1.2;
//aperature time in seconds
```

```
           //Enable Fifo for the channels
           mfunction->Counter->Channels->Item["CH2"]->FifoEnabled   = VARIANT_TRUE;

             mfunction->Trigger->Source = VTEXMultifunctionTriggerSourceImmediate;

             //Start measurement
             mfunction->Measurement->Initiate();
             int fifocount,loopcount = 0;
             do
             {
                  //Take measurement for 10 times.
                  fifocount = mfunction->Measurement->FifoCount;
                  Sleep(1000);
               if(loopcount > 10)
               {
                   printf("Expected Fifo count (10) is not Available, Exiting
loop\n");
                   break;
               }
               loopcount++;
             } while (fifocount < 10);

           printf("Fifo Count = %d\n",fifocount);
           if(fifocount <= 0)
           {  //Exiting function when no data available
              printf("No Fifo data, Exiting function...\n");
              return 0;
           }
             //No. of enabled channels(here it is two. Channel 1 and channel 2)
             int no_of_channels_enabled = 2;
             /*For each trigger all channels data will be stored. In our case 2
channels is enabled and 10 readings are taken.
               So we will get 20 data points in the data array*/
             SAFEARRAY *data = NULL;
             SAFEARRAY *time = NULL;
             SAFEARRAY *time_fraction = NULL;
             //Read the measurements
             mfunction->Measurement->ReadFifo(0, fifocount, &data, &time,
&time_fraction);
             for (long scan = 0; scan<fifocount; scan++)
             {
              double dSecond, dFraction;
              SafeArrayGetElement(time, &scan, (void *)&dSecond);
              SafeArrayGetElement(time_fraction, &scan, (void *)&dFraction);
                 fprintf(stderr, "@Time = %f\n", dSecond + dFraction);
                 for (int channel = 0; channel < no_of_channels_enabled; channel++)
                 {
                      long dataindex = scan * no_of_channels_enabled + channel;
                   double dData;
                   SafeArrayGetElement(data, &dataindex, (void *)&dData);
                       fprintf(stderr, "Channel  = %d, Frequency = %f\r\n", channel+1,
dData);
                 }

             }
             //Abort the measurement.
             mfunction->Measurement->Abort();

       }
       catch (_com_error& e)
       {
          ::MessageBox(NULL, e.Description(), e.ErrorMessage(), MB_ICONERROR);
       }
```

```
      if (mfunction != NULL && mfunction->Initialized)
      {
         // Close driver
         mfunction->Close();
      }
   }
   catch (_com_error& e)
   {
      ::MessageBox(NULL, e.Description(), e.ErrorMessage(), MB_ICONERROR);
   }

   ::CoUninitialize();

   printf("\nDone - Press Enter to Exit");
    getchar();

   return 0;
}
```

RPM FUNCTION

This example measures the RPM using an encoder or syncro resolver.

Visual Studio C++

```
// RPM measurement.cpp : Defines the entry point for the console application.

#include<stdio.h>
#include "stdafx.h"

#import "IviDriverTypeLib.dll" no_namespace
#import "VTEXMultifunction.dll" no_namespace



//This example measures the RPM using an encoder or syncro resolver(less voltage type)
int _tmain(int argc, _TCHAR* argv[])
{
   ::CoInitialize(NULL);

   try
   {
      IVTEXMultifunctionPtr mfunction(__uuidof(VTEXMultifunction));

      try
      {
         //Initialize a new session
           mfunction->Initialize("TCPIP::10.20.11.158::INSTR", VARIANT_TRUE,
VARIANT_TRUE, "");

           //Differential measurement
           //EX1200-1538 has only two voltage ranges - 48V and 100mV. Since the sine
wave is of ±10 volts, 48V range is being used.
           //This example uses AC coupling to cutoff offset voltage in the sinewave.
           //Analog channels Configuration
         mfunction->Counter->Channels->Item["CH1"]->ConfigureInput(VARIANT_TRUE,
VTEXMultifunctionCounterInputTypeAnalog,
            VTEXMultifunctionCounterInputModeDifferential, 48,
VTEXMultifunctionCounterCouplingAC);

         mfunction->Counter->Channels->Item["CH1"]->Function =
VTEXMultifunctionCounterFunctionRpm;
```

```
          //5V treshold for measuring the sine wave. Note that slope configuration
will NOT have any effect.
          mfunction->Counter->Channels->Item["CH1"]->ConfigureThreshold(0, 5,
VTEXMultifunctionCounterSlopePositive);

          //Enable Fifo for the channel
          mfunction->Counter->Channels->Item["CH1"]->FifoEnabled   = VARIANT_TRUE;

          //This measures the frequency between 10000Hz and 1Hz.
          mfunction->Counter->Channels->Item["CH1"]->UpperFrequencyLimit = 10000;
//Maximum frequency of 10000Hz
          mfunction->Counter->Channels->Item["CH1"]->LowerFrequencyLimit = 1;
//Lower frequency of 1Hz

          //This setting determines the no. of pulses for each revolution. In our case
it is 32, So 32 pulses makes one full revolution
          mfunction->Counter->Channels->Item["CH1"]->RpmToothCount = 32;
          //No missing count would be added or subtracted from actual count
          mfunction->Counter->Channels->Item["CH1"]->RpmToothState =
VTEXMultifunctionCounterRpmToothStateNormal;

          //Aperture time of 1 sencond. It averages the RPM for timespan for every
second.
          mfunction->Counter->Channels->Item["CH1"]->AverageMode =
VTEXMultifunctionCounterAverageModeApertureTime;
          mfunction->Counter->Channels->Item["CH1"]->ApertureTime = 1;
//aperature time in seconds

          //measuring frequency in a Digital channel
          //Only single measurement possible with EX1200 - 1538
          //EX1200-1538 has only TTL logic for Digital channels. So range setting is
ignored.


          //Digital channels configuration. Coupling setting is ignored for Digital
channel. Only DC coupling  is possible for Digital channel.
          mfunction->Counter->Channels->Item["CH2"]->ConfigureInput(VARIANT_TRUE,
VTEXMultifunctionCounterInputTypeDigital,
             VTEXMultifunctionCounterInputModeSingleEnded, 0,
VTEXMultifunctionCounterCouplingDC);

          //RPM measurement setting
          mfunction->Counter->Channels->Item["CH2"]->Function =
VTEXMultifunctionCounterFunctionRpm;

          // Threshold limits setting for Digital signal measurements are not required

          //This measures the frequency between 10000Hz and 1Hz.
          mfunction->Counter->Channels->Item["CH2"]->UpperFrequencyLimit = 10000;
//Maximum frequency of 10000Hz
          mfunction->Counter->Channels->Item["CH2"]->LowerFrequencyLimit = 1;
//Lower frequency of 1Hz

          //This setting determines the no. of pulses for each revolution. In our case
it is 32, So 32 pulses makes one full revolution
          mfunction->Counter->Channels->Item["CH2"]->RpmToothCount = 32;
          //No missing count would be added or subtracted from actual count
          mfunction->Counter->Channels->Item["CH2"]->RpmToothState =
VTEXMultifunctionCounterRpmToothStateNormal;

          //Aperture time of 1 sencond. It averages the RPM for timespan for every
second.
          mfunction->Counter->Channels->Item["CH2"]->AverageMode =
VTEXMultifunctionCounterAverageModeApertureTime;
```

```
          mfunction->Counter->Channels->Item["CH2"]->ApertureTime = 1;
//aperature time in seconds

          //Enable Fifo for the channel
          mfunction->Counter->Channels->Item["CH2"]->FifoEnabled   = VARIANT_TRUE;

            mfunction->Trigger->Source = VTEXMultifunctionTriggerSourceImmediate;

           //Start measurement
             mfunction->Measurement->Initiate();
             int fifocount,loopcount = 0;
             do
             {
                  //Take measurement for 10 times.
                  fifocount = mfunction->Measurement->FifoCount;
                  Sleep(1000);
               if(loopcount > 10)
               {
                   printf("Expected Fifo count (10) is not Available, Exiting
loop\n");
                   break;
               }
               loopcount++;
             } while (fifocount < 10);

           printf("Fifo Count = %d\n",fifocount);
           if(fifocount <= 0)
           {  //Exiting function when no data available
              printf("No Fifo data, Exiting function...\n");
              return 0;
           }
             //No. of enabled channels(here it is two. Channel 1 and channel 2)
             int no_of_channels_enabled = 2;
             /*For each trigger all channels data will be stored. In our case 2
channels is enabled and 10 readings are taken.
               So we will get 20 data points in the data array*/
             SAFEARRAY *data = NULL;
             SAFEARRAY *time = NULL;
             SAFEARRAY *time_fraction = NULL;
             //Read the measurements
             mfunction->Measurement->ReadFifo(0, fifocount, &data, &time,
&time_fraction);
             for (long scan = 0; scan<fifocount; scan++)
             {
              double dSecond, dFraction;
              SafeArrayGetElement(time, &scan, (void *)&dSecond);
              SafeArrayGetElement(time_fraction, &scan, (void *)&dFraction);
                 fprintf(stderr, "@Time = %f\n", dSecond + dFraction);
                 for (int channel = 0; channel < no_of_channels_enabled; channel++)
                 {
                      long dataindex = scan * no_of_channels_enabled + channel;
                   double dData;
                   SafeArrayGetElement(data, &dataindex, (void *)&dData);
                      fprintf(stderr, "Channel  = %d, RPM = %f\n", channel+1, dData);
                 }

             }
             //Abort the measurement.
             mfunction->Measurement->Abort();

       }
       catch (_com_error& e)
       {
          ::MessageBox(NULL, e.Description(), e.ErrorMessage(), MB_ICONERROR);
```

```
      }

      if (mfunction != NULL && mfunction->Initialized)
      {
         // Close driver
         mfunction->Close();
      }
   }
   catch (_com_error& e)
   {
      ::MessageBox(NULL, e.Description(), e.ErrorMessage(), MB_ICONERROR);
   }

   ::CoUninitialize();

   printf("\nDone - Press Enter to Exit");
    getchar();

   return 0;
}
```

TIME INTERVAL FUNCTION

This example measures the Time interval between two sine waves of ±10 V and ±5 V.

Visual Studio C++

```
// Time Interval.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"

#import "IviDriverTypeLib.dll" no_namespace
#import "VTEXMultifunction.dll" no_namespace

//This example measures the Time interval between two sinewaves of ±10 volts and ±5
volts.
//This example can be also used for measuring Phase measurement
//Note that for finding the phase measurement and time interval measurement of
continuous signal both the signal should be of same frequency
int _tmain(int argc, _TCHAR* argv[])
{
   ::CoInitialize(NULL);

   try
   {
      IVTEXMultifunctionPtr mfunction(__uuidof(VTEXMultifunction));

      try
      {
         //Initialize a new session
           mfunction->Initialize("TCPIP::10.30.1.16::INSTR", VARIANT_TRUE,
VARIANT_TRUE, "");
         //Differential measurement
           //EX1200-1538 has only two voltage ranges - 48V and 100mV. Since the sine
wave is of ±10 volts, 48V range is being used.
           //This example uses AC coupling to cutoff offset voltage in the sinewave.
           //Analog channels Configuration
           mfunction->Counter->Channels->Item["CH1"]->ConfigureInput(VARIANT_TRUE,
VTEXMultifunctionCounterInputTypeDigital,
               VTEXMultifunctionCounterInputModeSingleEnded, 48,
VTEXMultifunctionCounterCouplingDC);
           //Setting reference channel configuration
           mfunction->Counter->Channels->Item["CH2"]->ConfigureInput(VARIANT_TRUE,
VTEXMultifunctionCounterInputTypeDigital,
```

```
                VTEXMultifunctionCounterInputModeSingleEnded, 48,
VTEXMultifunctionCounterCouplingDC);

            mfunction->Counter->Channels->Item["CH1"]->Function =
VTEXMultifunctionCounterFunctionTimeInterval;
            //Time interval measurement setting
            /* You can also work with the following functions by changing the channel
function
             * Phase measurement
             */
            //Set channel 2 as the reference channel
            mfunction->Counter->Channels->Item["CH1"]->ReferenceChannel = "CH2";
            //Reference channel can be used for any measurement even though it is used
as the reference channel for channel 1 measurement.
            mfunction->Counter->Channels->Item["CH2"]->Function =
VTEXMultifunctionCounterFunctionFrequency;

          //Enable Fifo for the channels
          mfunction->Counter->Channels->Item["CH1"]->FifoEnabled   = VARIANT_TRUE;
          mfunction->Counter->Channels->Item["CH2"]->FifoEnabled   = VARIANT_TRUE;

            //In case of Digital channel type, Level and Hysteresis settings are
ignored.
            //5V treshold for measuring the sine wave. Measure on  the Rising edges
only.
            mfunction->Counter->Channels->Item["CH1"]->ConfigureThreshold(0, 5,
VTEXMultifunctionCounterSlopePositive);
            //2.5V treshold for measuring the sine wave. Note that this will affect
only the Channel 2 frequency measurement only NOT Time interval measurement.The slope
configuration will NOT have any effect.
            mfunction->Counter->Channels->Item["CH2"]->ConfigureThreshold(0, 2.5,
VTEXMultifunctionCounterSlopeNegative);

            //This measures the frequency between 1000Hz and 1Hz. Both the channel's
input signal frequency should be same
            mfunction->Counter->Channels->Item["CH1"]->UpperFrequencyLimit = 100000;
//Maximum frequency of 1000Hz
            mfunction->Counter->Channels->Item["CH1"]->LowerFrequencyLimit = 1;
//Lower frequency of 1Hz
            mfunction->Counter->Channels->Item["CH2"]->UpperFrequencyLimit = 1000;
//Maximum frequency of 1000Hz
            mfunction->Counter->Channels->Item["CH2"]->LowerFrequencyLimit = 1;
//Lower frequency of 1Hz

            //Aperture time of 2 seconds. It averages the timeinterval for 2 seconds
            mfunction->Counter->Channels->Item["CH1"]->AverageMode =
VTEXMultifunctionCounterAverageModeApertureTime;
            mfunction->Counter->Channels->Item["CH1"]->ApertureTime = 2;
//aperature time in seconds

            //Aperture time of 1 senconds. It averages the frequency for 1 second
            mfunction->Counter->Channels->Item["CH2"]->AverageMode =
VTEXMultifunctionCounterAverageModeApertureTime;
            mfunction->Counter->Channels->Item["CH2"]->ApertureTime = 1;
//aperature time in seconds

             mfunction->Trigger->Source = VTEXMultifunctionTriggerSourceImmediate;

             //Start measurement
           mfunction->Measurement->Abort();
             mfunction->Measurement->Initiate();
             int fifocount,loopcount = 0;
             do
             {
```

```
                  //Take measurement for 10 times.
                  fifocount = mfunction->Measurement->FifoCount;
                  Sleep(1000);
               if(loopcount > 10)
               {
                   printf("Expected Fifo count (10) is not Available, Exiting
loop\n");
                   break;
               }
               loopcount++;
             } while (fifocount < 10);

           printf("Fifo Count = %d\n",fifocount);
           if(fifocount <= 0)
           {  //Exiting function when no data available
              printf("No Fifo data, Exiting function...\n");
              return 0;
           }
             //No. of enabled channels(here it is two. Channel 1 and channel 2)
             int no_of_channels_enabled = 2;
             /*For each trigger all channels data will be stored. In our case 2
channels is enabled and 10 readings are taken.
               So we will get 20 data points in the data array*/
             SAFEARRAY *data = NULL;
             SAFEARRAY *time = NULL;
             SAFEARRAY *time_fraction = NULL;
             //Read the measurements
             mfunction->Measurement->ReadFifo(0, fifocount, &data, &time,
&time_fraction);
             for (long scan = 0; scan<fifocount; scan++)
             {
              double dSecond, dFraction;
              SafeArrayGetElement(time, &scan, (void *)&dSecond);
              SafeArrayGetElement(time_fraction, &scan, (void *)&dFraction);
                 fprintf(stderr, "@Time = %f\n", dSecond + dFraction);
                 for (int channel = 0; channel < no_of_channels_enabled; channel++)
                 {
                      long dataindex = scan * no_of_channels_enabled + channel;
                   double dData;
                   SafeArrayGetElement(data, &dataindex, (void *)&dData);
                      fprintf(stderr, "Channel  = %d, %s = %f\n", channel+1,
(channel==0)?"Time interval":"Frequency", dData);
                   //Channel 1 is measuring time interval and channel 2 is measuring
Frequency
                 }

             }
             //Abort the measurement.
             mfunction->Measurement->Abort();

       }
       catch (_com_error& e)
       {
          ::MessageBox(NULL, e.Description(), e.ErrorMessage(), MB_ICONERROR);
       }

       if (mfunction != NULL && mfunction->Initialized)
       {
          // Close driver
          mfunction->Close();
       }
   }
   catch (_com_error& e)
   {
```

```
        ::MessageBox(NULL, e.Description(), e.ErrorMessage(), MB_ICONERROR);
    }

    ::CoUninitialize();

    printf("\nDone - Press Enter to Exit");
     getchar();

    return 0;
}
```

PHASE DIFFERENCE FUNCTION

Time interval example can also be used to measure phase difference. To do so, change the function configured to "VTEXMultifunctionCounterFunctionPhase" (for example, mfunction-> Counter->Channels->Item["CH1"]->Function = VTEXMultifunctionCounterFunctionPhase).

QUADRATURE ENCODER FUNCTION

This example illustrates RPM measurement from TTL quadrature encoders using digital channels. To use analog quadrature encoders, change the counter input type to analog and set the threshold parameters accordingly.

Visual Studio C++

```
// Quadrature Encoder.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"

#import "IviDriverTypeLib.dll" no_namespace
#import "VTEXMultifunction.dll" no_namespace

//This example measures the RPM and shaft position(quadrature measurement) using TTL
quadrature encoders
int _tmain(int argc, _TCHAR* argv[])
{
    ::CoInitialize(NULL);

    try
    {
        IVTEXMultifunctionPtr mfunction(__uuidof(VTEXMultifunction));

        try
        {

             mfunction->Initialize("TCPIP::10.20.11.158::INSTR", VARIANT_TRUE,
VARIANT_TRUE, "");

             //Channel configuration. Note that coupling will NOT have any effect on
Digital inputs
           mfunction->Counter->Channels->Item["CH1"]->ConfigureInput(VARIANT_TRUE,
VTEXMultifunctionCounterInputTypeDigital,
              VTEXMultifunctionCounterInputModeSingleEnded, 0,
VTEXMultifunctionCounterCouplingDC);

             //Quadrature measurement setting
             mfunction->Counter->Channels->Item["CH1"]->Function =
VTEXMultifunctionCounterFunctionQuadrature;

             // Threshold limits setting for Digital signal measurements are not
required
```

```
            //This measures the frequency between 10000Hz and 1Hz.
            mfunction->Counter->Channels->Item["CH1"]->UpperFrequencyLimit = 10000;
//Maximum frequency of 10000Hz
            mfunction->Counter->Channels->Item["CH1"]->LowerFrequencyLimit = 1;
//Lower frequency of 1Hz


            //Aperture time of 1 seconds. This setting averages the measured frequency
for timespan of 10seconds
            mfunction->Counter->Channels->Item["CH1"]->AverageMode =
VTEXMultifunctionCounterAverageModeApertureTime;
            mfunction->Counter->Channels->Item["CH1"]->ApertureTime = 1;
//aperature time in seconds

            //Set Channel 2 as the reference channel[B signal of quadrature endcoder].
Channel 1 is for A signal
            mfunction->Counter->Channels->Item["CH1"]->ReferenceChannel = "CH2";

            //Using 1st index channel. Note that this refers to Index channel name not
measurement channel name.
            mfunction->Counter->Channels->Item["CH1"]->IndexChannel = "CH1";

            mfunction->Counter->Channels->Item["CH2"]->ConfigureInput(VARIANT_TRUE,
VTEXMultifunctionCounterInputTypeDigital,
                VTEXMultifunctionCounterInputModeSingleEnded, 0,
VTEXMultifunctionCounterCouplingDC);

            mfunction->Counter->Channels->Item["CH2"]->Function =
VTEXMultifunctionCounterFunctionRpm;
          //Enable Fifo for the channels
          mfunction->Counter->Channels->Item["CH1"]->FifoEnabled   = VARIANT_TRUE;
          mfunction->Counter->Channels->Item["CH2"]->FifoEnabled   = VARIANT_TRUE;

            //This setting determines the no. of pulses for each revolution. In our
case it is 32, So 32 pulses makes one full revolution
            mfunction->Counter->Channels->Item["CH2"]->RpmToothCount = 32;
            //No missing count would be added or subtracted from actual count
            mfunction->Counter->Channels->Item["CH2"]->RpmToothState =
VTEXMultifunctionCounterRpmToothStateNormal;

            //Aperture time of 1 sencond. It averages the RPM for timespan for every
second.
            mfunction->Counter->Channels->Item["CH2"]->AverageMode =
VTEXMultifunctionCounterAverageModeApertureTime;
            mfunction->Counter->Channels->Item["CH2"]->ApertureTime = 1;
//aperature time in seconds

            mfunction->Trigger->Source = VTEXMultifunctionTriggerSourceSoftware;

             //Start measurement
             mfunction->Measurement->Initiate();
             int fifocount;
             do
             {
             //Wait for 2 seconds for collecting no. of edges(rising and falling).
             printf("Waiting for 2 seconds\n");
             Sleep(2000);
             printf("sending software trigger\n");
             //send a software trigger for transferring data to get a snap short of
totalize count
             mfunction->Measurement->SendSoftwareTrigger();
             fifocount = mfunction->Measurement->FifoCount;
             } while (fifocount < 10);
```

```
            //No. of enabled channels(here it is two. Channel 1 and channel 2)
            int no_of_channels_enabled = 2;
            /*For each trigger all channels data will be stored. In our case 2
channels is enabled and 10 readings are taken.
              So we will get 20 data points in the data array*/
            SAFEARRAY *data = NULL;
            SAFEARRAY *time = NULL;
            SAFEARRAY *time_fraction = NULL;
            //Read the measurements
            mfunction->Measurement->ReadFifo(0, fifocount, &data, &time,
&time_fraction);
            for (long scan = 0; scan<fifocount; scan++)
            {
             double dSecond, dFraction;
             SafeArrayGetElement(time, &scan, (void *)&dSecond);
             SafeArrayGetElement(time_fraction, &scan, (void *)&dFraction);
                fprintf(stderr, "@Time = %f\n", dSecond + dFraction);
                for (int channel = 0; channel < no_of_channels_enabled; channel++)
                {
                     long dataindex = scan * no_of_channels_enabled + channel;
                  double dData;
                  SafeArrayGetElement(data, &dataindex, (void *)&dData);
                     fprintf(stderr, "Channel  = %d, %s = %f\n",
channel+1,(channel==0)? "Pulse count":"RPM", dData);
                }

            }
            //Abort the measurement.
            mfunction->Measurement->Abort();

       }
       catch (_com_error& e)
       {
          ::MessageBox(NULL, e.Description(), e.ErrorMessage(), MB_ICONERROR);
       }

       if (mfunction != NULL && mfunction->Initialized)
       {
          // Close driver
          mfunction->Close();
       }
   }
   catch (_com_error& e)
   {
      ::MessageBox(NULL, e.Description(), e.ErrorMessage(), MB_ICONERROR);
   }

   ::CoUninitialize();

   printf("\nDone - Press Enter to Exit");
    getchar();

   return 0;
}
```

DIGITAL I/O FUNCTION

This example illustrates the DIO functionality of EX1200-1538. Channel-1is configured as digital input and channel-2 is configured as digital output with inverse polarity. Channel-1 state will be read and print in the screen, and channel-2 state is set.

Visual Studio C++

```
// DIO example.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"

#import "IviDriverTypeLib.dll" no_namespace
#import "VTEXMultifunction.dll" no_namespace

//This example explains how to use Digital inputs and outputs

int _tmain(int argc, _TCHAR* argv[])
{
   ::CoInitialize(NULL);

   try
   {
      IVTEXMultifunctionPtr mfunction(__uuidof(VTEXMultifunction));

      try
      {
           mfunction->Initialize("TCPIP::10.30.1.16::INSTR", VARIANT_TRUE,
VARIANT_TRUE, "");
                  //Any channel can be configured as input or output

                  //Digital input configuration
                  mfunction->Dio->Channels->Item["DIO1"]->Direction =
VTEXMultifunctionDioDirectionInput;//Digital input operation
                  mfunction->Dio->Channels->Item["DIO1"]->Polarity =
VTEXMultifunctionDioPolarityNormal;// Data will be read without inverting
               printf("Channel 1: %s",(mfunction->Dio->Channels->Item["DIO1"]-
>Data)?"TRUE":"FALSE");

                  //Digital Output configuration
                  mfunction->Dio->Channels->Item["DIO2"]->Direction =
VTEXMultifunctionDioDirectionOutput;//Digital Output operation
                  mfunction->Dio->Channels->Item["DIO2"]->Polarity =
VTEXMultifunctionDioPolarityInverse;// Data will be inverted before writing
                mfunction->Dio->Channels->Item["DIO2"]->Data = VARIANT_FALSE; //This
will write the data as true, since this channel is being inverted
      }
      catch (_com_error& e)
      {
         ::MessageBox(NULL, e.Description(), e.ErrorMessage(), MB_ICONERROR);
      }

      if (mfunction != NULL && mfunction->Initialized)
      {
         // Close driver
         mfunction->Close();
      }
   }
   catch (_com_error& e)
   {
      ::MessageBox(NULL, e.Description(), e.ErrorMessage(), MB_ICONERROR);
   }
```

```
   ::CoUninitialize();

   printf("\nDone - Press Enter to Exit");
    getchar();

   return 0;
}
```

ANALOG OUTPUT FUNCTION

This example illustrates the DAC functionality of EX1200-1538. Channel-1is configured to produce voltage output and channel-2 is configured to produce current output. The channels are configured to produce 2 V and 10 mA respectively.

Visual Studio C++

```
// Analog Output.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include<iostream>

#import "IviDriverTypeLib.dll" no_namespace
#import "VTEXMultifunction.dll" no_namespace

using namespace std;
//This example explains how to program voltage and current output using Multifunction
DAC

int _tmain(int argc, _TCHAR* argv[])
{
   ::CoInitialize(NULL);

   try
   {
      IVTEXMultifunctionPtr mfunction(__uuidof(VTEXMultifunction));

      try
      {
           mfunction->Initialize("TCPIP::10.30.1.16::INSTR", VARIANT_TRUE,
VARIANT_TRUE, "");
         //Any channel can be configured for voltage or current
         mfunction->Dac->Channels->Item["DAC1"]->DriveMode =
VTEXMultifunctionDacDriveModeVoltage;
         mfunction->Dac->Channels->Item["DAC1"]->Voltage = 6.6;//Voltage in volts
         mfunction->Dac->Channels->Item["DAC1"]->Enabled = true;//Start giving out
the voltage

         cout<<mfunction->Dac->Channels->Item["DAC1"]->DriveMode<<endl;
         cout<<mfunction->Dac->Channels->Item["DAC1"]->Voltage<<endl;
         cout<<mfunction->Dac->Channels->Item["DAC1"]->Enabled<<endl;

         mfunction->Dac->Channels->Item["DAC2"]->DriveMode =
VTEXMultifunctionDacDriveModeCurrent;
         mfunction->Dac->Channels->Item["DAC2"]->Current = 0.003;//Current in Amperes
         mfunction->Dac->Channels->Item["DAC2"]->Enabled = true;//Start giving out
the voltage

      }
      catch (_com_error& e)
      {
         ::MessageBox(NULL, e.Description(), e.ErrorMessage(), MB_ICONERROR);
```

```
      }

      if (mfunction != NULL && mfunction->Initialized)
      {
         // Close driver
         mfunction->Close();
      }
   }
   catch (_com_error& e)
   {
      ::MessageBox(NULL, e.Description(), e.ErrorMessage(), MB_ICONERROR);
   }

   ::CoUninitialize();

   printf("\nDone - Press Enter to Exit");
    getchar();

   return 0;
}
```

SECTION 6

SFP OPERATION

INTRODUCTION

EX1200s offer an embedded web page which provides network configuration control, time configuration, and the ability to perform firmware upgrades. To facilitate discovery of the mainframe, VTI provides the **L**AN **In**strument **C**onnection and **U**pgrade (**LInC-U**) utility on the *VTI Instruments Corp. Drivers and Product Manuals CD* included with the EX1200 mainframe.

To open the embedded web page, start the **LInC-U** utility by navigating to **Start → Programs → VTI Instruments Corporation → LInC-U Utility → LInC-U Utility**. Once the utility is run, LInC-U will scan the network to discover all LAN-based VTI instruments. Once the scan is complete, the **Discovery Devices** tab will appear and show the instruments that were discovered, as shown in Figure 6-1. To open the web page, click on the hostname hyperlink in the **Discover Devices** tab. The IP address of the EX1200 can also be view from this window as well as its firmware version.

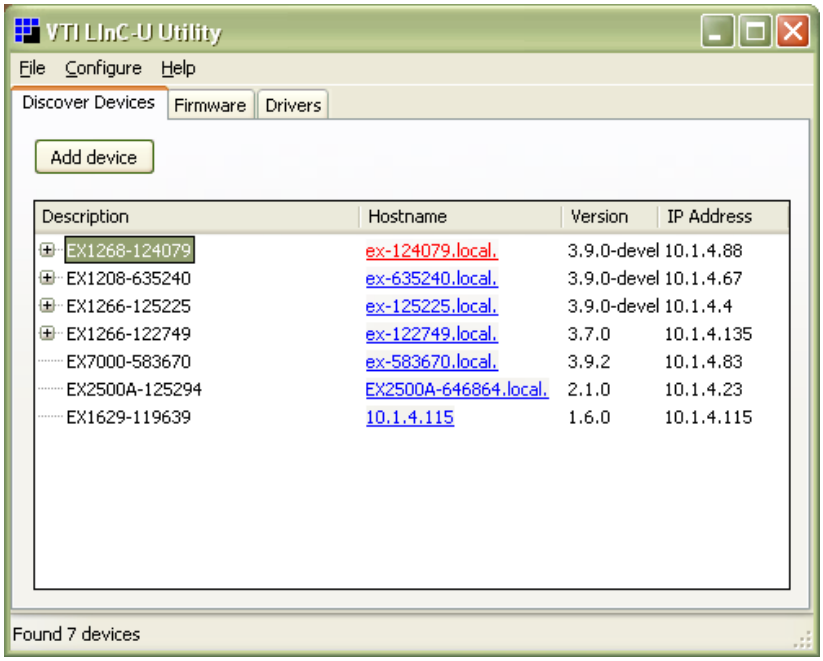

FIGURE 6-1: LINC-U DISCOVERY TAB WITH AN EX1268 SELECTED

Alternatively, the EX1200 may also be discovered using Internet Explorer's Bonjour for Windows plug-in, by entering the mainframe's IP address into the address bar of any web browser to view the embedded web page, or using VXI-11. For more information on discovery methods, refer to the *EX1200 Series User's Manual* (P/N: 82-0127-000).

GENERAL WEB PAGE OPERATION

When initial connection is made to the EX1200, the instrument home page, **Index**, appears (see Figure 6-2). This page displays instrument-specific information including:

- Model
- Manufacturer
- Serial Number
- Description
- LXI Class
- LXI Version
- Hostname
- MAC Address
- IP Address
- Netmask
- Instrument Address String
- Firmware Version
- IEEE-1588 Time

The **Index** is accessible from any other instrument page by clicking on the EX1200 web page header. The EX1200 **Command Menu** 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 **Index** 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 **Reboot** page is an example of an action page.

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

Use of the entry-type web pages in the EX1200 are governed by a common set of operational characteristics:

- Pages initially load with the currently-entered selections displayed.
- Each page contains a **Submit** button to accept newly entered changes. Leaving a page before submitting any changes has the effect of canceling the changes, leaving the instrument in its original state.
- Navigation through a parameter screen is done with the **Tab** key. The **Enter** key has the same function as clicking the **Submit** button and cannot be used for navigation.

Notes on Web Page Use

If a window needs to be resized, this should be done when the window opens. Resizing requires a refresh which causes the current state to be lost.

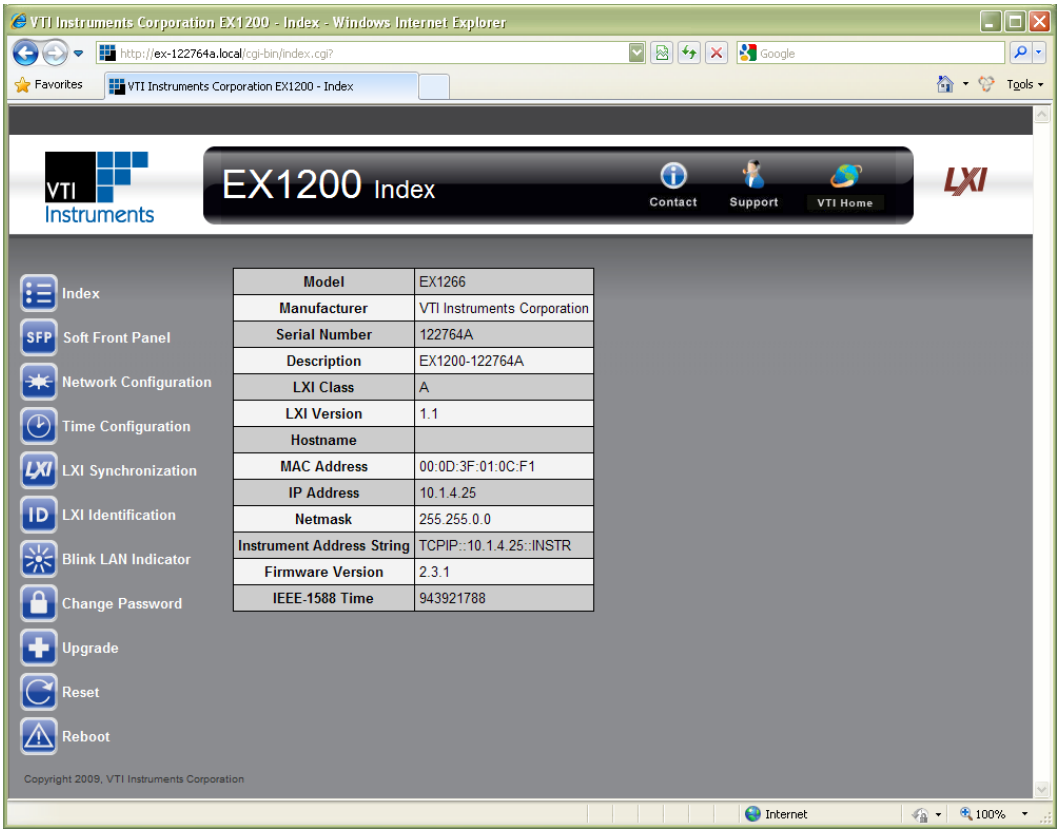

FIGURE 6-2: EX1200 MAIN WEB PAGE

VTI Instruments Logo

The VTI Instruments logo that appears on the upper left of all EX1200 web pages is a link to the VTI Instruments corporate website: http://www.vtiinstruments.com.

The remainder of this discussion will focus on the EX1200-1538 soft front panel. For more information on other EX1200 soft front panel elements, please refer to the *EX1200 Series User's Manual*.

EX1200-1538 SOFT FRONT PANEL

To navigate to the EX1200-1538 soft front panel, click on **Soft Front Panel** in the **Command Menu** (see Figure 6-3). Next, select **ex1200-1538 Multi Function** from the list of instruments installed in the EX1200.

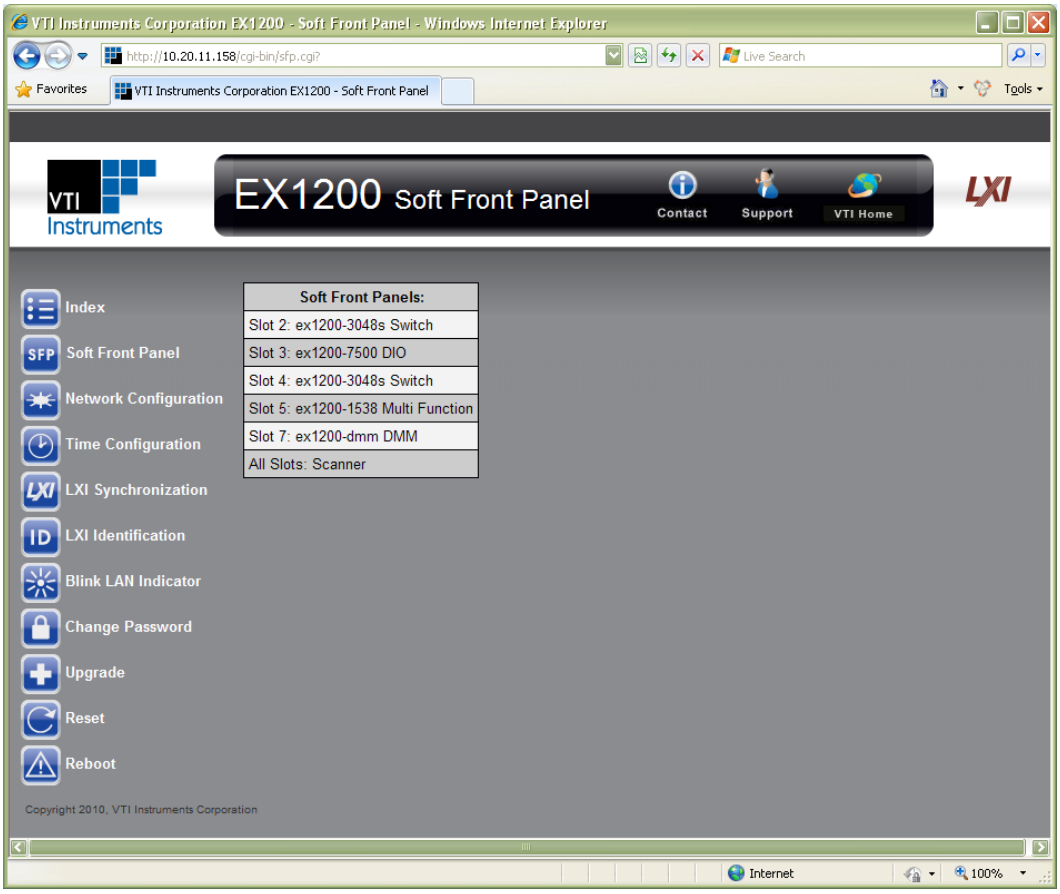

FIGURE 6-3: EX1200 SOFT FRONT PANEL MAIN PAGE

COUNTER CONTROL PAGE

By default, the EX1200-1538 SFP opens to the **Counter** page. From this view, the user can define a channel's function, input mode, aperture time, etc. A channel can only be configured with the **Enabled** checkbox is not selected. Once **Enabled** is selected for a channel, all configuration options are grayed out and cannot be modified.

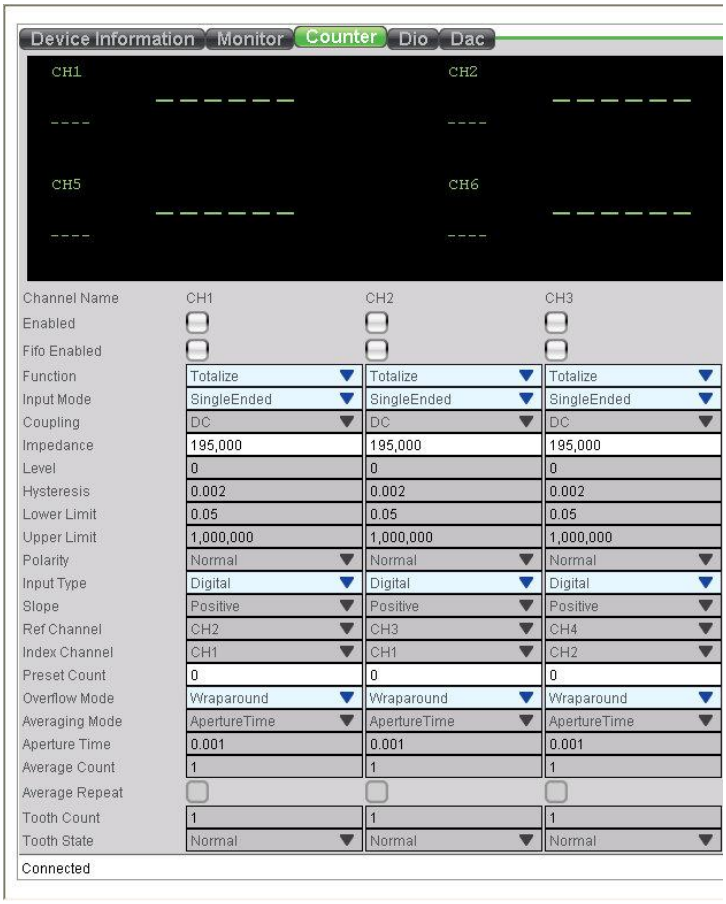

FIGURE 6-4: EX1200-1538 COUNTER PAGE (THREE CHANNELS SHOWN)

The following configuration options are provided for each counter channel. Note that some options may be dependent on the function selected or other parameters. Each parameter indicates when it is available for configuration.

- **Channel Name**: Indicates the name of the EX1200-1538 channel that is configured when the column parameters are modified.
- **Enabled**: When selected, this checkbox indicates that the counter channel is configured and ready for measurements. To modify any channel, the **Enabled** checkbox must be unselected.
- **FIFO Enabled**: When selected, data on the channel will be saved to FIFO memory and can be viewed using the **Data Log Table** on the **Monitor Page**.
- **Function**: Allows the user to select from one of the several functions offered by the counter. These functions are as follows: **Totalize, Frequency, Pulse Width**, **TimeInterval**, **Rpm**, **Quadrature**, **Duty Cycle**, **Phase**, **EdgeCount**, and **Period**. Each mode is discussed in detail in 0.
- **Input Mode**: Sets the channel for either **SingleEnded** or **Differential** operation. All channels are **SingleEnded** by default.
- **Coupling**: Sets the channel for either **DC** (default) or **AC** coupling mode. This parameter can only be configured for **Input Type** is set to **Analog**.

- **Impedance**: The input impedance for the channel. Default value is 195,000 ohms.
- **Level**: Sets the threshold level for an analog channel.
- **Hysteresis**: Sets the hysteresis level for an analog channel.
- **Lower Limit**: Sets the low frequency limit for measurement. Programmable from 0.05 to 1,000,000 (Hz). Not, the **Lower Limit** should be less than the **Upper Limit**. Not available for **Totalize**, **EdgeCount**, or **Quadrature** functions.
- **Upper Limit**: Sets the high frequency limit for measurements. 0.05 to 1,000,000 (Hz). Not available for **Totalize**, **EdgeCount**, or **Quadrature** functions**.**
- **Polarity**: Indicates whether the input signal will remain **Normal** (default) or if it will be **Inverted**.
- **Input Type**: Sets the channel as a **Digital** or **Analog** mode.
- **Slope**: Sets the channel for measurement on a **Positive** or **Negative** slope.
- **Ref Channel**: Sets the channel that the currently configured channel is referenced against when a comparative measurement is taken. Only available for the **Phase** and **Quadrature** functions.
- **Index Channel**: Indicates the index channel that will be used during a measurement comparison. Only available for **Quadrature** measurements.
- **Preset Count**: Sets the start value for counting functions. Only available for **Totalize**, **EdgeCount**, or **Quadrature** functions**.**
- **Overflow Mode**: Determines how a channel handles data if the count exceeds the maximum. Only available for **Totalize**, **EdgeCount**, or **Quadrature** functions**.**
  - **Stop**: When an overflow occurs, the instrument stops counting and returns an invalid value (NaN) as a result.
  - **Preset**: When an overflow occurs, the instrument counter rolls over and begins counting from the defined **Preset Count**.
  - **Wraparound** (default): When an overflow occurs, the instruments counter rolls over and begins counting from 0.
- **Averaging Mode**: Sets the channel to a specified averaging methodology. The user can select from **Aperture Time** or **Average Count**. Not available for **Totalize**, **EdgeCount**, or **Quadrature** functions**.**
- **Aperture Time**: Sets the duration over which the EX1200-1538 will make a measurement. Not available for **Totalize**, **EdgeCount**, or **Quadrature** functions. Also not available when **Average Count** is selected for **Averaging Mode**.
- **Average Count**: Sets the number of function events (i.e. periods, pulse widths, etc.) that will be counted and averaged before a value is returned.
- **Average Repeat**: When **Average Count** is selected, the **Average Count** checkbox can be selected. When selected, the EX1200-1538 average of consecutive, non-overlapping samples.
- **Tooth Count**: Indicates the nominal number of teeth on a toothed-wheel. Only available for **Rpm** measurements.
- **Tooth State**: Indicates whether the toothed-wheel has an **Extra** or **Missing** toothed-wheel as a point of reference. This allows for measurement compensation. The default for this parameter is **Normal**. Only available for **Rpm** measurements.

To begin a counter measurement, click on the **Monitor** button at the top of the **Counter** page. Once a measurement is initiated, the **LED Panel** will populate with the measurement values and indicate the type of measurement that was made on each channel, as shown in Figure 6-5. The **Monitor Page** is discussed in more detail below.

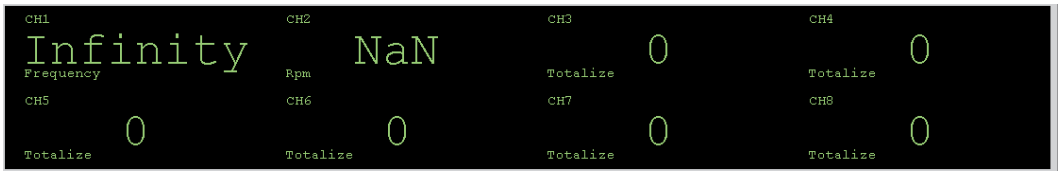

FIGURE 6-5: LED DISPLAY AFTER MEASUREMENT INITIATION

DIO CONTROL PAGE

By clicking on the **Dio** button on the SFP, the **DIO Control** page can be viewed where the DIO lines can be configured and enabled.

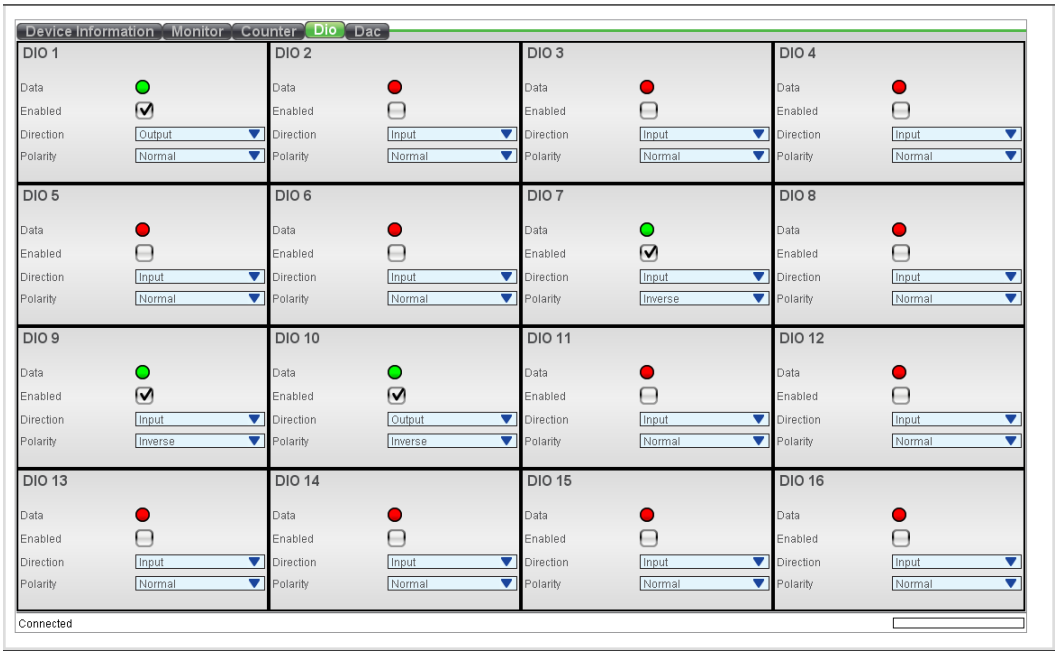

FIGURE 6-6: DIO CONTROL PAGE

- **Data LED**: Indicates the logical state of the DIO line. When the LED is green, the DIO line is a logical High. A red LED indicates a logical Low.
- **Enabled** checkbox: When selected, the channel acts as configured. For the DIO channels, it is automatically initiated once **Enabled**.
- **Direction**: Sets the DIO channel as either an **Input** or an **Output**.
- **Polarity**: Sets the DIO signal to be sent/received as being either **Inverted** or **Normal** (default).

DAC CONTROL PAGE

By clicking on the **DAC** button on the SFP, the **DAC Control** page can be viewed where the DAC lines can be configured and enabled.

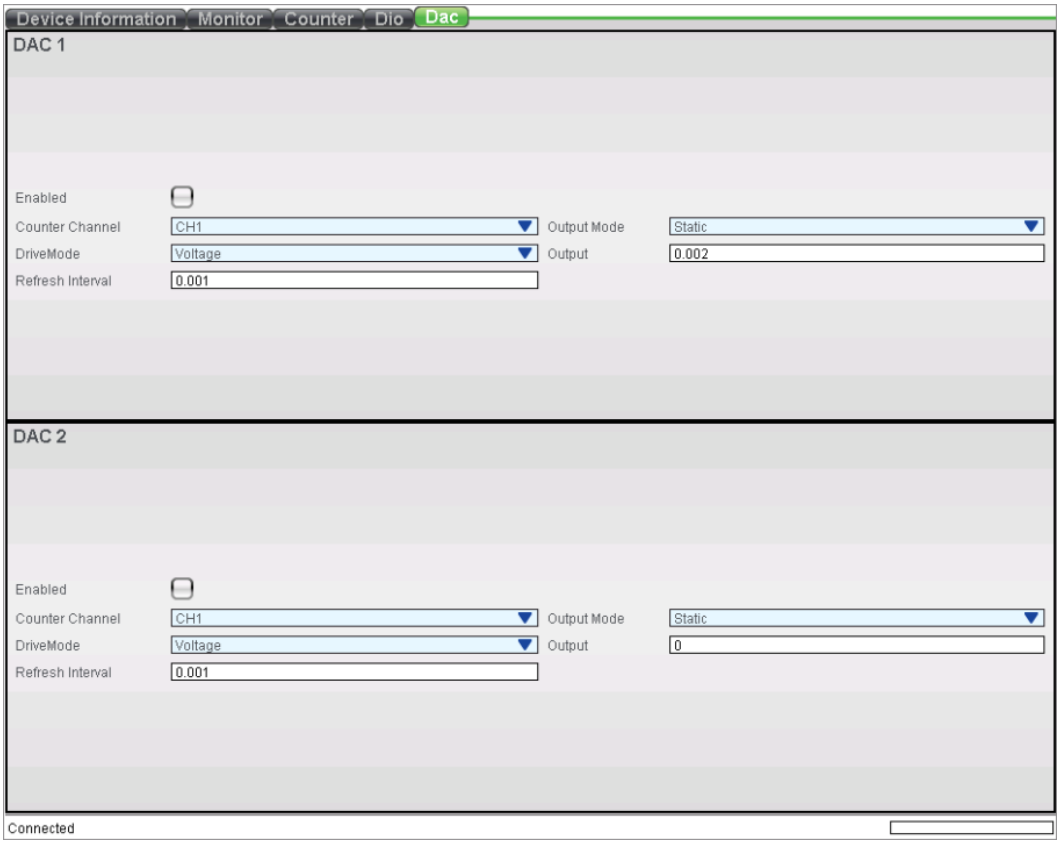

FIGURE 6-7: DAC CONTROL PAGE

- **Enabled** checkbox: When selected, the channel act as configured once the EX1200-1538 is initiated. Once a channel is **Enabled**, it is automatically initiated and its logical is monitored.
- **Drive Mode**: Sets the DAC channel as either a **Voltage** or **Current** output.
- **Output Mode**: Sets the DAC channel as either **Static** or **Dynamic**.
- **Counter Channel**: When the **Output Mode** is set to **Dynamic**, this parameter sets the counter channel for which the conversion will be performed.
- **Output**: When the **Output Mode** is set to **Dynamic,** this parameter sets the voltage/current level that is output by the DAC channel.
- **Refresh Interval**: When the **Output Mode** is set to **Dynamic**, this parameter sets the rate at which frequency is averaged and the counter channel is updated.

MONITOR PAGE

When the **Monitor** button of the EX1200-1538 web page is clicked, the **Monitor** page is viewed. From this page, the states of the counter, DIO, and DAC channels can be viewed and the EX1200-1538 can be initiated, data can be read, and card-level tasks independent of measurements can be performed, such as **Locking** and **Self-Test**.

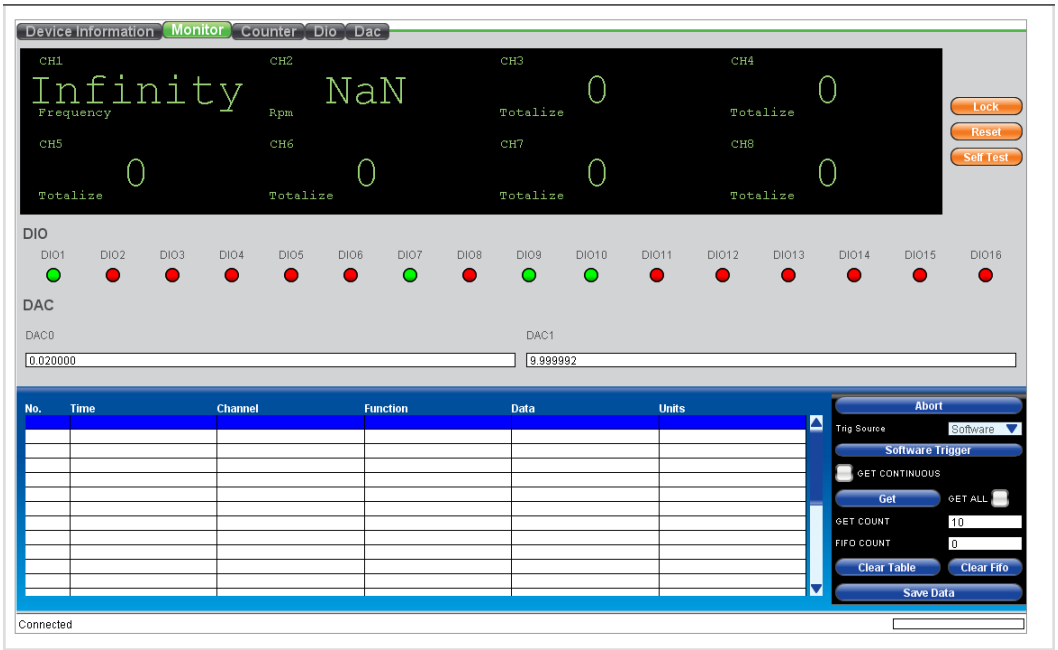

FIGURE 6-8: EX1200-1538 MONITOR PAGE

LED Panel

The **LED Panel** reflects shows the status of the EX1200-1538's counter channels. Using CH1 from Figure 6-8 as an example, the LED displays the channel (e.g. **CH1**), the return value (e.g. **Infinity**), and the channel's Function (e.g. **Frequency**). These fields are only populated when 1) the counter channel is enabled and 2) the EX1200-1538 has been initiated.

DIO Status Section

From the **DIO** section of the **Monitor** page, the user can determine the state of the DIO lines by examining the provided LEDs. When the LED is green, the DIO line is a logical High. A red LED indicates a logical Low. Status is returned independent of the **Enabled** state of the channel.

DAC Status Section

From the **DAC** section of the **Monitor** page, the user can view the output level of the two DAC channels. Status is returned independent of the **Enabled** state of the channel.

Data Log Table

Once a read of the FIFO is initiated, the data from FIFO becomes available in the **Event Log Table** at the bottom of the SFP.

- **No.**: Indicates the row number, for reference.
- **Time**: Indicates the IEEE-1588 time the event occurred.
- **Channel**: Indicates the channel where the measurement the Data was acquired at the specified **Time**. This data will be in hexadecimal format. For example, 0x1 (..0001) refers to the first channel, 0x2(..0010) refers to the second, channel, 2nd channel, etc.
- **Function**: Indicates the **Function** that the channel was set to when the **Data** was acquired.
- **Data**: Displays the value acquired by the **Channel** at the **Time** indicated.
- **Units**: Displays the unit of measure for the acquired **Data**.

Data Acquisition Section

When data is received, it is placed in FIFO memory and remains there until read from memory. The EX1200-7500 can be configured to read the FIFO data by using the **Data Acquisition Field**, which provides the following options.

- **Initiate/Abort** button: Selecting this button will start or stop data acquisition, depending on the current state of the EX1200-1538.
- **Trig Source**: Sets the external trigger source that will initiate/abort a data acquisition. The trigger can be **Immediate**, **Software**, or one of the EX1200 mainframe backplane lines (**BPL0** through **BPL7**).
- **Software Trigger**: Clicking on this button generates a software trigger that causes data the instrument to be initiated.
- **Get Continuous** checkbox: If enabled, this button disables the **Get** button and continually populates the data table as data is acquired.
- **Get** button: When clicked, data is immediately pulled from FIFO memory.
- **Get All** checkbox: When enabled, clicking the **Get** button will retrieve all of the current data. Once the FIFO is empty, it will stop acquiring retrieving data.
- **Get Count** field: Indicates the number of data points that will be returned when the **Get** button is clicked. Should be less than or equal to the **FIFO Count**.
- **FIFO Count** field: Indicates the number of data points available in the FIFO memory.
- **Clear Table** button: When clicked, all of the data currently in the table is erased.
- **Clear FIFO** button: When clicked, any data points currently stored in the FIFO memory will be erased.
- **Save Data** button: When clicked, the data in the table can be saved. A .csv file is generated by default.

Lock/Unlock Button

The **Lock** button requests (or releases) exclusive access to the EX1200-1538. If the function generator will be calibrated using the SFP, a lock should be established to prevent any unintended access to the instrument.

Reset Button

Clicking the **Reset** button returns the EX1200-1538 to its power-on default settings and values.

DEVICE INFORMATION PAGE

When the **Device Information** button is clicked, the user can access information regarding the EX1200-1538's version. This information includes the revision of the soft front panel, the firmware revision, the FPGA revision, and the hardware revision.

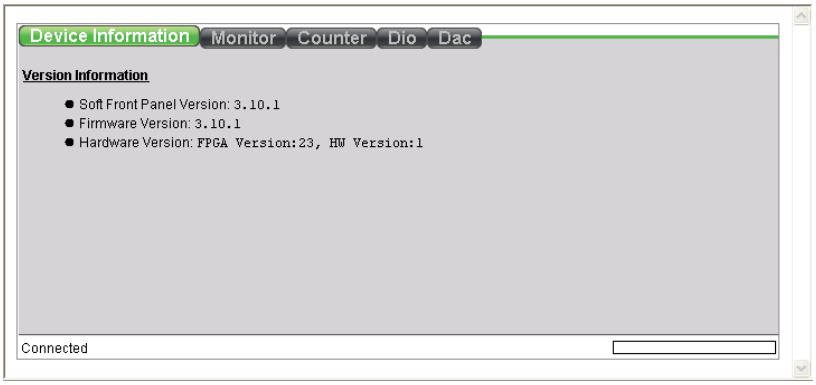

FIGURE 6-9: DEVICE INFORMATION PAGE

INDEX

A

accessories....12
  crimp tool....12
  mating connector....12
  mating terminal block....12
  terminal block....18
  unterminated wiring harness....12
accuracy calculations....13
  analog measurement using digital input....13
  duty cycle....14
  frequency measurement using analog input....13
  frequency measurement using digital input....13
  period measurement using digital input....13
  pulse width measurements using digital and analog inputs....14
  time interval using analog and digital inputs....14
analog output
  circuit protection....35
  Dynamic Update mode....34
  parallel operation....35
analog output function....55
analog output operation....34
aperture time mode....25
average count mode....26

C

calibration....19
connector pin/signal assignment....16
cooling....15
counter/timer
  aperture average count mode....26
  aperture time mode....25
  duty cycle measurements....28
  edge counting....25
  frequency measurements....28
  functions....24
  input coupling....22
  inputs....22
  overview....21
  period measurements....25
  phase measurements....31
  polarity conversion....24
  pulse width measurements....27
  quadrature measurements....31
  RPM measurements....29
  signal conversion....23
  time interval measurements....30
  totalizing....24
counter/timer operation....21–32

D

DAC....*See* analog output
digital I/O
  circuit protection....35
digital I/O function....54
digital I/O operation....33
DIO....*See* digital I/O
dynamic update mode formulas....34
  current formula....35
  voltage formula....35

E

edge counting function....42

F

firmware version....58
frequency function....42
front panel....17

I

IEEE-1588 time....58
index web page....58
initializing\closing the instrument....37
IP address....58

J

jitter....28

L

LAN Instrument Connection and Upgrade utility....57
LXI class....58
LXI version....58

M

moving average....*See* average count mode

O

option strings....38

P

phase difference function....51
plug-in module
  installation....15
power....15
power consumption....15
programming....37
  analog output function....55
  closing....37
  digitial I/O function....54
  edge counting function....42
  frequency function....42
  initializing....37
  option strings....38
  phase difference function....51
  Quadrature encoder function....51
  related software compnents....37
  RPM function....45
  time interval function....48
  totalize....39
  using the driver....37

Q

Quadrature encoder function....51

R

revolutions per minute....*See* RPM
RPM....29
  tooth count parameter....29
RPM function....45

S

simple averaging....*See* aperture time mode
soft front panel
  counter control page....61

DAC control page....64
DAC status....65
data acquisition data....66
data log table....66
device information....67
DIO control page....63
DIO status....65
LED panel....65
lock....66
monitor page....65
reset....66
specifications....11
analog output....12
counter input....11
DIO....12
general....11
support resources....7
system power requirements....15

T

terminal block....18, *See* accessories
terminal block receiver....18
time interval function....48
totalize function....39

U

unpacking....15

W

warm-up....16
WEEE....5