VM1602

TIME STAMP MODULE

USER'S MANUAL

82-0024-000
Release March 31, 2003

VXI Technology, Inc.

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

TABLE OF CONTENTS

CERTIFICATION

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

WARRANTY

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

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

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

LIMITATION OF WARRANTY

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

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

RESTRICTED RIGHTS LEGEND

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

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

DECLARATION OF CONFORMITY

Declaration of Conformity According to ISO/IEC Guide 22 and EN 45014

MANUFACTURER'S NAME	VXI Technology, Inc.
MANUFACTURER'S ADDRESS	2031 Main Street Irvine, California 92614-6509
PRODUCT NAME	Time Stamp Module
MODEL NUMBER(S)	VM1602
PRODUCT OPTIONS	All
PRODUCT CONFIGURATIONS	All

VXI Technology, Inc. declares that the aforementioned product conforms to the requirements of the Low Voltage Directive 73/23/EEC *and the* EMC Directive 89/366/EEC *(inclusive* 93/68/EEC*) and carries the "CE" mark accordingly. The product has been designed and manufactured according to the following specifications:*

SAFETY	EN61010 (2001)
EMC	EN61326 (1997 w/A1:98) Class A CISPR 22 (1997) Class A VCCI (April 2000) Class A ICES-003 Class A (ANSI C63.4 1992) AS/NZS 3548 (w/A1 & A2:97) Class A FCC Part 15 Subpart B Class A EN 61010-1:2001

The product was installed into a C-size VXI mainframe chassis and tested in a typical configuration.

I hereby declare that the aforementioned product has been designed to be in compliance with the relevant sections of the specifications listed above as well as complying with all essential requirements of the Low Voltage Directive.

March 2003

CE

Jerry Patton

Jerry Patton, QA Manager

General Safety Instructions

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

Service should only be performed by qualified personnel.

Terms and Symbols

These terms may appear in this manual:

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

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

These symbols may appear on the product:

⚠ ATTENTION - Important safety instructions

⏚ Frame or chassis ground

Warnings

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

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

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

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

WARNINGS (CONT.)

Avoid Electric Shock

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

Ground the Product

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

Operating Conditions

To avoid injury, electric shock or fire hazard:

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

Improper Use

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

SUPPORT RESOURCES

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

Internet Support

E-mail: support@vxitech.com
Web Address: http://www.vxitech.com

Telephone Support (U.S.)

Tel:	(949) 955-1894	**West Coast**
	(216) 447-8950	**East Coast**
Fax:	(949) 955-3041	**West Coast**
	(216) 447-8951	**East Coast**

VXI Technology Headquarters

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

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

SECTION 1

INTRODUCTION

INTRODUCTION

The VM1602 is a high-performance time stamp module that has been designed for accurate relative time measurements and flexibility of configuration. The instrument uses the message-based word serial interface for programming and data retrieval. The VM1602 command set conforms to the SCPI standard for consistency and ease of programming.

The VM1602 is a member of the VXI Technology VMIP™ (VXI Modular Instrumentation Platform) family and is available as a 32-, 64- or 96-channel, single-wide VXIbus instrument. Figure 1-1 shows the 96-channel version of the VM1602. The 64-channel version would not have J200 and its associated LEDs and nomenclature while the 32-channel version would eliminate J202 as well. In addition to these three standard configurations, the VM1602 may be combined with any of the other members of the VMIP family to form a customized and highly integrated instrument (see Figure 1-1). This allows the user to reduce system size and cost by combining the VM1602 with two other instrument functions in a single-wide C-size VXIbus module.

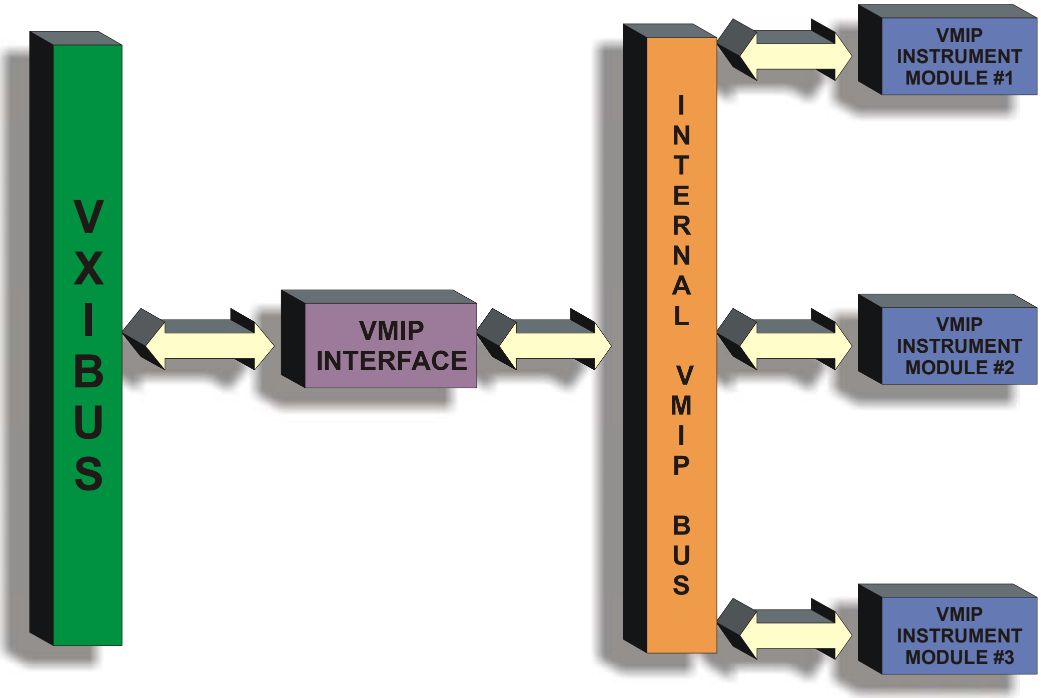

VXIBUS
VMIP INTERFACE
INTERNAL VMIP BUS
VMIP INSTRUMENT MODULE #1
VMIP INSTRUMENT MODULE #2
VMIP INSTRUMENT MODULE #3

FIGURE 1-1 VMIP™ PLATFORM

Regardless of whether the VM1602 is configured with other VM1602 modules or with other VMIP modules, each group of 32 channels is treated as an independent instrument in the VXIbus chassis. Each group has its own FAIL and ACCESS light.

DESCRIPTION

The VM1602 is a high-performance time stamp recording module. The instrument monitors 32 differential inputs for either rising or falling edges as programmed and records their relative time of occurrence. The instrument can collect a total of 128 k events or optionally 512 k events when the extended memory option is installed.

Each input may be configured for either differential or single-ended operation. In differential mode, a high input occurs when the positive input of a channel is at a higher potential than the negative input. In single-ended mode, a high input occurs when the positive input of a channel is at a higher potential than the programmed threshold voltage corresponding to that channel.

In addition to accepting inputs from the front panel, the VM1602 can monitor all of the VXIbus TTL Trigger bus lines. In order to keep the hardware to a reasonable level of complexity, TTL Trigger 0 may be routed to Channels 1, 2, 17 or 18. Similarly, TTL Trigger 1 may be routed to Channels 3, 4, 19, 20, etc. To further enhance the flexibility of the instrument, all even-numbered channels may be routed to their N-1 channel input. This allows both the rising and falling edges of a signal to be time stamped. For example, Channel 5 may be routed to TTL Trigger bus 2 and set to stamp on the rising edge and Channel 6 may be routed to Channel 5 and set to stamp on the falling edge. In this case, both rising and falling edges of the TTL Trigger bus 2 will be recorded.

The VM1602 allows each channel to be enabled or disabled. This prevents unused channels that may have signal activity from creating unwanted time stamp events. When a channel is enabled, a 1 value is recorded in RAM, in the corresponding channel bit position, every time an event occurs on that channel (a rising or falling edge as is programmed for that channel). The firmware of the instrument scans the RAM's data and reports relative times between events on the same or different channels. When a channel is disabled, the logic state of the input is recorded whenever an enabled channel has a time stamp event. This allows the disabled channels to function as a rudimentary logic analyzer.

Each time an enabled channel has a time stamp event, the value in a 40-bit counter is recorded along with the channel or channels which caused the event to occur. The rate at which the counter increments sets the overall resolution of the instrument and may be programmed for 1 μs, 10 μs, 100 μs or 1 ms.

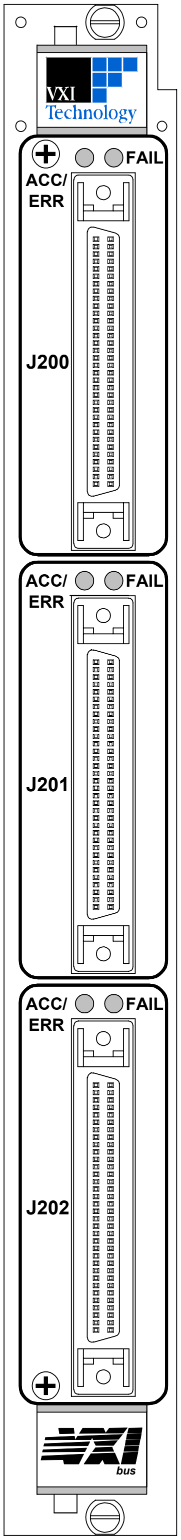


FIGURE 1-2 FRONT PANEL LAYOUT

VM1602 Specifications

General Specifications	
Channels	
VM1602-1	32
VM1602-2	64
VM1602-3	96
Input Sources	
	Front Panel TTL Trigger Bus
Input Type	
	Differential or Single-Ended
Input Range	
	±5 V
Input Impedance	
	20 kΩ Single-Ended 40 kΩ Differential
Input Threshold	
	±5 V, 39 mV resolution
Threshold Accuracy	
	±2% of range
Time Stamp Clock Source	
	VXIbus Clock 10
Clock Accuracy	
	± 0.01%, may be improved by supplying a superior clock source to the slot 0 controller
Time Stamp Clock Period	
	1 µs, 10 µs, 100 µs or 1 ms
Time Stamp Accuracy	
	±½ time stamp clock period
Maximum Number of Events	
	128 k events standard 512 k events optional
Time Stamp Counter Rollover	
	40-bit counter, 12.7 days with 1 µs resolution
Power Requirements	
VM1602-1	+5 V @ 2.15 A, -5.2 V @ 0.40 A
VM1602-2	+5 V @ 3.56 A, -5.2 V @ 0.75 A
VM1602-3	+5 V @ 4.97 A, -5.2 V @ 1.10 A
Cooling Requirements	
	See Power Cooling Table
Manufacturer's ID	
	3915
Module Model Code	
	265

SECTION 2

PREPARATION FOR USE

INSTALLATION

When the VM1602 is unpacked from its shipping carton, the contents should include the following items:

(1) VM1602 VXIbus module
(1) VM1602 Time Stamp Module User's Manual (this manual)

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

Once the VM1602 is assessed to be in good condition, it may be installed into an appropriate C-size or D-size VXIbus chassis in any slot other than slot zero. The chassis should be checked to ensure that it is capable of providing adequate power and cooling for the VM1602. Once the chassis is found adequate, the VM1602's logical address and the backplane jumpers of the chassis should be configured before the VM1602's installation.

CALCULATING SYSTEM POWER AND COOLING REQUIREMENTS

It is imperative that the chassis provide adequate power and cooling for this module. Referring to the chassis user's manual, confirm that the power budget for the system (the chassis and all modules installed therein) is not exceeded and that the cooling system can provide adequate airflow at the specified backpressure.

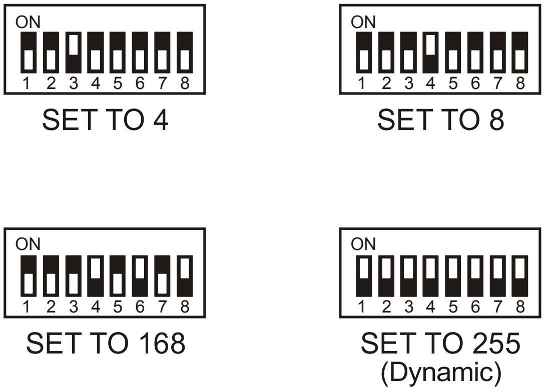
⚠ It should be noted that if the chassis cannot provide adequate power to the module, the instrument may not perform to specification or possibly not operate at all. In addition, if adequate cooling is not provided, the reliability of the instrument will be jeopardized and permanent damage may occur. Damage found to have occurred due to inadequate cooling would also void the warranty of the module.

Setting the Chassis Backplane Jumpers

Please refer to the chassis user manual for further details on setting the backplane jumpers.

Setting the Logical Address

The logical address of the VM1602 is set by a single 8-position DIP switch located near the module's backplane connectors (this is the only switch on the module). The switch is labeled with positions 1 through 8 and with an ON position. A switch pushed toward the ON legend will signify a logic 1; switches pushed away from the ON legend will signify a logic 0. The switch located at position 1 is the least significant bit while the switch located at position 8 is the most significant bit. See figure 2-1 for examples of setting the logical address switch.



Switch Position	Switch Value
1	1
2	2
3	4
4	8
5	16
6	32
7	64
8	128

Figure 2-1 Logical Address Switch Setting Examples

The VMIP may contain three separate instruments and will allocate logical addresses as required by the VXIbus specification (revisions 1.3 and 1.4). It is necessary that the address of the first instrument (the instrument closest to the top of the module) be set at an address which is divisible by 4 and not set to 0. Switch positions 0 and 1 must always be set to the OFF position. Therefore only addresses of 4, 8, 12, 16, ... 252 are allowed. The address switch should be set for one of these legal addresses and the address for the second instrument (the instrument in the center position) will automatically be set to the switch set address plus one; while the third instrument (the instrument in the lowest position) will automatically be set to the switch set address plus two.

If dynamic address configuration is desired, the address switch should be set for a value of 255. Upon power-up, the slot 0 resource manager will assign logical addresses to each instrument in the VMIP module.

FRONT PANEL INTERFACE WIRING

The VM1602's serial interface is made available on the front panel of the instrument. The 32-channel version (VM1602-1) will have J201 that contains all signals for this instrument. The 64-channel version (VM1602-2) will have J201 and J202 provided, while the 96-channel version (VM1602-3) will have J200, J201 and J202. The wiring for each of these connectors is identical and since each group of four channels is treated as a separate instrument, the module will have three Channel 1s, three Channel 2s, three Channel 3s, etc.

The connector used in the VM1602 is a commonly available 68-pin high-density type commonly known as a 68-pin version of the SCSI 2 connector. The mating connector is an IDC (Insulation Displacement Connector) component and is available from a variety of sources. The connector attaches to two 34-conductor 0.050 centers ribbon cable. Some manufacturers also allow the use of discrete 30 GA stranded wire. Contact the factory for more connector information.

The pin locations for J200, J201 and J202 are shown in Figure 2-2 and the pinout reference is shown in Table 2-1.

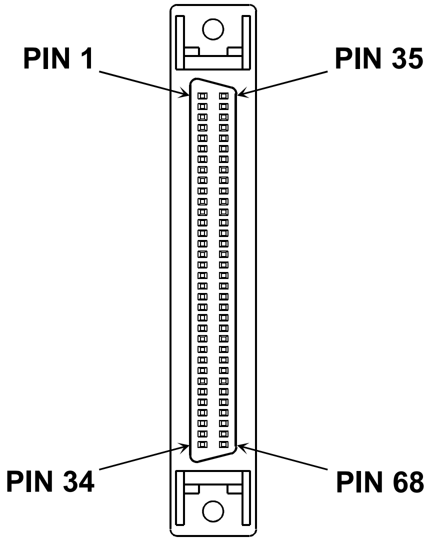


FIGURE 2-2 J200, J201 AND J202 PIN LOCATIONS

TABLE 2-1 J200, J201 AND J202 PINOUT

SIGNAL	FUNCTION	PIN	SIGNAL	FUNCTION	PIN
CH1+	INPUT	1	CH17+	INPUT	35
CH1-	INPUT	2	CH17-	INPUT	36
CH2+	INPUT	3	CH18+	INPUT	37
CH2-	INPUT	4	CH18-	INPUT	38
CH3+	INPUT	5	CH19+	INPUT	39
CH3-	INPUT	6	CH19-	INPUT	40
CH4+	INPUT	7	CH20+	INPUT	41
CH4-	INPUT	8	CH20-	INPUT	42
CH5+	INPUT	9	CH21+	INPUT	43
CH5-	INPUT	10	CH21-	INPUT	44
CH6+	INPUT	11	CH22+	INPUT	45
CH6-	INPUT	12	CH22-	INPUT	46
CH7+	INPUT	13	CH23+	INPUT	47
CH7-	INPUT	14	CH23-	INPUT	48
CH8+	INPUT	15	CH24+	INPUT	49
CH8-	INPUT	16	CH24-	INPUT	50
GND	REF	17	GND	REF	51
CH9+	INPUT	18	CH25+	INPUT	52
CH9-	INPUT	19	CH25-	INPUT	53
CH10+	INPUT	20	CH26+	INPUT	54
CH10-	INPUT	21	CH26-	INPUT	55
CH11+	INPUT	22	CH27+	INPUT	56
CH11-	INPUT	23	CH27-	INPUT	57
CH12+	INPUT	24	CH28+	INPUT	58
CH12-	INPUT	25	CH28-	INPUT	59
CH13+	INPUT	26	CH29+	INPUT	60
CH13-	INPUT	27	CH29-	INPUT	61
CH14+	INPUT	28	CH30+	INPUT	62
CH14-	INPUT	29	CH30-	INPUT	63
CH15+	INPUT	30	CH31+	INPUT	64
CH15-	INPUT	31	CH31-	INPUT	65
CH16+	INPUT	32	CH32+	INPUT	66
CH16-	INPUT	33	CH32-	INPUT	67
GND	REF	34	GND	REF	68

SECTION 3

PROGRAMMING

INTRODUCTION

The VM1602 module is a VXIbus message-based device whose command set is compliant with the SCPI (*Standard Command for Programmable Instrument*) programming language. See the Sample Program later in this section for specific programming examples and command usage. In addition, there are individual command descriptions located in the *Command Dictionary* section.

All module commands are sent over the VXIbus backplane to the module. Commands may be in upper, lower or mixed case. All numbers are sent in ASCII decimal unless otherwise noted.

The module recognizes SCPI commands. SCPI is a tree-structured language based on IEEE-STD-488.2 Specifications. It uses the IEEE-STD-488.2 Standard command, and the device dependent commands are structured to allow multiple branches off the same trunk to be used without repeating the trunk. To use this facility, terminate one branch with a semicolon and start the next branch with a colon. See the *Standard Command for Programmable Instruments (SCPI) Manual, Volume 1: Syntax & Style, Section 6*, for more information.

The SCPI commands in this section are listed in upper and lower case. Character case is used to indicate different forms of the same command. Keywords can have both a short form and a long form (some commands only have one form). The short form uses just the keyword characters in uppercase. The long form uses the keyword characters in uppercase plus the keyword characters in lowercase. Either form is acceptable. Note that there are no intermediate forms. All characters of the short form or all characters of the long form must be used. Short forms and long forms may be freely intermixed. The actual commands sent can be in upper case, lower case or mixed case (case is only used to distinguish long form and short form for the user). As an example, these commands are all correct and all have the same effect:

```
stat:oper:enab 0
stat:operation:enab 0
stat:oper:enable 0
stat:operation:enable 0
status:oper:enab 0
status:operation:enab 0
status:oper:enable 0
status:operation:enable 0
```

The following command is **not** correct because it uses part of the long form of STATus, but not all letters of the long form.

stat~~u~~:oper:enab 0 – incorrect syntax

All of the SCPI commands also have a query form unless otherwise noted. Query forms contain a question mark (?). The query form allows the system to ask what the current setting of a parameter is. The query form of the command generally replaces the parameter with the question mark. Query responses do not include the command header. This means only the parameter is returned; no part of the command is returned.

When character data is used for a parameter, both short and long forms are recognized. If the command has a query form with character response data, the short form is always returned in upper case. As an example, to find out what the current input source is for Channel 2, use the following command:

```
INP:SOUR? 2
```

The response could be:

```
FPAN
```

This tells the user that the Channel 2 input source is FPAN.

Multiple commands can also be combined on one line. To do this, terminate one command with a semicolon and start the next command with a colon. As an example, Channel 1 through Channel 3 input polarity and source could be set as follows:

```
INP:POL RIS;SOUR FPAN,(@1:3)
```

When combining commands, keep in mind the size of the input buffer. Command lines that are too long will generate an error and not be used.

The IEEE-STD-488.2 Common Commands can be placed anywhere set off from the rest of the command by a semicolon. They can also be placed alone on a line. For example:

```
*RST;INP:POL RIS;SOUR FPAN,(@1:3)
```

NOTATION

Keywords or parameters enclosed in square brackets ([ ]) are optional. If the optional part is a keyword, the keyword can be included or left out. Omitting an optional parameter will cause its default value to be used.

Parameters are enclosed by angle brackets (< >). Braces ({ }) are used to enclose one or more parameters that may be included zero or more times. A vertical bar (|), read as “or”, is used to separate parameter alternatives.

EXAMPLES OF SCPI COMMANDS

ABORT

The abort command will cause the instrument to stop collecting time stamp readings if the unit is currently running. It is important to note that this command does not change any other instrument settings. The data collected before the abort command is the available for query.

ABORt	***No command parameters***

EXAMPLES	
`ABORt`	*Stops the data collection in progress*

EVENT:COUNT?

The event count query returns the total number of registered events that have occurred on all of the selected channels. This query when used with the optional indices will return the number of events that happened between the lower and upper indices. The index values represent a specific time in which an event occurred. Index_1 represents the lower time and index_2 represents the upper time. The event count query is selected using the following SCPI command.

EVENt:COUNt? <index_1>,<index_2>[,<channel_list>]

Where <index_1>, <index_2> is a numeric value of 0 to 131,071

Where [<channel_list>] is standard channel list syntax forChannels 1 through 32

EXAMPLES

`EVEN:COUN?`
`105798`
	Since no index values or channel list is specified the query returns the total number of events that have occurred

`EVEN:COUN? (@2,4,6:15)`
`65`
	Returns the total number of events that have occurred forChannels 2, 4 and 6 through 15

`EVEN:COUN? 756,1024,(@9:27)`
`33`
	Returns the total number of events that have occurred forChannels 9 through 27 between the two index times

EVENT:DATA?

This query is used with the indices to return the source of events that happened between the lower and upper indices. The event data query returns a 32-bit number that correlates to the source or sources of the events. When index 1 is the only parameter specified the query will report the source of a single index point. The returned value of 4,294,967,295 equals all 32 channels. The event data query is selected using the following SCPI command.

EVENt:DATA? <index_1>[,<index_2>]

Where <index_1>, [<index_2>] are numeric values from 0 to 131,071

EXAMPLES

```
EVEN:DATA? 1028
256
```

Since index 1 is the only index specified, the returned value is a single point source of the channel that triggered the event. The returned value represents Channel 9 as the source

```
EVEN:DATA? 256,258
1024,13,1072,56
```

Returns the source of events that have occurred between index values 256, 257 and 258. The returned value represents Channels 11, 18 and 4, 5 and 6 as the sources respectively

EVENT:TIME?

This query is used to return the event number that occurred at a specified time. The event time query returns a value from 0 to 131,071 (524,287 if 512 k RAM option is installed) which correlates to the event or events address in RAM. The time parameter must be an exact match or an error is generated. The event time query is selected using the following SCPI command.

EVENt:TIMe? <time>	***Where <time>, is a numeric value of 0 to 1,099,511.62778 seconds***

EXAMPLES

`EVEN:TIM? 1.00566`	*Returns the location in RAM where the event*
`15,753`	*data is stored for the time of 1.00566*

Assuming that an event occurred at the time of 0.00276

`EVEN:TIM? 0.00275`
`“”`
	The “” designate that an error was generated. This is because no event occurred at the specified time

EVENT:TIME:NEXT?

This query is used to return the next event data occurring after the specified time. The event time next query returns a value from 0 to 4,294,967,295 that correlates to the event or events' number. The channel list parameter is used to exclude unwanted events during the search. The returned event data includes all channels regardless of the mask. The event time next query is selected using the following SCPI command.

EVENt:TIMe:NEXT? <time>[,<channel_list>]

Where <time>, is a numeric value of 0 to 1,099,511.62778 seconds.

Where [<channel_list>] is standard channel list syntax forChannels 1 through 32

EXAMPLES

```
EVEN:TIM:NEXT? 0
8,192
```

All channels are evaluated during the search. The returned value of 8,192 (Channel 14) is the first occurring event or events after the trace began.

```
EVEN:TIM:NEXT? 1.00000,(@3)
36
```

This command will search for the first occurrence of Channel 3 causing an event after the 1 second of run time. The returned value of 36 shows that Channel 3 and 6 triggered this event.

EVENT:TIME:PREVIOUS?

This query is similar to the event time next query, the difference being that it returns the event that occurred before the specified time. The event time previous query returns a value from 0 to 4,294,967,295 that correlates to the event or event's number. The channel list is used to exclude unwanted events during the search. The returned event data includes all channels regardless of the mask. The event time previous is selected using the following SCPI command:

EVENt:TIMe:PREVious? <time>[,<channel_list>]	***Where <time>, is a numeric value of 0 to 1,099,511.62778 seconds.*** ***Where [<channel_list>] is standard channel list syntax for Channels 1 through 32***

EXAMPLES

`EVEN:TIM:PREV? 60.00000`
`8,388,608`
	All channels are evaluated during the search. The returned value of 8,388,608 (Channel 24) is the last event or events occurring before one minute of execution time

`EVEN:TIM:PREV? 300.00000,(@12,15)`
`18,433`
	This command will search for the first occurrence of Channels 12 and 15 causing an event before 5 minutes of run time. The returned value of 18,433 shows that Channel 1, 12 and 15 triggered this event

FREQUENCY:DELTA?

This query is used with the indices to return the equivalent frequency for the time period between the two indices. The frequency returned is in hertz and is calculated as 1/t. This command enables the time stamp recorder to act as a low frequency counter. The event data query is selected using the following SCPI command.

FREQuency:DELTa? <index_1>[, <index_2>]	***Where <index_1>, [<index_2>] is a numeric value of 0 to 131,071***

EXAMPLES

`FREQ:DELT? 1028,-1`
`0.00005`
	Since Index 2 is denoted as a -1 the frequency is calculated using the points 1028 and the last recorded event

`FREQ:DELT? 256,257`
`100000.000000`
	Returns the frequency of events that have occurred between index values 256 and 257

INDEX:TIME?

This query is used to return the index number that correlates to the specified time. The index time query returns a value from 0 to 131,071 (524,287 if optional RAM is installed) which correlates to the address in RAM. The time parameter must be an exact match or an error is generated. The index time query is selected using the following SCPI command.

INDex:TIMe? <time>	***Where <time>, is a numeric value of 0 to 1,099,511.62778 seconds***

EXAMPLES

`IND:TIM? 100.00000`	*Returns the location in RAM where the event*
`15,753`	*data is stored for the time of 100.0000*

Assuming that an event occurred at the time of 10.00566

`IND:TIM? 10.05666`
`“”`
	The “” designate that an error was generated. This is because no event occurred at the specified time

INDEX:TIME:NEXT?

This query is used to return the next index occurring after the specified time. The index time next query returns a value from 0 to 131,071 that correlates to the index number. The channel list parameter is used to exclude unwanted events during the search. The index time next query is selected using the following SCPI command.

INDex:TIMe:NEXT? <time>[,<channel_list>]

Where <time>, is a numeric value of 0 to 1,099,511.62778 seconds.

Where [<channel_list>] is standard channel list syntax forChannels 1 through 32

EXAMPLES

```
IND:TIM:NEXT? 0.00250
1059
```

All channels are evaluated during the search. The returned index value of 1059 is the first occurring event or events after an execution time of 0.00250

```
IND:TIM:NEXT? 1.00000,(@3)
15074
```

This command will search for the first occurrence of Channel 3 causing an event after the 1 second of run time. The returned index value is 15074

INDEX:TIME:PREVIOUS?

This query is similar to the index time next query, the difference being that it returns the index number that occurred before the specified time. The index time previous query returns a value from 0 to 131,971 that correlates to the index number. The index time previous query returns a value from 0 to 131,071, that correlates to the index number. The channel list parameter is used to exclude unwanted events during the search. The index time previous is selected using the following SCPI command.

INDex:TIMe:PREVious? <time>[, <channel_list>]	***Where <time>, is a numeric value of 0 to 1,099,511.62778 seconds.*** ***Where [<channel_list>] is standard channel list syntax forChannels 1 through 32***

EXAMPLES

`IND:TIM:PREV? 60.00000`
`8608`
	All channels are evaluated during the search. The returned value of 8608 is the index value of the last event or events occurring before one minute of execution time.

`IND:TIM:PREV? 300.00000,(@12,15)`
`18433`
	This command will search for the first occurrence of Channels 12 and 15 causing an event before 5 minutes of run time. The returned index value of 18,433 shows the location of the occurrence.

INPUT:MASK

The input mask selects which channels will cause time stamp events to occur. The input mask register will not allow a signal to cause a time stamp event to occur if it is masked out. This command is used to disable unwanted inputs from generating time stamp events. An input channel must be enabled (INPut:MASK0) in order for the channel to cause a time stamp event to be recorded. The input mask is selected using the following SCPI command:

INPut:MASK ON / OFF / 1 / 0,<channel_list>	***Where ON /OFF will disable or enable an input.*** ***Where <channel_list> is standard channel list syntax for Channels 1 through 32***

EXAMPLES

`INP:MASK ON,(@1:7)`	*Disables Channels 1 through 7 to generate time stamp events*
`INP:MASK 0,(@9,11,13,15,16)`	*Enables Channels 9, 11, 13, 15 and 16 from generating time stamp events*

INPUT:MASK:ENABLE

The input mask enable command controls whether masked (disabled) input channels are included in the data searches which are performed when data queries are requested. For example, if Channels 15 through 32 are disabled using INPut:MASK command [INPut:MASK ON,(@15:32)], and then the INPut MASK enable is turned on, only the data for Channels 1 through 14 would be reported. If the INPut MASK ENABle is turned off, all channel data (1 through 32) would be reported. The input mask enable command is selected using the following SCPI command:

INPut:MASK:ENABle ON / OFF / 1 / 0	***Where ON /OFF will disable /enable a channel during data searches***

EXAMPLES

`INP:MASK:ENAB ON`	*Disables masked channels from affecting query responses*
`INP:MASK:ENAB 0`	*During data queries all events will be reported including masked channels*

Note: The *RST command enables the mask.

INPUT:POLARITY

The input polarity command will select which edge will cause the time stamp event to occur. When the input polarity is set to RISing, an event will be time stamped on the rising edge. When the input polarity is set to FALLing, an event will be time stamped on the falling edge. It is important to note that the *RST command sets all the channels for the rising edge. The input signal polarity is selected using the following SCPI command.

INPut:POLarity <polarity>[,<channel_list>]	***Where <polarity> is RISing or FALLing*** ***Where [<channel_list>] is standard channel list syntax forChannels 1 through 32***

EXAMPLES

`INP:POL RIS,(@1)`	*Sets input polarity for Channel 1 to RISing. This will generate a time stamp event on the rising edge of Channel 1 input signal*
`INP:POL FALL,(@3:5,16:21)`	*Sets input polarity for Channels 3 through 5 and Channels 16 through 21 for falling edge*

INPUT:SOURCE

The input source command will select where the input of the channel will originate. This command will select either the front panel or the VXIbus. The odd numbered channels can be sourced from the TTL triggers. The even channels can be sourced from the channels directly above, Channel 2 can be sourced from Channel 1. This allows an input signal to generate an event from both the rising and falling edges. This option is selected using ADJacent parameter. The *RST command sets all channels to FPAN. The input source is selected using the following SCPI command.

INPut:SOURce <source>[, <channel_list>]

Where <source> is FPAN, TTLTrig or ADJacent. TTLTrig is valid with odd-numbered channels. ADJacent is valid with even-numbered channels.

Where [<channel_list>] is standard channel list syntax for Channels 1 through 32.

EXAMPLES

```
INP:SOUR FPAN
```

Sets input source for all channels to the front panel input connector pins

```
INP:SOUR TTLT,(@1,3,5)
```

Sets input source for Channels 1, 3 and 5 as TTL triggers 0, 1 and 2 respectively

```
INP:SOUR ADJ,(@2,4,6)
```

Sets input source for Channels 2, 4 and 6 as Channels 1, 3 and 5 (TTL triggers 0, 1 and 2, respectively)

INPUT:TYPE

The input type command will select whether the input is in differential or single-ended mode. In differential mode, the inputs switch state when the voltage on one input side exceeds the other input side (i.e., a high level is recorded when the plus input of a channel is at a more positive voltage than the negative input of the same channel). In single-ended mode, the negative inputs are not used and the positive inputs are compared to an internal voltage generated by an 8-bit Digital-to-analog Converter (DAC). The *RST command sets all the inputs to single-ended mode. The input type is selected using the following SCPI command.

INPut:TYPE <type>,<channel_list>	***Where <type> is DIFF or SING***
	Where <channel_list> is standard channel list syntax for Channels 1 through 32

EXAMPLES

`INP:TYPE DIFF,(@1:8)`	*Sets the input type for Channels 1 through 8 as differential*
`INP:TYPE SING`	*Sets the input type for all channels (default) as single-ended*

INITIATE

The INITiate command starts the process of collecting time stamp events. The INITiate command clears the counters and registers on the VM1602. After power up or after an ABORt command, the VM1602 will not start measuring time or recording events until the INITiate command is received.

INITiate	***No command parameters***

EXAMPLES

`INIT`	*Clears the counters and registers. Enables the collection of time stamp events*

SWEEP:STEP

The sweep step command sets the time stamp resolution period. The sample rate sets the time interval that the time stamps can resolve and all times reported have an uncertainty equal to the sample rate. The purpose of reducing the sample rate from the highest resolution is to reduce the amount of data to consider and to increase the amount of time before the internal time stamp clocks over. The *RST command sets the time interval to 1E-6 seconds. The sweep step command is selected using the following SCPI command:

SWEep:STEP <time_interval>	***Where <time_interval> is 1E-3 (1 ms), 1E-4 (100 µs), 1E-5 (10 µs) or 1E-6 (1 µs)***

EXAMPLES

SWE:STEP 1E-6	*Selects the internal 10 MHz clock to be used as the time reference. This reference has a resolution of 1 µs.*
SWE:STEP 1E-3	*Selects the 1 kHz (divided 10 MHz) clock to be used as the time reference. This reference has a resolution of 1 ms.*

SYNC

The sync command will synchronize multiple VM1602 time stamp modules together. This is achieved by designating which module is the master and which is the slave. If only one VM1602 time stamp module is used, this command will designate it as a standalone. If two or three 32-channel VM1602 modules are to be used together, where all 64 or 96 channels are to be in sync in terms of sample time and start time, then the instrument to be configured as the Master is to be set up for the desired SWEep:STEP. First, **all** the instruments to be used are to be configured as Slaves and initiated; then, the designated instrument is to be changed to Master. Measurement data will not begin to be captured until the designated unit is set to Master. Once a master/slave setup has been established, the instrument set up as Master will set the sample rate for all the slaves.

Note: The *RST command sets the sync to the standalone mode. The sync command is selected using the following SCPI command:

SYNC <type>

Where <type> is STANdalone, MASTer, or SLAVe

EXAMPLES

`SYNC STAN`
Designates the module as the only VM1602 time stamp being used

`SWE:STEP`
Selects the internal 10 MHz clock to be used as the time reference. The SWEep:STEP setting on the instrument that is eventually to be designated as the Master will set the clock rate for the Master, and for the Slaves. The Slave units need to have the SWEep:STEP configured the same as the Master unit prior to initiating the instruments so that time data will be reported correctly.

`SYNC SLAV`
Designates the modules as slaves. (All instruments are initially set up as Slaves.)

`INIT`
Initiates the modules. All modules used in SYNC mode need to be initiated.

`SYNC MAST`
Designates the module as the master.

Note: If the Master is configured as the Master prior to initiating the Slaves units, the Slaves will begin capturing data as soon as they are initiated.

TIMe:DATA?

The Time Data query is used with the indices to return the time stamp data for single or multiple indices. The time data is returned in seconds with 1 microsecond resolution regardless of the time stamp interval selected. The time reported is relative to when the instrument was initiated and is in a fixed format numeric value.

It should be noted that the time stamp counter is reset with each INITiate or *TRG command. Therefore, there is no correlation between independent data collection runs. The time data is intended to be used to determine the time between events within a single collection run. The time data query is selected using the following SCPI command:

TIMe:DATA? <index_1>,[<index_2>]	***Where <index_1>, [<index_2>] is a numeric value of 0 to 131,071***

EXAMPLES

```
TIM:DATA? 10
0.002567
```

Since Index 2 is not specified, a single time is returned

```
TIM:DATA? 6526,6528
0.75890,0.87583,0.98956
```

Returns the time data of events that have occurred between index values 6526 and 6528

TIMe:DELTa?

The Time Delta query is used with the indices to return the time differences between the two indices. The time period is returned in seconds with 1 microsecond resolution regardless of the time stamp interval selected. The event data query is selected using the following SCPI command:

TIMe:DELTa? <index_1>,[<index_2>]

Where <index_1>, <index_2> is a numeric value of 0 to 131,071

EXAMPLES

```
TIM:DELT? 1028,-1
10.5378
```

Since Index 2 is denoted as a -1, the time period is calculated using the points 1028 and the last recorded event

```
TIM:DELT? 526,527
0.75890
```

Returns the time period of events that have occurred between index values 526 and 527

TRIGger:LEVel

The Trigger Level command sets the trigger threshold for a group of four channels. An 8-bit Digital-to-analog Converter (DAC) is used to provide the input voltage level to the signal comparators. The input signals are grouped in fours, bits 1-4 are set by trigger level 1, bits 5-9 are set by trigger level 5, and so on. To avoid redundancy of setting thresholds, this command only responds to the first channel number of a group. It is important to note that any channel may be queried for its threshold, but only specific channels may be programmed. Also, note that the trigger level applies only to channels that are set for single-ended operation. Differential inputs do not use the threshold DACs and switch state when the voltage on one input side exceeds the other input side. The *RST command will set the threshold to +1.80 volts for compatibility with standard TTL logic. The trigger level is selected using the following SCPI command:

TRIGger:LEVel <voltage>[, <channel_list>]	***Where <voltage> is +4.96 V to -5.0 V***
	Where [<channel_list>] is standard channel list syntax for Channels 1 through 32

EXAMPLES

`TRIG:LEV 0.8`	*Sets the input signal threshold for all channels to 0.8 V*
`TRIG:LEV -2.0,(@1,9,13)`	*Sets the input signal threshold for Channels 1 through 4, and 9 through 13 to -2.0 V*

APPLICATION EXAMPLES

This section contains examples of using SCPI command strings for programming the VM1602 module. The code is functional and will contain a brief description and block diagram of the operation.

TRIGGERING OF BOTH EDGES OF THE INPUT SIGNAL

In this example, the time stamp module is used to verify that the device under test is operating properly. With the device in the built-in test mode and operating, it will transmit a series of 3 pulses, 300 *μs* wide, every 2 seconds. VM1602 will be used to measure the duration and frequency that the pulses occur. Channel 2 will be sourced from Channel 1's input. Channel 1 will be initialized to trigger on the rising edge and Channel 2 the falling edge. Channels 3 through 32 are masked out to prevent unwanted event triggering. It is assumed that another VXI instrument is controlling the test duration of 10 to 15 seconds in length.

COMMANDS	DESCRIPTION
SWE:STEP 1E-6	Selects a resolution of 1 *μ*s
INP:TYPE DIFF,(@1,2)	Sets up Channel 1 and 2 as differential inputs
INP:SOUR ADJ,(@2)	Configures Channel 2 to be sourced from its adjacent channel, i.e., Channel 1
INP:MASK ON,(@3:32)	Masks out Channels 3 through 32 from generating an event
INP:POL RIS,(@1)	Sets up Channel 1 to trigger on rising edge
INP:POL FALL,(@2)	Sets up Channel 2 to trigger on falling edge
INIT	Starts the collection of time stamp events

Upon completion of the test (send the ABORt command), the following is used to verify the pulse widths and repetition rates.

<u>COMMANDS</u>	**<u>DESCRIPTION</u>**
TIMe:DATA? 1,7 1.0003, 1.0006, 1.0009, 1.0012, 1.0015, 1.0018, 3.0003	Queries for time between pointers 1 and 7 *Data shows that first event occurred at 1.0003 seconds into run and contained 3 pulses of 600 µs period 50% duty cycle*
TIMe:DELTa? 1,2 0.0003	Queries time delta between pointers 1 and 2 *Data shows pulse width to be 300 µs*
TIMe:DELTa? 1,7 2.0000	Queries time delta between pointers 1 and 7 *Data shows repetition rate to be 2 seconds*

Process Flow Monitoring

In this example, the VM1602 Time Stamp module will be used to monitor the steps in a process flow.

The process has 16 steps that are required before the finished product is ready for testing. The assembly steps take varying amounts of time to complete. The product travels down a conveyor belt and stops at each station where a different assembly process is performed. The conveyor belt has light beam switches that trigger as the product passes through the beam. The light switches are located at the entrance to the assembly station. The VM1602 will be used to gather the time and station ID number as product travels through the process flow. The data will be gathered at midnight everyday to identify possible bottlenecks.

The program for terminating time stamp collection and subsequent restarting will not be covered in this example. This example will concentrate on the actual time stamping as it pertains to the VM1602. The VM1602 will be set up to monitor Channels 1 through 16 only. The resolution will be 1 ms as the process is labor intensive. The light switch has a falling edge when the beam is broken.

COMMANDS	DESCRIPTION
SWE:STEP 1E-3	Selects a resolution of 1 ms
INP:TYPE SING,(@1:16)	Sets up Channel 1 through 16 as single ended inputs
TRIG:LEV 1.0,(@1:16)	Sets trigger level threshold for Channels 1 through 16 to 1.0 V
INP:POL FALL,(@1:16)	Sets Channel 1 through 16 for falling edge trigger
INP:SOUR FPAN,(@1:16)	Configures Channel 1 through 16 to be sourced from the front panel connector
INP:MASK ON,(@17:32)	Masks out Channel 17 through 32 from generating an event
INP:MASK:ENAB ON	Disables masked channels from event queries
INIT	Starts the collection of time stamp events

At midnight the program is terminated (send the ABORt command) and the following information gathered. For clarity, only the first five events and times will be examined.

COMMANDS	**DESCRIPTION**
TIMe:DATA? 1,10	Queries for time between pointers 1 and 10
10.0,910.0,1660.0, 1810.0,2530.0,2560.0,2710.0 910.0,3160.0,3460.0,3490.0	*Data shows that first event occurred at 10.0 seconds into run, the second occurrence was at 15 minutes and 10 seconds. The subsequent events occurred at 27 minutes and 40 seconds, 30 minutes and 10 seconds, 42 minutes and 10 seconds, etc.*
TIMe:DELTa? 2,3	Queries time delta between pointers 2 and 3.
750.0	*Data shows that step 2 took 750 seconds or 12 minutes and 30 seconds to complete*
EVEN:DATA? 1,5	Queries for events occurring between pointers 1 and 5
1,3,4,3,8	*Shows thatChannel 1 was the 1st occurrence, Channel 1 and 2 the 2nd, Channel 3 for the 3rd, Channel 1 and 2 for the 4th and Channel 4 for the 5th occurrence.*
EVENt:TIMe? 3160.0	Queries the events that happened at 3160.0 seconds
19	*Shows that Channels 1, 2, 5 were all active at 3160.0 seconds into the run*

REGISTER ACCESS

The VM1602 module provides pseudo register access for high-speed data transfers.

There are two different RAM configurations that the module supports. One allows up to 128 k events and the other allows up to 512 k events. Each event is 32 bits of information. Associated with each event is 40 bits of time information. This information is available through pseudo register access. Some of the information is simple and straightforward to use. Some of the information requires more post processing.

The number of events stored is called HIGHWATER MARK. This is available in the read only registers at offsets 0x20 and 0x22 (the information is up to 20 bits so there is a lower 16 bit part of the number and the upper 16 part of the number). The information is valid only after a data collection has stopped. While a data collection is in progress, this will most likely to be read as 0. This is the same number that is retrieved with the word serial “EVENt:COUNt?” query.

The INDEX is a pointer to the event and time data. It is available in the read/write registers at offset 0x24 and 0x26 (this information is up to 19 bits so there is a lower 16-bit part of the number and an upper 16-bit part of the number). It can be set (written) from 0 to HIGHWATER MARK - 1. It is auto incremented when the offset 0x2E is read. Offset 0x2E is a part of the EVENT information.

There are 40 bits of time information associated with each event. The lower 32 bits of information are in the read only registers at offsets 0x28 and 0x2A (the information is 32 bits wide so there is a lower 16 bit part of the number and upper 16 bit part of the number). To obtain the upper 8 bits requires some post processing which is described below. For most users, 32 bits of time information should be enough. The time of the start of the data collection is called 0. All times are relative to this time. To convert the 32 or 40 bits of time to a time value, multiply the time number by the sweep rate. For example, if the sweep rate is set at 1E-6 and a time value of 1234567, then the time of the associated event is 1.234567 seconds after the start of the data collection.

There are 32 channels that become the 32 bits of EVENt data at offsets 0x2C and 0x2E. The information is 32 bits so there is a lower 16 bit part of the number and an upper 16 part of the number. Channel 1 is the LSB and Channel 32 is the MSB. The way the data is recorded in memory is affected by the INPut:MASK. If a channel is enabled (mask = 0) then a one is recorded when there is a transition of the correct polarity in the time period preceding the clock. If a channel is disabled (mask = 1) then a one is recorded when there is a high when the clock occurs; this is called the logic analyzer mode.

Because the index will auto-increment when the event information is read at 0x2E, a user program to extract the information might do something as follows:

a) Read the HIGHWATER MARK at 0x20 and 0x22 to determine the number of points to process.

b) Set the INDEX at 0x24 and 0x26 to 0 (this is automatically done at the end of a data capture but it can be done in any case to be on the safer side).

c)
```
Loop for the number of points
    Read TIME HIGH at 0x28 and store
    Read TIME LOW at 0x2A and store
    Read EVENT HIGH at 0x2C and store
    Read EVENT LOW at 0x2E and store
End of loop
```

The hardware records 32 bits of time data but the module keeps track of 40 bits. To do this, the processor is interrupted every time there is a 31-bit overflow of the timer. When this interrupt occurs, the processor records the current event number in an array that will hold up to 512 entries. Because there are 32 bits of hardware time stored with each event, there is no ambiguity as to where a timer overflow occurred. With this extra information, it is possible to determine an extra 8 bits of timer information. The raw data for extracting this information is available through the pseudo register access.

The number of 31 bit timer overflows (the high water mark) that occurred during a data collection is recorded in the register at offset 0x30. This tells the user the maximum number of data items that can be retrieved from the overflow array. The number can range from 0 to 512.

The INDEX is a pointer to the overflow data. It is available in the read/write register at offset 0x32. It can be set (written) from 0 to the high water mark - 1. It is auto incremented when offset 0x36 is read. Offset 0x36 is part of the event index information.

The timer overflow data is really event index information recorded when the 31 bit timer overflows. The information is up to 19 bits (512 k - 1) so there is a lower 16 part of the number and an upper 16 part of the number. The information is read at offsets 0x34 and 0x36. Reading the data at offset 0x36 auto increments the INDEX for this information.

Because the index will auto increment when the event information is read at offset 0x36, a user program to extract the information might do something as follows:

a) Read the HIGHWATER MARK at 0x30 to determine the number of points to process

b) Set the INDEX at 0x32 to 0 (this is automatically done at the end of a data capture but it can be done anyway just to be safe).

c)
```
Loop for the number of points
    Read the event index high at 0x34 and store
    Read the event index low at 0x36 and store
End of Loop
```

REGISTER ACCESS

The model VM1602 Time Stamp Module supports access to various 32-bit data registers via the device dependent registers of the VXIbus interface. The specific registers are located in A16 memory. The following table shows A16 memory and the model VM1602 data register map.

TABLE 3-1 MEMORY MAP

3E	
3C	
3A	
38	
36	32 Bit Timer Overflow Data Low (Read Only)
34	32 Bit Timer Overflow Data High (Read Only)
32	16 Bit High Water Overflow Index (Read/Write)
30	16 Bit High Water Timer Overflow (Read Only)
2E	32 Bit Event Low (Read Only)
2C	32 Bit Event High (Read Only)
2A	32 Bit Time Low (Read Only)
28	32 Bit Time High (Read Only)
26	32 Bit Index Event/Time Low (Read/Write)
24	32 Bit Index Event/Time High (Read/Write)
22	32 Bit High Water Event/Time Low (Read Only)
20	32 Bit High Water Event/Time High (Read Only)
1E	
1C	
1A	
18	
16	[A32 Pointer Low]
14	[A32 Pointer High]
12	[A24 Pointer Low]
10	[A24 Pointer High]
E	Data Low
C	Data High
A	Response [/Data Extended]
8	Protocol [/Signal] Register
6	[Offset Register]
4	Status/Control Register
2	Device Type
0	ID Register

VXI*PLUG&PLAY* Driver Examples

```
/*
 *                            APPLICATION FUNCTION
 *                            --------------------
 */
/***************************************************************************
Function:         vtvm1602_setupAndCalcFreq

Formal
Parameters        ViSession instrHndl,
                        - A valid session handle to the instrument.

                  ViInt16 channelList[]
                        - This parameter specifies the channels which are to
                        be configured and enabled/disabled for time stamping.

                  Each channel number in the array has the valid range:
                        vtvm1602_MIN_CHANNEL_NO (1) to
                        vtvm1602_MAX_CHANNEL_NO (32)

                  ViInt16 numOfChannels
                        - This parameter specifies the number of channels in
                        the channel list.

                  Valid Range :
                        vtvm1602_MIN_CHANNEL_NO (1) to
                        vtvm1602_MAX_CHANNEL_NO (32)

                  ViBoolean mask[]
                        - This parameter specifies the mask to be used for the
                        specified channels.

                  Each element of the array must have the value
                        vtvm1602_MASK_ON        or
                        vtvm1602_MASK_OFF

                  ViBoolean polarity[]
                        - This parameter specifies the polarity to be
                        configured for the specified channels.

                  Each element of the array must have the value
                        vtvm1602_POLARITY_RIS   or
                        vtvm1602_POLARITY_FALL

                  ViBoolean inputType[]
                        - This parameter specifies the type, either
                        differential or single-ended operation to be
                        configured for the specified channels.
```

```
Each element of the array must have the value
      vtvm1602_INPUT_TYPE_DIFF      Differential
      vtvm1602_INPUT_TYPE_SING      Single-ended

ViReal32 voltage[]
      - This parameter specifies the threshold voltage to be
      configured for the specified channels.  It must be
      noted that if different voltage values are specified
      for the channels within the same group of 4 channels,
      only the last configured voltage for that group of
      channels will be in effect.

Each element of the array must be in range:
      vtvm1602_VOLTAGE_MIN (-5.0) to
      vtvm1602_VOLTAGE_MAX.(+4.96).

ViInt16 source[]
      - This parameter specifies the input source to be
      configured for the specified channels.

Each element of the array must have a value
      vtvm1602_SOURCE_FPAN          Front Panel
      vtvm1602_SOURCE_TTLT          TTLTrig
      vtvm1602_SOURCE_ADJ           Adjacent

ViReal64  timeInterval
      - This parameter specifies the time stamp resolution
      period to be configured.

Valid Values :
        vtvm1602_TIME_INTERVAL_1    0.001
        vtvm1602_TIME_INTERVAL_2    0.0001
        vtvm1602_TIME_INTERVAL_3    0.00001
        vtvm1602_TIME_INTERVAL_4    0.000001

ViInt16  transistionChannel
      - This parameter specifies the channel whose frequency
      of transition i.e. the frequency of time stamping is
      to be calculated.  If the channel specified has been
      disabled for time stamping events, then an error is
      returned.

Valid Range :
      vtvm1602_MIN_CHANNEL_NO (1) to
      vtvm1602_MAX_CHANNEL_NO (32).

ViPReal64  frequency
      - This parameter returns the frequency of transition
      for the specified transition channel.
```

```
                  Return Values:
                        Returns VI_SUCCESS if successful, else returns error
                        value.

                  Description
                        This function sets up the specified channels and
                        calculates the frequency of transition of the
                        specified transition channel.  This function allows 10
                        seconds for event captures.
***************************************************************/

ViStatus _VI_FUNC vtvm1602_setupAndCalcFreq(ViSession instrHndl,

            ViInt16     channelList[],
            ViInt16     numOfChannels,
            ViBoolean   mask[],
            ViBoolean   polarity[],
            ViBoolean   inputType[],
            ViReal32    voltage[],
            ViInt16     source[],
            ViReal64    timeInterval,
            ViInt16     transistionChannel,
            ViPReal64   frequency)

{

/* Variable used to store return status of the function */

      ViStatus status = VI_NULL;

/* Array used to read the results from the instrument*/

      ViChar readBuf[vtvm1602_READ_BUFF_SIZE+1];

      ViInt32 *sourceChannel = VI_NULL,
            count = VI_NULL,
            numElems = VI_NULL;

      ViInt32 i = VI_NULL,
            tmpIndex1 = VI_NULL,
            tmpIndex2 = VI_NULL;

      ViInt16 channelGroup = VI_NULL;

/*
 * The instrument is reset to its default state
 */

      status = vtvm1602_reset(instrHndl);
      if (status < VI_SUCCESS)
                  return status;
      for (i = 0; i < numOfChannels; i++)
      {
            if (channelList[i] == transistionChannel)
            {

/*
 * Validating the transition channel to ensure that it
 * has been enabled
 */

                  if (mask[i] != vtvm1602_MASK_ON)
                        return vtvm1602_TRANSITION_CHANNEL_NOT_ENABLED;
                  break;
            }
      }

/*
 * If the transition channel has not been set up, return an error
 */

      if (i == numOfChannels)
            return VI_ERROR_PARAMETER10;

/*
 * Enabling/Disabling the specified channels for time stamping
 */

      status = vtvm1602_maskUnmaskChannels(instrHndl, channelList,
                                                numOfChannels, mask,
                                                vtvm1602_MASK_DISABLE);
      if (status < VI_SUCCESS)
            return status;

/*
 * Configuring the polarity, source etc. for the specified
 * channels
 */

      status = vtvm1602_configChannels(instrHndl, channelList,
                                                numOfChannels, polarity,
                                                inputType, source);
      if (status < VI_SUCCESS)
            return status;

/*
 * Configuring the trigger level for the group of 4 channels
 * depending on which group the specified channels fall
 */

      for (i = 0; i < numOfChannels; i++)
      {
            if (channelList[i] > 28)
                  channelGroup = vtvm1602_CHANNEL_GROUP_8;
            else if (channelList[i] > 24)
                  channelGroup = vtvm1602_CHANNEL_GROUP_7;
            else if (channelList[i] > 20)
                  channelGroup = vtvm1602_CHANNEL_GROUP_6;
            else if (channelList[i] > 16)
                  channelGroup = vtvm1602_CHANNEL_GROUP_5;
            else if (channelList[i] > 12)
                  channelGroup = vtvm1602_CHANNEL_GROUP_4;
            else if (channelList[i] > 8)
                  channelGroup = vtvm1602_CHANNEL_GROUP_3;
            else if (channelList[i] > 4)
                  channelGroup = vtvm1602_CHANNEL_GROUP_2;
            else
                  channelGroup = vtvm1602_CHANNEL_GROUP_1;

/*
 * Configuring the trigger level for the specified channels
 */

            status = vtvm1602_configTriggerLevel(instrHndl, channelGroup,
                                                voltage[i]);
            if (status < VI_SUCCESS)
                  return status;
      }

/*
 * Configuring the sample rate of the time stamping
 */

      status = vtvm1602_configSweep(instrHndl, timeInterval);
      if (status < VI_SUCCESS)
            return status;

/*
 * Enabling the collection of time stamp events
 */

      status = vtvm1602_startStopData(instrHndl,
      vtvm1602_START_STOP_INIT);
            if (status < VI_SUCCESS)
                  return status;

/*
 * Providing a delay of 10 seconds for the module to collect time
 * stamp data
 */

      Delay(10);
```

```
/*
 * Stopping the data collection in progress
 */

      status = vtvm1602_startStopData(instrHndl, vtvm1602_START_STOP_ABOR);
            if (status < VI_SUCCESS)
                  return status;

/*
 * Querying the number of registered events on the specified
 * channels
 */

      status = vtvm1602_sendWSCmd(instrHndl, "EVEN:COUN?",
                                          strlen("EVEN:COUN?"));
      if (status < VI_SUCCESS)
            return status;

      status = vtvm1602_readInstrBuff(instrHndl, readBuf,
                                          vtvm1602_READ_BUFF_SIZE);
      if (status < VI_SUCCESS)
            return status;

      numElems = atol(readBuf);

/*
 * If no time stamp events have been captured, return an error to
 * the same effect
 */

      if (numElems == VI_NULL)
            return vtvm1602_NO_TIME_STAMP_DATA;

      sourceChannel = (ViInt32 *)calloc(numElems, sizeof(ViInt32));
      if (sourceChannel == VI_NULL) return vtvm1602_MALLOC_ERROR;

/*
 * Querying the source of channels for the registered time stamp
 * events
 */

      status = vtvm1602_querySourceChannel(instrHndl, 0, -1,
                                          sourceChannel, &count);
      if (status < VI_SUCCESS)

      {
            free(sourceChannel);
            return status;
      }
```

```
/*
 * Determining the first two consecutive index points on the
 * on-board memory where the specified transition channel has
 * caused time stamp events
 */

      for(i = 0; i < count; i++)

      {
                  if(sourceChannel[i] & (int)pow(2,transistionChannel - 1))
            {
                  tmpIndex1 = i;
                  break;
            }
      }

/*
 * Returning an error if the specified transition channel has not
 * sourced any time-stamp events
 */

      if (i == count)

      {
            free(sourceChannel);
            return vtvm1602_NOT_ENOUGH_EVENTS;
      }

      for(i++;i < count;i++)

      {
                  if(sourceChannel[i] & (int)pow(2,transistionChannel - 1))
            {
                  tmpIndex2 = i;
                  break;
            }
      }

/*
 * Returning an error if the specified transition channel has not
 * sourced at least two events
 */

      if (i == count)

      {
            free(sourceChannel);
            return vtvm1602_NOT_ENOUGH_EVENTS;
      }

      free(sourceChannel);
```

```
/*
 * Calculating the frequency of transition between the first two
 * consecutive events sourced by the specified transition channel
 */

      status = vtvm1602_queryFrequency( instrHndl, tmpIndex1, tmpIndex2,
                                          frequency);
      if (status < VI_SUCCESS)
            return status;

      return VI_SUCCESS;
}
```

SECTION 4

COMMAND DICTIONARY

INTRODUCTION

This section presents the instrument command set. It begins with an alphabetical list of all the commands supported by the VM1602 divided into three sections: IEEE 488.2 commands, the instrument specific SCPI commands and the required SCPI commands. With each command is a brief description of its function, whether the command's value is affected by the *RST command and its default value.

The remainder of this section is devoted to describing each command, one per page, in detail. The description is presented in a regular and orthogonal way assisting the user in the use of each command. Every command entry describes the exact command and query syntax, the use and range of parameters and a complete description of the command's purpose.

ALPHABETICAL COMMAND LISTING

The following tables provide an alphabetical listing of each command supported by the VM1602 along with a brief description. If an X is found in the column titled *RST, then the value or setting controlled by this command is possibly changed by the execution of the *RST command. If no X is found, then *RST has no effect. The default column gives the value of each command's setting when the unit is powered up or when a *RST command is executed.

TABLE 4-1 IEEE 488.2 COMMON COMMANDS

Command	Description	*RST	Reset Value
*CLS	Clear the Status Register	X	
*ESE	Set the Event Status Enable Register		N/A
*ESR?	Query the Standard Event Status Register		N/A
*IDN?	Query the module identification string		N/A
*OPC	Set the OPC bit in the Event Status Register	X	0
*RST	Reset the module to a known state		N/A
*SRE	Set the Service Request Enable Register		N/A
*STB?	Query the Status Byte Register		N/A
*TST?	Starts and reports a self-test procedure		N/A
*WAI	Halts execution and queries		N/A

TABLE 4-2 INSTRUMENT SPECIFIC SCPI COMMANDS

Command	Description	*RST	*RST Value
ABORt	Stops a data collection in progress		N/A
EVENt:COUNt?	Queries the number of events registered	X	0
EVENt:DATA?	Queries the source of events between two indices	X	Data cleared
EVENt:TIMe?	Queries the event data which occurred at a given time	X	Data cleared
EVENt:TIMe:NEXT?	Queries the event data which occurred after a given time	X	Data cleared
EVENt:TIMe:PREVious?	Queries the previous event data which occurred before a given time	X	Data cleared
FREQuency:DELTa?	Queries the equivalent frequency for the time period between two indices	X	Data cleared
INDex:TIMe?	Queries the index which corresponds with a given time value	X	Data cleared
INDex:TIMe:NEXT?	Queries the next index which occurred after a given time	X	Data cleared
INDex:TIMe:PREVious?	Queries the previous index which occurred before a given time	X	Data cleared
INPut:MASK	Selects which channels will cause time stamp events to occur	X	All enabled
INPut:MASK:ENABle	Selects if masked channels will be used in time stamp data queries	X	Enabled
INPut:POLarity	Sets the input polarity of one or more channels	X	RISing edge
INPut:SOURce	Sets the input source for a list of channels	X	Front panel
INPut:TYPE	Sets the input for either differential or single-ended mode of operation	X	Single ended
INITiate	Enables the collection of time stamp events		N/A
MFGTEST:MEMory?	Reports the memory available on the VM1602		
SWEep:STEP	Sets the time stamp resolution period	X	1 μs
SYNC	Synchronizes multiple VM1602 boards within a VMIP host	X	Stand alone
TIMe:DATA?	Queries the time stamp data for one or more indices	X	Data cleared
TIMe:DELTa?	Queries the time difference between time stamp indices	X	Data cleared
TRIGger:LEVel	Sets the trigger level of a group of four channels	X	+1.79 volts

TABLE 4-3 SCPI REQUIRED COMMANDS

Command	Description	*RST	*RST Value
STATus:OPERation:CONDition?	Queries the Operation Status Condition Register	X	
STATus:OPERation:ENABle	Sets the Operation Status Enable Register	X	
STATus:OPERation[:EVENt]?	Queries the Operation Status Event Register	X	
STATus:PRESet	Presets the Status Register	X	
STATus:QUEStionable:CONDition?	Queries the Questionable Status Condition Register	X	
STATus:QUEStionable:ENABle	Sets the Questionable Status Enable Register	X	
STATus:QUEStionable[:EVENt]?	Queries the Questionable Status Event Register	X	
SYSTem:ERRor?	Queries the Error Queue	X	Clears queue
SYSTem:VERSion?	Queries which version of the SCPI standard the module complies with		N/A

COMMAND DICTIONARY

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

Purpose	Describes the purpose of the command.
Type	Describes the type of command such as an event or setting.
Command Syntax	Details the exact command format.
Command Parameters	Describes the parameters sent with the command and their legal range.
Reset Value	Describes the values assumed when the *RST command is sent.
Query Syntax	Details the exact query form of the command.
Query Parameters	Describes the parameters sent with the command and their legal range. The default parameter values are assumed the same as in the command form unless described otherwise.
Query Response	Describes the format of the query response and the valid range of output.
Description	Describes in detail what the command does and refers to additional sources.
Examples	Present the proper use of each command and its query (when available).
Related Commands	Lists commands that affect the use of this command or commands that are affected by this command.

IEEE 488.2 Common Commands

*CLS

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Clears all status and event registers</td></tr>
<tr><td><b>Type</b></td><td colspan="2">IEEE 488.2 Common Command</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">*CLS</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Description</b></td><td colspan="2">This command clears the Status Event Register, Operation Status Register and the Questionable Data/Signal Register. It also clears the OPC flag and clears all queues (except the output queue).</td></tr>
<tr><td rowspan="2"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response / Descriptions</b></td></tr>
<tr><td><code>*CLS</code></td><td><i>(Clears all status and event registers)</i></td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">N/A</td></tr>
</table>

*ESE

<table>
<tr><td><b>Purpose</b></td><td>Sets the bits of the Event Status Enable Register</td></tr>
<tr><td><b>Type</b></td><td>IEEE 488.2 Common Command</td></tr>
<tr><td><b>Command Syntax</b></td><td>*ESE <mask></td></tr>
<tr><td><b>Command Parameters</b></td><td><mask> = numeric ASCII value</td></tr>
<tr><td><b>*RST Value</b></td><td>N/A, the parameter is required</td></tr>
<tr><td><b>Query Syntax</b></td><td>*ESE?</td></tr>
<tr><td><b>Query Parameters</b></td><td>N/A</td></tr>
<tr><td><b>Query Response</b></td><td>Numeric ASCII value from 0 to 255</td></tr>
<tr><td><b>Description</b></td><td>The Event Status Enable (ESE) command is used to set the bits of the Event Status Enable Register. See ANSI/IEEE 488.2-1987 section 11.5.1 for a complete description of the ESE register. A value of 1 in a bit position of the ESE register enables generation of the Event Status Bit (ESB) in the Status Byte by the corresponding bit in the Event Status Register (ESR). If the ESB is set in the Service Request Enable (SRE) register, then an interrupt will be generated. See the *ESR? command for details regarding the individual bits. The ESE register layout is:

Bit 0 - Operation Complete
Bit 1 - Request Control
Bit 2 - Query Error
Bit 3 - Device Dependent Error
Bit 4 - Execution Error
Bit 5 - Command Error
Bit 6 - User Request
Bit 7 - Power On

The Event Status Enable query reports the current contents of the Event Status Enable Register.</td></tr>
<tr><td><b>Examples</b></td><td><table><tr><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr><tr><td>*ESE 36</td><td></td></tr><tr><td>*ESE?</td><td>36 <i>(Returns the value of the event status enable register)</i></td></tr></table></td></tr>
<tr><td><b>Related Commands</b></td><td>*ESR?</td></tr>
</table>

*ESR?

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Queries and clears the Standard Event Status Register</td></tr>
<tr><td><b>Type</b></td><td colspan="2">IEEE 488.2 Common Command</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">ESR?</td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">Numeric ASCII value from 0 to 255</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Event Status Register (ESR) query queries and clears the contents of the Standard Event Status Register. This register is used in conjunction with the ESE register to generate the Event Status Bit (ESB) in the Status Byte. The layout of the ESR is:

Bit 0 - Operation Complete
Bit 1 - Request Control
Bit 2 - Query Error
Bit 3 - Device Dependent Error
Bit 4 - Execution Error
Bit 5 - Command Error
Bit 6 - User Request
Bit 7 - Power On

The Operation Complete bit is set when it receives an *OPC command.

The Query Error bit is set when data is over-written in the output queue. This could occur if one query is followed by another without reading the data from the first query.

The Execution Error bit is set when an execution error is detected. Errors that range from -200 to -299 are execution errors.

The Command Error bit is set when a command error is detected. Errors that range from -100 to -199 are command errors.

The Power On bit is set when the module is first powered on or after it receives a reset via the VXI Control Register. Once the bit is cleared (by executing the *ESR? command) it will remain cleared.</td></tr>
<tr><td rowspan="2"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td>*ESR?</td><td>4</td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">*ESE</td></tr>
</table>

*IDN?

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Queries the module for its identification string</td></tr>
<tr><td><b>Type</b></td><td colspan="2">IEEE 488.2 Common Command</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">*IDN?</td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">ASCII character string</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Identification (IDN) query returns the identification string of the module. The response is divided into four fields separated by commas. The first field is the manufacturer’s name, the second field is the model number, the third field is an optional serial number and the fourth field is the firmware revision number. If a serial number is not supplied, the third field is set to 0 (zero).</td></tr>
<tr><td rowspan="2"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td><code>*IDN</code></td><td>VXI Technology, Inc.,1602,0,1.xx
<i>(The revision listed here is for reference only; the response will always be the current revision of the instrument.)</i></td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">N/A</td></tr>
</table>

*OPC

Purpose	Sets the OPC bit in the Event Status Register	
Type	IEEE 488.2 Common Command	
Command Syntax	*OPC	
Command Parameters	N/A	
***RST Value**	N/A	
Query Syntax	*OPC?	
Query Parameters	N/A	
Query Response	1	
Description	The Operation Complete command sets the OPC bit in the Event Status Register (ESR) when all pending operations have completed. The VM1602 will continue to process other commands after receiving the *OPC command and the ESR register should be polled to determine when the operation has completed. The Operation Complete query will return a 1 to the output queue when all pending operations have completed. While waiting for the operation to complete, the VM1602 suspends the processing of further commands, preventing changes from occurring before the current operation is complete. The only pending operation on the VM1602 is the collection of time stamp data that can literally take years to complete. In the case that the query is used, the slot 0 controller should be set up to time-out in a reasonable period of time.	
Examples	**Command / Query**	**Response *(Description)***
	`*OPC`	*(Sets the OPC bit in the Event Status Register)*
	`*OPC?`	1 *(Returns the value of the Event Status Register)*
Related Commands	*WAI	

*RST

Purpose	Resets the module's hardware and software to a known state	
Type	IEEE 488.2 Common Command	
Command Syntax	*RST	
Command Parameters	N/A	
***RST Value**	N/A	
Query Syntax	N/A	
Query Parameters	N/A	
Query Response	N/A	
Description	The Reset (RST) command resets the module's hardware and software to a known state. See the command index at the beginning of this chapter for the default parameter values used with this command.	
Examples	**Command / Query**	**Response *(Description)***
	`*RST`	*(Resets the module)*
Related Commands	N/A	

*SRE

Purpose	Sets the service request enable register
Type	IEEE 488.2 Common Command
Command Syntax	*SRE <mask>
Command Parameters	<mask> = Numeric ASCII value from 0 to 255
***RST Value**	None – Required Parameter
Query Syntax	*SRE?
Query Parameters	N/A
Query Response	Numeric ASCII value from 0 to 255
Description	The Service Request Enable (SRE) mask is used to control which bits in the status byte generate back plane interrupts. If a bit is set in the mask that newly enables a bit set in the status byte and interrupts are enabled, the module will generate a REQUEST TRUE event via an interrupt. See the *STB? Command for the layout of bits. **Note:** Bit 6 is always internally cleared to zero as required by IEEE 488.2 section 11.3.2.3. The layout of the Service Request Enable Register is: Bit 0 – Unused Bit 1 – Unused Bit 2 – Error Queue Has Data Bit 3 – Questionable Status Summary (Not Used) Bit 4 – Message Available Bit 5 – Event Status Summary Bit 6 – 0 (per IEEE 488.2 section 11.3.2.3) Bit 7 – Operation Status Summary

Examples	**Command / Query**	**Response *(Description)***
	`*SRE 4`	*(Sets the service request enable register)*
	`*SRE?`	4 *(Returns the value of the SRE register)*

Related Commands	N/A

*STB?

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Queries the Status Byte Register</td></tr>
<tr><td><b>Type</b></td><td colspan="2">IEEE 488.2 Common Command</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">*STB?</td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">Numeric ASCII value from 0 to 255</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Read Status Byte (STB) query fetches the current contents of the Status Byte Register. See the IEEE 488.2 specification for additional information regarding the Status byte Register and its use. The layout of the Status Register is:

Bit 0 – Unused
Bit 1 – Unused
Bit 2 – Error Queue Has Data
Bit 4 – Questionable Status Summary (not used)
Bit 5 – Message Available
Bit 6 – Master Summary Status
Bit 7 – Operation Status Summary</td></tr>
<tr><td rowspan="2"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td><code>*STB?</code></td><td>16 (<i>Queries the Status Byte Register)</i></td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">N/A</td></tr>
</table>

*TST?

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Causes a self-test procedure to occur and queries the results</td></tr>
<tr><td><b>Type</b></td><td colspan="2">IEEE 488.2 Common Command</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">*TST?</td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">Numeric ASCII value from 0 to 143</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Self-Test query causes the VM1602 to run a self-test and report on the results. The self-test checks the time stamp RAM and reports either a pass or a fail status. If the self-test passes, the query returns a 0 value. If the self-test fails, the query returns a non-zero value.</td></tr>
<tr><td rowspan="2"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td><code>*TST</code></td><td>0 (<i>Begins the self-test procedure returns the result)</i></td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">N/A</td></tr>
</table>

*WAI

Purpose	Halts execution of additional commands and queries until the No Operation Pending message is true	
Type	IEEE 488.2 Common Command	
Command Syntax	*WAI	
Command Parameters	N/A	
*RST Value	N/A	
Query Syntax	N/A	
Query Parameters	N/A	
Query Response	N/A	
Description	The Wait to Continue command halts the execution of commands and queries until the No Operation Pending message is true. This command makes sure that all previous commands have been executed before proceeding. It provides a way of synchronizing the module with its commander.	
Examples	Command / Query	Response (Description)
	`*WAI`	(Pauses the execution of additional commands until the No Operation Pending message is true.)
Related Commands	*OPC	

INSTRUMENT SPECIFIC SCPI COMMANDS

ABORt

Purpose	Stops a data collection in progress	
Type	Command	
Command Syntax	ABORt	
Command Parameters	N/A	
***RST Value**	N/A	
Query Syntax	None	
Query Parameters	N/A	
Query Response	N/A	
Description	The Abort command will cause the instrument to stop taking time stamp readings if the unit is currently running. The Abort command does not change any other instrument settings. The data collected before the Abort command is then available for query.	
Examples	**Command / Query**	**Response *(Description)***
	ABOR	*(Pauses the execution of additional commands until the No Operation Pending message is true.)*
Related Commands	INITiate	

EVENt:COUNt?

Purpose	Queries the number of events registered
Type	Query
Command Syntax	None
Command Parameters	N/A
***RST Value**	0
Query Syntax	EVENt:COUNt? [<index_1>,<index_2>[,<channel_list>]]
Query Parameters	<index_1> = 0 to 131,071 (0 to 524,287 if 512 k RAM option is installed) <index_2> = 0 to 131,071 (0 to 524,287 if 512 k RAM option is installed) A value of -1 indexes to the end of the data set (the end of RAM or the last recorded event, whichever is less) <channel_list>= Standard channel list format supporting Channels 1 through 32 If the channel list is not supplied, all channels are used
Query Response	Single ASCII numeric value
Description	The Event Count query returns the total number of time stamp events that have occurred on all of the selected channels. In the event that more than one channel has a time stamp event at a given time, the event will only be counted once. The optional channel list selects the channels of interest. If the channel list is omitted, then all channels will be included. Index 1 and Index 2 are specific time stamp events in the collection RAM. The numeric value indicates a specific RAM location that contains the time and source information on a specific event. If the optional indices are not used, the event count returned covers the entire data set collected. If the optional indices are included in the query, then the returned number of events is constrained between the specified index points. Index 1 should precede Index 2 in time and therefore the value of Index 1 would be less than Index 2. If Index 1 and Index 2 are the same, only one event will be considered.
Examples	**Command / Query** | **Response *(Description)*** `EVEN:COUN?` | 131-72 `EVEN:COUN? (@1,3,7:9)` | 53 `EVEN:COUN? 25,256,(@1,3,7:9)` | 27
Related Commands	None

EVENt:DATA?

Purpose	Queries the source of events between two indices	
Type	Query	
Command Syntax	None	
Command Parameters	N/A	
***RST Value**	Data Cleared	
Query Syntax	EVENt:DATA? <index_1>[,<index_2>]	
Query Parameters	<index_1> = 0 to 131,071 (0 to 524,287 if 512 k RAM option is installed) <index_2> = 0 to 131,071 (0 to 524,287 if 512 k RAM option is installed) A value of -1 indexes to the end of the data set (the end of RAM or the last recorded event, whichever is less)	
Query Response	A comma-separated string of ASCII numeric values from 0 to 4,294,967,295	
Description	The Event Data query reports the sourcing channels of events between the index points specified. Each channel is represented by a bit in a 32-bit integer value where Channel 1 is in the LSB position and Channel 32 is in the MSB position. If the optional second index is not included, then a single index point source is reported. If Index 1 and Index 2 are the same value, only one event will be reported.	
Examples	**Command / Query**	**Response *(Description)***
	`EVEN:DATA? 122`	32
	`EVEN:DATA? 122,126`	32,1,64,65,256
Related Commands	None	

EVENt:TIMe?

Purpose	Queries the event data which occurred at a given time	
Type	Query	
Command Syntax	None	
Command Parameters	N/A	
***RST Value**	Data Cleared	
Query Syntax	EVENt:TIMe? <time>	
Query Parameters	<time> = ASCII numeric value ranging from 0 to 1,099,511.62778 seconds	
Query Response	ASCII numeric value from 0 to 131,071 (0 to 524,287 if 512 k RAM option is installed)	
Description	The Event Time query reports the event number that occurred at a specific time. The specified time must be an exact match with an event or an error is generated.	
Examples	**Command / Query**	**Response *(Description)***
	`EVEN:TIM? 0.001022`	32
Related Commands	EVENt:TIMe:NEXT? EVENt:TIMe:PREVious?	

EVENt:TIMe:NEXT?

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Queries the event data which occurred after a given time</td></tr>
<tr><td><b>Type</b></td><td colspan="2">Query</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">None</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">Data Cleared</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">EVENt:TIMe:NEXT? <time>[,<channel_list>]</td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2"><time> = ASCII numeric value ranging from 0 to 1,099,511.62778 seconds
<channel_list> = Standard channel list format supporting Channels 1 through 32 If the channel list is not supplied, all channels are used</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">ASCII numeric value from 0 to 4,294,967,295</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Event Time Next query reports the event data on the next event that occurs after the specified time. If the time exactly matches an event, the reported event data is for the following event index.

The optional channel list is provided to mask the search for the next event to include only events caused by one or more of the specified channels in the list. The returned event data includes all channels regardless of the mask.</td></tr>
<tr><td><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td></td><td><code>EVEN:TIM:NEXT? 12.075</code></td><td>48</td></tr>
<tr><td></td><td><code>EVEN:TIM:NEXT? 13.000102,(@1,3,5,9)</code></td><td>53</td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">EVENt:TIMe?
EVENt:TIMe:PREVious?</td></tr>
</table>

EVENt:TIMe:PREVious?

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Queries the previous event data which occurred before a given time</td></tr>
<tr><td><b>Type</b></td><td colspan="2">Query</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">None</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">Data Cleared</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">EVENt:TIMe:PREVious? <time>[,<channel_list>]</td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2"><time> = ASCII numeric value ranging from 0 to 1,099,511.62778 seconds
<channel_list> = Standard channel list format supporting Channels 1 through 32. If the channel list is not supplied, all channels are used</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">ASCII numeric value from 0 to 4,294,967,295</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Event Time Previous command reports the event data on the previous event that occurs before the specified time. If the time exactly matches an event, the reported event data is for the previous event index.

The optional channel list is provided to mask the search for the previous event to include only events caused by one or more of the specified channels in the list. The returned event data includes all channels regardless of the mask.</td></tr>
<tr><td><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td></td><td>EVEN:TIM:PREV? 12.075</td><td>4096</td></tr>
<tr><td></td><td>EVEN:TIM:PREV? 13.00102,(@1,3,5,9)</td><td>48</td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">EVENt:TIMe?
EVENt:TIMe:NEXT?</td></tr>
</table>

FREQuency:DELTa?

Purpose	Queries the equivalent frequency for the time period between two indices	
Type	Query	
Command Syntax	None	
Command Parameters	N/A	
***RST Value**	Data Cleared	
Query Syntax	FREQuency:DELTa? <index_1>,<index_2>	
Query Parameters	<index_1> = 0 to 131,071 (0 to 524,287 if 512 k RAM option is installed) <index_2> = 0 to 131,071 (0 to 524,287 if 512 k RAM option is installed) A value of -1 indexes to the end of the data set (the end of RAM or the last recorded event, whichever is less)	
Query Response	ASCII numeric floating point value	
Description	The Frequency Delta command calculates and reports the equivalent frequency for the time period between the specified indices. The frequency is calculated as 1/t, where t is the time period in seconds between the specified index points. The frequency is reported in Hertz (cycles per second).	
Examples	**Command / Query**	**Response *(Description)***
	`FREQ:DELT? 120,121`	1012.150000
Related Commands	TIMe:DELTa?	

INDex:TIMe?

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Queries the index which corresponds with a given time value</td></tr>
<tr><td><b>Type</b></td><td colspan="2">Query</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">None</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">Data Cleared</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">INDex:TIMe? <time></td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2"><time> = ASCII numeric value ranging from 0 to 1,099,511.62778 seconds</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">ASCII numeric value from 0 to 131,071
(0 to 524,287 if 512 k RAM option is installed)</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Index Time query reports the index number that corresponds with a specified time. If the specified time does not exactly match an event, then no index exists for that time and an error is generated</td></tr>
<tr><td rowspan="2"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td><code>IND:TIM? 0.011004</code></td><td>13</td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">INDex:TIMe:NEXT?
INDex:TIMe:PREVious?</td></tr>
</table>

INDex:TIMe:NEXT?

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Queries the next index which occurred after a given time</td></tr>
<tr><td><b>Type</b></td><td colspan="2">Query</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">None</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">Data Cleared</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">INDex:TIMe:NEXT? <time>[,<channel_list>]</td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2"><time> = ASCII numeric value ranging from 0 to 1,099,511.62778 seconds
<channel_list> = Standard channel list format supporting Channels 1 through 32 If the channel list is not supplied, all channels are used</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">ASCII numeric value from 0 to 131,071
(0 to 524,287 if 512 k RAM option is installed)</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Index Time Next query reports the next index number that occurs after a specified time. If the specified time exactly matches an event, then the next index is reported.

The optional channel list is provided to mask the search for the next index to include only events caused by one or more of the specified channels in the list.</td></tr>
<tr><td rowspan="3"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td><code>IND:TIM:NEXT? 0.011004</code></td><td>1024</td></tr>
<tr><td><code>IND:TIM:NEXT? 0.11004,(@1:8)</code></td><td></td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">INDex:TIMe?
INDex:TIMe:PREVious?</td></tr>
</table>

INDex:TIMe:PREVious?

<table>
<tr><td><b>Purpose</b></td><td>Queries the previous index which occurred before a given time</td></tr>
<tr><td><b>Type</b></td><td>Query</td></tr>
<tr><td><b>Command Syntax</b></td><td>None</td></tr>
<tr><td><b>Command Parameters</b></td><td>N/A</td></tr>
<tr><td><b>*RST Value</b></td><td>Data Cleared</td></tr>
<tr><td><b>Query Syntax</b></td><td>INDex:TIMe:PREVious? <time>[,<channel_list>]</td></tr>
<tr><td><b>Query Parameters</b></td><td><time> = ASCII numeric value ranging from 0 to 1,099,511.62778 seconds
<channel_list> = Standard channel list format supporting Channels 1 through 32 If the channel list is not supplied, all channels are used</td></tr>
<tr><td><b>Query Response</b></td><td>ASCII numeric value from 0 to 131,071
(0 to 524,287 if 512 k RAM option is installed)</td></tr>
<tr><td><b>Description</b></td><td>The Index Time Previous query reports the index number that corresponds with an event occurring before the specified time. If the specified time exactly matches an event, then the preceding index to that time is reported.

The optional channel list is provided to mask the search for the previous index to include only events caused by one or more of the specified channels in the list.</td></tr>
<tr><td><b>Examples</b></td><td><table>
<tr><th>Command / Query</th><th>Response <i>(Description)</i></th></tr>
<tr><td><code>IND:TIM:PREV? 0.011004</code></td><td>8192</td></tr>
<tr><td><code>IND:TIM:PREV? 0.011004,(@1:8)</code></td><td>8183</td></tr>
</table></td></tr>
<tr><td><b>Related Commands</b></td><td>INDex:TIMe?
INDex:TIMe:NEXT?</td></tr>
</table>

INPut:MASK

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Selects which channels will cause time stamp events to occur</td></tr>
<tr><td><b>Type</b></td><td colspan="2">Setting</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">INPut:MASK <boolean>[,<channel_list>]</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2"><boolean> = 0 | 1 | OFF | ON
<channel_list> = Standard channel list format supporting Channels 1 through 32 If the channel list is not supplied, all channels are used</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">All channels enabled</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">INPut:MASK? <channel></td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2"><channel> = 1 to 32</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">0 | 1</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Input Mask command controls which input channels are enabled/disabled to cause time stamp events to occur. An input channel must be enabled in order for that channel to cause a time stamp event to be recorded.

All channels which are disabled will have their input state recorded when a time stamp event occurs. This allows the VM1602 to act like a simple logic analyzer and only record levels when specific inputs make transitions.

It should be noted that a set bit in the capture RAM represents a transition (rising or falling edge as programmed) for an enabled channel, while a set bit represents a high logic level (or low level if programmed for inversion) on disabled (masked) channels.

The Input Mask query reports the enabled channels using the standard channel list format.

The *RST command enables all channels.</td></tr>
<tr><td rowspan="3"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td><code>INP:MASK 0,(@8:20)</code></td><td></td></tr>
<tr><td><code>INP:MASK? 13</code></td><td>0</td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">INPut:MASK:ENABle</td></tr>
</table>

INPut:MASK:ENABle

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Selects if masked channels will be used in time stamp data queries</td></tr>
<tr><td><b>Type</b></td><td colspan="2">Setting</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">INPut:MASK:ENABle <boolean&gt;</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">&lt;boolean> = 0 | 1 | OFF | ON</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">Enabled</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">INPut:MASK:ENABle?</td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2">None</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">0 | 1</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Input Mask Enable command controls whether masked (disabled) input channels are included in the data searches that are performed when data queries are requested.
If the input mask is enabled, then all disabled channels will not affect the response to queries made to the time stamp data. If the input mask is disabled, then a logic high in the time stamp data will be interpreted as an event and allows the user to determine when a disabled input logic level goes to the true state.
The *RST command enables the mask.
The module records time stamp events only when a transition occurs on any of the enabled channels. It records the particular instance of time when the event occurred. It also records the state of transition of the disabled channels. However, a transition occurring on a disabled channel does not cause any events. Events can be generated only on enabled channels. Since the transition state of disabled channels is also recorded when an event occurs on an enabled channel, the user can configure the module to include or exclude the events on the disabled channels while performing time stamp data queries using the INPut:MASK:ENABle command.</td></tr>
<tr><td><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td></td><td><code>INP:MASK:ENAB 1</code></td><td></td></tr>
<tr><td></td><td><code>INP:MASK:ENAB?</code></td><td>1</td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">INPut:MASK</td></tr>
</table>

INPut:POLarity

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Sets the input polarity of one or more channels</td></tr>
<tr><td><b>Type</b></td><td colspan="2">Setting</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">INPut: POLarity <polarity>[,<channel_list>]</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2"><polarity> = RISing | FALLing
<channel_list> = Standard channel list format supporting Channels 1 through 32 If the channel list is not supplied, all channels are used</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2"><polarity> = RISing
<channel_list> = All channels selected</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">INPut:POLarity? <channel></td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2"><channel> = 1 to 32</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">RIS | FALL</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Input Polarity command selects which edge of an input signal is recorded as an event. If a polarity of a channel is RISing, then a time stamp event will be recorded on the rising edge. If a polarity of a channel is FALLing, then the event will be recorded on the falling edge of the input signal.

The *RST command sets all channels for the rising edge.</td></tr>
<tr><td rowspan="5"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td><code>INP:POL FALL</code></td><td>1012.150000</td></tr>
<tr><td><code>INP:POL? 12</code></td><td>FALL</td></tr>
<tr><td><code>INP:POL RIS,(@1:16)</code></td><td></td></tr>
<tr><td><code>INP:POL? 1</code></td><td>RIS</td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">None</td></tr>
</table>

INPut:SOURce

Purpose	Sets the input source for a list of channels
Type	Setting
Command Syntax	INPut:SOURce <source>[,<channel_list>]
Command Parameters	<source> = FPAN | TTLTrig | ADJacent TTLTrig is only valid with odd-numbered channels ADJacent is only valid with even-numbered channels <channel_list> = Standard channel list format supporting Channels 1 through 32 If the channel list is not supplied, all channels are used
***RST Value**	<source> = FPAN <channel_list> = All channels selected
Query Syntax	INPut:SOURce? <channel>
Query Parameters	<channel> = 1 to 32
Query Response	ASCII string Odd channels respond with FPAN | TTLT Even channels respond with FPAN | ADJ
Description	The Input Source command selects the channel input source. All channels may come from the front panel. The odd numbered channels have the option of being sourced from the VXIbus TTL trigger bus. Channels 1 and 17 may source from TTL Trigger 0, while Channels 3 and 19 source from TTL Trigger 1, etc. This option is selected using the TTLTrig parameter. The even-numbered channels have the option of being sourced from the input connections of the channel above it (i.e., Channel 2 sources from Channel 1's inputs). This allows a single signal to be stamped on both rising and falling edges. This option is selected using the ADJacent parameter. The *RST command sets all channels to FPAN.
Examples	
Related Commands	None

Examples

Command / Query	Response *(Description)*
`INP:SOUR FPAN,(@1:3)`	
`INP:SOUR? 2`	PFAN

INPut:TYPE

Purpose	Sets the input for either differential or single-ended mode of operation	
Type	Setting	
Command Syntax	INPut:TYPE < type>[,<channel_list>]	
Command Parameters	<type> = DIFFerential | SINGle <channel_list> = Standard channel list format supporting Channels 1 through 32 If the channel list is not supplied, all channels are used	
***RST Value**	<type> = SINGle <channel_list> = All channels selected	
Query Syntax	INPut:TYPE? <channel>	
Query Parameters	<channel> = 1 to 32	
Query Response	DIFF | SING	
Description	The Input Type command selects differential or single-ended mode for a specified input. In differential mode, the inputs switch state when the voltage on one input side exceeds the other input side (i.e., a high level is recorded when the plus input of a channel is at a more positive voltage than the negative input of the same channel). In single-ended mode, the negative inputs are not used and the positive inputs are compared to an internal voltage generated by an 8-bit DAC. Any input that exceeds the programmed threshold voltage is recorded as a high state and any input below the threshold is recorded as a low state. The *RST command sets all inputs to the single-ended mode.	
Examples	**Command / Query**	**Response *(Description)***
	`INP:TYP SING (@12:24)`	
	`INP:TYPE? 15`	SING
Related Commands	None	

INITiate

Purpose	Enables the collection of time stamp events	
Type	Event	
Command Syntax	INITiate[:IMMediate]	
Command Parameters	None	
***RST Value**	N/A	
Query Syntax	None	
Query Parameters	N/A	
Query Response	N/A	
Description	The Initiate command starts the process of collecting time stamp events. The Initiate command clears the counters and registers on the VM1602. After power up or after an Abort command, the VM1602 will not start measuring time or recording events until the Initiate command is received. This allows for other equipment in the system to be set up before recording data and prevents the collection of unnecessary data.	
Examples	**Command / Query**	**Response *(Description)***
	`INIT`	
Related Commands	ABORt	

MFGTEST:MEMory?

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Reports the memory available on the VM1602</td></tr>
<tr><td><b>Type</b></td><td colspan="2">Query</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">None</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">MFGTEST:MEMory?</td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2">None</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">131071 | 524287</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The MFGTEST:MEMory query allows the user to determine the memory option installed on the VM1602. The standard unit has 128 k (131071), and the available addition memory option provides 512 k (524287).</td></tr>
<tr><td rowspan="2"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td><code>MFGTEST:MEM?</code></td><td>131071</td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">N/A</td></tr>
</table>

SWEep:STEP

<table>
<tr><td><b>Purpose</b></td><td>Sets the time stamp resolution period</td></tr>
<tr><td><b>Type</b></td><td>Setting</td></tr>
<tr><td><b>Command Syntax</b></td><td>SWEep:STEP <time_interval></td></tr>
<tr><td><b>Command Parameters</b></td><td><time_interval> = 1E-3 | 1E-4 | 1E-5 | 1E-6</td></tr>
<tr><td><b>*RST Value</b></td><td>1 μs</td></tr>
<tr><td><b>Query Syntax</b></td><td>SWEep:STEP?</td></tr>
<tr><td><b>Query Parameters</b></td><td>None</td></tr>
<tr><td><b>Query Response</b></td><td>1E-3 | 1E-4 | 1E-5 | 1E-6</td></tr>
<tr><td><b>Description</b></td><td>The Sweep Step command sets the sample rate of the time stamp module. The sample rate sets the time interval that the time stamps can resolve, and all the times reported have an uncertainty equal to the sample rate (i.e., with a 10 μs sample rate, a reported time will have a 0 to 10 μs uncertainty relative to another event).

The purpose of reducing the sample rate from the highest resolution is to reduce the amount of data to consider, and to increase the amount of time before the internal time stamp clock rolls over. The time stamp clock rollover periods are as follows:

<table>
<tr><th>Step Size</th><th>Roll Over Period</th></tr>
<tr><td>1 μs</td><td>12.7 days</td></tr>
<tr><td>10 μs</td><td>127 days</td></tr>
<tr><td>100 μs</td><td>3.479 years</td></tr>
<tr><td>1 ms</td><td>34.79 years</td></tr>
</table>

The *RST command sets the time interval to 1E-6 seconds.</td></tr>
<tr><td><b>Examples</b></td><td>
<table>
<tr><th>Command / Query</th><th>Response <i>(Description)</i></th></tr>
<tr><td><code>SWE:STEP 1E-3</code></td><td></td></tr>
<tr><td><code>SWE:STEP?</code></td><td>0.001000</td></tr>
</table>
</td></tr>
<tr><td><b>Related Commands</b></td><td>None</td></tr>
</table>

SYNC

<table>
<tr><td><b>Purpose</b></td><td>Synchronizes multiple VM1602 boards within a VMIP host</td></tr>
<tr><td><b>Type</b></td><td>Setting</td></tr>
<tr><td><b>Command Syntax</b></td><td>SYNC <type></td></tr>
<tr><td><b>Command Parameters</b></td><td><type> = STANdalone | MASTer | SLAVe</td></tr>
<tr><td><b>*RST Value</b></td><td>STANdalone</td></tr>
<tr><td><b>Query Syntax</b></td><td>SYNC?</td></tr>
<tr><td><b>Query Parameters</b></td><td>None</td></tr>
<tr><td><b>Query Response</b></td><td>STAN | MAST | SLAV</td></tr>
<tr><td><b>Description</b></td><td>The Sync command allows up to three VM1602 modules to be synchronized within a VMIP host. If only one VM1602 module is installed, or if each module should function independently, the default standalone parameter is used.

If two or three 32-channel VM1602 modules are to be used together, where all 64 or 96 channels are to be in sync in terms of sample time and start time, then one of the instruments must be set to Master mode, and all other modules set to Slave mode.

In order to use the instruments in the Master/Slave setup, all instruments to be synced must first be set up as Slaves. The instrument to be set up as the Master will set the sample rate for all Slaves, but first, the SWEep:STEP needs to be set at the same value for all the units together.

All instruments must then be initiated, and then the instrument to be used as the Master is set to Master to begin the measurement cycle. No data will be collected until the Master unit is set. Data collection will continue until all instruments are aborted.

It should be noted that the memory resource for each instrument is used independently. This means that one instrument is likely to deplete its RAM resource before the other instruments. It also means that a given index for one group of 32 channels will not correspond to the same index for another group. The time between events caused by channels on different 32-channel groups must be calculated in time and not referenced by their indices. See Section 3 for more detail on synchronizing multiple instruments.

The *RST command sets the SYNC to STANdalone mode.</td></tr>
<tr><td><b>Examples</b></td><td><table><tr><th>Command / Query</th><th>Response <i>(Description)</i></th></tr><tr><td>SYNC MAST</td><td></td></tr></table></td></tr>
<tr><td><b>Related Commands</b></td><td>ABORt
INITiate</td></tr>
</table>

TIMe:DATA?

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Queries the time stamp data for one of more indices</td></tr>
<tr><td><b>Type</b></td><td colspan="2">Query</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">None</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">Data Cleared</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">TIMe:DATA? <index_1>[,<index_2>]</td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2"><index_1> = 0 to 131,071 (0 to 524,287 if 512 k RAM option is installed)
<index_2> = 0 to 131,071 (0 to 524,287 if 512 k RAM option is installed)
A value of -1 indexes to the end of the data set (the end of RAM or the last recorded event, whichever is less)</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">ASCII numeric value ranging from 0.000000 to 1,099,511.627780 seconds</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Time Data query reports the time stamp data (when the specific time stamp occurred) for a single index or series of indices. The time reported is relative to when the instrument was initiated and is in a fixed format numeric value. The time value is reported in seconds with 1 μs resolution, regardless of the time stamp interval selected.

It should be noted that the time stamp counter is reset with each initiate or *TRG command. Therefore, there is no correlation between independent data collection runs. The time data is intended to be used to determine the time between events within a single collection run. The time data may be used between instruments if they are synchronized and is actually the only way to correlate events between synchronized instruments.</td></tr>
<tr><td rowspan="2"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td><code>TIM:DATA? 5</code>
<code>TIM:DATA? 5,7</code></td><td>0.000123
0.00123,0.002005,0.101134</td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">None</td></tr>
</table>

TIMe:DELTa?

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Queries the time difference between time stamp indices</td></tr>
<tr><td><b>Type</b></td><td colspan="2">Query</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">None</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">Data Cleared</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">TIMe:DELTa? <index_1>,<index_2></td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2"><index_1> = 0 to 131,071 (0 to 524,287 if 512 k RAM option is installed)
<index_2> = 0 to 131,071 (0 to 524,287 if 512 k RAM option is installed).
A value of -1 indexes to the end of the data set (the end of RAM or the
last recorded event, whichever is less)</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">ASCII numeric value ranging from 0.000000 to 1,099,511.627780 seconds</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Time Delta query reports the time period between two time stamp indices. The time is reported in seconds with 1 μs resolution, regardless of the time stamp interval selected. The response is represented in a fixed format numeric value.</td></tr>
<tr><td rowspan="2"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td><code>TIM:DELT? 5,10325</code></td><td>121.0001234</td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">FREQuency:DELTa?</td></tr>
</table>

TRIGger:LEVel

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Sets the trigger level for a group of four channels</td></tr>
<tr><td><b>Type</b></td><td colspan="2">Setting</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">TRIGger:LEVel <voltage>[,<channel_list>]</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2"><voltage> = -5.0 to +4.96
<channel_list> = Standard channel list format supporting Channels 1, 5, 9, 13, 17, 21, 25 and 29 only</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2"><voltage> = +1.79 V
<channel_list> = All channels selected</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">TRIGger:LEVel? <channel></td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2"><channel> = 1 to 32</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">ASCII numeric value from -5.0 to +5.0 volts for single-ended channels
OFF for channels which are differential</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Trigger Level command sets the trigger threshold for a group of four channels. Because each group of four consecutive channels share a common threshold DAC, it is only necessary to set eight levels. To avoid redundancy of setting thresholds, this command only responds to the first channel number of a group.
Note that any channel may be queried for its threshold, but only specific channels may be programmed. Also, note that the trigger level applies only to channels that are set for single-ended operation. Differential inputs do not use the threshold DACs, and switch state when the voltage on one input side exceeds the other input side (i.e., a high level is recorded when the plus input of a channel is at a more positive voltage than the negative input of the same channel).
The *RST command will set the threshold to +1.79 volts for compatibility with standard TTL logic.</td></tr>
<tr><td rowspan="2"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td><code>TRIG:LEV 1.68,(@1,5,9)</code>
<code>TRIG:LEV? 5</code></td><td>
1.68</td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">None</td></tr>
</table>

SCPI REQUIRED COMMANDS

STATus:OPERation:CONDition?

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Queries the Operation Status Condition Register</td></tr>
<tr><td><b>Type</b></td><td colspan="2">Required SCPI command</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">None – Query Only</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">STATus:OPERation:CONDition?</td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2">None</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">Bit 0 - Calibrating - not used on this instrument
Bit 1 - Settling - not used on this instrument
Bit 2 - Ranging - not used on this instrument
Bit 3 - Sweeping - not used on this instrument
Bit 4 - Measuring
Bit 5 - Waiting for trigger
Bit 6 - Waiting for arm
Bit 7 - Correcting - not used on this instrument</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Operation Status Condition Register query is provided for SCPI compliance.</td></tr>
<tr><td rowspan="2"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td><code>STAT:OPER:COND?</code></td><td>0</td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">None</td></tr>
</table>

STATus:OPERation:ENABle

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Sets the Operation Status Enable Register</td></tr>
<tr><td><b>Type</b></td><td colspan="2">Required SCPI command</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">STATus:OPERation:ENABle <NRf></td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2"><NRf> = numeric ASCII value from 0 to 32767</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">NRf must be specified</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">STATus:OPERation:ENABle?</td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2">None</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">Numeric ASCII value from 0 to 32767</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Operation Status Enable Register is included for SCPI compatibility and the VM1602 does not alter any of the bits in this register. The register layout is as follows:

Bit 0 - Calibrating
Bit 1 - Setting
Bit 2 - Ranging
Bit 3 - Sweeping
Bit 4 - Measuring
Bit 5 - Waiting for trigger
Bit 6 - Waiting for arm
Bit 7 - Correcting</td></tr>
<tr><td rowspan="2"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td><code>STAT:OPER:ENAB 0</code></td><td>0</td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">None</td></tr>
</table>

STATus:OPERation[:EVENt]?

Purpose	Queries the Operation Status Event Register	
Type	Required SCPI command	
Command Syntax	None – Query Only	
Command Parameters	N/A	
***RST Value**	N/A	
Query Syntax	STATus:OPERation[:EVENt]?	
Query Parameters	None	
Query Response	0	
Description	The Status Operation Event Register query is included for SCPI compliance.	
Examples	**Command / Query**	**Response *(Description)***
	`STAT:OPER?`	
Related Commands	None	

STATus:PRESet

Purpose	Presets the Status Registers	
Type	Required SCPI command	
Command Syntax	STATus:PRESet	
Command Parameters	None	
***RST Value**	N/A	
Query Syntax	None – Command Only	
Query Parameters	N/A	
Query Response	N/A	
Description	The Status Preset command presets the Status Registers. The Operational Status Enable Register is set to 0 and the Questionable Status Enable Register is set to 0. This command is provided for SCPI compliance only.	
Examples	**Command / Query**	**Response *(Description)***
	`STAT:PRES`	*(Presets the Status Registers)*
Related Commands	None	

STATus:QUEStionable:CONDition?

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Queries the Questionable Status Condition Register</td></tr>
<tr><td><b>Type</b></td><td colspan="2">Required SCPI command</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">None – Query Only</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">STATus:QUEStionable:CONDition?</td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2">None</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">0</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The Questionable Status Condition Register query is provided for SCPI compliance only. The VM1602 does not alter any of the bits in this register and a query always reports a 0.</td></tr>
<tr><td rowspan="2"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td><code>STAT:QUES:COND?</code></td><td>0</td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">None</td></tr>
</table>

STATus:QUEStionable:ENABle

<table>
<tr><td><b>Purpose</b></td><td>Sets the Questionable Status Enable Register</td></tr>
<tr><td><b>Type</b></td><td>Required SCPI command</td></tr>
<tr><td><b>Command Syntax</b></td><td>STATus:QUEStionable:ENABle <NRf></td></tr>
<tr><td><b>Command Parameters</b></td><td><NRf> = numeric ASCII value from 0 to 32767</td></tr>
<tr><td><b>*RST Value</b></td><td>NRf must be supplied</td></tr>
<tr><td><b>Query Syntax</b></td><td>STATus:QUEStionable:ENABle?</td></tr>
<tr><td><b>Query Parameters</b></td><td>None</td></tr>
<tr><td><b>Query Response</b></td><td>Numeric ASCII value from 0 to 32767</td></tr>
<tr><td><b>Description</b></td><td>The Status Questionable Enable command sets the bits in the Questionable Status Enable Register. This command is provided only to comply with the SCPI standard.

The Status Questionable Enable query reports the contents of the Questionable Status Enable Register. The VM1602 does not alter the bit settings of this register and will report the last programmed value.</td></tr>
<tr><td><b>Examples</b></td><td><table>
<tr><th>Command / Query</th><th>Response <i>(Description)</i></th></tr>
<tr><td><code>STAT:QUES:ENAB 64</code></td><td></td></tr>
<tr><td><code>STAT:QUES:ENAB?</code></td><td>64</td></tr>
</table></td></tr>
<tr><td><b>Related Commands</b></td><td>None</td></tr>
</table>

STATus:QUEStionable[:EVENt]

Purpose	Queries the Questionable Status Event Register	
Type	Required SCPI command	
Command Syntax	None – Query Only	
Command Parameters	N/A	
***RST Value**	N/A	
Query Syntax	STATus:QUEStionable[:EVENt]?	
Query Parameters	None	
Query Response	0	
Description	The Questionable Status Event Register is provided for SCPI compliance only. The VM1602 does not alter the bits in this register and queries always report a 0.	
Examples	**Command / Query**	**Response *(Description)***
	`STAT:QUES?`	0
Related Commands	None	

SYSTem:ERRor?

<table>
<tr><td><b>Purpose</b></td><td colspan="2">Queries the Error Queue</td></tr>
<tr><td><b>Type</b></td><td colspan="2">Required SCPI command</td></tr>
<tr><td><b>Command Syntax</b></td><td colspan="2">None – Query Only</td></tr>
<tr><td><b>Command Parameters</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>*RST Value</b></td><td colspan="2">N/A</td></tr>
<tr><td><b>Query Syntax</b></td><td colspan="2">SYSTem:ERRor?</td></tr>
<tr><td><b>Query Parameters</b></td><td colspan="2">None</td></tr>
<tr><td><b>Query Response</b></td><td colspan="2">ASCII string</td></tr>
<tr><td><b>Description</b></td><td colspan="2">The System Error query is used to retrieve error messages from the error queue. The error queue will maintain the two error messages. If additional errors occur, the queue will overflow and the second and subsequent error messages will be lost. In the case of an overflow, an overflow message will replace the second error message. See the SCPI standard Volume 2: Command Reference for details on errors and reporting them. Refer to the “Error Messages” section of this manual for specific details regarding the reported errors.</td></tr>
<tr><td rowspan="2"><b>Examples</b></td><td><b>Command / Query</b></td><td><b>Response <i>(Description)</i></b></td></tr>
<tr><td><code>SYST:ERR?</code></td><td>-350, “Queue overflow”</td></tr>
<tr><td><b>Related Commands</b></td><td colspan="2">None</td></tr>
</table>

SYSTem:VERSion?

Purpose	Queries the SCPI version number to which the VM1602 complies	
Type	Required SCPI command	
Command Syntax	None – Query Only	
Command Parameters	N/A	
***RST Value**	N/A	
Query Syntax	SYSTem:VERSion?	
Query Parameters	None	
Query Response	Numeric ASCII value	
Description	The System Version query reports version of the SCPI standard to which the VM1602 complies.	
Examples	**Command / Query**	**Response *(Description)***
	`SYST:VERS?`	1994.0
Related Commands	None	

SECTION 5

THEORY OF OPERATION

INTRODUCTION

The VM1602 is a high-performance Time Stamp module that has been designed for accurate relative time measurements and flexibility of configuration, see Figure 5-1. The instrument uses the message-based word serial interface for programming and data retrieval.

The instrument monitors 32 inputs for either rising or falling edges as programmed and records their relative time of occurrence.

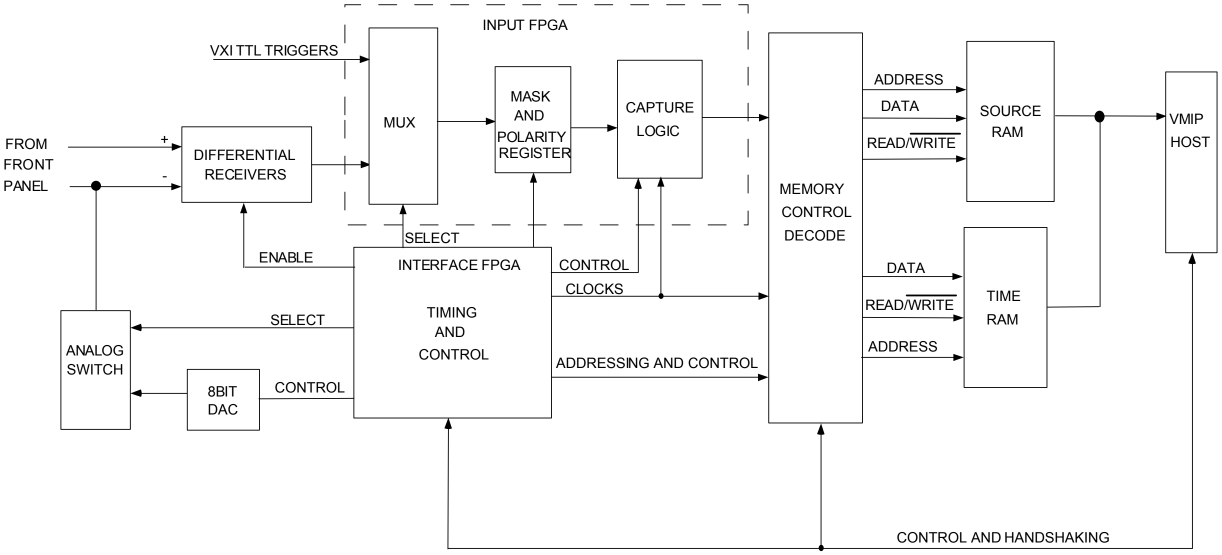


FIGURE 5-1 VM1602 FUNCTIONAL BLOCK DIAGRAM

In addition to accepting inputs from the front panel, the VM1602 can also monitor all the VXIbus TTL Trigger bus lines. To keep the hardware to a reasonable level of complexity, TTL Trigger 0 may be routed to Channels 1, 2, 17 or 18. Similarly, TTL Trigger 1 may be routed to Channels 3, 4, 19, 20, etc. To further enhance the flexibility of the instrument, all even-numbered channels may be routed to their N-1 channel input. This allows both the rising and falling edges of a signal to be time stamped. For example, Channel 5 may be routed to TTL Trigger bus 2 and set to stamp on the rising edge and Channel 6 may be routed to Channel 5 and set to stamp on the falling edge. In this case, both rising and falling edges of the TTL Trigger bus 2 will be recorded.

The VM1602 allows each channel to be enabled or disabled. This prevents unused channels that may have signal activity from creating unwanted time stamp events. When a channel is enabled, a 1 value is recorded in RAM in the corresponding channel bit position every time an event occurs on that channel (a rising or falling edge as is programmed for that channel). The firmware of the instrument scans the RAM's data and reports relative times between events on the same or different channels. When a channel is disabled, the logic state of the input is recorded whenever an enabled channel has a time stamp event. This allows the VM1602 to function as a rudimentary logic analyzer. The time and event data are available to the user through the VMIP message-based interface.

Each time an enabled channel has a time stamp event, the value in a 40-bit counter is recorded, along with the channel or channels that caused the event to occur. The rate at which the counter increments sets the overall resolution of the instrument and may be programmed for 1 μs, 10 μs, 100 μs or 1 ms.

INPUT FPGA

All channel pairs operate identically; therefore, only Channel 1 and 2 are shown in Figure 5-2.

By using op amps as input receivers, the VM1602 can receive either differential or single-ended signals, see Figure 5-2. A Digital-to-analog Converter (DAC) is tied to the negative or inverting input of the receiver to provide signal threshold. The DAC has an output of +2.5 V to -2.5 V and a *RST Value of +1.8 V. The capture FPGA selects an input signal for triggering an event from the front panel, the VXI TTL triggers, or the front panel input's adjacent channel. This is accomplished through a pair of muxes that are user definable.

The signal is then verified to be the proper polarity. This is done by "exclusively-oring" the input signal with the user defined rising or falling polarity command bit. The signal is then routed through a mask register. If the signal is masked it will not be capable of generating an event trigger, but will be stored in RAM if another non-masked signal triggers an event. If the signal is indeed the proper polarity and enabled (non-masked) it will be synchronized to the user-defined clock through a series of latches and captured. The capture FPGA notifies the interface FPGA that an event has occurred by generating the BEENHIT signal. The data is latched and output onto the data bus.

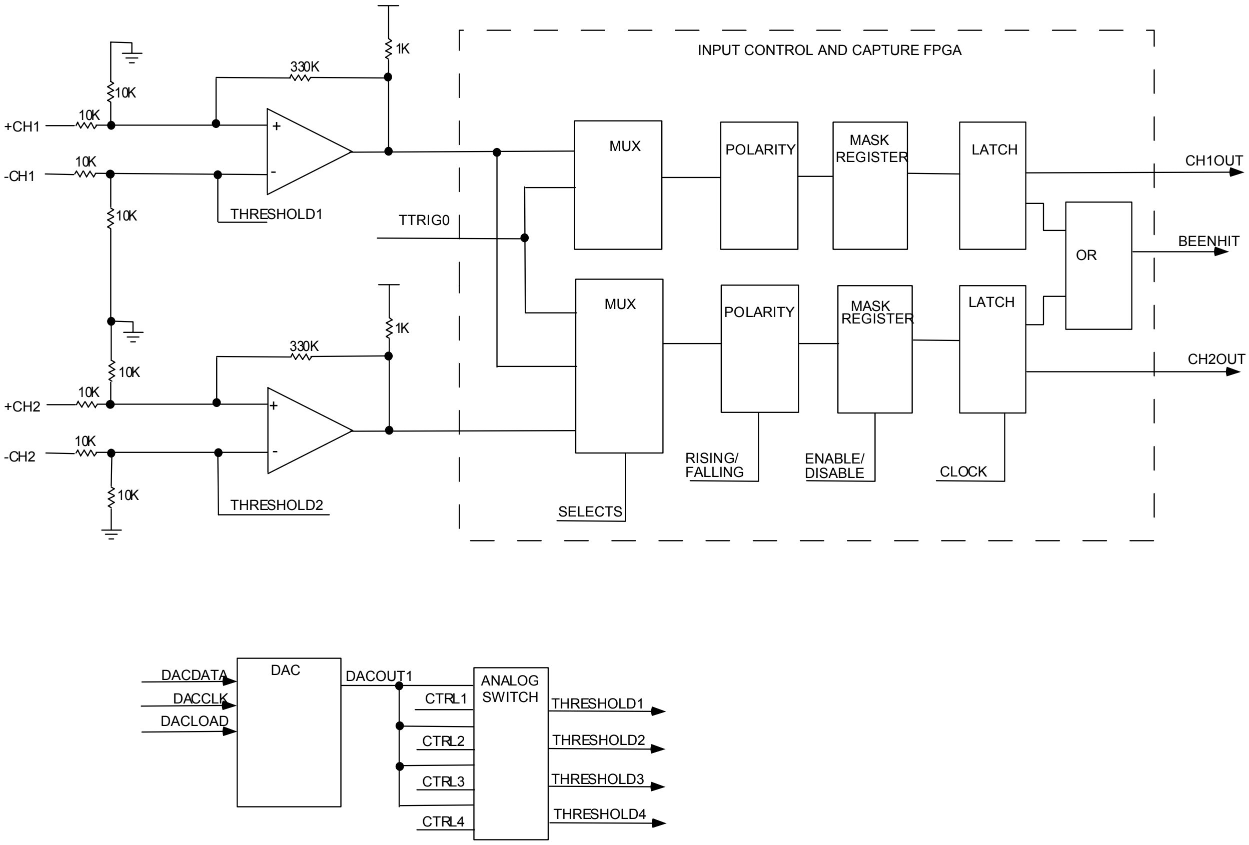


FIGURE 5-2 INPUT CONTROL AND CAPTURE FPGA BLOCK DIAGRAM

INTERFACE FPGA

The interface FPGA generates the write enables, addresses and strobes to the appropriate memories for logging of the event time and data; see Figure 5-3. The event time is calculated by a 40-bit counter that is started when the VM1602 is initialized. The interface FPGA also controls all functions needed to interface with the VMIP host. When the VM1602 operation is aborted the event time and data is available to the user through the VMIP CPU.

The interface FPGA receives the BEENHIT signal from the capture FPGA signifying that an event has occurred. The memory control logic will verify that memory space exists based on the last address signal. This signal is generated by the address register logic when the last address has been reached. If the last memory location has not been reached, the memory control logic will generate the next signal that will increment the address register. The memory control logic will then generate write enables to the event and time stamp data registers. The memory control logic then generates the control word that is decoded by the memory read/write decoder. This decoder provides the write and output enables to the source and time stamp RAMs based on the decoder's input. The time stamp and event data is recorded into their respective memories and is available to the CPU upon completion or termination of the program being executed.

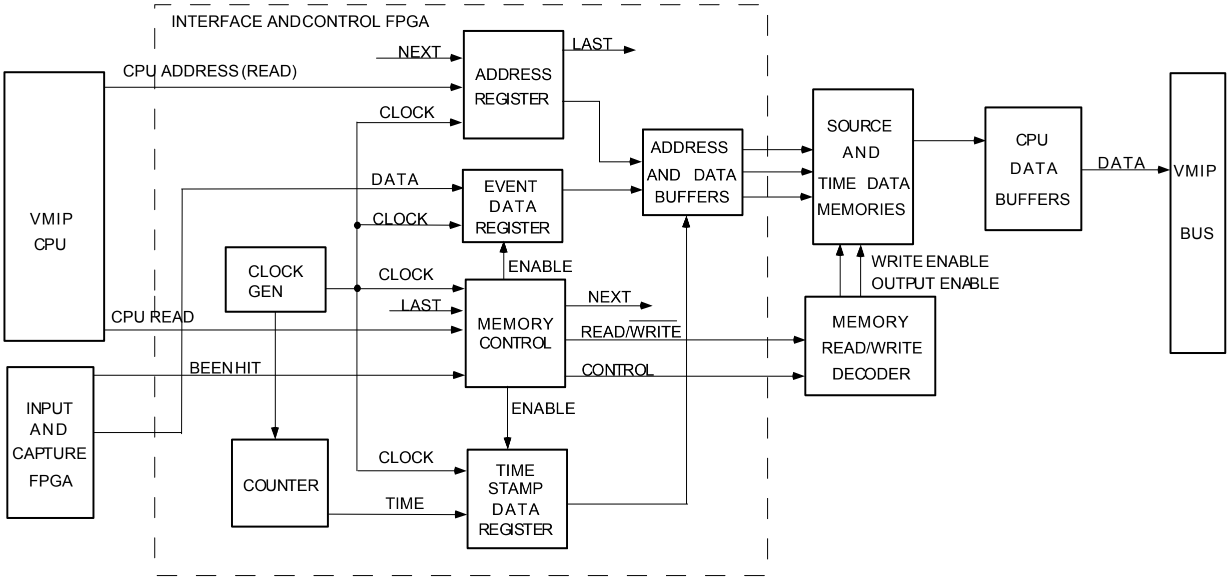


FIGURE 5-3 INTERFACE AND CONTROL FPGA

DATA RETRIEVAL

The CPU notifies the interface FPGA which data is requested and at what address. The address register is loaded with the requested address and the memory control logic generates the proper control word to the read/write decoder. The read/write decoder generates an output enable and the data is placed onto the data bus. The interface FPGA generates the enable and clock signal to latch the data into the CPU data buffer.

INDEX